Ruprecht-Karls-Universität Heidelberg
Institut für Informatik
Visual Computing Group

Bachelor Thesis

Visualization of Streamline Distributions in Uncertain 2D Vector Fields

Name: Philip Hausner
Matrikelnummer: 3220550
Betreuer: Prof. Dr. Filip Sadlo
Datum der Abgabe: 18.04.2018

Ich versichere hiermit, dass Ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit ist in gleicher oder vergleichbarer Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Heidelberg, den 18. April 2018 _____

Zusammenfassung

Stromlinien sind ein geläufiges Visualisierungskonzept, um die Struktur von Vektorfeldern zu beschreiben. Insbesondere in der numerischen Strömungsdynamik werden sie verwendet. Mit der Einführung von Unsicherheit in Vektorfeldern werden neue Techniken für die auf Stromlinien basierende Visualisierung benötigt. Mithilfe von zweidimensionalen Verteilungsgittern verfolgen wir Partikeln in einem zellbasierenden Gitter. Dabei starten wir mit einer vorgegebenen Partikeldichte in einer einzigen Zelle im ersten Zeitschritt. Außerdem stellen wir eine neue Art von Unsicherheit, sogenannte Domänenunsicherheit, vor. Während unsichere Vektorfelder eine Wahrscheinlichkeitsfunktion von Vektoren an jeder Position speichern, definieren domänen-unsichere Daten einen eindeutigen sicheren Vektor an jeder Position, jedoch ist die Position selbst unsicher. In dieser Arbeit werden domänen-unsichere Daten definiert und Probleme, die beim Arbeiten mit solchen Daten aufkommen, werden näher diskutiert. Zudem analysieren wir das Verhalten von unsicheren Stromlinien und das von Stromlinien, welche auf einer unsicheren Domäne definiert sind. Danach machen wir Gebrauch von Verteilungsgittern, um die Struktur von Stromlinien hinsichtlich verschiedener zweidimensionaler unsicherer Vektorfelder zu analysieren und wenden unsere Methode auf einen Testdatensatz an. Schlussenedlich integrieren wir dies in einen Raum-Zeit-Würfel, um eine überblicksvisualsierung von 2D Stromlinien auf unischeren Vektorfeldern zu erstellen.

Abstract

Streamlines are a common visualization concept to describe the structure of vector fields, especially in computational fluid dynamics. With the introduction of uncertainty to vector fields, new techniques are required for streamline-based visualization. With the help of two-dimensional distribution fields, we track the particle density distributions on a cell-based grid over time, starting with a fixed particle density in a single cell. Additionally, we investigate a new aspect of uncertainty, so called domain-uncertainty. While uncertain vector fields store a probability distribution of vectors at every position, domain-uncertain data defines a unique certain vector at every position, however, the position itself is uncertain. In this thesis, domain-uncertain data is defined and problems that arise when dealing with such data are discussed. Furthermore, we analyse the behaviour of uncertain streamlines and of streamlines that are defined on an uncertain domain. Subsequently, we make use of distribution grids to analyse the structure of streamlines in regard to different, two-dimensional uncertain vector fields and apply our method to a test data set. Eventually, we employ space-time representation to provide an at-a-glance visualization of 2D streamlines in uncertain vector fields.

Contents

1 Introduction

1.1 Motivation

Streamlines are an important tool to visualize vector fields in flow visualization to analyse diverse real-world processes, like the flow behaviour around a car in a wind tunnel, or the structure of a sea current. In this context, many researchers regard the visualization of uncertainty as one of the most important topics in the field of scientific visualization. However, most previous work focused on the visualization of local uncertainties by designing special glyphs to indicate noise, randomness of a process, or simulation errors. Only recently also the notion of the transport of uncertainty in vector fields attracted the interest of the scientific community, and new approaches were developed to track the behaviour of moving particles in uncertain vector fields. Yet, in general not only the topological features of vector fields are of interest but also the behaviour of particles over time. Additionally, in previous studies only the uncertainty of the data was considered, disregarding the fact that the underlying domain can be uncertain, too. This implies that in a measurement, the data may be certain while the position or time, at which the measurement was made, may be uncertain. This thesis has the goal to help in closing this gap and to give a detailed explanation on how to track particles in uncertain vector fields.

1.2 Goals of this Thesis

This thesis aims to describe the behaviour of particles and particle densities in 2D uncertain vector fields, as well as to introduce the concept of domain-uncertain data. Therefore, we first track single particles in an uncertain vector field; and afterwards expand this concept to uncertain vector fields that are defined on an uncertain domain, i.e., that our data is not stored at certain points but has only a probability to be at certain positions. Furthermore, the distribution of particles in an uncertain vector field is traced, and the

behaviour of particle distributions in different uncertain vector fields is compared. In this context, we consider streamline distributions in uncertain vector fields in contrast to streamlines in certain vector fields. The contributions of this work include:

- The introduction of domain-uncertain data independent from the concept of uncertain vector fields.
- A tracking of particle densities in different uncertain vector fields.
- An at-a-glance representation of streamline distributions in 2D uncertain vector fields.

1.3 Structure

Chapter 2 presents the fundamental tools for this work. Besides a short introduction to probability theory and necessary statistics, Chapter 2 also highlights the fundamental concepts of scalar fields and vector fields. Hereby, we will differentiate between certain and uncertain vector fields. Finally, an overview of previous work is presented. Chapter 3 gives an introduction to uncertainty and where it stems from, then traces single particles in uncertain vector fields and concludes with the concept of domain-uncertain data. In Chapter 4, the approach is expanded to particle distributions and an at-a-glance visualization of streamline distributions in uncertain vector fields is presented. With the means of Chapter 3 and Chapter 4, plugins are provided for *ParaView* [1]; and with the help of these plugins, in Chapter 5, the results of this work are visualized and discussed. Chapter 6 concludes this work and gives an outlook on potential future projects.

2 Fundamentals

This chapter introduces the fundamentals necessary for this work. Firstly, an introduction to probability theory is given. Afterwards typical tools and methods used in scientific visualization are presented.

2.1 Probability Theory and Statistics

In the following statistics section, a countable *sample space* $\Omega = \{\omega_1, \ldots, \omega_n\}$ is assumed where elements of Ω represent outcomes of a random experiment. Furthermore, a subset of Ω is called an *event*.

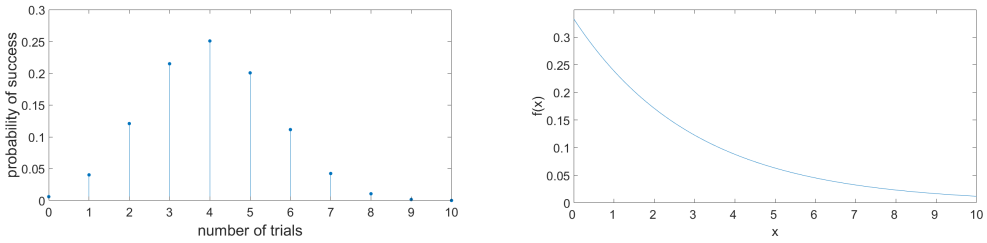
2.1.1 Probability Axioms

Let P be a function $P : \mathcal{P}(\Omega) \to \mathbb{R}$ that assigns a real number to each event E. Instead of $\mathcal{P}(\Omega)$ any Sigma-algebra on Ω is valid, however, in this work we restrict the definition of P to the power set, and do not introduce Sigma-algebras. P is called a *probability function* if it satisfies the following three *axioms of probability*, often also referred to as Kolmogorov axioms:

1. $P(E) \geq 0, \forall\, E \subseteq \Omega$,
2. $P(\Omega) = 1$,
3. If $E_1, E_2, \ldots$ is a countable sequence of disjoint events, then

$$P\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} P(E_i).$$

For events $A, B \subseteq \Omega$, the following properties of probability functions can be deducted from these axioms:



(a) (b)

Figure 2.1: (a) Probability function of the binomial distribution with probability $p = 0.4$ and $n = 10$ trials, (b) probability density function of exponential distribution with $\lambda = 3$.

- $P(\varnothing) = 0$,
- $P(A^c) = 1 - P(A) \quad$ with $A^c = \Omega \setminus A$,
- $A \subset B \Rightarrow P(A) \leq P(B)$,
- $A \cap B = \varnothing \Rightarrow P(A \cup B) = P(A) + P(B)$,
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

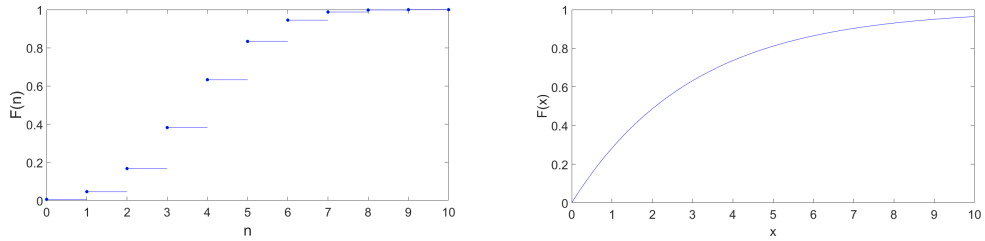
2.1.2 Random Variables

A *random variable* is defined as a mapping $X : \Omega \to E$ that assigns a certain probability to each event. In this work, E is always assumed to be $\mathbb{R}$, however, in general, any measurable space is valid. The common notation to indicate that X has a certain distribution D is $X \sim D$.

We can furthermore differentiate between *discrete* and *continuous* random variables. Discrete random variables take on only a countable number of values. The probability function of a discrete random variable is then defined as

$$f_X(x) = P(X = x),$$

with $x \in \mathbb{R}$. Figure 2.1a shows the probability function of a discrete random variable, the binomial distribution.



(a) (b)

Figure 2.2: (a) Cumulative distribution function (CDF) of the binomial distribution with probability $p = 0.4$ and $n = 10$ trials. (b) CDF of exponential distribution with $\lambda = 3$. Both illustrations show the CDFs corresponding to the probability functions in Figure 2.1 with the same bounds.

A continuous random variable X takes on an uncountably infinite number of values. It has the following properties:

- $\exists\, f_X$ s.t. $f_X(x) \geq 0 \quad \forall\, x \in \mathbb{R}$,
- $\int\limits_{x \in \mathbb{R}} f_X(x)\, \mathrm{d}x = 1$,
- $P(a < X < b) = \int\limits_a^b f_X\, \mathrm{d}x \quad \forall\, a \leq b$,
- $P(X = x) = 0 \quad \forall\, x \in \mathbb{R}$.

In case of a continuous random variable, f_X is called the *probability density function* (PDF). Figure 2.1b shows the PDF of the exponential distribution, a common continuous distribution.

2.1.3 Cumulative Distribution Function

The *cumulative distribution function* (CDF) of a given real-valued random variable X is a function $F_X : \mathbb{R} \to [0, 1]$ defined by:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(t)\,\mathrm{d}t. \tag{2.1}$$

Given the CDF it is uncomplicated to determine the probability that X lies in a certain interval $[a, b]$:

$$P(a \leq X \leq b) = F_X(b) - F_X(a). \tag{2.2}$$

Figure 2.2 shows the CDFs corresponding to the probability functions in Figure 2.1.

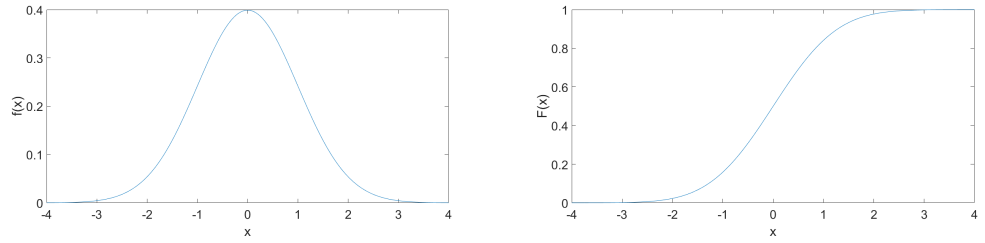
2.1.4 Normal Distribution

One of the most commonly used continuous distributions is the *normal* (or *Gaussian*) *distribution*. Its importance stems from observations found in natural and social sciences where it is often the best model available to describe certain phenomena since it has a good trade-off between accuracy and complexity. The PDF of the Gaussian distribution is:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \forall\, x \in \mathbb{R}. \tag{2.3}$$

Here μ denotes the *mean* of the distribution, σ its *standard deviation*, and σ^2 its *variance*. To indicate that a random variable X is normally distributed, we write $X \sim N(\mu, \sigma^2)$. The normal distribution has a few important key characteristics. Firstly, it is axially symmetrical around μ, i.e., $f(x - \mu) = f(-(x - \mu))$, where it is maximal. This leads to the typical bell curve of its probability density function which can be observed in Figure 2.3a. Furthermore, values drawn from a Gaussian distribution have a probability of about

- $68,3\%$ to lie in the interval $[\mu - \sigma, \mu + \sigma]$,
- $95,4\%$ to lie in the interval $[\mu - 2\sigma, \mu + 2\sigma]$,
- $99,7\%$ to lie in the interval $[\mu - 3\sigma, \mu + 3\sigma]$.



(a) (b)

Figure 2.3: (a) PDF and (b) CDF of the normal distribution with parameters $\mu = 0$ and $\sigma = 1$.

The normal distribution's CDF is

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \mathrm{d}t \tag{2.4}$$

and especially

$$\frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \mathrm{d}t = 1. \tag{2.5}$$

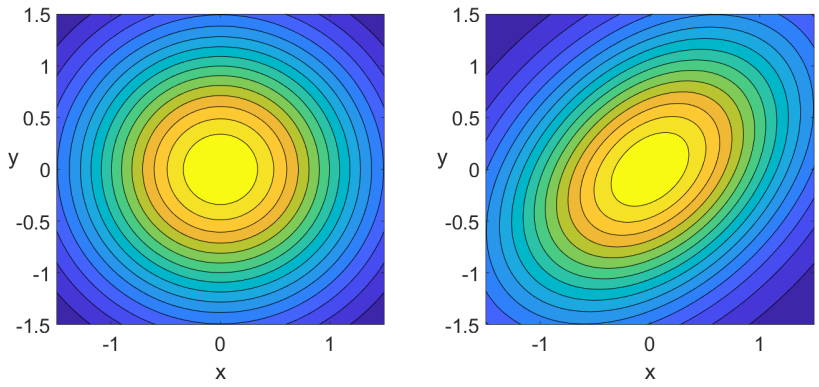
Equation (2.5) implies that the peak height of the normal distribution around μ is directly correlated with the width of the distribution, i.e., the value of σ, where a smaller σ implies a higher peak at μ.

Figure 2.3 is an illustration of the so-called *standard normal distribution*, i.e., $N(0, 1)$. The probabilities of every Gaussian distribution $X \sim N(\mu, \sigma^2)$ can be calculated by converting X to a standard normal distribution. When calculating

$$Z = \frac{X - \mu}{\sigma}, \tag{2.6}$$

Z becomes a standard normal distributed random variable. Knowledge about the CDF of the standard normal distribution can then be used to calculate the CDF of X:

$$P(Z \leq z) = P\left(\frac{X - \mu}{\sigma} \leq z\right) = P(X \leq \sigma z + \mu) = F(\sigma z + \mu). \tag{2.7}$$



(a) (b)

Figure 2.4: Contour plot of the PDF of a bivariate normal distribution with parameters $\mu = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$ and (a) $\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ respectively (b) $\Sigma = \begin{pmatrix} 1 & 0.4 \\ 0.4 & 1 \end{pmatrix}$.

This standardisation is often called the *z-transform*. For the standard normal distribution so called *z-tables* are available in which the values of its CDF are given[2]. This is often necessary since there exists no closed form of the CDF.

2.1.5 Multivariate Normal Distribution

The normal distribution can be generalized to more than one dimension. It is then called a *multivariate normal* (or *Gaussian*) *distribution*. Instead of specifying the dimension, usually we call it *univariate* normal distribution (or only normal distribution) in 1D, and *bivariate* normal distribution in 2D. The density of a d-variate Gaussian distribution is:

$$f_X(x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \cdot e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)}, \tag{2.8}$$

with $\mathbf{x} \in \mathbb{R}$, μ being a d-dimensional mean vector and Σ a symmetric, positive definite $d \times d$ *covariance matrix*. In case of $d = 2$, the covariance matrix has entries:

$$\Sigma = \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix}, \tag{2.9}$$

[2] www.stat.ufl.edu/~athienit/Tables/Ztable.pdf

where ρ is the correlation between X and Y in the interval $(-1, 1)$. Figure 2.4 shows the influence of the correlation: The contour lines of a bivariate Gaussian distribution with $\rho = 0$ are perfect circles while the distribution in Figure 2.4b has a correlation $\rho = 0.4$ and, hence, its contour lines are elliptic. Stronger correlation yields a more elliptic form while a lower correlation yields more circular forms.

The CDF of a random variable $X \sim N_X(\mu, \Sigma)$ is usually defined as:

$$F(\mathbf{x}) = P(X \leq \mathbf{x}), \tag{2.10}$$

with $x \in \mathbb{R}^n$. Like in the univariate case, the CDF of the multivariate generalisation has no closed form, however, it can be approximated numerically.
Additionally, it should be mentioned that the marginal distributions of a multivariate normal distribution are again (multivariate) normal distributions. For a random variable $X \sim N(\mu, \Sigma)$ with

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}, \tag{2.11}$$

the marginal distributions $X_i, i \in 1, 2$, are given by

$$X_i \sim N(\mu_i, \Sigma_{ii}). \tag{2.12}$$

Note that this expression is independent of the correlation ρ.

2.2 Scalar Fields

Given a domain Ω with $\Omega \subset \mathbb{R}^n$, a *scalar field* f assigns a scalar value to each point $\mathbf{x} \in \Omega$:

$$f : \Omega \to \mathbb{R}. \tag{2.13}$$

2.2.1 Contours

Given a scalar field f, a *contour*, regarding isovalue c and dimension n, is defined as the set

$$\mathcal{I}_c = \{\mathbf{x} \in \mathbb{R}^n | f(\mathbf{x}) = c\}. \tag{2.14}$$

In 2D, a contour is usually called an *isoline*, while in 3D it is referred to as an *isosurface*.

2.2.2 Gradient

Given a scalar function $\rho(\mathbf{x}, t), \rho : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, the gradient is defined as the vector whose components are the n first-order partial derivatives of ρ:

$$grad(\rho(\mathbf{x}, t)) = \nabla\rho(\mathbf{x}, t) = \frac{\partial}{\partial x_1}\rho(\mathbf{x}, t) \cdot \mathbf{e}_1 + \cdots + \frac{\partial}{\partial x_n}\rho(\mathbf{x}, t) \cdot \mathbf{e}_n, \tag{2.15}$$

with $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$.

2.2.3 Laplace Operator

Given a twice differentiable scalar function $\rho(\mathbf{x}, t), \rho : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, the *Laplace operator* (or *Laplacian*) is defined as the divergence (Section 2.3.2) of the scalar field's gradient:

$$\Delta\rho(\mathbf{x}, t) = div(\nabla\rho(\mathbf{x}, t)) = \nabla \cdot \nabla\rho(\mathbf{x}, t) = \frac{\partial^2}{\partial x_1^2}\,\rho(\mathbf{x}, t) + \cdots + \frac{\partial^2}{\partial x_n^2}\,\rho(\mathbf{x}, t), \tag{2.16}$$

with $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$. The Laplacian is also applicable to vector fields (Section 2.3). In the case of a scalar field the Laplace operator yields a scalar; in case of a vector field a vector.

2.3 Vector Fields

A *vector field* is a function $f : \Omega \to \mathbb{R}^m$ that maps a point $\mathbf{x} \in \Omega$ to a vector $\mathbf{u} \in \mathbb{R}^m$. Often times it is assumed that $\Omega = \mathbb{R}^m$. It can be distinguished between *steady*

$$\mathbf{u}(\mathbf{x}) \tag{2.17}$$

and *unsteady*

$$\mathbf{u}(\mathbf{x}, t) \tag{2.18}$$

vector fields. Unsteady vector fields are not only dependent on the domain Ω, but also on time t.
A vector field is a map $\mathbf{u}(\mathbf{x})$ defined by:

$$\begin{gathered} \mathbf{u} : \mathbb{R}^n \to \mathbb{R}^m \\ f_1 : \mathbb{R}^n \to \mathbb{R} \quad \dots \quad f_m : \mathbb{R}^n \to \mathbb{R} \\ \mathbf{u}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \dots \\ f_m(\mathbf{x}) \end{pmatrix}, \end{gathered} \tag{2.19}$$

with $\mathbf{x} \in \mathbb{R}^n$. To expand this definition to unsteady vector fields f_m is additionally parametrized by time t.

2.3.1 Jacobian Matrix

Given a vector field $\mathbf{u} : \mathbb{R}^n \to \mathbb{R}^m$ the *Jacobian matrix* (also called the *velocity gradient*, *gradient tensor*, or only *Jacobian*) is defined as follows:

$$\mathbf{J} = \nabla \mathbf{u}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}, \tag{2.20}$$

with $\mathbf{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$. Similar to the gradient of a scalar field, the Jacobian is the matrix of all first-order partial derivatives of a vector field.

2.3.2 Divergence

The *divergence* of a vector field $\mathbf{u} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ is a scalar field that describes for every point how much the vectors in the points' vicinity diverge. It is defined by:

$$div\ \mathbf{u}(\mathbf{x}, t) = \nabla \mathbf{u}(\mathbf{x}, t) = \frac{\partial}{\partial x_1} f_1 + \cdots + \frac{\partial}{\partial x_n} f_n = tr(\mathbf{J}), \tag{2.21}$$

with $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. The divergence can be interpreted as the net oriented flow trough the surface enclosing a given point. In this regard, positive divergence corresponds to net outflow and negative divergence to net inflow.

2.3.3 Curl

The *curl*, in CFD often also called *vorticity*, describes the local rotation at a point $\mathbf{x} \in \mathbb{R}^n$. In three dimensions the curl is defined as

$$\boldsymbol{\omega}(\mathbf{x}, t) = curl\ \mathbf{u}(\mathbf{x}, t) = \nabla \times \mathbf{u}(\mathbf{x}, t) = \begin{pmatrix} \frac{\partial}{\partial x_2} f_3 - \frac{\partial}{\partial x_3} f_2 \\ \frac{\partial}{\partial x_3} f_1 - \frac{\partial}{\partial x_1} f_3 \\ \frac{\partial}{\partial x_1} f_2 - \frac{\partial}{\partial x_2} f_1 \end{pmatrix}, \tag{2.22}$$

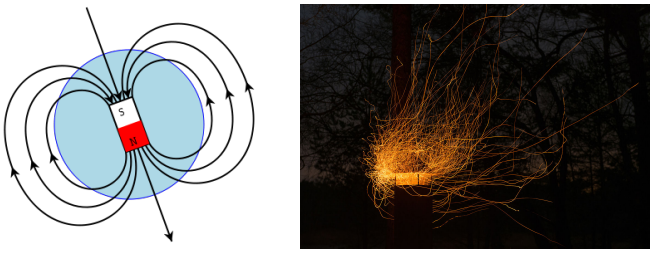
with $\mathbf{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3$.

2.3.4 Streamlines

In velocity fields, *streamlines* are curves that are instantaneously tangent to the underlying vector field, and describe the movement of massless particles in a flow. A common example are magnetic field lines (Figure 2.5a). Given an unsteady vector field $\mathbf{u}$, a streamline is defined as a solution to the initial value problem of the ordinary differential equation (ODE)

$$\mathbf{L}(0) = \mathbf{x}_0 \quad , \qquad \frac{\mathrm{d}\mathbf{L}(s)}{\mathrm{d}s} = \mathbf{u}(\mathbf{L}(s), \bar{t}), \tag{2.23}$$

with $\mathbf{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$ for a fixed time step $\bar{t}$. The solution is the resulting integral curve $\mathbf{L}(s)$ given by



(a) (b)

Figure 2.5: (a) Streamlines of a magnetic field,[3] (b) Long-exposure photo from a campfire shows pathlines of sparks.[4]

$$\mathbf{x}(s) = \mathbf{x}_0 + \int_{s_0}^{s} \mathbf{u}(\mathbf{L}(\tau))\mathrm{d}\tau. \tag{2.24}$$

Since streamlines are defined for a fixed time step, for unsteady vector fields they can only be computed for a given "snapshot" in time, which means

$$\mathbf{u}_t(\mathbf{x}) = \mathbf{u}(\mathbf{x}, \bar{t}). \tag{2.25}$$

2.3.5 Pathlines

Another concept is that of pathlines. They describe the trajectory of massless particles in a time-dependent vector field (Figure 2.5b). A path line is a solution to the initial value problem of an ODE

$$\mathbf{L}(0) = \mathbf{x}_0 \quad , \quad \frac{\mathrm{d}\mathbf{L}(s)}{\mathrm{d}s} = \mathbf{u}(\mathbf{L}(s), s), \tag{2.26}$$

[3] https://en.wikipedia.org/wiki/Magnetic_field#/media/File:Earths_Magnetic_Field_Confusion.svg

[4] https://en.wikipedia.org/wiki/%Streamlines,_streaklines,_and_pathlines#/media/File:Kaberneeme_campfire_site.jpg

with $\mathbf{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$. The resulting integral curve is then

$$\mathbf{x}(s) = \mathbf{x}_0 + \int_{s_0}^{s} \mathbf{u}(\mathbf{L}(\tau), \tau) \mathrm{d}\tau. \tag{2.27}$$

It should be noted that for steady vector fields path- and streamlines are identical.

2.4 Uncertain Vector Fields

In Section 2.3, (*certain*) vector fields were defined. However, the focus of this work evolves around so-called *uncertain vector fields*. Otto et al. [24] define steady 2D uncertain vector fields as 4D scalar fields $\rho(x, y; u, v)$ with:

1. $(x, y) \in \Omega$ and $(u, v) \in \mathbb{R}^2$,
2. $\rho(x, y; u, v) \geq 0$ with $(x, y) \in \Omega$ and $(u, v) \in \mathbb{R}^2$,
3. $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \rho(x, y; u, v) \mathrm{d}u \, \mathrm{d}v = 1 \quad \forall \, (x, y) \in \Omega.$

This definition is similar to those of the axioms of probability (Section 2.1.1), if we interpret $(x, y; u, v)$ as sample space and ρ as a probability function. As with certain vector fields, Ω is usually equal to $\mathbb{R}^2$. Simply spoken, a 2D uncertain vector field does not map a point $(x, y) \in \Omega$ to a vector $u \in \mathbb{R}$, but instead to a distribution of vectors. If this distribution is a degenerate (delta) distribution, the uncertain vector field is essentially a certain one, since for every point (x, y), there is only one vector that has an assigned probability of 1, while all other vectors have a probability of 0. Otto et al. [25] also showed that this concept can be generalized to 3D vector fields, however, this is out of the scope of this thesis.

2.4.1 Particle Density Function

Since for uncertain vector fields it is useful to analyse particle densities instead of single particles, Otto et al. [24], furthermore, define *particle density functions* over the domain

Ω. In order to track distribution changes over time, these functions are also parametrized by time t. They are 2D unsteady scalar fields $p(x, y; t)$ with $(x, y) \in \Omega$ and

1. $p(x, y; t) \geq 0 \quad \forall (x, y) \in \Omega$,
2. $\int\int_{\Omega} p(x, y; t)\, \mathrm{d}x\, \mathrm{d}y \leq 1$.

In this definition, the integral over all elements is only ≤ 1 instead of $= 1$, since it is possible that particles leave the domain. However, it is mandatory that in the first time step the particle density is equal to 1, so the integral in all following time steps is a ratio in regard to the starting particle density.

2.4.2 Streamlines on Uncertain Vector Fields

Otto et al. [24] have shown that the transport of particle densities in an uncertain vector field is described by:

$$\begin{aligned} p(x, y; t + \Delta t) &= \int\int_{D} p(r, s; t) \rho\Big(r, s; \frac{x - r}{\Delta t}, \frac{y - s}{\Delta t}\Big)\, \mathrm{d}\Big(\frac{x - r}{\Delta t}\Big)\, \mathrm{d}\Big(\frac{y - s}{\Delta t}\Big) \\ &= \frac{1}{\Delta t^2} \int\int_{D} p(r, s; t) \rho\Big(r, s; \frac{x - r}{\Delta t}, \frac{y - s}{\Delta t}\Big)\, \mathrm{d}r\, \mathrm{d}s, \end{aligned} \tag{2.28}$$

with a transport of particles from (r, s) to (x, y) in time Δt and on the domain $D = \mathbb{R}^2$. Furthermore, they define a streamline on an uncertain vector field $\rho(x, y; u, v)$ as an initial value problem with

$$\begin{aligned} p(x, y, t_0) &= p_0(x, y), \\ \frac{\mathrm{d}p(x, y; t)}{\mathrm{d}t} &= \lim_{\Delta t \to 0} \frac{p(x, y; t + \Delta t) - p(x, y; t)}{\Delta t}, \\ p(x, y; t + \Delta t) &= \textit{Equation (2.28)}, \end{aligned} \tag{2.29}$$

where $p_0(x, y)$ is the starting particle density function and $p(x, y; t)$ a time-dependent particle density function. Backward integration can be acquired by integrating p with the underlying uncertain vector field $\rho(x, y; -u, -v)$.

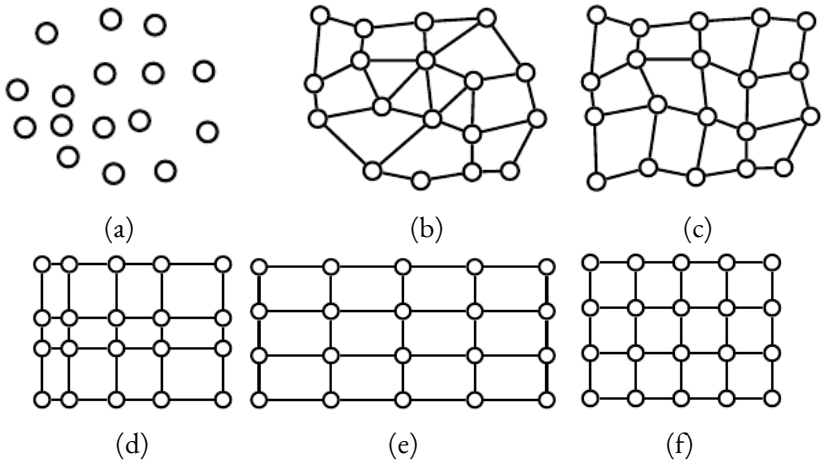


Figure 2.6: Grid types. (a) Scattered data. (b) Unstructured grid. (c) Structured grid. (d) Rectilinear grid. (e) Uniform grid. (f) Cartesian grid. Images from Sadlo [35].

2.5 Grid Representations

The concepts of vector and scalar fields assume a continuous structure, so that for every point in its domain a value is defined. However, in numerical approaches, an analytic representation is often not available, since data from measurements or simulations is usually only provided in a discrete way. The representation of such discrete data differs from case to case.

A common way is to define the data as a set of points with corresponding values. Such data is usually called scattered and is most often a result from measurements (Figure 2.6a). Scattered data has no connectivity between single points and, hence, no given topology.

However, in simulations, the resulting data often already has a certain structure which implies a topology. Such data can be discretised and fit into a grid. Grids consist of cells and nodes; typical cell types are triangles and quads in 2D, or tetrahedrons and hexahedrons in 3D. Furthermore, these grids can be classified either as unstructured or structured. In unstructured grids (Figure 2.6b) a topology has to be explicitly provided while structured grids (Figure 2.6c) have an implicit regular topology. Consequently, structured grids can only consist of one cell type while in unstructured

grids different cell types can be combined. Structured grids can be distinguished even further: A so-called rectilinear grid (Figure 2.6d) only consists of cells, so that every edge is parallel to one of the coordinate axes. Uniform grids (Figure 2.6e) have the additional constraint that the spacing between grid nodes in each dimension has to be constant, and Cartesian grids (Figure 2.6f) need a constant spacing in every dimension.

Different grid types allow for different computational methods, e.g., scattered data needs to be interpolated differently than structured data, and more regular grids can usually be stored more effectively. As mentioned already, grid data consists of cells and nodes. The data of our vector or scalar field can be stored at the nodes (*node-based data*) or at the cells (*cell-based data*) (or possibly even both).

2.6 Dual Grids

To convert cell-based data of a grid to node-based data or vice-versa a so called dual grid can be computed. Therefore, a new point is initialised at the centre of each cell where the new value is interpolated. Afterwards, depending on the original face-connectivity of the grid, new edges are inserted, so that a new grid emerges. Drawbacks of this method are the decreasing resolution of the grid as well as the smoothing effect that occurs due to the interpolation.

2.7 Interpolation

Section 2.5 explained that in numerical approaches often only discrete data is available and how such data is represented on a grid. This section introduces two common techniques to obtain a continuous representation for each cell. Therefore, it is assumed that data is stored node-based. For unstructured grids, interpolation by inverse distance weighting is presented; for rectilinear quad based 2D grids bilinear interpolation.

2.7.1 Inverse Distance Weighting

Inverse distance weighting (IDW) is an interpolation method for scattered point sets. Given N sample points $\mathbf{x}_i, i = 1, 2, ...N, \mathbf{x}_i \in \mathbb{R}^n$ and corresponding values

$u_i = u(\mathbf{x}_i)$, with $u_i \in \mathbb{R}$, the interpolated value $u(\mathbf{x})$ at point $\mathbf{x}$ is in general given by:

$$u(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^{N} w_i(\mathbf{x}) u_i}{\sum_{i=1}^{N} w_i(\mathbf{x})} & \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ for all } i, \\ u_i & \text{if } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ for some i,} \end{cases} \tag{2.30}$$

with d being a given distance function, and w_i an inverse distance weighting function. A commonly used IDW function was defined by Shepard [38]:

$$w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^p}, \tag{2.31}$$

with $p \in \mathbb{R}_+$. Larger values of p increase the weight of points that have a closer distance to point $\mathbf{x}$.

N is called the neighbourhood size and can be specified in regard to:

- The radius around the interpolated point $\mathbf{x}$,
- a fixed number of points around $\mathbf{x}$,
- a combination of the two.

2.7.2 Bilinear Interpolation

Given a rectangular cell defined by the four grids points $(x_i, y_j), \ldots, (x_{i+1}, y_{j+1})$ (Figure 2.7) with corresponding scalar values $f_{i,j} = f(x_i, y_j), \ldots, f_{i+1,j+1} = f(x_{i+1}, y_{j+1})$, for an arbitrary point (x, y) inside the rectangle, bilinear interpolation yields the following result:

$$\begin{aligned} f(x,y) &= (1-\alpha)(1-\beta) f_{i,j} + \alpha(1-\beta) f_{i+1,j} + (1-\alpha)\beta f_{i,j+1} + \alpha\beta f_{i+1,j+1} \\ &= (1-\beta)[(1-\alpha) f_{i,j} + \alpha f_{i+1,j}] + \beta[(1-\alpha) f_{i,j+1} + \alpha f_{i+1,j+1}] \\ &= (1-\beta) f_j + \beta f_{j+1} \end{aligned} \quad , \tag{2.32}$$

with

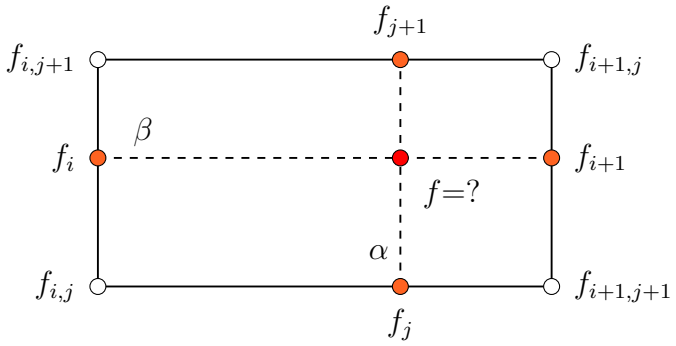


Figure 2.7: Bilinear interpolation inside of a rectangular cell.

$$\begin{aligned} f_j &= (1-\alpha) f_{i,j} + \alpha f_{i+1,j} \\ f_{j+1} &= (1-\alpha) f_{i,j+1} + \alpha f_{i+1,j+1} \end{aligned} \tag{2.33}$$

and the local coordinates

$$\alpha = \frac{x - x_i}{x_{i+1} - x_i}, \qquad \beta = \frac{y - y_i}{y_{i+1} - y_i}. \tag{2.34}$$

Despite its name, bilinear interpolation is not linear but quadratic. For cubes the concept of bilinear interpolation can easily be expanded to three dimensions: The method is then called trilinear interpolation and needs a third local coordinate γ.

2.8 Numerical Integration of Ordinary Differential Equations

To compute streamlines, it is necessary to solve the initial value problem for ordinary differential equations, which is given by

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}, t) \quad \text{with} \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.35}$$

and $\mathbf{x} \in \mathbb{R}^n$.

To solve this initial value problem, it is necessary to choose an adequate numerical solver that *integrates* the ODE, and by doing so, produces a streamline. The simplest approach for this integration is the explicit Euler method defined by

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot f(\mathbf{x}, t). \tag{2.36}$$

The Euler method is a first-order method, i.e., the global error is proportional to the chosen step size Δt. Therefore, it is mandatory to choose a small step size to yield meaningful results; otherwise the method is too inaccurate.

In the case of uncertain vector fields, we will restrict ourselves to the Euler method, but it is worth mentioning the fourth-order Runge-Kutta (RK) method, too, since it is one of the most common integration methods for certain vector fields. It is defined by:

$$\begin{aligned} k_1 &= \Delta t f(\mathbf{x}, t), \\ k_2 &= \Delta t f(\mathbf{x} + \frac{k_1}{2}, t + \frac{\Delta t}{2}), \\ k_3 &= \Delta t f(\mathbf{x} + \frac{k_2}{2}, t + \frac{\Delta t}{2}), \\ k_4 &= \Delta t f(\mathbf{x} + k_3, t + \Delta t), \\ \mathbf{x}(t + \Delta t) &= \mathbf{x} + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(\Delta t^5). \end{aligned} \tag{2.37}$$

While the fourth-order Runge-Kutta method is one of the most used RK methods, there exists a complete family of explicit Runge-Kutta methods. A detailed introduction is provided by Butcher [4].

2.9 Related Work

In the scientific visualization community, the representation of uncertainty is considered as one of the most important topics [5][12]. Uncertainty can have various sources [16], and is not only limited by the quality of the data, but also of the used method, i.e., the visualization itself [3]. An initial overview of uncertainty visualization can be found in the work of Brodlie et al. [3], Johnson and Sanderson [13], Pang et al. [27] or Potter et al. [32]. For different domains, a variety of problems occur when dealing with uncer-

tainty: In scalar fields, data is often visualized with the help of volume rendering [8], isosurfaces are extracted [29][33], but also gradients [30] and critical points [20] are extracted. Uncertainty in vector fields is often visualized with the help of glyphs [17][40], however, Botchen et al. [2] also presented a texture-based technique to visualize uncertainty, and Petz et al. [28] took spatial correlation into account and proposed new methods to extract local features in uncertain vector fields. Despite the fact that only little work is dedicated to visualize uncertainty in tensor fields [7], there are projects that visualize uncertainty in fiber orientation [14] and tracking [10][37]. Furthermore, uncertainty has also been considered in the visualization of ensemble data, especially in regard to meteorology [11][21][31][36], and in the visualization of geospatial data [19][26]. In contrast to these previous approaches, Otto et al. do not consider only local uncertainty but also include the transport of uncertainty in a flow. They apply the approach of vector field topology to uncertain 2D [24] and 3D [25] vector fields, and by doing so, segment the uncertain vector field into regions with different flow behaviour [23]. Additionally, Otto and Theisel introduced a novel method to track vortices in uncertain vector fields [22]. In this thesis, we base our definition of uncertain vector fields on the work by Otto et al., and expand our model from there on. For the definition of domain-uncertain data, we use concepts by Shu et al. [39].

3 Tracing Single Uncertain Streamlines

In this chapter, typical sources of uncertainty and a technique to compute uncertain streamlines in 2D are introduced. For this, we refer to the methods proposed by Otto et al. [24]. Afterwards, the concept of domain-uncertain data is presented. In this work, we differentiate between (single) uncertain streamlines, which are presented in this chapter, and streamline distributions in uncertain vector fields, which are described in Chapter 4.

3.1 Sources of Uncertainty

The concept of uncertainty comes into play when one or several properties of data cannot be determined exactly. Uncertainty can either be statistical, i.e., the results of a random experiment will differ, even if initial conditions are (nearly) identical; or it can be systematic, i.e., that there is a fixed error that is inherent in every measurement of the experiment. Systematic errors can, for example, occur if the underlying model of a simulation is wrong, or if the same mistake is made in every measurement of an experiment.

Based on the work of Kennedy and O'Hagan [16], one can identify the following possible sources of uncertainty:

- *Parameter uncertainty*:
  In mathematical models, in general, not all parameters can be determined exactly, since they are not known, e.g., in weather forecasts, it is nearly impossible to know all relevant parameters beforehand, and even if we did, a small measurement error would be enough, such that the prediction would be completely off in the near future.

- *Model uncertainty*:
  Another uncertain factor is the model itself. Models are based on observations in nature, however, often-times they are not completely accurate, or make assumptions to simplify the problem and, therefore, results may not fully reflect the real-world problem.

- *Numerical uncertainty*:
  This uncertainty stems from numerical errors in the calculations of a computer program. In every computational step, e.g., every interpolation, or every integration step to determine a streamline, there is an error due to discretisation, or used method.

- *Experimental uncertainty*:
  This is the most typical example for statistical uncertainty. Even if initial conditions of an experiment are identical, results will differ. Usually, many repetitions of the same experiment are made to get an accurate result, but even then, the accuracy of the result cannot be determined exactly.

This list does not claim to be complete, but is supposed to give an idea about the most common sources of uncertainty. Furthermore, it should be mentioned that typical sources of uncertainty are dependent on the use case. In chemical experiments, different problems will be encountered than in computational simulations.

3.2 Integration of Uncertain Streamlines

In this section, we will focus on uncertain vector fields (see Section 2.4), and track the movement of single particles over time (cf. Otto et al. [24]). This is also called the integration of particles. If not mentioned otherwise, we assume that the given data is stored in an uniform grid, where each grid point has its own distribution.

To integrate an uncertain streamline, we use integration methods known from classical streamline integration, and use a Monte Carlo approach [34], in which we use random samples to integrate streamlines. In contrast to classical streamline integration inside of cells, we cannot interpolate the vector at a given position $\mathbf{x} = (x, y)$, but have to

interpolate the parameters of the given distribution. Therefore, it is mandatory that the same distribution model is used at all nodes, such that a meaningful interpolation is possible. The parameters of this distribution can differ at each node. For interpolation, a standard bilinear interpolation is performed. From this interpolated distribution, a vector $\mathbf{v}$ is drawn from the distribution function $\rho(x, y; u, v)$. $\mathbf{v}$ is then used for the next integration step, and a Monte Carlo based integration is performed. Usually, we use only small Euler steps instead of more complex integration methods like the Runge-Kutta methods, since due to the random nature of the method, the improved accuracy is unnecessary. It might even be undesirable if symmetric distributions are in use, since in many cases, this would prevent the vector from taking on outliers of the distribution.

A backwards integration of uncertain streamlines is also possible, which is usually accomplished by multiplying the vector field by (-1), and then performing a normal forward integration.

3.3 Domain-Uncertain Data

The first contribution of this work is the concept of *domain-uncertain data*. In the case of domain-uncertain data, it is assumed that the given values are certain, however, the underlying domain is not. In other words, the measured data is correct, but it is not known where, or when, the data point existed. This problem often arises in geoinformatics, and in this context, Shu et al. [39] defined so-called *random points*.

A random point $p : \mathbb{R}^2 \rightarrow [0, 1]$ maps to each point in the domain a probability that the random point exists at this position. A common example in geoinformatics is GPS data, in cases where the exact position is of importance.

This definition of random points is restricted to two-dimensional data. Of course, this can be generalised to higher (or lower) dimensions, but in the context of this work, we will stick to the 2D case.
Shu et al. [39] also defined the concept of a *random instant*, which is a mapping from time space to a probability in the interval $[0, 1]$. Since the focus of this work lies on vector

fields, we can merge this with the concept of random points. Therefore, we define an *uncertain point* to be a mapping $p(x, y, t)$:

$$\mathbb{R}^2 \times \mathbb{R} \to [0, 1]. \tag{3.1}$$

To be in line with the axioms of probability, the following constraint is introduced:

$$\Upsilon_p = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x, y, t) \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}t = 1. \tag{3.2}$$

This implies that an uncertain point has to exist somewhere in space and time. In case of steady vector fields, uncertain points and random points are equal; for domains where only time is uncertain, uncertain points simplify to random instants.

If one wants to define a grid that stores uncertain points, we encounter the problem that possibly more than one cell edge exists between two uncertain points. Therefore, the concept of *uncertain edges* is defined. In conformity with Shu et al. [39], we denote a set of deterministic lines as $LSpace$. Deterministic lines are also time-dependent, i.e., the $LSpace$ can span over a whole time interval. An uncertain line is a mapping

$$LSpace \to [0, 1]. \tag{3.3}$$

The probability for the realisation of a specific uncertain line l is denoted as p_l, and it holds that

$$\int_{l \in LSpace} p_l \, \mathrm{d}l = 1. \tag{3.4}$$

The vector field domain is then defined as a set of uncertain points p, where each point has stored a vector. It is also possible to combine domain-uncertain data with uncertain vector fields. This would imply that not only the domain D is uncertain, but also the data, which is defined on the domain, such that at every uncertain point, a probability distribution of vectors ρ is defined. We can then refer to the definitions in Section 2.4, however, our domain D is a set of uncertain points. In the following, we will refer to a

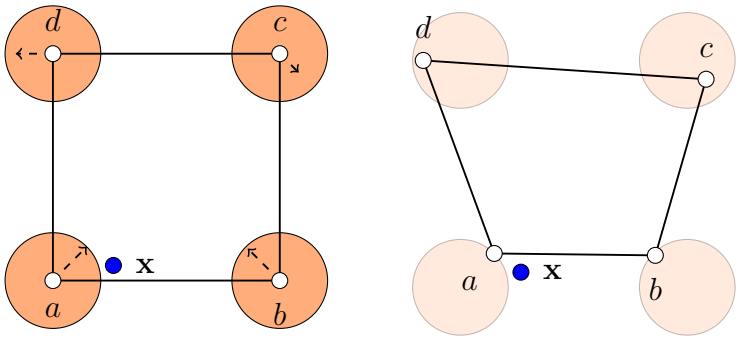


Figure 3.1: The value at the blue point $\mathbf{x}$ needs to be interpolated. (a) The orange circles indicate which values the uncertain points a, b, c, d can take on. (b) One possible outcome of randomly choosing points out of the uncertain point set.

domain, that is defined on domain-uncertain data, as an *uncertain domain*.

3.3.1 Streamlines on Uncertain Domains

In order to compute streamlines on an uncertain domain, the already presented methods for streamline integration are applicable. This integration works for certain, as well as for uncertain vector fields, the following description, however, assumes a certain vector field.

We start the integration at a certain point $\mathbf{x}$ in our domain. To be consistent with our uncertain domain, this certain point is an uncertain point that has a probability of 1 to be at a specific position in space and time. At the start of each integration step, for each uncertain point, its position for this integration step is randomly chosen, such that after this step, every uncertain point has a distinct (certain) position, where it is located. In the next integration step, this position can alter again. We will call this the randomization step. For the integration, it is necessary to interpolate the value f_x at point $\mathbf{x}$ in regard to the point set. Figure 3.1 indicates a few problems that arise when processing an uncertain domain:

- Firstly, even if we have a quad-based topology defined on our grid, cells can take on an arbitrary form, because, in general, cells are not necessarily homogeneous.

If one wants to use approaches like bilinear interpolation, it is necessary to find a mapping ϕ that transforms arbitrary quad cells to a *computational space* in which local coordinates can be computed. This mapping step is computationally very expensive and, hence, not preferable if one needs to compute many integration steps.

- Secondly, also assuming a quad-based topology, if our point $\mathbf{x}$ is near an uncertain point ,or an uncertain edge between two points, it is never known in which cell the point is at the moment. In case of our interpolation problem, this means that after each randomization step, it has to be checked in which cell the point is at the moment. This could even lead to a situation where a point, that was in the domain before the integration step, has left the domain in the next time step, because it has left the outer bounds of the grid. To intercept such boundary cases, it is necessary to compute the maximum values the boundary can take on, or at least to define a boundary beyond which a point is considered as outside of the uncertain domain.

- Thirdly, it cannot be guaranteed that after the randomization step, the grid remains topologically the same, e.g., two uncertain points could be so close to each other that in some scenarios pre-defined cells are overlapping. One solution to this problem is to determine a new cell structure, e.g., via triangulation, after each randomization step, however, this is a time-consuming procedure that negates one of the key advantages of structured grids.

Because of these reasons, it is preferable to handle the data as if it were scattered, and use inverse distance weighting to interpolate the value at point a $\mathbf{x}$. If a topology is defined on the data, it can be assumed that the nearest points, in regard to $\mathbf{x}$, are the points that define the cell $\mathbf{x}$ is currently contained in, as well as the directly surrounding cells. However, this approach becomes less accurate the smaller the cells are. If this assumption cannot be made, or if no topology is defined, the distance from $\mathbf{x}$ to all other points in the data set has to be calculated. Especially for large data sets, this is very expensive and can lead to high computation times, if multiple or very long uncertain streamlines have to be computed. After the interpolation step, the integration is not different from methods that perform on domain-certain data.

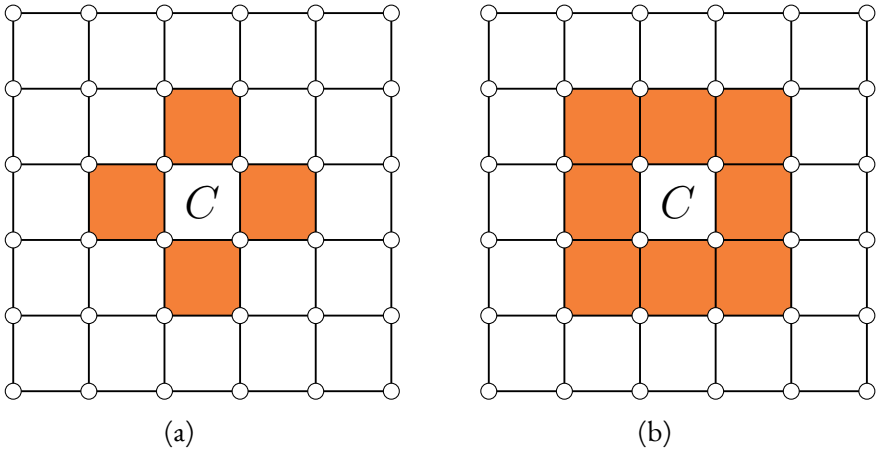


Figure 3.2: Orange cells indicate the cells belonging to cell C's (a) 4-neighbourhood and (b) 8-neighbourhood.

An algorithm based on the previous insights was implemented. The algorithm works on node-based uniform grids, and it is assumed that every uncertain point is normally distributed, with every node position of the grid representing the mean position μ of each uncertain point. Except for the interpolation method, the algorithm is designed the same way as the algorithm that was implemented to produce uncertain streamlines in Section 3.2. In the algorithm, different integration methods (Euler, Midpoint, fourth-order Runge-Kutta), maximum integration time, maximum number of integration steps, step size for integration, integration direction (forward, backward, both), and a starting point $\mathbf{x}$ can be chosen. The algorithm uses Shepard interpolation with a weight $p = 2$, and the Euclidean distance function (compare Equation (2.31)). The neighbourhood size is specified in regard to the 4-neighbourhood of cell C, in which $\mathbf{x}$ lies. All cell nodes that are part of C's 4-neighbourhood are taken into account when interpolating the value at point $\mathbf{x}$. For cells C at the border of the domain, only the existing cells in their neighbourhood are chosen, such that only 2 or 3 neighbouring cells are taken into account.

4 Streamline Distributions in Uncertain Vector Fields

The previous chapter described how to track single uncertain streamlines, however, the visualization of single streamlines cannot reflect the complete behaviour of an uncertain vector field. Therefore, in this chapter, we will discuss an algorithm to track distributions of particles in uncertain vector fields, so-called streamline distributions, and present a visualization that indicates the structure of the distribution over time.

4.1 Distribution Grids

To characterize the streamline of an uncertain vector field, it would be necessary to seed multiple streamlines at the same seeding spot. This, however, would lead to a much too cluttered representation of the streamline distribution, and no meaningful insight into the structure of the flow would be gained. Since the flow is hardly identifiable by this approach, other visualization techniques are needed to tackle the problem of this agglomeration of uncertain streamlines. Therefore, we decided to visualize the distribution of particles in an uncertain vector field instead of visualizing the particles themselves.

The first thing to consider, when tracking distributions instead of single particles, is the representation of our data. In Chapter 3, we considered a node-based grid representation. However, to track particle densities, it is beneficial to use cell-based representations. With that in mind, we use the concept of particle density functions (Section 2.4.1), but instead of tracking the density of particles at a certain position (x, y), we consider the density inside of quad grid cells with an extent of $[c_x - 0.5 \cdot r, c_x + 0.5 \cdot r; c_y - 0.5 \cdot s, c_y + 0.5 \cdot s]$, with (c_x, c_y) being the centre of the cell, r being the cell spacing in x-direction, and s the spacing in y-direction. As a result, it is no longer possible to seed particles

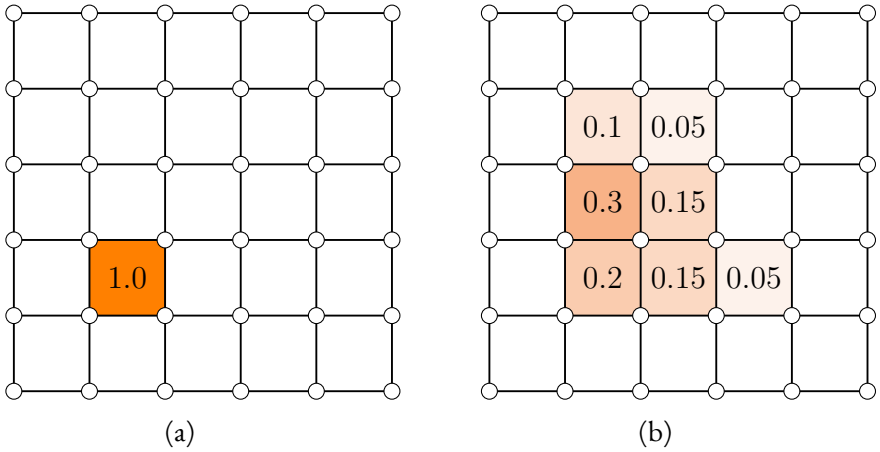


Figure 4.1: Distribution Grid. (a) At the start of the integration only one cell has density 1.0. (b) After the first integration step the distribution of particles is altered. All white cells have density 0.

at a single point, but only inside of a cell region, in which the particles are usually assumed to be equally distributed. To be exact, we will no longer seed particles, but assign a density to each cell of the grid before the integration starts. In the following, the resulting grid at an arbitrary time step of the integration will simply be called the *distribution grid*, or *distribution field*; the grid at start time t_0 is called the *start (distribution) grid*.

Potentially, we can also assign densities greater than 0 to multiple cells of the start grid, but since we are mainly interested in the structure of streamline distributions started in one cell, generally we will assign a density of 1 to one cell while the rest of the cells have a density of 0. For numerical reasons, the presented algorithm does not actually assign a value of 1 to the start cell, but a higher, user-assigned value α. After the integration of densities, the real density at any given position is then given by $\frac{\tau_x}{\alpha}$, with τ_x being the stored value in the cell with index x. By visualizing the cell densities, it can be observed which ratio of particles, seeded in start cell (x, y), moved to another cell (x_{end}, y_{end}) of the grid after a certain time Δt. This ratio is equivalent to the probability of an uncertain streamline, seeded at a random point inside of cell (x, y), ending in cell (x_{end}, y_{end}) after time Δt. As already indicated, we have to integrate the cell densities as presented in Section 2.4.2, in order to determine the cell densities after a certain time. Figure 4.1 shows a possible distribution of particles before and after an integration step. To determine

which ratio of particles moves from one cell to another, in every step of the integration, we have multiple options. In the following sections, the options considered in this work are discussed.

4.2 Tracking Particle Distributions

To track particle distributions over time, we need to compute a *kernel* k, that stores values $k_{g,h}$, that describes which ratio of particles flows from cell g to cell h in every integration step. It holds that

$$k_{g,h} \geq 0 \quad \forall\, g,\, \forall\, h, \tag{4.1}$$

and

$$\sum_{i \in h} k_{g,i} \leq 1 \quad \forall\, g. \tag{4.2}$$

In the case of steady uncertain vector fields, we have two primary options at which point to compute this kernel:

1. In every integration step, for every cell g, it is determined how much of its density is flowing to every other cell h. While we can improve the speed of this method with a few tricks, e.g., ignoring cells that have a density of 0, the major downside of this approach is the runtime. Especially for long integration times with many steps, this can lead to long computation times. The key advantage, however, is the amount of memory that is needed for this calculations. Even high resolution grids can be processed without running out of memory.

2. Computation before the integration. We iterate over all cells g, and determine which ratio will flow into every other cell h. Consequently, the kernel only needs to be computed once, and afterwards, is just applied in every integration step. This yields a much faster runtime than the first approach, but the memory usage is multiple times higher. For high resolution grids, this memory usage is not feasible, and the first, computation heavy, method is preferable. One way to accommodate this

technique is to introduce a *kernel size* s. The kernel size limits the kernel entries for each cell g to the s entries with the highest value. The kernel size needs to be determined individually, in regard to the grid resolution and data. It is also possible to define a *kernel threshold* t. Values lower than the threshold are ignored, and so the size of the kernel can be reduced, too. This can further reduce the runtime of the algorithm.

Another detail we have to consider, is which distribution of particles inside of one cell is assumed. There are four major options that should be taken into account:

1. Instead of assuming a special distribution, we go back to the concept of tracking single particles at certain positions instead of only storing cell densities. The density of each cell is then determined by the number of particles in the respective cell divided by the number of particles seeded. However, in this work, we want to concentrate on the tracking of distributions instead of particles, so this approach is not suited for the task at hand.

2. Inside of every cell, the distribution of particles is assumed to be concentrated at the cell centre. This would imply that all particles are manifested at the same infinitesimal small point inside of one cell. This approach can speed up computation times. Nonetheless, this approach simplifies the model a lot, and in the worst case, this can lead to situations where particles do not leave a cell at all. In general, it can be assumed that the slight improvement regarding runtime is not worth the inaccuracy introduced by this simplification.

3. We can also consider a continuous uniform particle distribution inside of each cell. For an arbitrary particle, the chance of being at a certain position is the same at every point inside a cell. If we have no additional information about the data, this is the most unbiased approach, and if not specified otherwise, is the method performed in this work.

4. Instead of an uniform distribution, any distribution can be chosen to represent the behaviour of particles in a cell. It is doubtful whether many use cases exist, in which this is applicable. Nonetheless, the possibility exists that there may occur cases, in which the data implies another certain behaviour inside of predefined cells that demands another distribution than the ones presented above.

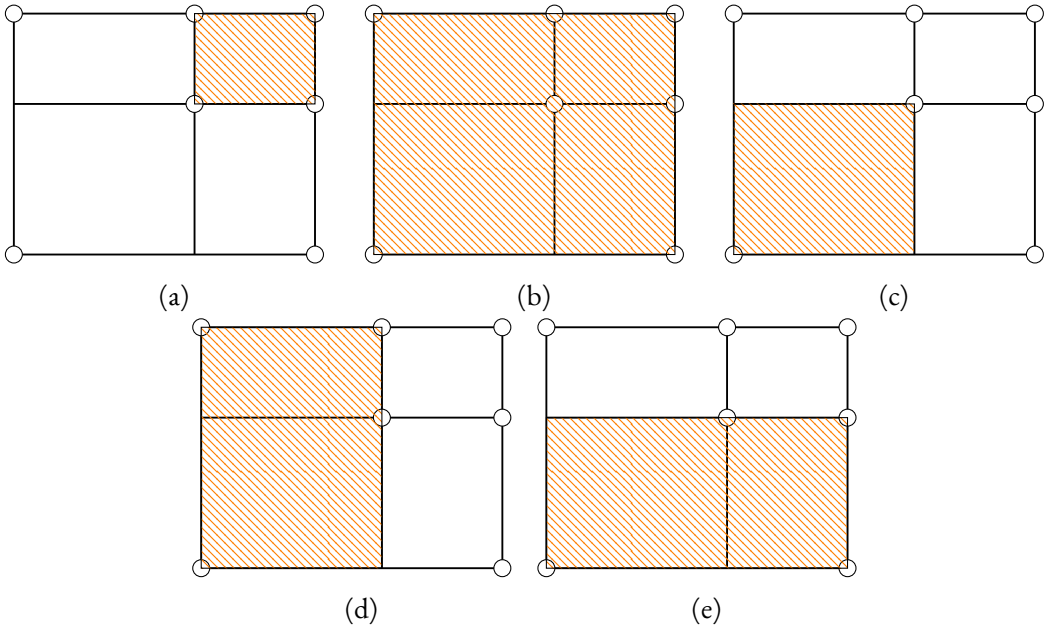


Figure 4.2: Area of (a) can be computed with knowledge about the areas indicated in (b), (c), (d) and (e).

4.3 Approximating Gaussian Kernels

Since most available data is certain, it is necessary to produce data that is uncertain to test our methods. Therefore, we generally assume that our uncertain data consists of vectors that are bivariate normally distributed. For a bivariate normal distribution $X \sim N(\mu, \Sigma)$, μ is defined as the vector of the underlying certain vector field given by the data, and Σ is set to a user-defined value. Since we are interested in the movement of particle distributions instead of particles, we are mostly interested in the cumulative distribution function of $N(\mu, \Sigma)$. This is, because in our case of flow from cell g to a cell h with an extent of $[h_x - r, h_x + r; h_y - s, h_y + s]$, where (h_x, h_y) is the centre of cell h, the kernel can be computed by

$$\begin{aligned} k_{g,h} &= F((h_x - r, h_y - s) \leq X \leq (h_x + r, h_y + s)) \\ &= F((h_x + r, h_y + s)) - F((x - r, y + s)) - \\ &\quad F((x + r, y - s)) + F((h_x - r, h_y - s)). \end{aligned} \tag{4.3}$$

Figure 4.2 illustrates this equation where the area in Figure 4.2a equals $k_{g,h}$, Figure 4.2b equals $F((h_x + r, h_y + s))$, Figure 4.2c equals $F((h_x - r, h_y - s))$, Figure 4.2d equals $F((x - r, y + s))$, and Figure 4.2e equals $F((x + r, y - s))$.

Since this CDF does not have a closed form, we can only approximate the value of the CDF. One method for this approximation is to use Gaussian quadrature formulas as described by Drezner and Wesolowsky [9]. They also provide an explicit algorithm to solve the problem. However, when using an approach that relies on calculating the cumulative distribution functions of multiple bivariate distributions, the runtime becomes unreasonably long. Calculating $k_{g,h}$ by these means will usually yield very small values, since we have to consider that Equation (4.2) still has to hold. Especially for high resolution grids, this becomes a problem due to the fact that we have to assure a very low error tolerance for this computation, because in comparison to the true value of the integral, we have to assure an insignificant error.

A more recent approach by Derakhshan and Deutsch [6] uses knowledge about the correlation ρ to integrate the bivariate Gaussian distribution. Especially for $\rho = 0$, $\rho = \pm\frac{1}{2}$, and $\rho = \pm 1$, this can speed up the runtime by a large margin. Nonetheless, computation times are still immense if we need to calculate this integral for a high number of cells, specifically, if we need to assure an error of insignificant magnitude. Consequently, this approach is only feasible if we can precompute the kernel, and do not need to compute it on demand in every time step. This, however, leads back to the problem that high resolution grids are not suitable for this kind of approach, since at some point, it is not possible to store such huge kernels.

While the method of computing the integral of the bivariate normal distribution is mathematically reasonable, in practice, they proved to be either too slow or too imprecise. Therefore, we use a simpler, much faster approach to compute kernel k that exploits the fact that the marginal distributions of a bivariate Gaussian distribution are also normal distributed (see Section 2.1.5). Figure 4.3 describes an algorithm that approximates k by drawing a certain number of samples from the PDF of a bivariate normal distribution for each cell. If we do not precompute the kernel, but compute it on demand in every step

of the density integration, we restrict the calculation to cells that have a particle density greater than 0 to minimize computation time.

Using this algorithm, we only need to draw vectors from the probability density function of a bivariate Gaussian distribution, i.e., we need to find a vector X with $\mathbf{X} \sim N(\mu, \Sigma)$ for an arbitrary but fixed μ and Σ. To achieve this, first we have to introduce the Cholesky decomposition: Every symmetric, positive definite matrix $M_{n\times n}$ can be decomposed by

$$M_{n\times n} = L \cdot L', \tag{4.4}$$

where L is a lower triangular matrix. The ansatz we use to draw a vector $\mathbf{X}$ from the PDF is

$$\mathbf{X} = L\varepsilon + \mu, \tag{4.5}$$

with $\Sigma = LL'$, and $\varepsilon \sim N(0, \mathbb{1})$. Σ can usually be Cholesky decomposed, since every covariance matrix we encounter, is symmetric and positive definite. If we expect covariance matrices that are positive-semidefinite, a different decomposition would be needed, e.g., we could then use the LDL decomposition. To compute a vector $\mathbf{X}$, we execute the following steps:

1. Calculate lower triangular matrix L in regard to Σ.

2. Generate a vector $\varepsilon \sim N(0, \mathbb{1})$ with length n by using a standard generator for univariate normal distributed random variables.

3. Evaluate the expression $\mathbf{X} = L\varepsilon + \mu$.

It can be proven that $\mathbf{X}$ is in fact normal distributed with mean μ and variance Σ. The expected value E of $\mathbf{X}$ is given by

$$E(\mathbf{X}) = E(L\varepsilon + \mu) = L \cdot E(\varepsilon) + \mu = \mu, \tag{4.6}$$

Algorithm 1 Approximate kernel with CDF of bivariate normal distribution

```
1:  procedure ESTIMATE_KERNEL_WITH_BVN_CDF
2:      num_samples ← number of samples to draw
3:      for g ∈ cells do
4:          ▷ Initialize kernel with 0 values
5:          for h ∈ cells do
6:              k_{g,h} ← 0
7:          i ← 1
8:          ▷ Draw samples form bivariate Gaussian distribution
9:          for i ≤ num_samples do
10:             next_cell ← step_to_cell(g)
11:             if next_cell ∈ domain then
12:                 k_{g,next_cell} ← k_{g,next_cell} + 1
13:         ▷ Scale kernel to values between 0 and 1
14:         for h ∈ cells do
15:             k_{g,h} ← k_{g,h}/num_samples
16:
17: procedure STEP_TO_CELL(cell c)
18:     ▷ Random position inside of cell drawn from a continuous uniform distribution
19:     cell_pos ← c.random_position
20:     ▷ Get vector from PDF of bivariate normal distribution
21:     rand_vector ← bvn_pdf(cell_pos.μ, cell_pos.Σ)
22:     ▷ Compute which cell rand_vector points to
23:     cell h ← cell(cell_pos + rand_vector)
24:     return h
```

Figure 4.3: Pseudocode for a possible kernel estimation.

since $E(\varepsilon) = 0$ by definition. Furthermore, the variance of $\mathbf{X}$ is

$$\begin{aligned} V(\mathbf{X}) &= E[(\mathbf{X} - \mu)(\mathbf{X} - \mu)'] = E[(L\varepsilon)(L\varepsilon)'] \\ &= E[(L\varepsilon)(\varepsilon' L')] = LE[\varepsilon\varepsilon']L' \\ &= L\mathbb{1}L' = LL' = \Sigma, \end{aligned} \tag{4.7}$$

which is the expected covariance matrix Σ.

This approach can be parallelized, since in the algorithm, both the cells and different samples are independent from each other. In this work, we parallelized this algorithm using Thrust[5], a parallel algorithms library that supports an integration of CUDA (and other parallelization tools) in C++. The number of samples drawn from the Gaussian distribution can be specified by the user.

4.4 Streamline Distributions in Uncertain Vector Fields

The preceding sections covered how to compute a kernel to describe the flow, given a Gaussian distribution at every cell. The theoretical background for the transport of particle densities was given by Otto et al. (see Section 2.4.2). In practice, this means, for every entry of kernel $k_{g,h}$, the current particle density ζ_g in cell g is multiplied by $k_{g,h}$, and added to the particle density ζ_h of cell h, i.e.,

$$\zeta_h = k_{g,h} \cdot \zeta_g. \tag{4.8}$$

Since all particles are supposed to flow at the same time, a buffer is needed, such that during one integration step, the particle densities of the previous step do not alter. This is also indicated in Figure 4.4. To compute the distribution at a certain time, this code is applied multiple times. It is not possible to just specify a larger step size, because this does neither consider a change of direction of the flow nor does it account for diffusion, i.e., particles can also flow back into a cell, they originally came from. The visualization of the distribution of particles after a certain number of integration steps

[5] https://thrust.github.io/

Algorithm 2 One step of particle density integration

```
1:  procedure APPLY_KERNEL
2:      new_densities[number_of_cells]
3:      for g ∈ cells do
4:          new_densities_g ← 0
5:      for g ∈ cells do
6:          if g.density > 0 then
7:              for h ∈ cells do
8:                  new_densities_h ← new_densities_h + k_{g,h} · g.density
9:      for g ∈ cells do
10:         g.density = new_densities_g
```

Figure 4.4: Pseudocode for a kernel application with buffer.

is, subsequently, straightforward. We can simply visualize the grid with the help of an appropriate colour map.

With the help of this approach, it is possible to examine the distribution of particles at a certain time, however, in this work, we propose a technique to visualize the distribution over time. One possible method for achieving this, is an animation, in which every frame represents one integration step. The second possibility is to visualize the 2D slices by stacking them atop of one another. Therefore, we take use of the so-called space-time cube. The space-time cube introduces time as a third dimension to our domain, such that every x-y-slice represents one 2D data slice, while the z-axis describes the progress of time. Figure 4.5 illustrates an example space-time cube, in which a fluid is spreading over time which can be observed in typical diffusion processes. The single slices indicate data slices of the underlying certain vector field. In our use-case, we treat the distribution at every time step as one slice of the space-time cube, such that advancing time indicates a longer integration time. For a meaningful visualization of this streamline distribution, we apply an isosurface to the data, such that we generate a surface which surrounds all regions that have a higher density than the specified value of the isosurface. It is also possible to employ multiple isosurfaces to differentiate between areas with different particle densities. In these cases, it is useful to reduce the opacity of the single contours. However, before

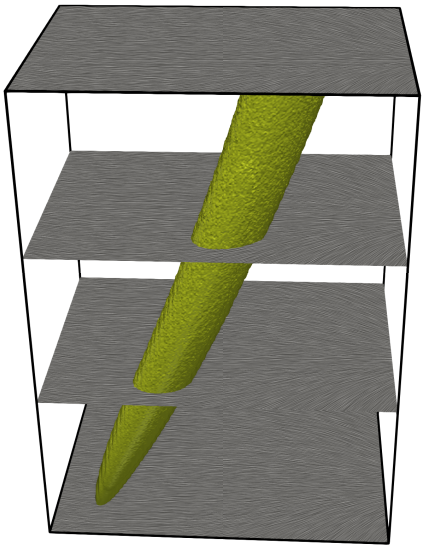


Figure 4.5: Space-time cube. The LIC of the different slices indicate the vector field at different time steps.

applying an isosurface to our data, we first need to convert our grid, such that instead of cell-based data, node-based data is stored. This is due to the fact that the marching cube algorithm [18] is used to create those contours. Nonetheless, this method allows for an at-a-glance visualization of streamline distributions in an uncertain vector field. It is also possible to seed particles in more than one cell. In this case, multiple streamline distributions are not overlapping each other, instead, we can observe at which points the streamline distributions do or do not intertwine, and can identify critical regions of the flow. However, if we seed streamline distributions at too many cells at once, it becomes impossible to identify single instances.

4.5 Description of Algorithm

The used algorithm consists of two – if the kernel is computed on demand – or three steps, if the kernel is precomputed. If we assume a given vector field, as well as a given uncertainty at every node, stored in a structured grid, the steps are the following:

1. Initialization of the distribution grid. Depending on the bounds of input grid and the user-specified dimensions of the distribution grid, the distribution grid is constructed, and all cells are filled with a particle density of 0. Afterwards cells that are dedicated to be starting points of a streamline integration are set to a particle density greater than 0. Therefore, the user specifies a certain position in the grid, and the algorithm determines the correct cell that belongs to the given point. For numerical reasons, the start density in such a cell is set to $10,000,000$ by default.

2. The second step is the optional precomputation of the kernel, as described in the previous sections. Usually, this step is skipped since this limits the method to small grid dimensions.

3. Application of the kernel. The user has multiple options to influence this step. Firstly, the number of iterations, i.e., the number of successive executions of the kernel, can be specified. Secondly, the number of samples *num_samples* that are used to approximate the CDF of the bivariate normal distribution can be altered. However, in the algorithm, the total number of possible samples $samples_{all}$ for all cells is limited, i.e., if $num\_samples \times num\_cells_{density > 0} > samples_{all}$, the number of samples per cell is decreased, such that the overall number of samples is less or equal to $samples_{all}$. $num\_cells_{density > 0}$ is the number of cells that have a density greater than 0. The sampling step itself is parallelized with the help of Thrust and CUDA. Thirdly, the used Gaussian distribution that is used to represent the uncertainty, can be scaled by a factor n. For a given $N \sim (\mu, \Sigma)$, this means $N \sim N(n\mu, n\Sigma)$ is processed internally. This is the equivalent of the step size in the computation of certain streamlines.

We can store only the final distribution grid, or a slice is saved for every integration step, in cases where we want to visualize a streamline distribution by the means of a space-time cube. Every slice then represents one layer of a rectilinear grid.

5 Results

In this chapter, we apply the presented algorithms to several analytic vector fields, as well as to a real test data set. Firstly, we describe the test data set. Secondly, we present results for the tracking of single uncertain streamlines. Thirdly, results for streamline distributions in uncertain vector fields are presented and, finally, the results are discussed.

5.1 Description of the Test Data Set

The data set used in this chapter is from a 3D buoyant flow simulation, in which the bottom is heated and the top is cooled, while the sides are adiabatic surfaces, resulting in a flow between the top and bottom boundary. The complete dataset consists of $61 \times 31 \times 61$ nodes, and there exists data for multiple time steps of the simulation. However, we are only interested in steady 2D vector fields. The used dataset is extracted from a time step in the middle of the simulation, and we took only one slice along the x-y-plane. The resulting 2D dataset has an extent of 61×31 nodes. The domain bounds range from 0 to 0.10002 along the x-axis, and 0 to 0.05001 along the y-axis. Figure 5.1 shows the streamlines of the used dataset. To simulate uncertainty in the data set, we assume a bivariate Gaussian distribution $X \sim N(\mu, \Sigma)$ at every node, with the vector being the mean μ of the distribution. The variance σ^2 is not defined by the data itself, but an arbitrary value is assigned for σ_x and σ_y. The correlation ρ is assigned 0, i.e., the uncertainty in x-direction is independent from the uncertainty in y-direction. Consequently, the covariance matrix Σ_X at every node presents as follows:

$$\Sigma_X = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}. \tag{5.1}$$

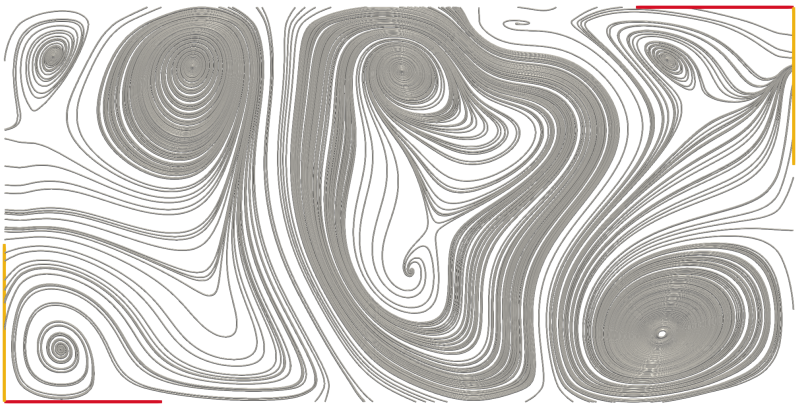
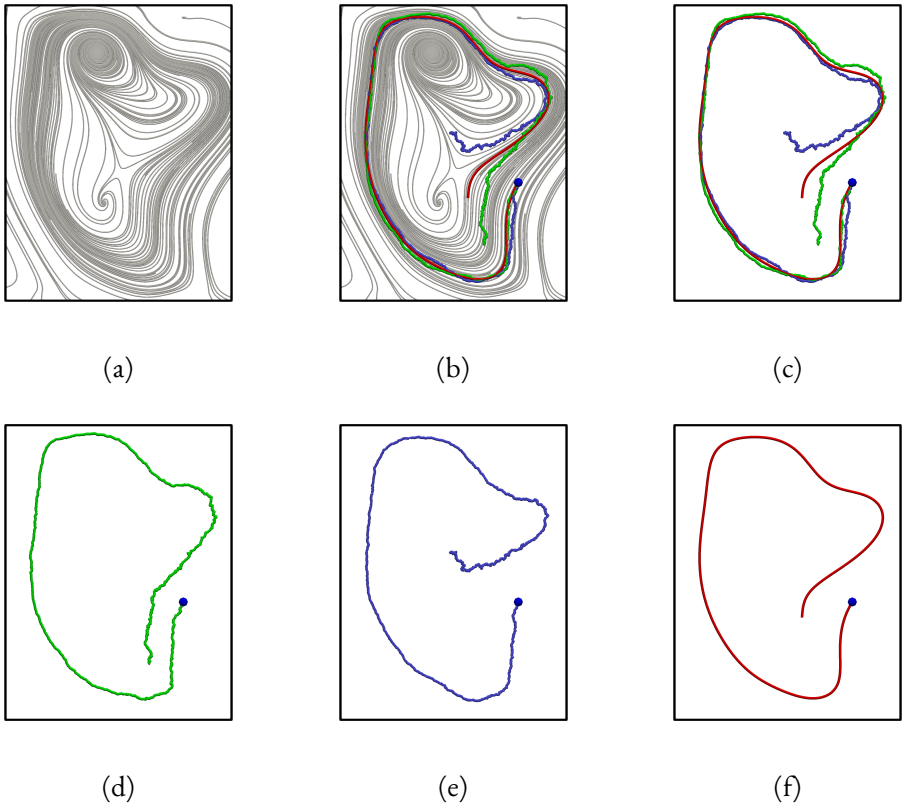


Figure 5.1: Certain streamlines of the used test data set derived from a buoyant flow without any added uncertainty.

5.2 Single Uncertain Streamlines

The algorithm presented in Section 3.2 has been applied to the dataset, and the results are presented in Figure 5.2. It should be noted that Figure 5.2 has been cropped, such that only the relevant area of the domain is seen. In Figure 5.2a, the streamlines of the underlying certain vector field are shown, i.e., $\sigma_x = \sigma_y = 0$ and, hence, the integration depends only on the mean μ. In Figure 5.2d and Figure 5.2e, a direct comparison of two different uncertain streamlines can be observed, while in Figure 5.2f the certain streamline is shown. Consequently, this line represents the correct streamline, as if no uncertainty were present. The green, blue, and red streamlines are all seeded at position $u = (0.065, 0.02)$, indicated by the blue glyph, and then forward integrated. For the two uncertain streamlines in Figure 5.2e and Figure 5.2d a fixed variance is assumed at every node, and $\sigma_x^2 = \sigma_y^2 = 0.0025$. In direct comparison, it can be observed that the general behaviour of uncertain streamlines is similar in most regions of the field, however, especially near saddle points the streamlines' behaviour can differ greatly. Figure 5.2b and Figure 5.2c show the three different streamlines in direct comparison, either with or without the underlying certain vector field.



(a) (b) (c)

(d) (e) (f)

Figure 5.2: Visualization of different uncertain streamlines. (a) Streamlines of the underlying vector field, (b) streamlines seeded at location $u = (0.065, 0.02)$ (blue glyph) and streamlines of the underlying vector field, (c) streamlines seeded at u, (d) and (e) uncertain streamlines seeded at u, and (f) certain streamline seeded at u.

(a) (b) (c)

Figure 5.3: Visualization of uncertain streamlines. (a) and (b) uncertain streamlines seeded at location $u = (0.065, 0.02)$ (blue glyph), (c) comparison with certain streamline.

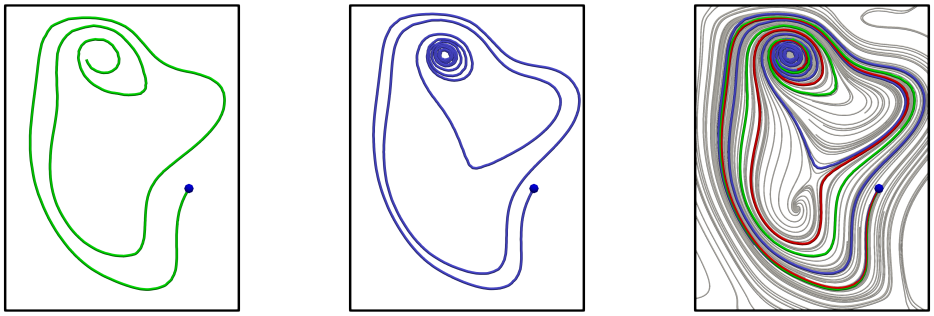
In Figure 5.3 the same case as in Figure 5.2a is analysed, but the integration time is increased. It is observed that, even though at the saddle point the uncertain streamlines have a different local behaviour, after a longer integration both streamlines eventually converge to a critical sink point, which is the result one would expect. Figure 5.3c also indicates the certain streamline, which is in line with the behaviour of the uncertain streamlines, and confirms the behaviour of the uncertain streamlines.

5.3 Domain-Uncertain Streamlines

Figure 5.4 shows a comparison between two different streamlines with an underlying uncertain domain. The vector field is the same as the test data set. However, in contrast to Section 5.1, the vector field itself is treated certain, i.e., that at every node only the mean μ is considered while Σ is dropped, i.e., $\Sigma = 0_{2\times2}$. Instead, a 2D covariance matrix Σ_D is defined for every node. Σ_D describes the distribution of the uncertain point at the respective position. In every randomization step, the node position p of all uncertain points is then determined by drawing a random position out of a bivariate Gaussian distribution $X \sim N_2(\mu_D, \Sigma_D)$, where μ_D is the mean position of the respective uncertain point. Since the vector field itself is not time-dependent, only the physical position of the points is uncertain. Interpolation is then treated as described in Section 3.3, i.e., the seeding point is assumed to be certain. In Figure 5.4 the chosen Σ is

$$\Sigma_D = \begin{pmatrix} 0.0004 & 0 \\ 0 & 0.0004 \end{pmatrix}. \tag{5.2}$$

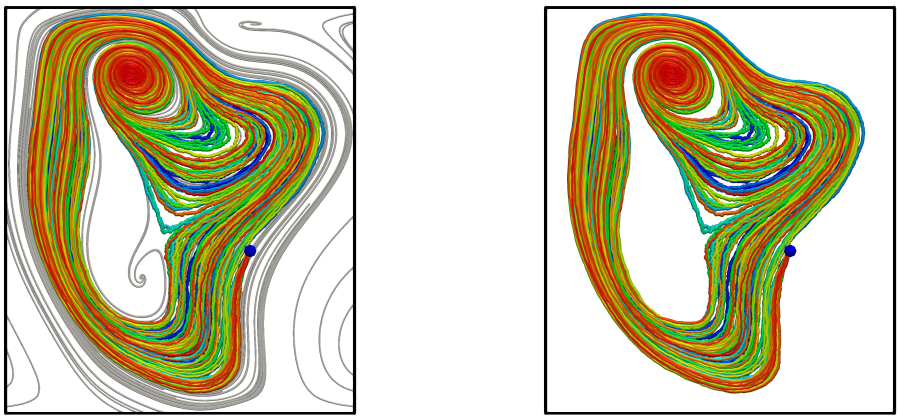
Figure 5.4a and Figure 5.4b both show streamlines seeded at position $u = (0.065, 0.02)$. We can observe a similar behaviour of both streamlines, which is the expected result, however, like with uncertain streamlines at saddle points, different streamlines on an uncertain domain can split up. After a sufficiently long integration, streamlines on an uncertain domain will also converge to critical sink points, if those exist in the flow field. Streamlines produced with the presented algorithm are much smoother than uncertain streamlines, and do not show a zig-zag-pattern, as we could observe in Figure 5.2 and Figure 5.3. It can be assumed that this is due to the interpolation method: Especially in homogeneous



(a) (b) (c)

Figure 5.4: Visualization of streamlines on an uncertain domain. (a) and (b) streamlines seeded at location $u = (0.065, 0.02)$ (blue glyph), (c) comparison with streamline on certain domain.

flow regions, the randomization of the surrounding nodes does not matter much, since all vectors point in a similar direction and, thus, also the resulting vector goes in the same direction; only in areas like a saddle point, where the vectors at two near node points differ greatly, the vector has a large possible range of values. And even then, it is unlikely that the vector will take on one of the two extremes, but some value in between. Another matter is that up to 12 points are taken into account during interpolation, which makes outliers less likely, since often-times an extreme outlier of one uncertain point will be compensated by 11 other points and, hence, its influence is restricted. For these reasons, we can conclude that in regions of a steady flow, streamlines on uncertain domains will only have a few outliers while most streamlines will stay in a close vicinity to each other. Only when they encounter regions, where the flow splits up, e.g., at saddle points, the streamlines will strongly diverge from each other. Figure 5.4c confirms this: Until the streamlines on the uncertain domain encounter, the saddle point they are nearly identical to the red streamlines, which is integrated on the certain domain ,i.e., no randomization of the node points. However, after the flow has passed by the saddle point the streamlines differ a lot.



(a) (b)

Figure 5.5: 50 uncertain streamlines seeded at location $u = (0.065, 0.02))$ (blue glyph) (a) with and (b) without streamlines of the underlying certain vector field. Every colour indicates a different streamline.

5.4 Sets of Uncertain Streamlines

Beforehand, we only analysed the behaviour of single uncertain streamlines, however, in the case of uncertain streamlines, this generally does not reflect the behaviour of the streamline distribution. To get a better grasp of the behaviour of an uncertain streamline distribution, one could simply visualize multiple streamlines at once. In Figure 5.5, we can observe a visualization of 50 different uncertain streamlines in the same image. All 50 streamlines are seeded at the same position $u = (0.065, 0.02)$, and the variance was set to $\sigma_x = \sigma_y = 0.0025$, such that the illustration of Figure 5.5 is comparable to the visualization in Figure 5.2f and Figure 5.3c. Figure 5.5 indicates that seeding a multitude of streamlines at a certain position, is not a suitable way to visualize the behaviour of a streamline distribution in an uncertain vector field. The visualization is much too cluttered, the path a particle went along is not recognizable at all and, hence, an analysis of Figure 5.5 would not offer too many meaningful insights. Important details, like the most probable path a particle will take, cannot be monitored, and at best, this visualization serves as a rough first impression of a streamline distribution on an uncertain domain.

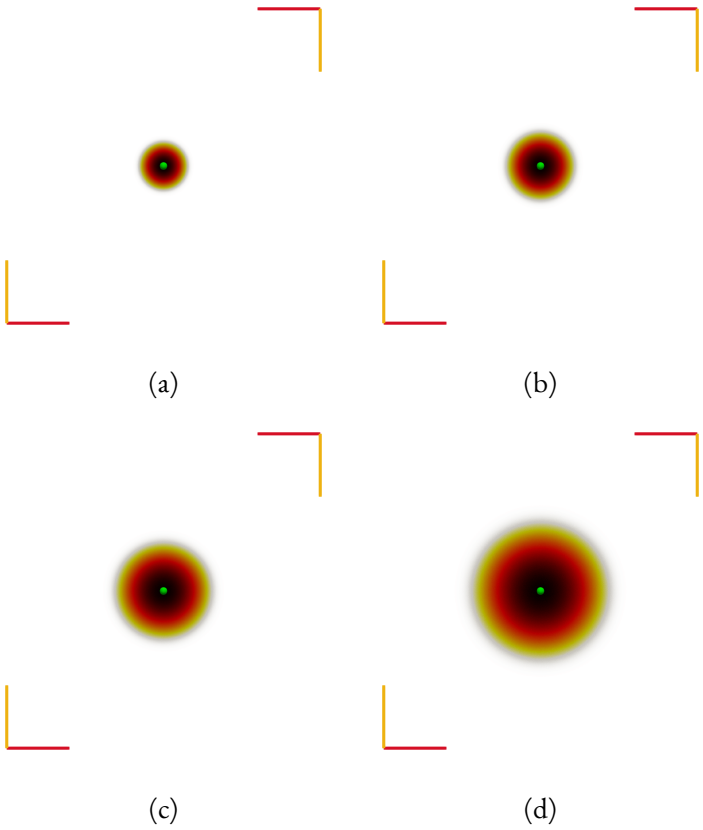


Figure 5.6: Distribution grid initialized at position $u = (0, 0)$ (green glyph) after (a) 50, (b) 100, (c) 200 and (d) 400 integration steps. The underlying uncertain vector field has velocity $v = (0, 0)$ at all nodes. Darker areas indicate a higher particle density.

5.5 Distribution Grids

In the following section, we analyse the behaviour of the integration of particle densities applied to different 2D analytical fields. Every field is assumed to be Gaussian distributed, such that the vector field consists of μ, which is considered to be the "real" vector field, and an added covariance matrix Σ. Therefore, we use distribution grids to visualize the distribution of particles at different time steps.

5.5.1 Zero-Velocity Vector Field

The first analytic field, we define, is a vector field $v(x, y)$ without any flow, such that

$$v(x, y) = 0 \cdot \begin{pmatrix} x \\ y \end{pmatrix}, \quad \text{and} \quad \Sigma = \begin{pmatrix} 0.005 & 0 \\ 0 & 0.005 \end{pmatrix}, \tag{5.3}$$

over the domain $[-1, 1]^2$. The field is discretized on a grid of 50×50 nodes. Without the influence of the introduced uncertainty, we would observe no streamlines at all, since there is no velocity that transports any particles.

Figure 5.6 shows the results after a varying number of integration steps. The distribution grid has a resolution of 801×801 cells, and it holds that $\textit{num\_samples} = \textit{samples}_{\textit{all}} = 400.000.000$ (see Section 4.5). The initial particles are seeded at $u = (0, 0)$. The used colour map indicates higher particle densities at darker areas, and makes use of a logarithmic scale. In all four figures, we can observe an isotropic Gaussian distribution. Over time the particles diffuse and distribute over the domain. This is the behaviour one would expect in such a field. If no velocity is present, then the only influence on the particles is the uncertainty, and the distribution forms a shape given by Σ around the point where the particle density was originally seeded.

5.5.2 Constant Vector Field

We define an uncertain vector field $v(x, y)$ with a constant flow, such that

$$v(x, y) = \begin{pmatrix} 0.005 \\ 0 \end{pmatrix}, \quad \text{and} \quad \Sigma = \begin{pmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{pmatrix}, \tag{5.4}$$

over the domain $[-1, 1]^2$ and with a grid size of 50×50 nodes. The initial particles are seeded at $u = (-0.75, 0)$; otherwise the specifications are the same as in Section 5.5.1.

In Figure 5.7, we can observe that the behaviour of the distribution in this vector field is similar to the behaviour of the distribution in a vector field with no velocity at all. Again, we can identify the Gaussian shape, we could recognize in the previous section, but the distribution moves with the flow, such that the cell with the highest density is not the

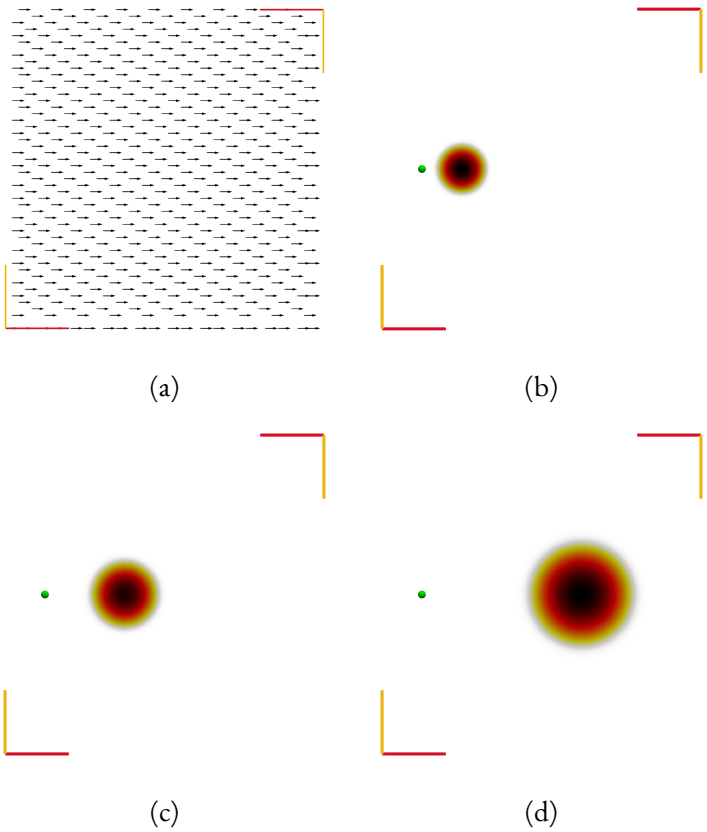


Figure 5.7: Distribution grids on constant uncertain vector field. (a) Underlying certain vector field. Distribution grid initialized at position $u = (-0.75, 0)$ (green glyph) after (b) 50, (b) 100, and (c) 200 integration steps. Darker areas indicate a higher particle density.

initial seeding cell. Instead, this point of highest density moves the same way as we would expect from a certain streamline.

5.5.3 Diverging Vector Field

Next, we want to examine a vector field $v(x, y)$ with a flow diverging from the x-axis. The uncertain vector field is given by

$$v(x, y) = 0.005 \cdot \begin{pmatrix} 1 \\ sgn(y) \end{pmatrix}, \quad \text{and} \quad \Sigma = \begin{pmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{pmatrix}, \tag{5.5}$$

over the domain $[-1, 1]^2$ and with a grid size of 65×65 nodes, and otherwise with the parameters used in Section 5.5.2. The signum function is defined as

$$\forall\, x \in \mathbb{R}: \quad sgn(x) = \begin{cases} +1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases} \tag{5.6}$$

Figure 5.8 shows the results of applying our algorithm to this diverging vector field. The particle density distribution does no longer resemble the typical shape of a bivariate Gaussian distribution. Instead, the distribution starts to flow away from the x-axis and splits up, leaving two separate areas of particle aggregations. Additionally, we can observe a tail both particle density distributions are leaving behind. We assume that this behaviour is due to the fact that near the x-axis, the velocity in y-direction is lower than at a distance from the x-axis This is a typical case that can be observed when a streamline encounters a saddle point. While in this scenario the vector field is perfectly symmetrical, in real-world problems, this behaviour is less perfect. However, the general behaviour of the separating flow, and the tail we detect, can be assumed to be also observed in a real-world process.

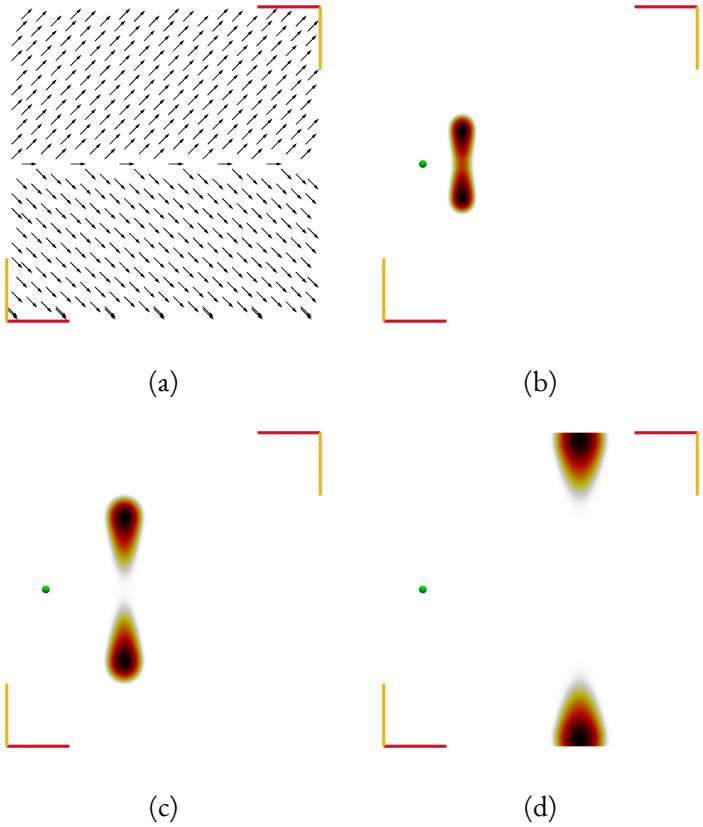
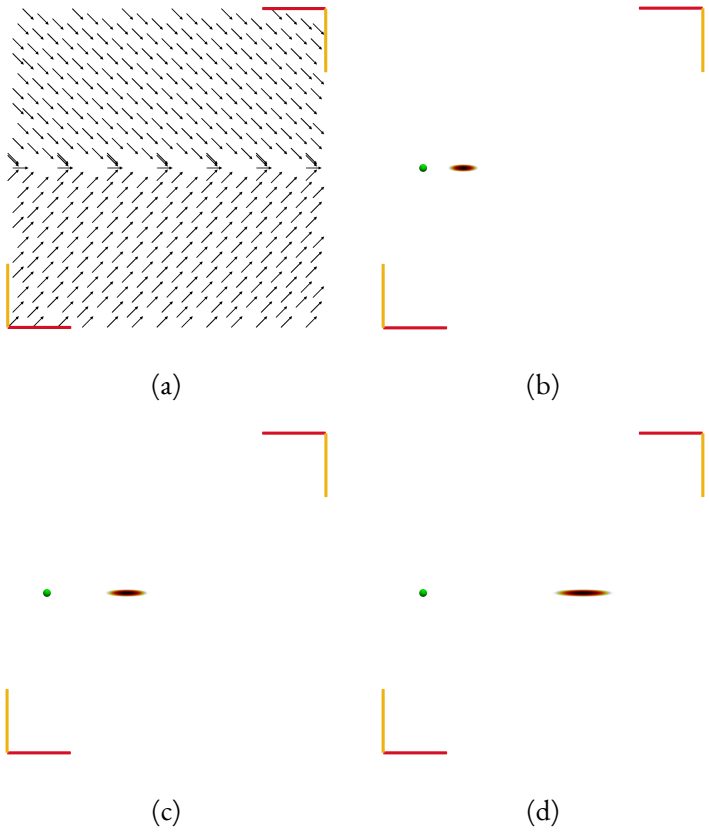


Figure 5.8: Distribution grids on diverging uncertain vector field. (a) Underlying certain vector field. Distribution grid initialized at position $u = (-0.75, 0)$ (green glyph) after (b) 50, (b) 100, and (c) 200 integration steps. Darker areas indicate a higher particle density.



(a) (b) (c) (d)

Figure 5.9: Distribution grids on converging uncertain vector field. (a) Underlying certain vector field. Distribution grid initialized at position $u = (-0.75, 0)$ (green glyph) after (b) 50, (b) 100, and (c) 200 integration steps. Darker areas indicate a higher particle density.

5.5.4 Converging Vector Field

We define the uncertain vector field $v(x, y)$

$$v(x, y) = 0.005 \cdot \begin{pmatrix} 1 \\ -sgn(y) \end{pmatrix}, \quad \text{and} \quad \Sigma = \begin{pmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{pmatrix}, \tag{5.7}$$

over the domain $[-1, 1]^2$, and with a grid size of 65×65 nodes. The vector field converges towards the x-axis as indicated in Figure 5.9a. Otherwise, we used the same specifications as in Section 5.5.2.

In Figure 5.9, we can observe that in areas of an uncertain vector field that shows converging behaviour, the resulting particle density distribution is not isotropic, and does not solely rely on the form of Σ. Instead, the shape is influenced by the inflow that compresses the distribution in direction of the y-axis. The resulting shape becomes anisotropic, and does no longer resemble the form of a Gaussian distribution. Therefore, we can conclude that in a converging flow, the resulting distribution also converges to the centre of the flow.

5.5.5 Rankine Vortex

A two-dimensional rigid body rotation $u(x, y)$ is defined as

$$u = \begin{pmatrix} -\Omega y \\ \Omega x \end{pmatrix}, \tag{5.8}$$

where Ω is the angular velocity. We calculate a Rankine vortex by setting the magnitude of every vector in the vector field of the rigid body rotation to

$$\|u\| = \begin{cases} \frac{r}{R} \cdot v_{max} & \text{if} \quad r \leq R, \\ \left(\frac{R}{r}\right)^2 \cdot v_{max} & \text{if} \quad r > R, \end{cases}, \tag{5.9}$$

in regard to the radius r from the centre of the vortex, the radius R where velocity is maximal, and the maximum velocity magnitude v_{max}. In a two-dimensional Rankine vortex, the velocity of the flow increases linearly from the centre of the vortex to elements of the fluid that have distance R to the core. Outside of radius R, the velocity decreases super-hyperbolically. Figure 5.10 shows results using a Rankine vortex with $R = 0.5$, and $v_{max} = 0.025$ over the domain $[-1, 1]^2$ with a grid resolution of 65×65 nodes. Additionally, we set Σ to

$$\Sigma = \begin{pmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{pmatrix}. \tag{5.10}$$

Figure 5.10a shows the general flow of the vortex, i.e., the vectors are not scaled by magnitude. Figure 5.10b depicts the magnitude at different regions of the flow, indicating that the vortex really represents a Rankine vortex. The distribution grid visualized in Fig-

ure 5.10c to Figure 5.10f consists of 801×801 cells, and it holds that $num\_samples = samples_{all} = 100.000.000$. We can observe that the shape of the streamline distribution becomes a crescent after several integration steps, and the distribution is slightly pushed to the outside of the vortex. Consequently, we can assume that in such vortex like regions with a high velocity, the streamline distribution gets a longish shape that does no longer resemble a Gaussian distribution.

5.5.6 Varying Uncertainty and Distribution Grid Resolutions

In Figure 5.11, we can observe the influence of varying uncertainties, as well as different grid resolutions. From left to right, the distribution grid resolution increases from 801×801 to 1601×1601 to 3201×3201 cells. From top to bottom, the uncertainty increases from $\sigma_y = \sigma_x = 0$ to $\sigma_y = \sigma_x = 0.001$ to $\sigma_y = \sigma_x = 0.0025$. It is apparent that a higher uncertainty yields distribution grids in which the particles are more spread out, while a lower uncertainty results in areas with concentrated particle densities. However, the same effect emerges with different grid resolutions. A higher resolution yields more condensed particle densities. This is a behaviour inherent to the method, since we do not seed particles at a single infinitesimal small point in our domain, but in one cell of a distribution grid, i.e., we set an area of our domain to a certain particle density. Because the distribution inside of one cell is assumed to be continuously uniform, this method can no longer track certain streamlines, even if the uncertainty is considered to be 0. Nonetheless, increasing the grid resolution can limit this effect, as can be seen in Figure 5.11. This is due to the fact that an increase of the grid resolution also implies a decrease of the cell size and, hence, the seeding area becomes smaller and smaller, such that the seeding area converges to the given seeding point itself.

5.5.7 Test Data Set

Finally, we apply our algorithm to the test data set described in Section 5.1. We start our integration at position $u = (0.065, 0.02)$ on a grid with 800×400 cells, and $num\_samples = samples_{all} = 400.000.000$. Additionally, we scale our Gaussian distribution by 0.005 at every position, which is equivalent to reducing the step size when

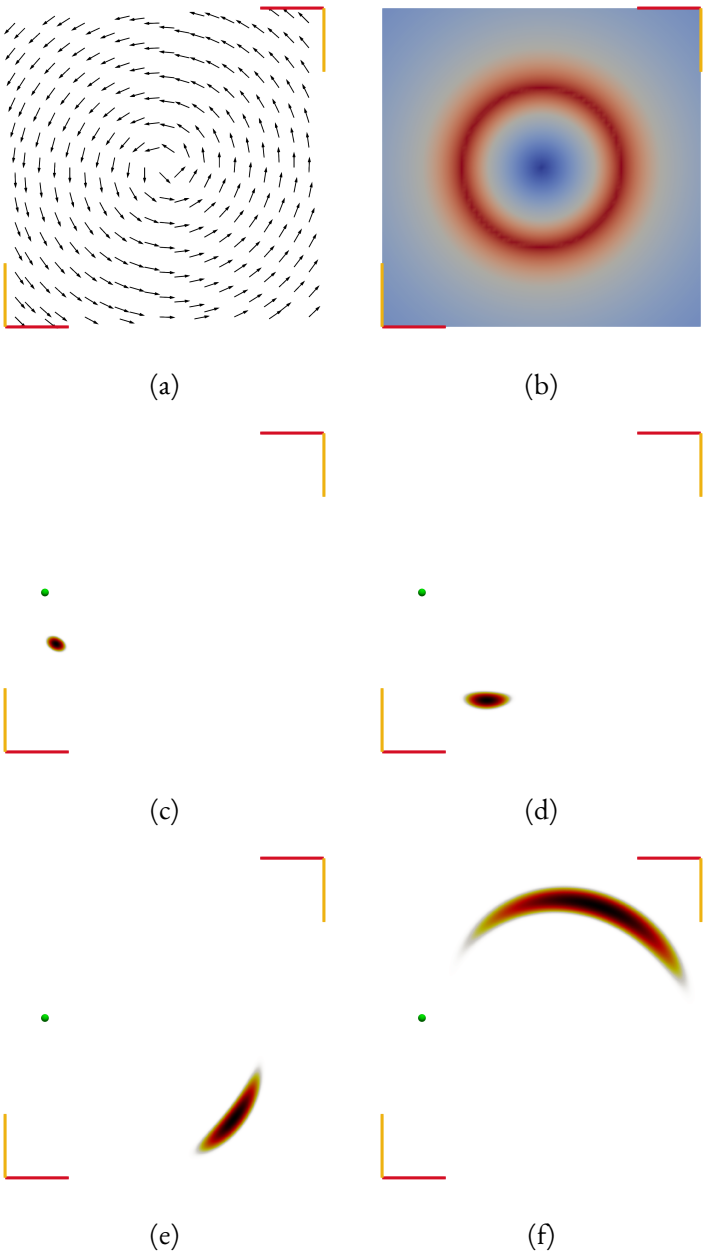


Figure 5.10: Distribution grids on uncertain vector field using a Rankine vortex model. (a) Underlying certain vector field. (b) Magnitude of vectors: Red indicates a higher magnitude. Distribution grid initialized at position $u = (-0.75, 0)$ (green glyph) after (c) 20, (d) 50, (e) 100 and (f) 200 integration steps. Darker areas indicate a higher particle density.

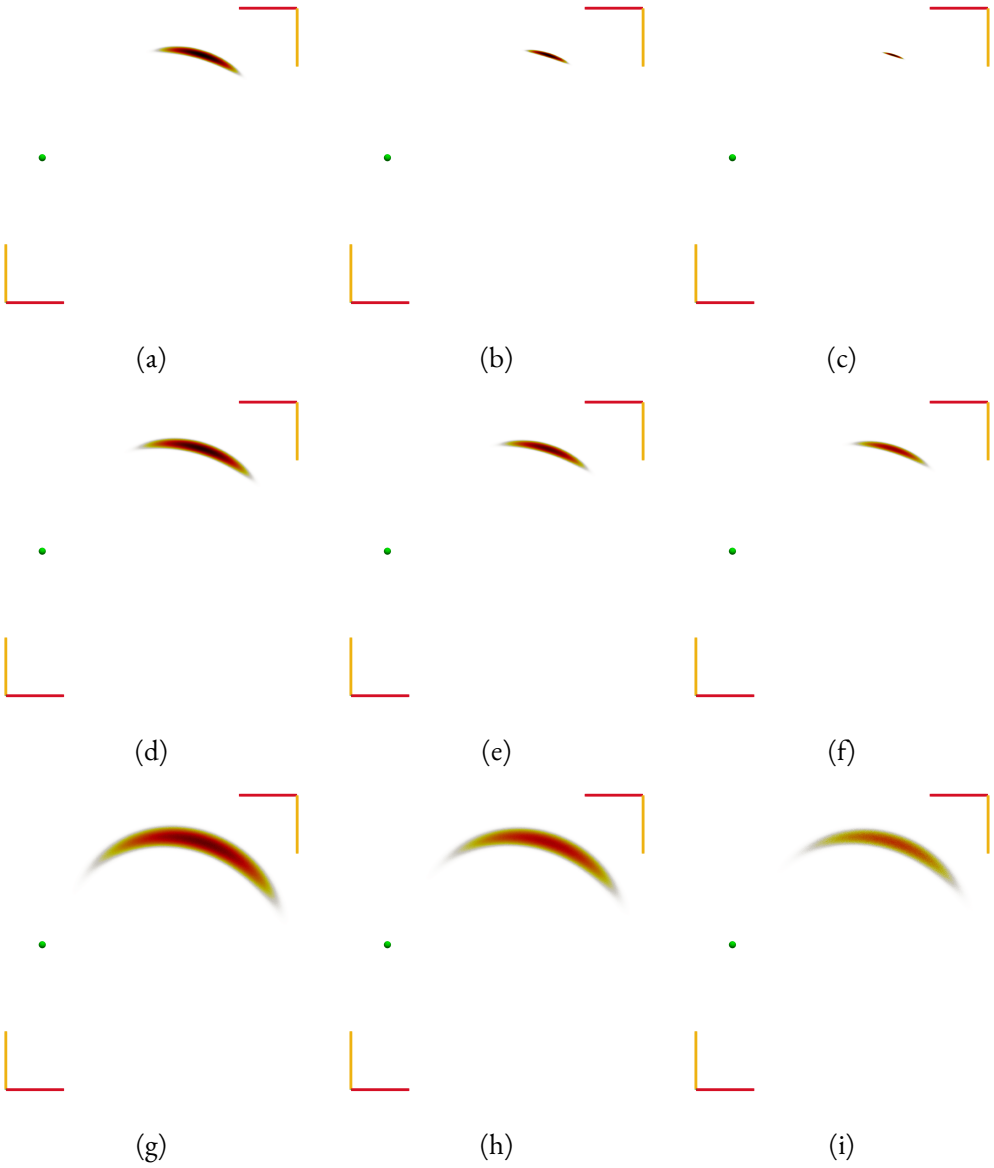
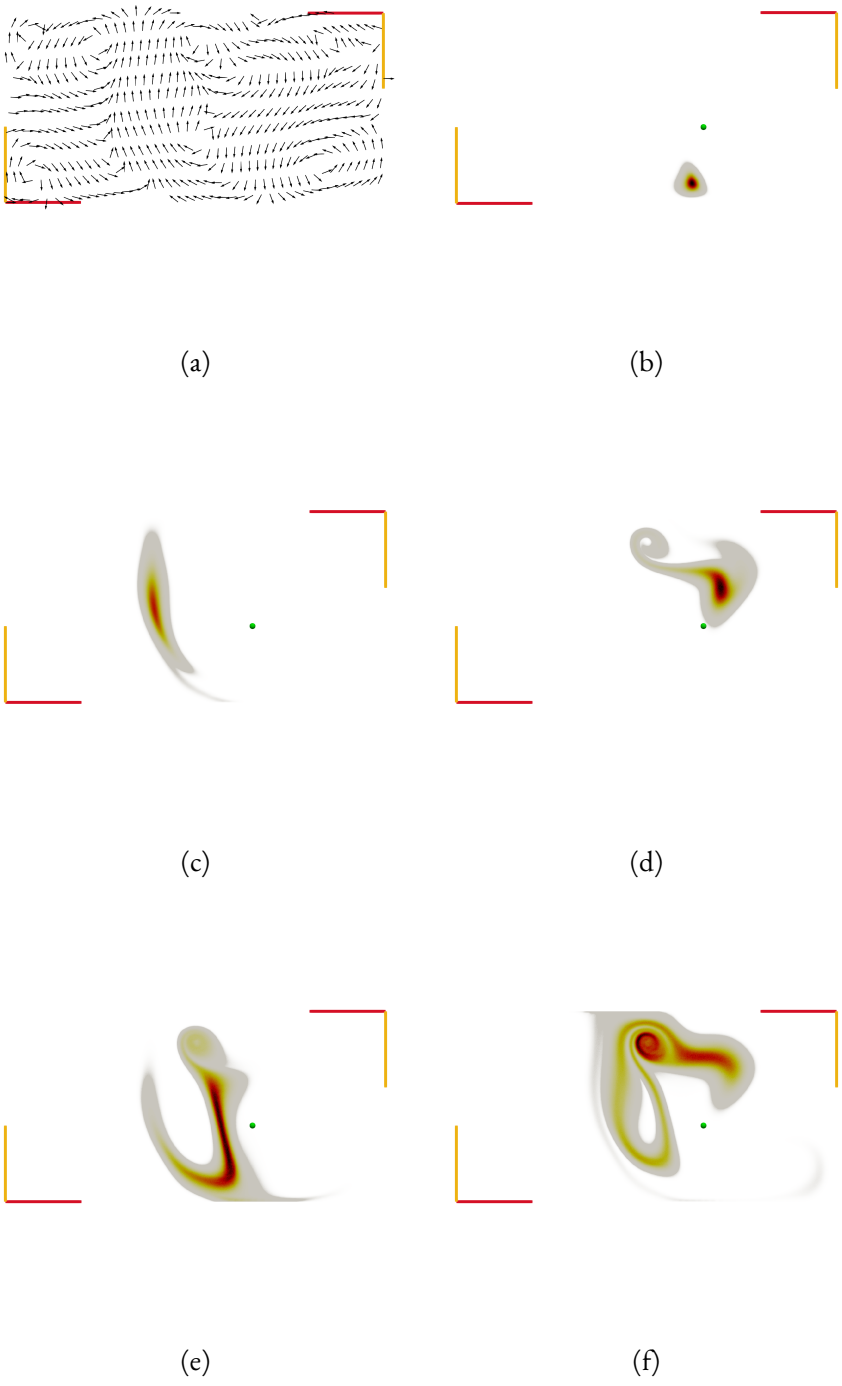


Figure 5.11: Distribution grids on uncertain vector field using a Rankine vortex model (compare Figure 5.10) with varying uncertainty and grid resolution. All streamline distributions seeded at position $u = (-0.75, 0)$. Row 1: $\sigma_y = \sigma_x = 0$, row 2: $\sigma_y = \sigma_x = 0.001$, row 3: $\sigma_y = \sigma_x = 0.025$, column 1: 801×801 cells, column 2: 1601×1601 cells, column 3: 3201×3201 cells.



(a) (b)

(c) (d)

(e) (f)

Figure 5.12: Distribution grids on uncertain vector field with the underlying test data set. (a) Underlying certain vector field. Distribution grid initialized at position $u = (0.065, 0.02)$ (green glyph) after (b) 250, (c) 500, (d) 1000, (e) 1500, and (f) 2000 integration steps. Darker areas indicate a higher particle density. Note that in contrast to previous representations, the colour map is not logarithmic.

computing certain streamlines. At every node of the data grid, we define the covariance matrix:

$$\Sigma = \begin{pmatrix} 0.005 & 0 \\ 0 & 0.005 \end{pmatrix}. \tag{5.11}$$

The results are shown in Figure 5.12, with Figure 5.12a indicating the general flow behaviour, and Figure 5.12b to Figure 5.12f representing the distribution grid after a varying number of integration steps. When comparing the distribution of particles over time with the behaviour of the single uncertain streamlines extracted in Section 5.2, we can see that both illustrations generally coincide. However, the representation provided in Figure 5.12 gives a clearer insight into the particle distribution after a certain time. By examining the streamlines in Figure 5.1, we can also conclude that in this field, the behaviour of the streamline distributions is in accordance with our previous analyses. Generally, the density distribution is growing, however, the shape gets deformed differently in separate areas of the flow. In areas where the flow converges, the distribution is compressed and becomes longish (Figure 5.12c). In areas where the flow diverges, the density distribution drifts apart and, especially near saddle points, can split up (Figure 5.12d to Figure 5.12f). Nonetheless, this visualization is not perfect. To get complete insight into the behaviour of the streamline distribution, we would need many successive images of the distribution grid. In computer applications, this can be done by creating animations that show the behaviour of the flow over time frame by frame. However, this illustration is not suitable to visualize the structure of a streamline in one single image. Therefore, we present an alternative in the next section.

5.6 Streamline Distributions in Space-Time

In this section, we present an at-a-glance visualization of streamline distributions for uncertain two-dimensional vector fields. We will observe the results of the technique using vector fields already presented in the previous section, and integrate the single slices into space-time representation as described in Section 4.4.

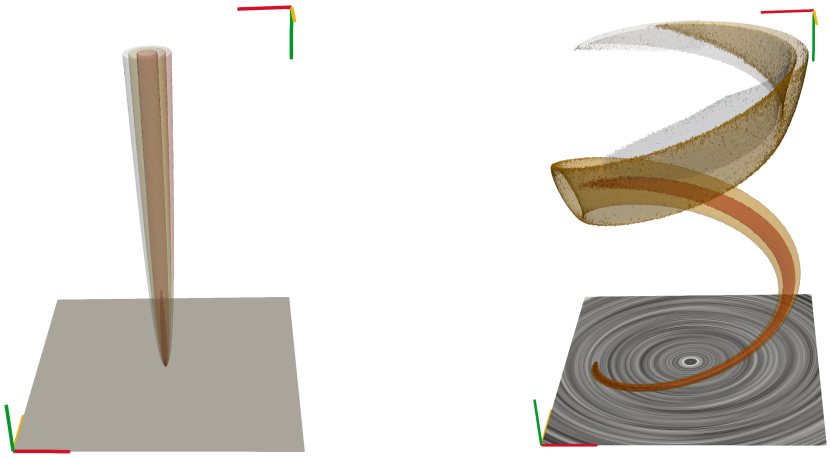
5.6.1 Analytic Vector Fields

Figure 5.13a shows a streamline distribution visualization of the same vector field as in Section 5.5.1, such that no μ-flow is present in the vector field. Time is increasing from the bottom to the top of the space-time cube, and our seeding cell is at position $u = (0, 0)$, such that our simulation is the same as in Figure 5.6. By visualizing the density of particles in certain areas, with the help of contours using varying isovalues, we can separate areas with varying particle density. Subsequently, we can observe that the distribution spreads out over time, and that most particles remain in the centre of the distribution. Figure 5.13b indicates the streamline distribution on the same Rankine vortex as used in Section 5.5.5 with a seeding cell at position $u = (-0.75, 0)$. In this representation, the circular structure of the streamline distribution is apparent at-a-glance, and we can observe that the distribution gets stretched along the stream, resulting in a longish shape, swirling around the centre of the vortex. When analysing the different isolevels, we can identify less layers at the end of the integration than right after the start. This is due to the fact that, with progressing time, the particle density in each cell gets thinned out, and is more equally distributed at later time steps. Especially in Section 5.5.5, it becomes apparent that at the end of the integration, there are no cells with a high particle density left.

Figure 5.14 shows the same cases as Figure 5.13. However, an additional seeding cell is chosen, such that in the initial distribution grid, two cells have a density greater than 0. In Figure 5.14a, the second seeding cell is located close to the first one, such that after a few iterations, the two distributions start to overlap. In areas where two streamline distributions overlap, we can observe that the density of both streamline distributions adds up, while the structure remains unchanged in locations, in which the distributions do not overlap. In Figure 5.14b, the second seeding cell is at the opposing side of the vortex. It is observed that the two streamline distributions do not overlap at all, but are independent of each other. Furthermore, both streamline distributions are identical, only mirrored at the vortex core.

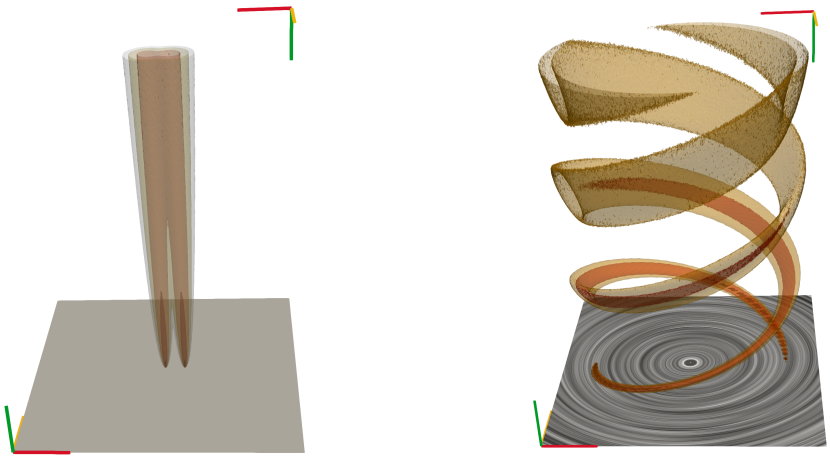
5.6.2 Test Data Set

Finally, we apply the proposed technique to our test data set. The single slices of the space-time cube are calculated as in Section 5.5.7. In contrast to the representation using



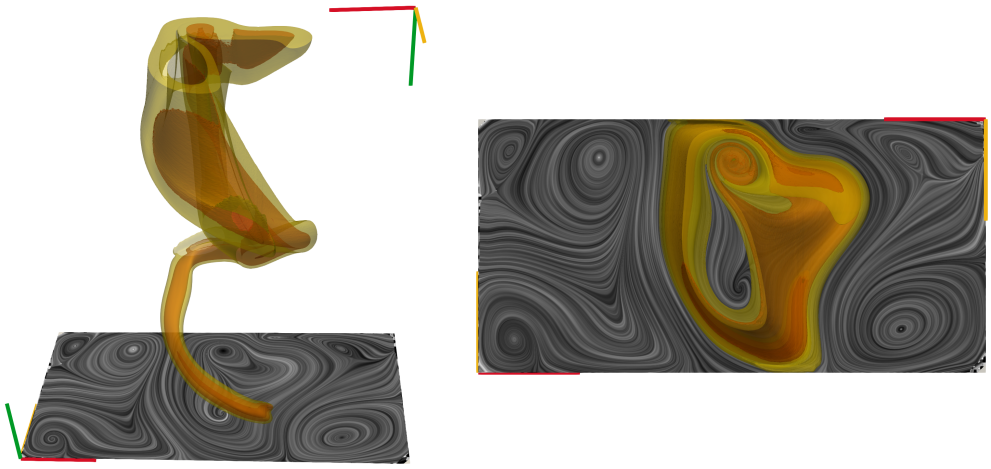
(a) (b)

Figure 5.13: Streamline distributions of (a) same vector field as in Figure 5.6, starting point $u = (0, 0)$ and 200 integration steps, (b) same vector field as in Figure 5.10, starting point $u = (-0.75, 0)$ and 500 integration steps. Darker areas indicate a higher particle density.



(a) (b)

Figure 5.14: Streamline distributions as in Figure 5.13 but with additional seeding points (a) $q = (0.15, 0)$ and (b) $q = (0.75, 0)$.



(a) (b)

Figure 5.15: Streamline distribution of the test data set with $\sigma_x^2 = \sigma_y^2 = 0.005$ and starting point $u = 0.065, 0.02$. (a) View from the side and (b) view from the top of the space-time cube.

single slides, the streamline distribution can be depicted in one image. In Figure 5.15a, the streamline distribution is represented within the space-time cube. For computation 2000 integration steps were performed, otherwise the same parameters were chosen as in Section 5.5.7. It is observed that the streamline distribution does in fact behave similarly to the observed structure that was observed, when visualizing single uncertain streamlines of the field. In this case, it becomes additionally apparent in which areas the particle density is the highest. This is a huge improvement to the visualization technique presented in Section 5.4, where we computed a whole set of uncertain streamlines. With the means of this visualization, it is possible to easily identify areas where the streamline distribution encounters a saddle point, i.e., where the streamline distribution splits up. Areas with a high particle density at the end of the integration can indicate areas to which the flow converges to, hence, representing critical sinks. However, these areas do not necessarily represent sinks, e.g., in cases where no sinks are present in the vector field. Additionally, segments of the flow with high velocity are equal to parts where the streamline distribution moves fast as well. Figure 5.15b shows the space-time cube from the top. The image emphasizes the fact that the computed streamline distribution has the same general structure as the uncertain streamlines presented in Section 5.4

6 Conclusion

The first concept presented in this thesis was domain uncertainty. In this context, uncertain domains offer a promising addition to uncertain streamlines. To the best of our knowledge, there is no direct transformation from domain-uncertain vector data to uncertain vector fields. Hence, domain-uncertain data can be of use in certain scenarios, and in the future could give insight into new problems, e.g., in geoinformatics. However, at the moment, it is not known to us that appropriate field data is available, which is defined on uncertain domains and, thus, there are not many use-cases, for which this approach is suitable at this time. One reason for this could be that not much work was done in the area of domain-uncertainty, or similar concepts and, hence, it is more convenient to produce data that can be processed by already established methods. By providing a fundament and algorithms that can work with uncertain domains, this can hopefully be changed in the future, such that more benefits of uncertain domain representations will be identified. We presented and applied an algorithm using the concept of domain-uncertainty to a test data set, compared those with results yielded by computing uncertain streamlines, and confirmed our assumption that the behaviour of those two concepts is not identical.

Secondly, we addressed streamline distributions. Streamline distributions in uncertain vector fields aim to be the equivalent of streamlines in certain vector fields. Therefore, we track the particle distribution over time, and analyse the distribution at different integration times. The method was applied to a set of different analytical uncertain vector fields, and the structure of streamline distributions in those fields was characterized. Additionally, the influence of distribution grid resolution was analysed by comparing different grid resolutions. It was shown that higher resolutions approximate the streamline distribution of smaller areas in the domain. If the grid resolution is chosen infinitesimally small, we assume that the method yields streamline distributions of a single starting point.

Eventually, we applied the algorithm to a test data set, observing similar behaviour as in the analytical cases. By employing a space-time representation, we visualize the streamline distributions of two analytical fields, and the test data set, and provide an at-a-glance visualization of 2D streamline distributions. To reduce numerical diffusion, which is inherent in methods like the proposed one, future work could include improved interpolation techniques, e.g., by integrating WENO reconstruction as proposed by Karch et al. [15]. Furthermore, we restricted ourselves to uncertain 2D vector fields. In the future, this work could be expanded to three-dimensional vector fields. In this case distribution fields can no longer be integrated into a 3D space-time cube. This problem yields new challenges to visualize streamline distributions. Finally, we constructed our uncertainty by means of a normal distribution. Different results can be anticipated by applying different distributions, e.g., bimodal distributions would probably largely alter the structure of streamline distributions.

List of Figures

Bibliography

1. U. Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., USA, 2015. ISBN: 1930934300, 9781930934306.
2. R. P. Botchen, D. Weiskopf, and T. Ertl. "Texture-based visualization of uncertainty in flow fields". In: *Visualization, 2005. VIS 05. IEEE*. IEEE. 2005, pp. 647–654.
3. K. Brodlie, R. A. Osorio, and A. Lopes. "A review of uncertainty in data visualization". In: *Expanding the frontiers of visual analytics and visualization*. Springer, 2012, pp. 81–109.
4. J. C. Butcher. "The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods", 1987.
5. C. Chen. "Top 10 unsolved information visualization problems". *IEEE computer graphics and applications* 25:4, 2005, pp. 12–16.
6. S. Derakhshan and C. Deutsch. *Numerical Integration of Bivariate Gaussian Distributions*. 2011.
7. A. Dienstfrey and R. Boisvert. *Uncertainty Quantification in Scientific Computing: 10th IFIP WG 2.5 Working Conference, WoCoUQ 2011, Boulder, CO, USA, August 1-4, 2011, Revised Selected Papers*. Vol. 377. Springer, 2012.
8. S. Djurcilov, K. Kim, P. Lermusiaux, and A. Pang. "Visualizing scalar volumetric data with uncertainty". *Computers & Graphics* 26:2, 2002, pp. 239–248.
9. Z. Drezner and G. O. Wesolowsky. "On the computation of the bivariate normal integral". *Journal of Statistical Computation and Simulation* 35:1-2, 1990, pp. 101–107.
10. H.-H. Ehricke, U. Klose, and W. Grodd. "Visualizing MR diffusion tensor fields by dynamic fiber tracking and uncertainty mapping". *Computers & Graphics* 30:2, 2006, pp. 255–264.

11. F. Ferstl, K. Bürger, and R. Westermann. "Streamline variability plots for characterizing the uncertainty in vector field ensembles". *IEEE Transactions on Visualization and Computer Graphics* 22:1, 2016, pp. 767–776.
12. C. Johnson. "Top scientific visualization research problems". *IEEE Computer Graphics and Applications* 24:4, 2004, pp. 13–17.
13. C. R. Johnson and A. R. Sanderson. "A next step: Visualizing errors and uncertainty". *IEEE Computer Graphics and Applications* 23:5, 2003, pp. 6–10.
14. D. K. Jones. "Determining and visualizing uncertainty in estimates of fiber orientation from diffusion tensor MRI". *Magnetic Resonance in Medicine* 49:1, 2003, pp. 7–12.
15. G. K. Karch, F. Sadlo, D. Weiskopf, C.-D. Munz, and T. Ertl. "Visualization of Advection-Diffusion in Unsteady Fluid Flow". In: *Computer Graphics Forum*. Vol. 31. 3pt2. Wiley Online Library. 2012, pp. 1105–1114.
16. M. C. Kennedy and A. O'Hagan. "Bayesian calibration of computer models". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63:3, 2001, pp. 425–464.
17. S. K. Lodha, A. Pang, R. E. Sheehan, and C. M. Wittenbrink. "UFLOW: Visualizing uncertainty in fluid flow". In: *Proceedings of the 7th conference on Visualization'96*. IEEE Computer Society Press. 1996, 249–ff.
18. W. E. Lorensen and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics*. Vol. 21. 4. ACM. 1987, pp. 163–169.
19. A. M. MacEachren, A. Robinson, S. Hopper, S. Gardner, R. Murray, M. Gahegan, and E. Hetzler. "Visualizing geospatial information uncertainty: What we know and what we need to know". *Cartography and Geographic Information Science* 32:3, 2005, pp. 139–160.
20. M. Mihai and R. Westermann. "Visualizing the stability of critical points in uncertain scalar fields". *Computers & Graphics* 41, 2014, pp. 13–25.
21. M. Mirzargar, R. T. Whitaker, and R. M. Kirby. "Curve boxplot: Generalization of boxplot for ensembles of curves". *IEEE transactions on visualization and computer graphics* 20:12, 2014, pp. 2654–2663.

22. M. Otto and H. Theisel. "Vortex analysis in uncertain vector fields". In: *Computer Graphics Forum*. Vol. 31. 3pt2. Wiley Online Library. 2012, pp. 1035–1044.

23. M. Otto, T. Germer, and H. Theisel. "Closed stream lines in uncertain vector fields". In: *Proceedings of the 27th Spring Conference on Computer Graphics*. ACM. 2011, pp. 87–94.

24. M. Otto, T. Germer, H.-C. Hege, and H. Theisel. "Uncertain 2D vector field topology". In: *Computer Graphics Forum*. Vol. 29. 2. Wiley Online Library. 2010, pp. 347–356.

25. M. Otto, T. Germer, and H. Theisel. "Uncertain topology of 3d vector fields". In: *Visualization Symposium (PacificVis), 2011 IEEE Pacific*. IEEE. 2011, pp. 67–74.

26. A. Pang et al. "Visualizing uncertainty in geo-spatial data". In: *Proceedings of the Workshop on the Intersections between Geospatial Information and Information Technology*. National Research Council Arlington, VA. 2001, pp. 1–14.

27. A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. "Approaches to uncertainty visualization". *The Visual Computer* 13, 1997, pp. 370–390.

28. C. Petz, K. Pöthkow, and H.-C. Hege. "Probabilistic local features in uncertain vector fields with spatial correlation". In: *Computer Graphics Forum*. Vol. 31. 3pt2. Wiley Online Library. 2012, pp. 1045–1054.

29. T. Pfaffelmoser, M. Reitinger, and R. Westermann. "Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields". In: *Computer Graphics Forum*. Vol. 30. 3. Wiley Online Library. 2011, pp. 951–960.

30. T. Pfaffelmoser, M. Mihai, and R. Westermann. "Visualizing the variability of gradients in uncertain 2d scalar fields". *IEEE transactions on visualization and computer graphics* 19:11, 2013, pp. 1948–1961.

31. K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson. "Ensemble-vis: A framework for the statistical visualization of ensemble data". In: *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*. IEEE. 2009, pp. 233–240.

32. K. Potter, P. Rosen, and C. R. Johnson. "From quantification to visualization: A taxonomy of uncertainty visualization approaches". In: *Uncertainty Quantification in Scientific Computing*. Springer, 2012, pp. 226–249.

33. P. J. Rhodes, R. S. Laramee, R. D. Bergeron, T. M. Sparr et al. "Uncertainty visualization methods in isosurface rendering". In: *Eurographics*. Vol. 2003. 2003, pp. 83–88.

34. C. P. Robert. *Monte carlo methods*. Wiley Online Library, 2004.

35. F. Sadlo. "Computational Visualization of Physics and Topology in Unsteady Flow". PhD Dissertation No. 19284. ETH Zurich, 2010.

36. J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. "Noodles: A tool for visualization of numerical weather model ensemble uncertainty". *IEEE Transactions on Visualization and Computer Graphics* 16:6, 2010, pp. 1421–1430.

37. T. Schultz, H. Theisel, and H.-P. Seidel. "Topological visualization of brain diffusion MRI data". *IEEE Transactions on Visualization and Computer Graphics* 13:6, 2007.

38. D. Shepard. "A two-dimensional interpolation function for irregularly-spaced data". In: *Proceedings of the 1968 23rd ACM national conference*. ACM. 1968, pp. 517–524.

39. H. Shu, S. Spaccapietra, and D Quesada Sedas. "Uncertainty of Geographic Information and its Support in MADS". In: *Proceedings of the 2ncdInternational Symposium on Spatial DataQuality*. LBD-CONF-2003-006. 2003.

40. C. M. Wittenbrink, A. T. Pang, and S. K. Lodha. "Glyphs for visualizing uncertainty in vector fields". *IEEE transactions on Visualization and Computer Graphics* 2:3, 1996, pp. 266–279.