FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 22
30614 Plzeň
Czech Republic

Visualization methods for large n-dimensional and t-variant data

State of the Art and Concept of Ph.D. Thesis

Michal Šmolík

Technical Report No. DCSE/TR-2016-06
July, 2016

Distribution: public

Technical Report No. DCSE/TR-2016-06
July, 2016

Visualization methods for large n-dimensional and t-variant data

Michal Šmolík

Abstract

This work focuses on vector field visualization and approximation or interpolation. We summarize the knowledge about vector fields and its critical points as they are the most important property of all vector fields. Next, we present the interpolation and approximation techniques together with some of our published work about vector fields and radial basis functions. Vector fields can be results of numerical simulations, so we present algorithms for solving PDE with meshless techniques as well. Next, we summarize some of the visualization techniques for vector field visualization.

In the last chapter we discuss the possible directions of future research in the fields of vector field visualization and interpolation or approximation.

Keywords: Vector field, critical point, visualization, interpolation, approximation, Radial basis functions

This work has been supported by MSMT CR projects LG13047, LH12181, SGS 2013-029 and SGS 2016-013.

Acknowledgements

The author would like to thank his supervisor Prof. Ing. Václav Skala, CSc. and the colleagues at the University of West Bohemia, Plzen, for their discussions, suggestions, valuable comments and hints provided. This work has been supported by MSMT CR projects LG13047, LH12181, SGS 2013-029 and SGS 2016-013.

Contents

1 Introduction

The concept of flow plays an important role in many fields of science. Classical application fields are, for example, the automotive and aerospace industry, where the investigation of the air flow around vehicles is an important task. However, the same concepts are used in the simulation and analysis of water flow in turbines of power plants, of blood flow in vessels, the propagation of smoke in buildings, and weather simulations, to mention just a few. The visualization of data gained from the simulation/measurement of such processes is relevant for the domain users as visualization has the potential to ease the understanding of such complex flow phenomena. In this context, topological flow visualization methods have been developed, with the aim to give insight into the overall behaviour of the flow. A characteristic of this class of methods is the segmentation of the flow domain into regions of substantially different flow behaviour, providing a topology of the flow domain.

The theory of dynamical systems goes back to the 19th century work of Henri Poincare. An introduction can be found, for example in Katokand and Hasselblatt [Kat97]. In our context, the case of deterministic, continuous and autonomous dynamical systems is most interesting, because such systems can be used to formulate velocity fields of a steady fluid flow. Many patterns in a flow can be described and analysed by concepts from dynamical systems theory, such as critical points, separatrices and periodic orbits. Perry and Chong [Per87] give a comprehensive overview of such $2D$ and $3D$ flow patterns. Helman and Hesselink introduced these methods to the scientific visualization community, and used them under the notion of vector field topology for the visualization of computed and measured velocity fields, first in $2D$ [Hel89] and later in $3D$ [Hel91]. Vector field topology was further popularized both by Asimov's excellent tutorial [Asi93] and by Globus et al.'s TOPO module [Glo91] for NASA's FAST visualization software.

Mesh methods are standard tools for the simulation of flow problems, they enable efficient and reliable approximations of the differential equations in fluid flow. However, in certain applications, for example in the presence of large geometric deformations of the boundary or rotating and moving obstacles, i.e. situations which may frequently occur in the context of fluid with structure interaction problems, the maintenance of a conforming mesh may be almost impossible. Different techniques have been developed to deal with these problems in the mesh-based context; the fictitious domain and fictitious boundary methods, techniques employing overlapping grids, sliding mesh, level-set methods, or standard arbitrary Lagrangian-Eulerian formulations with frequent remeshing.

A different way to handle complex flow problems is to employ a comparably new and innovative class of methods, which enables the approximation of partial differential equations based on a set of nodes, without the need for an additional mesh. In recent years meshless/meshfree methods have gained considerable attention in engineering and applied mathematics. The variety of problems that are now being addressed by these techniques continues to expand and the quality of the obtained results demonstrates the effectiveness of many of the methods currently available. These meshfree methods are generally able to solve problems where meshes bring up difficulties. However, they are comparably time-consuming, which limits their usefulness in the simulation of challenging real-life problems.

2 Vector fields

The term flow denotes an abstract concept adopted in many application fields. Fluid dynamics, for instance, is concerned with the study of fluid flows, i.e. the motion of fluids: typical examples include the motion of water in a pump or a turbine, the stream of air around a car or an airplane, blood in a vessel, oil or gas in a pipe, and many other. Flow visualization usually deals with data generated via measurements, simulations or modelling, and the results are commonly expressed as vector fields.

Vector fields on surfaces are important objects, which appear frequently in scientific simulation in CFD (Computational Fluid Dynamics) or modelling by FEM (Finite Element Method). To be visualized, such vector fields are usually linearly interpolated for the sake of simplicity and performance considerations. Namely, the vector field is sampled at each vertex of the underlying piecewise linear surface and interpolated linearly, similarly to what is done for the geometry.

Vector fields can be described for general vectors in E^n, but for our purposes it is sufficient to consider E^2 or E^3. A vector field in E^3 is a map that for each point $\boldsymbol{x} = [x, y, z]^T$ in a domain assigns a vector

$$\boldsymbol{v}(\boldsymbol{x}(t), t) = [v_1(\boldsymbol{x}(t), t), v_2(\boldsymbol{x}(t), t), v_3(\boldsymbol{x}(t), t)]^T \,, v_i: E^3 \rightarrow E^1 \,, \tag{2.1}$$

where each of the function v_i is a scalar field and is dependent on the position and time, i.e. $\boldsymbol{v}$ time-dependent vector field, or does not depend on time, i.e. $\boldsymbol{v}$ is time-independent vector field.

A field line is a line that is everywhere tangent to a given vector field at a particular time. That is, every point on the line has a tangent that coincides with the vector at that location in the vector field. It can be constructed by tracing a path in the direction of the vector field while keeping time fixed. Field lines will depict the direction of the vector field, but not the magnitude.

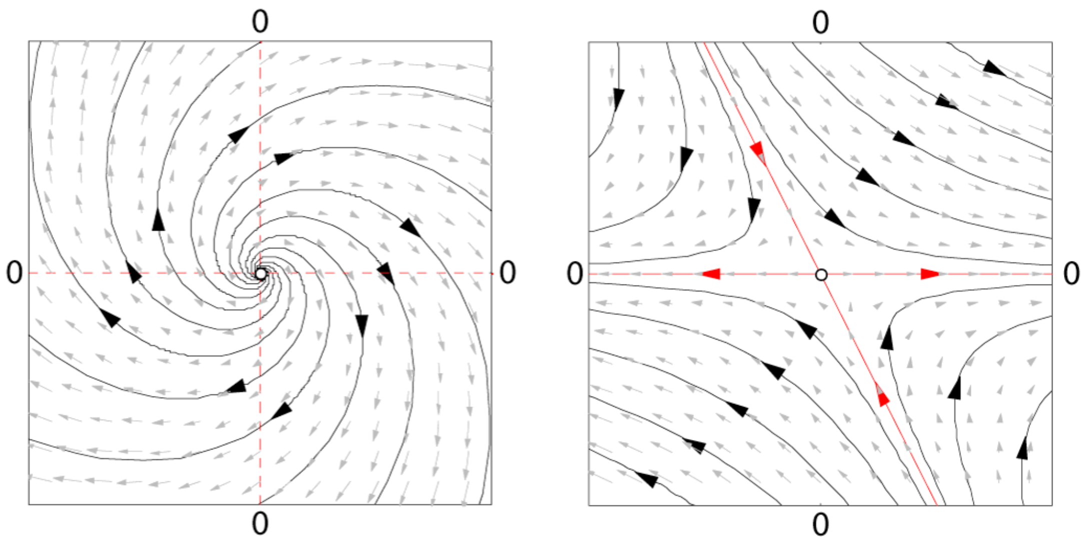


Figure 2.1: Example of linear vector fields with oriented field lines and vector field arrows located in a regular grid.

Let $\boldsymbol{x}(t)$ be a field line with parameter t representing the time. The field line is then given by a system of ordinary differential equations, which can be written as

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{v}\big(\boldsymbol{x}(t)\big)\,, \boldsymbol{x}(0) = \boldsymbol{x}_0\,, \tag{2.2}$$

where $\boldsymbol{x}_0$ is an initial point. When numerical integration is used to solve the system of equations for visualization purposes, the point $\boldsymbol{x}_0$ is often referred to as a seed point. In terms of differential equations, the visualization of the field lines of a vector field correspond to phase portraits of solutions of the system (2.2). This will be an important concept in the discussion of classification of critical points later.

2.1 Vector field integration

The integration through a vector field is a computational challenge, and since it is crucial for a lot of methods, it should be explained at the beginning. The explanation will be carried out on a flow data – an important, intuitive and probably the most frequent case of a vector field. Integrating flow brings the respective path $\boldsymbol{x}(s)$ of an imaginary particle traveling through the field. This path would be analytically defined by

$$\boldsymbol{x}(s) = \boldsymbol{x}_0 + \int_{t=0}^{s} \boldsymbol{v}(\boldsymbol{x}(t), t + t_0)dt\,, \tag{2.3}$$

where $\boldsymbol{x}_0$ represents the seed location and t_0 is the starting time when the particle was seeded. This particle trace (2.3) through a vector field is called a pathline.

Usually we cannot calculate the equation (2.3) analytically but we need to perform a numerical integration. The simplest approach is to use a first-order Euler method to compute an approximation. One iteration of the curve integration is calculated as

$$\boldsymbol{x}(t+\Delta t)=\boldsymbol{x}(t)+\Delta t\cdot\boldsymbol{v}(\boldsymbol{x}(t),t)\,, \tag{2.4}$$

where Δt is a small time step. A more accurate but also more costly technique is the second order Runge-Kutta method, which uses the Euler approximation (2.4) as a hint to compute a better approximation of the integral curve described in (2.3) as

$$\boldsymbol{x}(t+\Delta t)=\boldsymbol{x}(t)+\Delta t\frac{\boldsymbol{v}(\boldsymbol{x}(t),t)+\boldsymbol{v}(\boldsymbol{x}(t)+\Delta t\cdot\boldsymbol{v}(\boldsymbol{x}(t),t),t)}{2}\,. \tag{2.5}$$

Obviously, the choice of the time step Δt is an issue. Too large step will lead to loss of accuracy, and small step will increase the time needed for calculation of the vector field integration.

2.2 Vector field interpolation

Let's have a vector field given on a uniform discrete grid. A uniform discrete grid is a grid where the grid points are evenly spaced in all dimensions. A vector is associated with every grid point.

Many of the analysis methods and visualization techniques use vectors at positions not directly on the grid points of the discrete representation of a vector field. Therefore, it is necessary to use an interpolation scheme to approximate the vectors at intermediate points between grid cells. A suitable interpolation scheme is the bilinear (in E^2) or trilinear (in E^3) interpolation method. For this interpolation method to give a good approximation, it is assumed that the resolution of the grid is sufficient, so that the vector field is linear, or nearly linear, within every grid cell.

2.2.1 2D bilinear interpolation

The bilinear interpolation is an extension of the linear interpolation for interpolating functions of two variables (e.g., x and y) on a rectilinear $2D$ grid.

The key idea is to perform linear interpolation first in one direction, and then again in the other direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location.

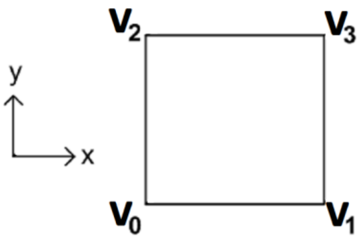


Figure 2.2: Indexing of vector field data in a $2D$ cell.

The known values are in grid points and to calculate the value inside a cell (see Figure 2.2 for indexing), we have to calculate the following formula

$$\boldsymbol{v}(x,y) = \boldsymbol{a} + \boldsymbol{b}x + \boldsymbol{c}y + \boldsymbol{d}xy\,, \tag{2.6}$$

where $\boldsymbol{a} = [a_x, a_y]$, $\boldsymbol{b} = [b_x, b_y]$, $\boldsymbol{c} = [c_x, c_y]$ and $\boldsymbol{d} = [d_x, d_y]$ are interpolation coefficients. To perform an interpolation, the values of interpolation coefficients must be determined so we have to solve a system of linear equations. The system is simplified by introducing a local parametric coordinate system for the cell. In this local coordinate system each component lies in the closed interval [0; 1], i.e. $\boldsymbol{v}_0 = [0, 0]^T$ and $\boldsymbol{v}_3 = [1, 1]^T$. This will remove many terms from the system because of the zero factor that is introduced, and we can determine the interpolation coefficients by solving a simpler system of equations

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \end{bmatrix}. \tag{2.7}$$

Solving this system of linear equations we get the interpolation coefficients

$$\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_0 \\ -\boldsymbol{v}_0 + \boldsymbol{v}_1 \\ -\boldsymbol{v}_0 + \boldsymbol{v}_2 \\ \boldsymbol{v}_0 - \boldsymbol{v}_1 - \boldsymbol{v}_2 + \boldsymbol{v}_3 \end{bmatrix}. \tag{2.8}$$

Interpolated value $\boldsymbol{v}(x,y)$ of a point inside the cell is calculated first by computing coefficients from (2.8) and then by inserting them into (2.6).

2.2.2 3D trilinear interpolation

Trilinear interpolation is a method of multivariate interpolation on a 3-dimensional regular grid. It approximates the value of an intermediate point $[x, y, z]^T$.

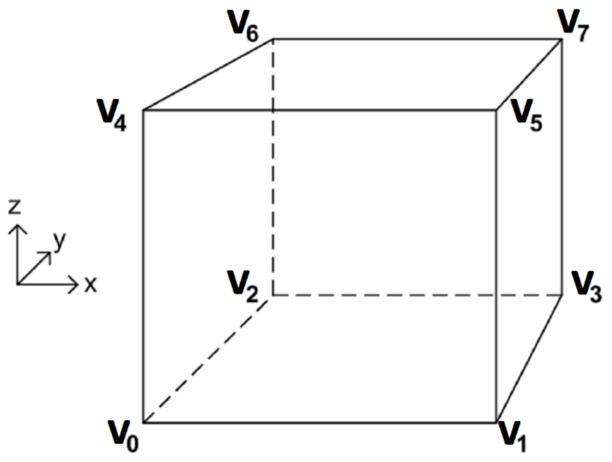


Figure 2.3: Indexing of vector field data in a $3D$ cell.

The known values are in grid points and to calculate the value inside a cell (see Figure 2.3 for indexing), we have to calculate the following formula

$$\boldsymbol{v}(x,y,z) = \boldsymbol{a} + \boldsymbol{b}x + \boldsymbol{c}y + \boldsymbol{d}z + \boldsymbol{e}xy + \boldsymbol{f}xz + \boldsymbol{g}yz + \boldsymbol{h}xyz\,, \tag{2.9}$$

where $\boldsymbol{a}$, ..., $\boldsymbol{h}$ are interpolation coefficients. To perform an interpolation, the values of interpolation coefficients must be determined in the same way as when doing the $2D$ interpolation. The system of linear equations that has to be solved is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{c} \\ \boldsymbol{d} \\ \boldsymbol{e} \\ \boldsymbol{f} \\ \boldsymbol{g} \\ \boldsymbol{h} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \\ \boldsymbol{v}_4 \\ \boldsymbol{v}_5 \\ \boldsymbol{v}_6 \\ \boldsymbol{v}_7 \end{bmatrix}. \tag{2.10}$$

Solving this system of linear equations we get the interpolation coefficients

$$\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{c} \\ \boldsymbol{d} \\ \boldsymbol{e} \\ \boldsymbol{f} \\ \boldsymbol{g} \\ \boldsymbol{h} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_0 \\ -\boldsymbol{v}_0 + \boldsymbol{v}_1 \\ -\boldsymbol{v}_0 + \boldsymbol{v}_2 \\ -\boldsymbol{v}_0 + \boldsymbol{v}_4 \\ \boldsymbol{v}_0 - \boldsymbol{v}_1 - \boldsymbol{v}_2 + \boldsymbol{v}_3 \\ \boldsymbol{v}_0 - \boldsymbol{v}_1 - \boldsymbol{v}_4 + \boldsymbol{v}_5 \\ \boldsymbol{v}_0 - \boldsymbol{v}_2 - \boldsymbol{v}_4 + \boldsymbol{v}_6 \\ -\boldsymbol{v}_0 + \boldsymbol{v}_1 + \boldsymbol{v}_2 - \boldsymbol{v}_3 + \boldsymbol{v}_4 - \boldsymbol{v}_5 - \boldsymbol{v}_6 + \boldsymbol{v}_7 \end{bmatrix}. \tag{2.11}$$

Interpolated value $\boldsymbol{v}(x,y,z)$ of a point inside the cell is calculated first by computing coefficients from (2.11) and then by inserting them into (2.9).

3 Critical points

Critical points ($\boldsymbol{x}_0$) of the vector field are points at which the magnitude of the vector vanishes

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{v}(\boldsymbol{x}(t)) = \boldsymbol{0}\,, \tag{3.1}$$

i.e. all components are equal to zero

$$\begin{bmatrix} \frac{dx(t)}{dt} \\ \frac{dy(t)}{dt} \\ \frac{dz(t)}{dt} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.2}$$

A critical point is said to be isolated, or simple, if the vector field is non-vanishing in an open neighbourhood around the critical point. Thus for all surrounding points $\boldsymbol{x}_\varepsilon$ of the critical point $\boldsymbol{x}_0$ the equation (3.1) does not apply, i.e.

$$\frac{d\boldsymbol{x}_\varepsilon(t)}{dt} \neq \boldsymbol{0}\,. \tag{3.3}$$

At critical points, the direction of the field line is indeterminate, and they are the only points in the vector field were field lines can intersect (asymptotically). The terms singular point, null point, neutral point or equilibrium point are also frequently used to describe critical points.

Critical points in vector fields can be classified according to their order. The order of the critical point is identical to the topological degree of the critical point. For a closed surface $\tau(\boldsymbol{x}_0)$ surrounding a single critical point $\boldsymbol{x}_0$ in a continuous $3D$ vector field, the topological degree of $\boldsymbol{x}_0$ is defined as the following surface integral [Wan06]

$$I(\boldsymbol{x}_0) = \frac{1}{4\pi}\int_{\tau(\boldsymbol{x}_0)} d\theta \tag{3.4}$$

and for $2D$ vector field as the following curve integral

$$I(\boldsymbol{x}_0) = \frac{1}{2\pi}\int_{\tau(\boldsymbol{x}_0)} d\theta\,, \tag{3.5}$$

where θ is the angle between the vector field and a constant reference vector. The value of $I(\boldsymbol{x}_0)$ is always an integer, and the order of the critical point is the absolute value of this integer (see Figure 3.1 and Figure 3.2 for few examples). A surface, resp. curve, τ enclosing a region which does not contain a critical point has a topological degree of zero.

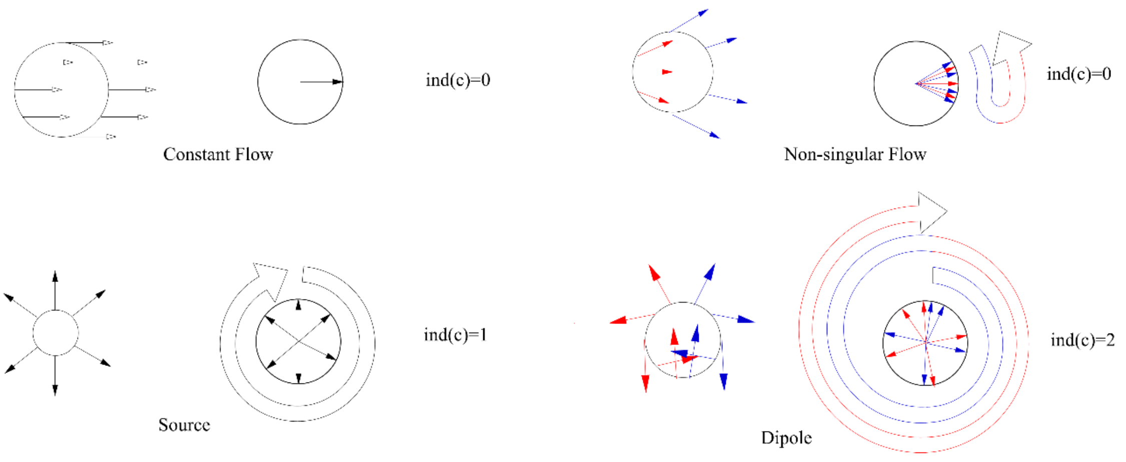


Figure 3.1: Vector fields on circles around critical points, their Gauss maps and the order of critical points ($ind(\boldsymbol{c})$) from [Man02].

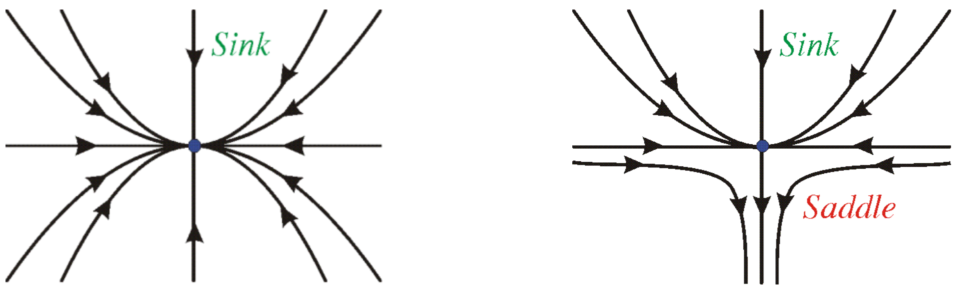


Figure 3.2: Example of first order critical point (left) and a second order critical point (right).

Given a curve in a vector field, the Poincaré-index is defined as the total rotation of the vectors of the vector field along the curve. For a closed curve in a continuous $2D$ vector field the number of rotations will always be an integer since the start and the end vectors will be identical. See Figure 3.3 for an example in $2D$ for an open curve. The angle between two vectors is calculated using the formula

$$\theta = \cos^{-1}(\boldsymbol{u} \cdot \boldsymbol{v}) \,, \tag{3.6}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are normalized vectors.

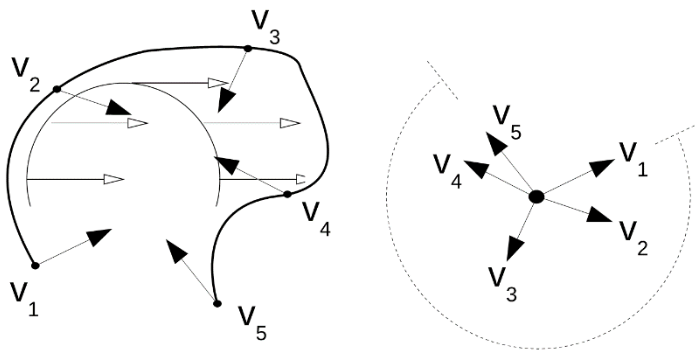


Figure 3.3: Calculating the order of $2D$ critical point ($ind(\boldsymbol{c})$).

The topological degree of a discrete $3D$ vector field can be calculated as following. We need to create a triangulation that creates a closed surface. The second step is to sum all the solid angles formed with three vectors for each triangle (see Figure 3.4) to calculate the integral $\int_{\tau(\boldsymbol{x}_0)} d\theta$ discretely. The solid angle θ can be calculated using the following formula

$$\tan\left(\frac{1}{2}\theta\right) = \frac{|\boldsymbol{B}_1\,\boldsymbol{B}_2\,\boldsymbol{B}_3|}{\|\boldsymbol{B}_1\|\|\boldsymbol{B}_2\|\|\boldsymbol{B}_3\| + (\boldsymbol{B}_1\cdot\boldsymbol{B}_2)\|\boldsymbol{B}_3\| + (\boldsymbol{B}_1\cdot\boldsymbol{B}_3)\|\boldsymbol{B}_2\| + (\boldsymbol{B}_2\cdot\boldsymbol{B}_3)\|\boldsymbol{B}_1\|}, \tag{3.7}$$

where $|\boldsymbol{B}_1\,\boldsymbol{B}_2\,\boldsymbol{B}_3|$ denotes the determinant of the matrix that results when writing the vectors together in a row; this is also equivalent to the scalar triple product of the three vectors, i.e.

$$|\boldsymbol{B}_1\,\boldsymbol{B}_2\,\boldsymbol{B}_3| = (\boldsymbol{B}_1\times\boldsymbol{B}_2)\cdot\boldsymbol{B}_3\,. \tag{3.8}$$

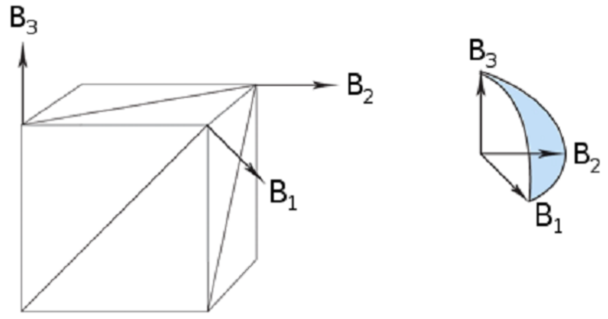


Figure 3.4: Illustration of the solid angle for three points forming a tringle.

3.1 Location of critical points

Location of critical points is important step in flow field visualization and feature extraction. There exist several approaches for extracting isolated zeros of scalar, vector and tensor fields [Ben66], [Gjø04], [Gre92], [Man02], [Pre92]. Combinatorial methods become particularly attractive, as they are not sensitive to numerical instabilities or the details of a particular implementation. [Bha14] introduced a robust method for detecting singularities in vector fields. They establish, in combinatorial terms, necessary and sufficient conditions for the existence of a critical point in a cell of a simplicial mesh for a large class of interpolation functions. These conditions are entirely local and lead to a provably consistent and practical algorithm to identify cells containing singularities. Other selected algorithms for critical points location are presented in the following subchapters.

3.1.1 Newton-Raphson method

In this section we will discuss the simplest multidimensional root finding method, Newton-Raphson [Ben66], [Pre92]. This method gives you a very efficient means of converging to a root, if we have a sufficiently good initial guess. It can also spectacularly fail to converge, indicating (though not proving) that our putative root does not exist nearby.

Let $\boldsymbol{p}_0$ be a good estimate of critical point $\boldsymbol{x}_0$ and let $\boldsymbol{x}_0 = \boldsymbol{p}_0 + \boldsymbol{h}$. Since the true critical point is $\boldsymbol{x}_0$, and $\boldsymbol{h} = \boldsymbol{x}_0 - \boldsymbol{p}_0$, the vector $\boldsymbol{h}$ measures how far the estimate $\boldsymbol{p}_0$ is from the truth critical point.

In the neighbourhood of $\boldsymbol{p}_0$ each of the functions $\boldsymbol{v} = \left[v_x, v_y\right]^T$ can be expanded in Taylor series

$$\boldsymbol{v}(\boldsymbol{p}) = \boldsymbol{v}(\boldsymbol{p}_0) + \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}(\boldsymbol{p} - \boldsymbol{p}_0)\,, \tag{3.9}$$

where

$$\boldsymbol{J} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}, \tag{3.10}$$

and $\boldsymbol{J}$ is the Jacobian matrix. By setting $\boldsymbol{v}(\boldsymbol{p}) = \boldsymbol{0}$, we obtain a set of linear equations for the corrections $\boldsymbol{h} = (\boldsymbol{p} - \boldsymbol{p}_0)$ that move each function closer to zero simultaneously, namely

$$\boldsymbol{J} \cdot \boldsymbol{h} = -\boldsymbol{v}(\boldsymbol{p}_0)\,. \tag{3.11}$$

Matrix equation (3.11) can be solved by the well-known LU decomposition. The corrections are then added to the solution vector

$$\boldsymbol{p}_{new} = \boldsymbol{p}_{old} + \boldsymbol{h}\,. \tag{3.12}$$

and the process is iterated to convergence. In general it is a good to check the degree to which both functions and variables have converged.

3.1.2 Analytic method

[Gjø04] described a new analytical method for locating critical points. The $3D$ vector field can be interpolated using trilinear interpolation with the following definition

$$\begin{aligned} v_x(x,y,z) &= a_1 + b_1x + c_1y + d_1z + e_1xy + f_1xz + g_1yz + h_1xyz \\ v_y(x,y,z) &= a_2 + b_2x + c_2y + d_2z + e_2xy + f_2xz + g_2yz + h_2xyz \\ v_z(x,y,z) &= a_3 + b_3x + c_3y + d_3z + e_3xy + f_3xz + g_3yz + h_3xyz\,. \end{aligned} \tag{3.13}$$

When finding a critical point, the three equations must fulfil the following equations

$$\begin{aligned} v_x(x,y,z) &= 0 \\ v_y(x,y,z) &= 0 \\ v_z(x,y,z) &= 0\,. \end{aligned} \tag{3.14}$$

This system of equations can be solved using an analytic method. The strategy is as follows: First the interpolation coefficients $(a_1, a_2, a_3, b_1, \dots, h_3)$ must be computed. Variable z is then eliminated from the first equation in (3.13).

$$z = \frac{a_1 + b_1x + c_1y + e_1xy}{d_1 + f_1x + g_1y + h_1xy}\,. \tag{3.15}$$

The expression (3.15) for z is inserted into the two other equations in (3.13) giving two functions $\xi(x,y)$ and $\phi(x,y)$.

$$\begin{aligned} \xi(x,y) &= (A_1x^2 + B_1x + C_1)y^2 + (A_2x^2 + B_2x + C_2)y + (A_3x^2 + B_3x + C_3) \\ \phi(x,y) &= (A_4x^2 + B_4x + C_4)y^2 + (A_5x^2 + B_5x + C_5)y + (A_6x^2 + B_6x + C_6)\,, \end{aligned} \tag{3.16}$$

where

$$\begin{aligned} A_1 &= e_{2h_1} - h_2e_1 \\ B_1 &= e_2g_1 - g_2e_1 + c_2h_1 - h_2c_1 \\ C_1 &= c_2g_1 - g_2c_1 \\ A_2 &= e_2f_1 - f_2e_1 + b_2h_1 - h_2b_1 \\ B_2 &= e_2d_1 - d_2e_1 + c_2f_1 - f_2c_1 + b_2g_1 - g_2b_1 + a_2h_1 - h_2a_1 \\ C_2 &= c_2d_1 - d_2c_1 + a_2g_1 - g_2a_1 \\ A_3 &= b_2f_1 - f_2b_1 \\ B_3 &= b_2d_1 - d_2b_1 + a_2f_1 - f_2a_1 \\ C_3 &= a_2d_1 - d_2a_1\,, \end{aligned} \tag{3.17}$$

and

$$\begin{aligned}
A_4 &= e_{3h_1} - h_3 e_1 \\
B_4 &= e_3 g_1 - g_3 e_1 + c_3 h_1 - h_3 c_1 \\
C_4 &= c_3 g_1 - g_3 c_1 \\
A_5 &= e_3 f_1 - f_3 e_1 + b_3 h_1 - h_3 b_1 \\
B_5 &= e_3 d_1 - d_3 e_1 + c_3 f_1 - f_3 c_1 + b_3 g_1 - g_3 b_1 + a_3 h_1 - h_3 a_1 \\
C_5 &= c_3 d_1 - d_3 c_1 + a_3 g_1 - g_3 a_1 \\
A_6 &= b_3 f_1 - f_3 b_1 \\
B_6 &= b_3 d_1 - d_3 b_1 + a_3 f_1 - f_3 a_1 \\
C_6 &= a_3 d_1 - d_3 a_1 \,.
\end{aligned} \tag{3.18}$$

These two equations are solved so that

$$\begin{aligned}
\xi(x,y) &= 0 \\
\phi(x,y) &= 0 \,.
\end{aligned} \tag{3.19}$$

For a given x there must be a common value y that solves both equations (3.19). To determine this common factor, if any, one possibility is to use the determinant of the Sylvester matrix to find a polynomial which can be solved for x, see [Gar92].

When the roots of the polynomial are determined using the numerical method, the next step is to determine which roots are the candidates for the x component of the critical points. The candidates are the values that are real and inside the closed unit interval. The candidate values of x are inserted into $\xi(x,y)$ and $\phi(x,y)$ to see if there is at least one common root for y. For each pair (x,y) that solves both equations, the values are inserted in (3.15) to find the corresponding z value.

Not all values (x, y, z) that are computed correspond to true critical points in the cell. Every candidate point is therefore forwarded to a validation step to test if it is a true critical point.

3.1.3 Octree and topological degree method

According to [Gre92], [Man02] the space is subdivided into several cubes in a regular grid. If the index of a cube is nonzero, then a critical point was found, or perhaps a collection of critical points inside the cube. Then the cube has to be subdivided and thus create an octree structure. The subdivision step takes place until the cube has nonzero index and the size of cube is smaller than some defined smallest resolution. At this point a center of the cube is said to be the critical point.

Some of the critical points can be miss as the topological degree of critical points is additive. When two critical points with topological degree $+1$ and -1 are in the same cube, then the resulting topological degree of the cube is 0.

3.2 Linearization of vector field

Critical points can be characterized according to the behaviour of nearby tangent curves. We can use a particular set of these curves to define a skeleton that characterizes the global behaviour of all other tangent curves in the vector field. An important feature of differential equations is that it is often possible to determine the local stability of a critical point by approximating the system by a linear system. These approximations are aimed at studying the local behaviour of a system, where the nonlinear effects are expected to be small. In this section we discuss how to locally approximate a system by its linearization. The Taylor series expansion must be utilized locally to find the relation between $\boldsymbol{v}$ and position $\boldsymbol{x}$, supposing the flow $\boldsymbol{v}$ to be sufficiently smooth and differentiable. In such case, the expansion of $\boldsymbol{v}$ around the critical points $\boldsymbol{x}_0$ is

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{v}(\boldsymbol{x}_0) + \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}_0)\,. \tag{3.20}$$

As $\boldsymbol{v}(\boldsymbol{x}_0)$ is according to (3.1) equal zero for critical points, we can rewrite equation (3.20) using matrix notation

$$\begin{gathered} \boldsymbol{v} = \boldsymbol{J} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) \\ \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{bmatrix} \cdot \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}. \end{gathered} \tag{3.21}$$

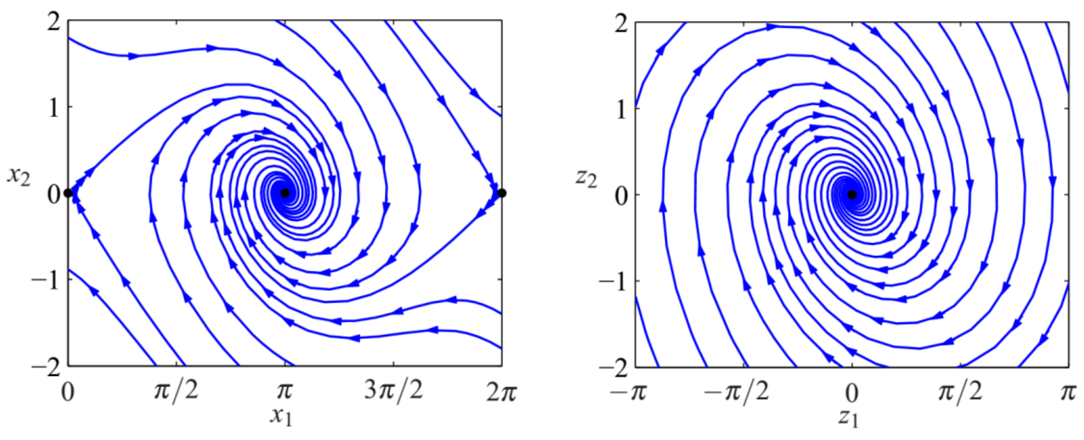
If we consider an inverted pendulum whose open loop dynamics are given by

$$\frac{d\boldsymbol{x}(t)}{dt} = \begin{bmatrix} y \\ \sin x - \gamma y \end{bmatrix}, \tag{3.22}$$

where a coefficient of viscous friction is γ, $x = \theta$ is the angle and $y = \dot{\theta}$ is the angular rate. One of the critical points is the point $\boldsymbol{x}_0 = [\pi, 0]^T$ and the linearization of the dynamical system at this point is according to (3.21) equal to

$$\begin{aligned}
\begin{bmatrix} v_x \\ v_y \end{bmatrix} &= \left[\begin{matrix} \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} \\ \frac{\partial(\sin x - \gamma y)}{\partial x} & \frac{\partial(\sin x - \gamma y)}{\partial y} \end{matrix}\right]\Bigg|_{x_0=[\pi,0]^T} \cdot \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 1 \\ \cos x & -\gamma \end{bmatrix}\Bigg|_{x_0=[\pi,0]^T} \cdot \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 1 \\ -1 & -\gamma \end{bmatrix} \cdot \begin{bmatrix} x - \pi \\ y \end{bmatrix}.
\end{aligned} \tag{3.23}$$

We call the system (3.23) the linear approximation of the original nonlinear system or the linearization at the critical point $\boldsymbol{x}_0$. The phase portrait of this linear approximation and original nonlinear phase portrait can be seen in (Figure 3.5), where $\boldsymbol{x} - \boldsymbol{x}_0 = \boldsymbol{z}$.



a) Nonlinear approximation b) Linear approximation

Figure 3.5: Comparison between the phase portraits for the full nonlinear system (a) and its linear approximation around the critical point at $\boldsymbol{x_0} = [\pi, 0]^T$ (b). Notice that near the critical point at the center of the plots, the phase portraits are almost identical, i.e. the dynamics in both phase portraits are almost identical [Ast10].

3.3 Vector field approximation using second derivative

Vector fields are approximated using only linear approximation to determine the local behaviour of the vector field. However, linearization gives as basic classification of the critical points and about the flow around them, the approximation using second order derivatives will give us some more information as we propose in [Smo16c].

The approximation of vector field around a critical point using the second order derivative must be written for each vector component (v_x and v_y) separately, see the following equations

$$v_x = \begin{bmatrix} \dfrac{\partial v_x}{\partial x} \\ \dfrac{\partial v_x}{\partial y} \end{bmatrix}^T \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^T \cdot \begin{bmatrix} \dfrac{\partial^2 v_x}{\partial x^2} & \dfrac{\partial^2 v_x}{\partial x \partial y} \\ \dfrac{\partial^2 v_x}{\partial y \partial x} & \dfrac{\partial^2 v_x}{\partial y^2} \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{3.24}$$

$$v_y = \begin{bmatrix} \dfrac{\partial v_y}{\partial x} \\ \dfrac{\partial v_y}{\partial y} \end{bmatrix}^T \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^T \cdot \begin{bmatrix} \dfrac{\partial^2 v_y}{\partial x^2} & \dfrac{\partial^2 v_y}{\partial x \partial y} \\ \dfrac{\partial^2 v_y}{\partial y \partial x} & \dfrac{\partial^2 v_y}{\partial y^2} \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \tag{3.25}$$

where $\Delta x = x - x_0$ and $\Delta y = y - y_0$. These two equations can be written in matrix notation as well

$$v_x = \boldsymbol{J}_x \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}_0)^T \cdot \boldsymbol{H}_x \cdot (\boldsymbol{x} - \boldsymbol{x}_0) \tag{3.26}$$

$$v_y = \boldsymbol{J}_y \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}_0)^T \cdot \boldsymbol{H}_y \cdot (\boldsymbol{x} - \boldsymbol{x}_0), \tag{3.27}$$

where $\boldsymbol{H}_x$ and $\boldsymbol{H}_y$ are Hessian matrices, $\boldsymbol{J}_x$ is the first row of Jacobian matrix and $\boldsymbol{J}_y$ is the second row of Jacobian matrix.

The Hessian matrix is a square matrix of second-order partial derivatives of a scalar-valued function, or scalar field. It describes the local curvature of a function of many variables.

Approximation of vector field using (3.24) and (3.25) gives us more detailed description than approximation of vector field using linear approximation, see Figure 3.6. The approximation in Figure 3.6 (right) gives us the same information like in Figure 3.6 (left), although we can see the curvature of the two main axis for the saddle.

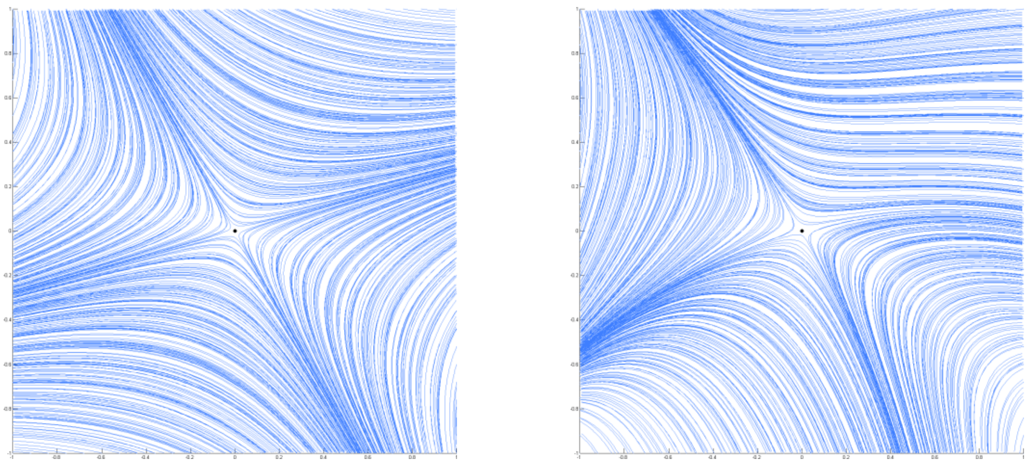


Figure 3.6: Comparison between the phase portraits for the vector field approximated using linear approximation (left) and using second order derivative (right).

Equations (3.24) and (3.25) can be rewritten in different formulas as follows

$$v_x = \frac{1}{2}[\Delta x \quad \Delta y \quad 1] \cdot \begin{bmatrix} \frac{\partial^2 v_x}{\partial x^2} & \frac{\partial^2 v_x}{\partial x \partial y} & \frac{\partial v_x}{\partial x} \\ \frac{\partial^2 v_x}{\partial y \partial x} & \frac{\partial^2 v_x}{\partial y^2} & \frac{\partial v_x}{\partial y} \\ \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ 1 \end{bmatrix} \tag{3.28}$$

$$v_y = \frac{1}{2}[\Delta x \quad \Delta y \quad 1] \cdot \begin{bmatrix} \frac{\partial^2 v_y}{\partial x^2} & \frac{\partial^2 v_y}{\partial x \partial y} & \frac{\partial v_y}{\partial x} \\ \frac{\partial^2 v_y}{\partial y \partial x} & \frac{\partial^2 v_y}{\partial y^2} & \frac{\partial v_y}{\partial y} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ 1 \end{bmatrix}. \tag{3.29}$$

These two equations have some geometrical background. When v_x and v_y are equal zero, each equation describes some conic section.

Approximation of the vector field using Hessian matrix, i.e. using second order derivatives, is a bit more computationally expensive than the standard linear approximation but gives us more detailed description of the vector field as will be seen in the following chapters.

3.3.1 Conic Section.

A conic is the curve obtained as the intersection of a plane, called the cutting plane, with a double cone, see Figure 3.7. Planes that pass through the vertex of the cone will intersect the cone in a point, a line or a pair of intersecting lines. These are called degenerate conics and some authors do not consider them to be conics at all.

There are three types of non-degenerated conics, the ellipse, parabola, and hyperbola, Figure 3.7. The circle is a special kind of ellipse. The circle and the ellipse arise when the intersection of the cone and plane is a closed curve. The circle is obtained when the cutting plane is parallel to the plane of the generating circle of the cone, this means that the cutting plane is perpendicular to the symmetry axis of the cone. If the cutting plane is parallel to exactly one generating line of the cone, then the conic is unbounded and is called a parabola. In the remaining case, the figure is a hyperbola. In this case, the plane will intersect both halves of the cone, producing two separate unbounded curves.

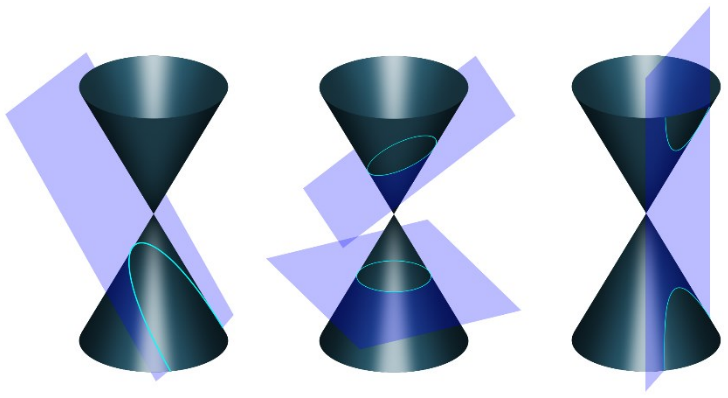


Figure 3.7: Types of conic sections, i.e. parabola, circle and ellipse, and hyperbola.[1]

A conic section is described by the following implicit equation

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0\,. \tag{3.30}$$

where $a_{ij}\ i,j \in \{1,2,3\}$ are coefficients of conic section. Depending on these values, we can classify the types of conic sections. To do that, we need to compute two determinants

$$\Omega = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \tag{3.31}$$

$$\omega = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}\,. \tag{3.32}$$

When knowing determinants Ω and ω we can easily classify the type of conic section using the following table

[1] image source: https://en.wikipedia.org/wiki/Conic_section

Table 3.1: Classification of conic section (from [Bar71]).

	$\omega \neq 0$		$\omega = 0$
	$\omega > 0$	$\omega < 0$	
$\Omega \neq 0$	ellipse	hyperbola	parabola
$\Omega = 0$	pair of intersecting lines		pair of parallel lines

Equations (3.28) and (3.29) are the same as (3.30) when $v_x = 0$ and $v_y = 0$ and therefore they geometrically represent conic sections.

3.4 Classification of critical points

There exist a finite set of fundamentally different critical points, defined by the number of inflow and outflow directions, spiralling structures etc., and combinations of these. Since the set is finite, each critical point can be classified. Such a classification defines the field completely in a close neighbourhood around the critical point. By knowing the location and classification of critical points in a vector field, the topology of the field is known in small areas around these. Assuming a smooth transition between these areas, one can construct a simplified model of the whole vector field. Such a simplified representation is useful, for instance, in compressing vector field data into simpler building blocks [Phi97].

The critical points are classified based on the vector field around that point. The information derived from the classification of critical points aids the information selection process when it comes to visualizing the field. By choosing seed points for field lines based on the topology of critical points, field lines encoding important information is ensured. A more advanced approach is to connect critical points, and use the connecting lines and surfaces to separate areas of different flow topology [Hel89], [Wei05].

3.4.1 Standard classification using a linear approximation

In the study of steady flow/autonomous dynamical systems certain features such as critical points, separatrices and closed orbits play an important role. In 1989, Helman and Hesselink introduced these concepts to the visualization community under the name of vector field topology [Hel89]. Methods for visualizing steady flow fields, especially planar flow fields, have achieved a high level of proficiency, whereas the unsteady case is still challenging and by no means complete [Lar07], [Sal08].

The fact that a linear model can be used to study the behaviour of a nonlinear system near a critical point is a powerful one. If we consider the Taylor series expansion of the field in the neighbourhood of such a point, then the first order partial derivatives of the field (with respect to position) determine the vector field's behaviour. Thus for nondegenerate critical point, we can use the Jacobian matrix (see equation (3.33)) to characterize the vector field and the behaviour of nearby tangent curves.

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{bmatrix} \quad \text{or} \quad \boldsymbol{J} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} \end{bmatrix}. \tag{3.33}$$

The eigenvalues and eigenvectors of Jacobian matrix are very important for vector field classification and description. For the eigenvalues and eigenvectors the following equation applies

$$\boldsymbol{Ju} = \lambda \boldsymbol{u}\,, \tag{3.34}$$

where λ is the eigenvalue and $\boldsymbol{u}$ is the corresponding eigenvector. To compute the eigenvalues, we have to solve

$$(\boldsymbol{J} - \lambda \boldsymbol{I})\boldsymbol{u} = \boldsymbol{0}\,, \tag{3.35}$$

where $\boldsymbol{I}$ is the identity matrix. Knowing Cramer's rule, a linear system of equations has nontrivial solution if the determinant vanishes, so the solutions of equation (3.35) are given by

$$det(\boldsymbol{J} - \lambda \boldsymbol{I}) = 0\,. \tag{3.36}$$

A real eigenvector of the Jacobian matrix defines a direction such that if we move slightly from the critical point in that direction, the field is parallel to the direction we moved. Thus, at the critical point, the real eigenvectors are tangent to the trajectories that end on the point. The sign of the corresponding eigenvalue determines whether the trajectory is outgoing (repelling) or incoming (attracting) at the critical point. The imaginary part of an eigenvalue denotes circulation about the point.

Now suppose that $\boldsymbol{x}$ is on the line determined by an eigenvector $\boldsymbol{u}$ for an eigenvalue λ. That is, $\boldsymbol{x} = t\boldsymbol{u}$ for some scalar t. Then

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{Jx} = \boldsymbol{J} \cdot (t\boldsymbol{u}) = t \cdot \boldsymbol{Ju} = t\lambda \boldsymbol{u}\,. \tag{3.37}$$

The derivative is a multiple of $\boldsymbol{u}$ and hence points along the line determined by $\boldsymbol{u}$. As $\lambda > 0$, the derivative points in the direction of $\boldsymbol{u}$ when t is positive and in the opposite direction when t is negative. The lines determined by the eigenvectors, and arrows on the lines to indicate the directions can be seen in Figure 3.8.

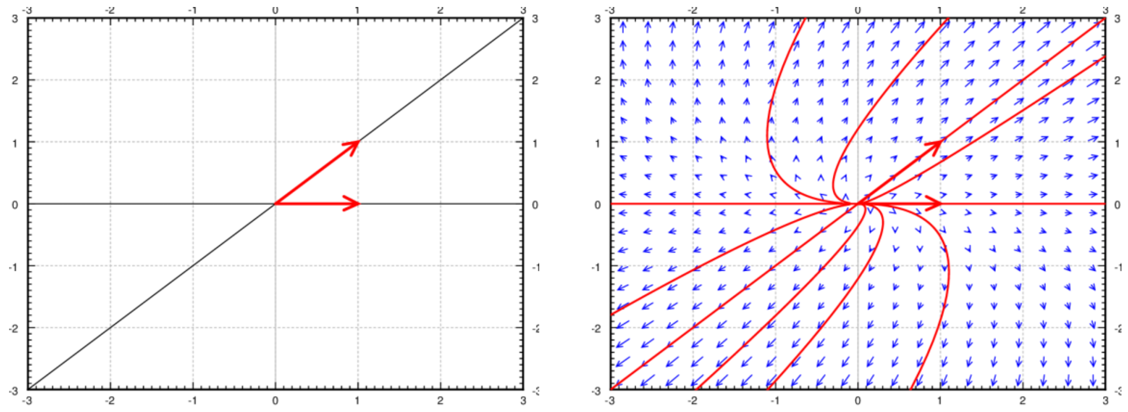


Figure 3.8: Eigenvectors of $\boldsymbol{J}$ with directions (left). Example source vector field with eigenvectors and solutions (right).

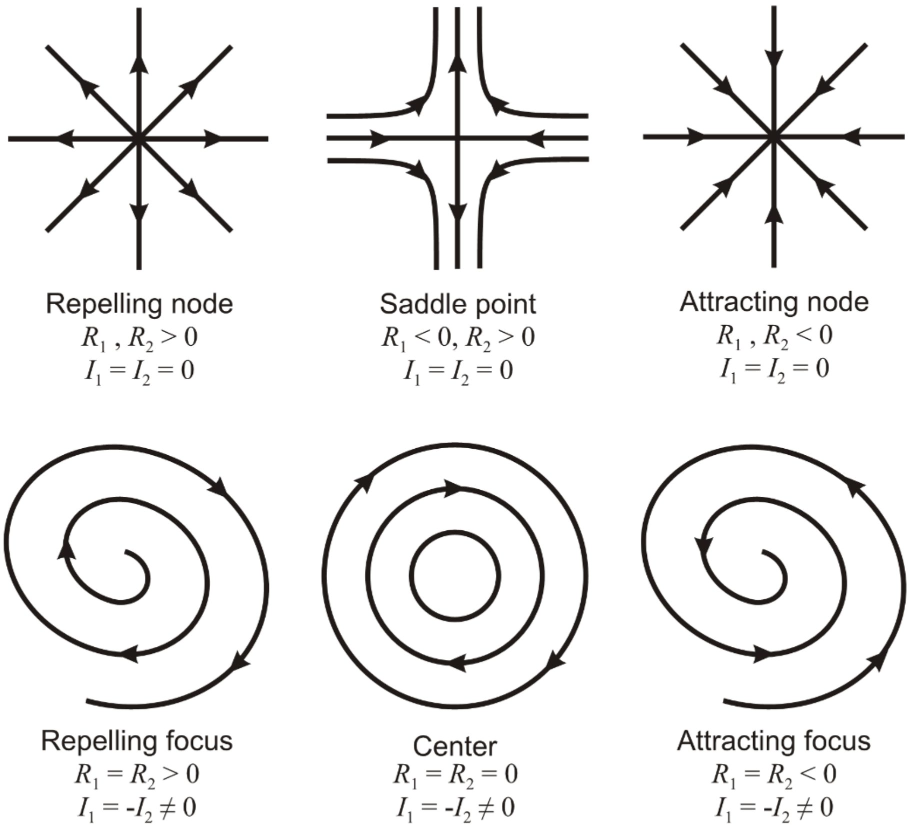


Figure 3.9: Classification of $2D$ first order critical points. R_1, R_2 denote the real parts of the eigenvalues of the Jacobian matrix while I_1, I_2 denote their imaginary parts (from [Hel89]).

Classification of first order critical points can be done using eigenvalues of Jacobian matrix. We have 6 types of first order critical points in $2D$ (see Figure 3.9) and 8 in $3D$ (see Figure 3.10).

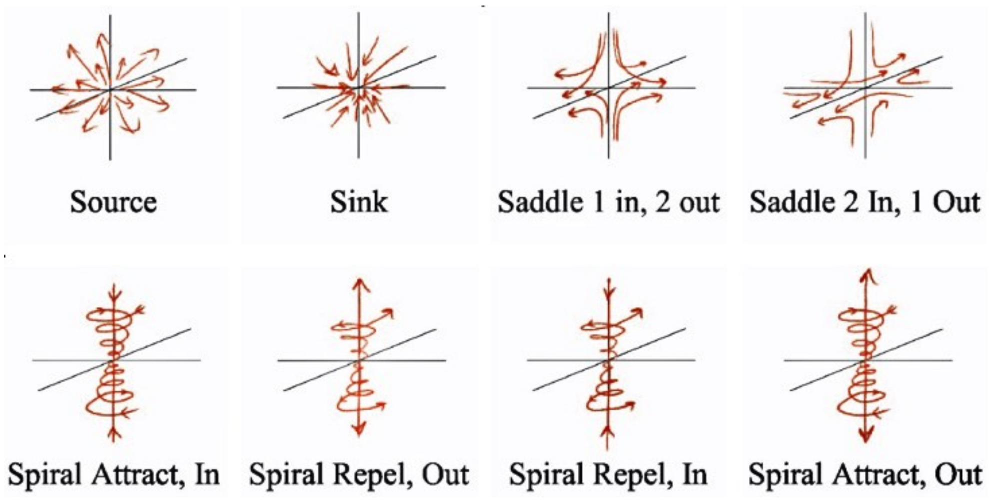


Figure 3.10: Classification of $3D$ first order critical points (from [For09]).

3.4.2 Curvature of vector field

Approximated vector field using (3.26) and (3.27) is not only linear but contains the Hessian matrices that describe the local curvature of the vector field. In this chapter, we will introduce our approach [Smo16c] how to compute the local curvature of a vector field that is approximated with Jacobian and Hessian matrices.

Using an approximation of the vector field with second order derivatives gives us the opportunity to compute a Jacobian matrix $\boldsymbol{J}_\varepsilon$ in the neighborhood of a critical point from approximated vector field (3.26) and (3.27) as

$$\boldsymbol{J}_\varepsilon = \begin{bmatrix} \left.\frac{\partial v_x}{\partial x}\right|_{x_0+\varepsilon} & \left.\frac{\partial v_x}{\partial y}\right|_{x_0+\varepsilon} \\ \left.\frac{\partial v_y}{\partial x}\right|_{x_0+\varepsilon} & \left.\frac{\partial v_y}{\partial y}\right|_{x_0+\varepsilon} \end{bmatrix}, \tag{3.38}$$

where $\boldsymbol{\varepsilon} = \left[e_x, e_y\right]^T$ is an arbitrary direction vector pointing from the critical point $\boldsymbol{x}_0$. The matrix $\boldsymbol{J}_\varepsilon$ (2×2) in (3.38) can be rewritten using elements of $\boldsymbol{J}$ in (3.21), elements of $\boldsymbol{H}_x$ in (3.24) and elements of $\boldsymbol{H}_y$ in (3.25) as

$$\boldsymbol{J}_\varepsilon = \begin{bmatrix} \left.\frac{\partial v_x}{\partial x}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_x}{\partial x^2}\right|_{\boldsymbol{x}_0} \varepsilon_x + \frac{1}{2}\left(\left.\frac{\partial^2 v_x}{\partial x \partial y}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_x}{\partial y \partial x}\right|_{\boldsymbol{x}_0}\right)\varepsilon_y & \left.\frac{\partial v_x}{\partial y}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_x}{\partial y^2}\right|_{\boldsymbol{x}_0} \varepsilon_x + \frac{1}{2}\left(\left.\frac{\partial^2 v_x}{\partial x \partial y}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_x}{\partial y \partial x}\right|_{\boldsymbol{x}_0}\right)\varepsilon_x \\ \left.\frac{\partial v_y}{\partial x}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_y}{\partial x^2}\right|_{\boldsymbol{x}_0} \varepsilon_x + \frac{1}{2}\left(\left.\frac{\partial^2 v_y}{\partial x \partial y}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_y}{\partial y \partial x}\right|_{\boldsymbol{x}_0}\right)\varepsilon_y & \left.\frac{\partial v_y}{\partial y}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_y}{\partial y^2}\right|_{\boldsymbol{x}_0} \varepsilon_x + \frac{1}{2}\left(\left.\frac{\partial^2 v_y}{\partial x \partial y}\right|_{\boldsymbol{x}_0} + \left.\frac{\partial^2 v_y}{\partial y \partial x}\right|_{\boldsymbol{x}_0}\right)\varepsilon_x \end{bmatrix}, \tag{3.39}$$

Assuming that $\boldsymbol{v}$ has continuous second partial derivatives at any given point, the mixed derivatives of v_x and v_y in the Hessian matrix are commutative (Schwarz's theorem)

$$\left.\frac{\partial^2 v_x}{\partial x \partial y}\right|_{\boldsymbol{x}_0} = \left.\frac{\partial^2 v_x}{\partial y \partial x}\right|_{\boldsymbol{x}_0} \quad \text{and} \quad \left.\frac{\partial^2 v_y}{\partial x \partial y}\right|_{\boldsymbol{x}_0} = \left.\frac{\partial^2 v_y}{\partial y \partial x}\right|_{\boldsymbol{x}_0}. \tag{3.40}$$

Equation (3.39) can be rewritten using (3.40) as

$$\boldsymbol{J}_\varepsilon = \begin{bmatrix} \left.\frac{\partial v_x}{\partial x}\right|_{\boldsymbol{x}_0} & \left.\frac{\partial v_x}{\partial y}\right|_{\boldsymbol{x}_0} \\ \left.\frac{\partial v_y}{\partial x}\right|_{\boldsymbol{x}_0} & \left.\frac{\partial v_y}{\partial y}\right|_{\boldsymbol{x}_0} \end{bmatrix} + \begin{bmatrix} \left.\frac{\partial^2 v_x}{\partial x^2}\right|_{\boldsymbol{x}_0} & \left.\frac{\partial^2 v_x}{\partial x \partial y}\right|_{\boldsymbol{x}_0} \\ \left.\frac{\partial^2 v_y}{\partial x^2}\right|_{\boldsymbol{x}_0} & \left.\frac{\partial^2 v_y}{\partial x \partial y}\right|_{\boldsymbol{x}_0} \end{bmatrix} \varepsilon_x + \begin{bmatrix} \left.\frac{\partial^2 v_x}{\partial x \partial y}\right|_{\boldsymbol{x}_0} & \left.\frac{\partial^2 v_x}{\partial y^2}\right|_{\boldsymbol{x}_0} \\ \left.\frac{\partial^2 v_y}{\partial x \partial y}\right|_{\boldsymbol{x}_0} & \left.\frac{\partial^2 v_y}{\partial y^2}\right|_{\boldsymbol{x}_0} \end{bmatrix} \varepsilon_y \tag{3.41}$$

Elements of $\boldsymbol{J}$, $\boldsymbol{H}_x$ and $\boldsymbol{H}_y$ matrices in (3.26) and (3.27) are elements used to calculate $\boldsymbol{J}_\varepsilon$ in (3.39) and (3.41), so there is no need to compute any additional derivatives than those in (3.26) and (3.27). Note that the Jacobian matrix $\boldsymbol{J}_\varepsilon$ is for $\boldsymbol{\varepsilon} = [0,0]^T$ equal to Jacobian matrix $\boldsymbol{J}$ in (3.21).

The Jacobian matrix $\boldsymbol{J}_\varepsilon$ can be computed for any point $(\boldsymbol{x_0} + \boldsymbol{\varepsilon})$. Therefore, we start by computing the eigenvectors of Jacobian matrix $\boldsymbol{J}$ in a critical point $\boldsymbol{x}_0$.

There exist two eigenvectors ($\boldsymbol{u}_1$ and $\boldsymbol{u}_2$) for a Jacobian matrix in $2D$. In the case, that the vector field is circular around the critical point, we will use only the real part of the eigenvectors, i.e.

$$\begin{array}{l} Re(a+bi) = a \\ Im(a+bi) = b \end{array} \qquad a, b \in \mathbb{R} \tag{3.42}$$

To calculate the curvature of the vector field we need to compute the eigenvectors in the near surroundings of the critical point as we explain later. First, we need to compute vectors pointing from $\boldsymbol{x}_0$ in the direction of main axes of the vector field, i.e.

$$\begin{aligned} \boldsymbol{\varepsilon}_{1L} &= -\frac{Re(\boldsymbol{u}_1)}{\|Re(\boldsymbol{u}_1)\|}h & \boldsymbol{\varepsilon}_{1R} &= \frac{Re(\boldsymbol{u}_1)}{\|Re(\boldsymbol{u}_1)\|}h \\ \boldsymbol{\varepsilon}_{2L} &= -\frac{Re(\boldsymbol{u}_2)}{\|Re(\boldsymbol{u}_2)\|}h & \boldsymbol{\varepsilon}_{2R} &= \frac{Re(\boldsymbol{u}_2)}{\|Re(\boldsymbol{u}_2)\|}h\,, \end{aligned} \tag{3.43}$$

where h is some small number (e.g. $h = 10^{-3}$ for the vector field in Figure 3.12).

In the next step, we calculate Jacobian matrix $\boldsymbol{J}_\varepsilon$ for all vectors computed in (3.43), i.e. Jacobian matrix at points $(\boldsymbol{x}_0 + \boldsymbol{\varepsilon}_{**})$

$$\begin{aligned} \boldsymbol{J}_{1L} &= \boldsymbol{J}_\varepsilon(\boldsymbol{\varepsilon}_{1L}) & \boldsymbol{J}_{1R} &= \boldsymbol{J}_\varepsilon(\boldsymbol{\varepsilon}_{1R}) \\ \boldsymbol{J}_{2L} &= \boldsymbol{J}_\varepsilon(\boldsymbol{\varepsilon}_{2L}) & \boldsymbol{J}_{2R} &= \boldsymbol{J}_\varepsilon(\boldsymbol{\varepsilon}_{2R})\,. \end{aligned} \tag{3.44}$$

For each Jacobian matrix in (3.44) we need to calculate real parts of both eigenvectors and determine which one is pointing in the almost same direction like original eigenvector $Re(\boldsymbol{u}_1)$ for $\boldsymbol{J}_{1L}$ and $\boldsymbol{J}_{1R}$ and determine $\boldsymbol{u}_{1L}$ and $\boldsymbol{u}_{1R}$, resp. similarly eigenvector $Re(\boldsymbol{u}_2)$ for $\boldsymbol{J}_{2L}$ and $\boldsymbol{J}_{2R}$ and determine $\boldsymbol{u}_{2L}$ and $\boldsymbol{u}_{2R}$. This test can be done using the dot product between original eigenvector $Re(\boldsymbol{u}_i)$, where $i = \{1, 2\}$, and both real parts of eigenvectors for Jacobian matrix $\boldsymbol{J}_{i*}$ in (3.44). The closest two vectors, i.e. vectors with minimal angle between them, have the greatest dot product. Therefore, for each directional vector in (3.43) we get one vector, thus four vectors $\boldsymbol{u}_{1L}$, $\boldsymbol{u}_{1R}$, $\boldsymbol{u}_{2L}$ and $\boldsymbol{u}_{2R}$, i.e. for example $\boldsymbol{u}_{1L}$ is computed as the following procedure

$$\begin{gathered} \{{}^1\boldsymbol{\xi}_{1L}, {}^2\boldsymbol{\xi}_{1L}\} = Re\big(eigenvectors(\boldsymbol{J}_{1L})\big) \\ \boldsymbol{u}_{1L} = \begin{cases} {}^1\boldsymbol{\xi}_{1L} & {}^1\boldsymbol{\xi}_{1L} \cdot Re(\boldsymbol{u}_1) > {}^2\boldsymbol{\xi}_{1L} \cdot Re(\boldsymbol{u}_1) \\ {}^2\boldsymbol{\xi}_{1L} & {}^1\boldsymbol{\xi}_{1L} \cdot Re(\boldsymbol{u}_1) > {}^2\boldsymbol{\xi}_{1L} \cdot Re(\boldsymbol{u}_1) \end{cases} \end{gathered} \tag{3.45}$$

The curvature vectors of a vector field are computed as follows

$$\begin{aligned} \boldsymbol{c}_1 &= \frac{1}{2h}\left(\frac{\boldsymbol{u}_{1R}}{\|\boldsymbol{u}_{1R}\|} - \frac{\boldsymbol{u}_{1L}}{\|\boldsymbol{u}_{1L}\|}\right) \\ \boldsymbol{c}_2 &= \frac{1}{2h}\left(\frac{\boldsymbol{u}_{2R}}{\|\boldsymbol{u}_{2R}\|} - \frac{\boldsymbol{u}_{2L}}{\|\boldsymbol{u}_{2L}\|}\right)\,. \end{aligned} \tag{3.46}$$

This is a discrete formula for curvature calculation using the difference of two unit vectors.

The important property of the curvature vectors in (3.46) is that they are perpendicular to $Re(\boldsymbol{u}_1)$, resp. $Re(\boldsymbol{u}_2)$, i.e.

$$\begin{aligned} \boldsymbol{c}_1 \cdot Re(\boldsymbol{u}_1) &= 0 \\ \boldsymbol{c}_2 \cdot Re(\boldsymbol{u}_2) &= 0\,. \end{aligned} \tag{3.47}$$

The length of the curvature vectors in (3.46) is a number that characterizes how much is each the main axis of the vector field curved. In the case that both curvatures are equal zero, then matrices $\boldsymbol{H}_x$ and $\boldsymbol{H}_y$ must be zero matrices otherwise at least one of the curvatures is nonzero.

3.4.2.1 Examples of vector field curvature

Vector field around a critical point can be classified as one of the vector type, see Figure 3.9 and Figure 3.10. In this chapter, we will show examples how the vector field approximated with the same Jacobian matrix changes when changing the Hessian matrices used to approximate the vector field around a critical point.

Vector field around a saddle point

An example of the vector field around a saddle point can be characterized with the following approximation

$$\begin{aligned} v_x &= \begin{bmatrix} 1.2 \\ 1.4 \end{bmatrix}^T \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \frac{1}{2} t \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^T \cdot \begin{bmatrix} 1.2 & 0.84 \\ 0.84 & 1.2 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\ v_y &= \begin{bmatrix} 0.7 \\ -0.9 \end{bmatrix}^T \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \frac{1}{2} t \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^T \cdot \begin{bmatrix} -2 & 0.6 \\ 0.6 & 2 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \end{aligned} \tag{3.48}$$

where $t \in \mathbb{R}$ is a parameter. If we will continuously change the parameter t, the vector field will change continuously as well, i.e. there will be no discontinuity.

As an example, the parameter $t \in \langle -1; 1 \rangle$ was changed and both curvatures of the main axes of the vector field were calculated. It can be seen that both the curvatures change continuously (see results in Figure 3.11). For parameter $t = 0$ is the vector field approximated only with Jacobian matrix part of (3.48), i.e.

$$\begin{aligned} v_x &= \begin{bmatrix} 1.2 \\ 1.4 \end{bmatrix}^T \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\ v_y &= \begin{bmatrix} 0.7 \\ -0.9 \end{bmatrix}^T \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \end{aligned} \tag{3.49}$$

and both the curvatures are thus equal to 0.

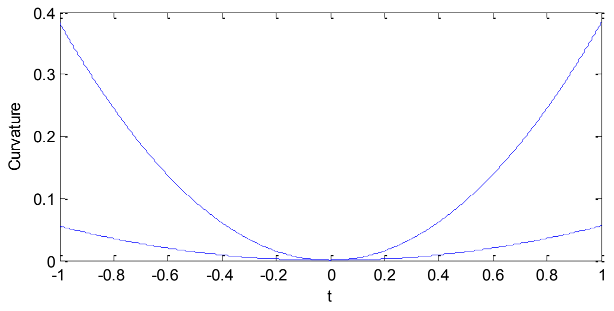


Figure 3.11: Progress of both curvatures when changing the parameter $t \in \langle -1; 1 \rangle$ in (3.48). One curvature grows faster with greater absolute value of t. This means that one main axis is more curved that the other.

The approximated vector field represented by (3.48) can be seen for different values of parameter t in Figure 3.12. It can be seen that the vector field has a different phase portrait for a different values t, however all of them have the same description using a linear approximation of the vector field around a critical point.

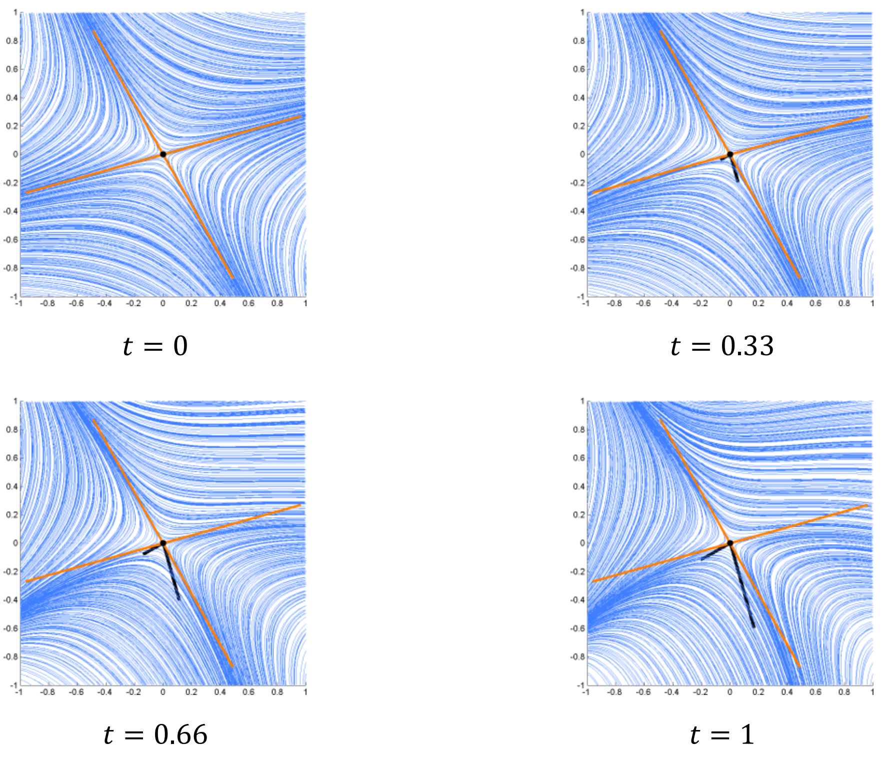


Figure 3.12: Vector fields and their curvatures for different parameters t in (3.48). The orange lines visualize the main axes of vector field obtained from the linear part of the approximation. The black lines visualize vectors of the curvature of the main axes (note, that they are perpendicular to the main orange axes).

Vector field around an attracting focus point

An example of the vector field around an attracting focus point can be characterized with the following approximation

$$\begin{aligned} v_x &= \begin{bmatrix}-0.8\\-1.3\end{bmatrix}^T \cdot \begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} + \frac{1}{2}t\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}^T \cdot \begin{bmatrix}-1.7 & 0.6\\ 0.6 & 1.2\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \\ v_y &= \begin{bmatrix}0.9\\-1.1\end{bmatrix}^T \cdot \begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} + \frac{1}{2}t\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}^T \cdot \begin{bmatrix}0.7 & 0.45\\ 0.45 & 0.95\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}, \end{aligned} \tag{3.50}$$

where $t \in \mathbb{R}$ is a parameter.

In this case, the parameter $t \in \langle -1; 1\rangle$ was changed and the curvature of the real part of the main axis of the vector field was calculated. It can be seen that the curvature changes continuously (see results in Figure 3.13). For parameter $t = 0$ is the vector field approximated only with Jacobian matrix part of (3.50), i.e.

$$\begin{aligned} v_x &= \begin{bmatrix}-0.8\\-1.3\end{bmatrix}^T \cdot \begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \\ v_y &= \begin{bmatrix}0.9\\-1.1\end{bmatrix}^T \cdot \begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \end{aligned} \tag{3.51}$$

and the curvature is thus equal to 0.

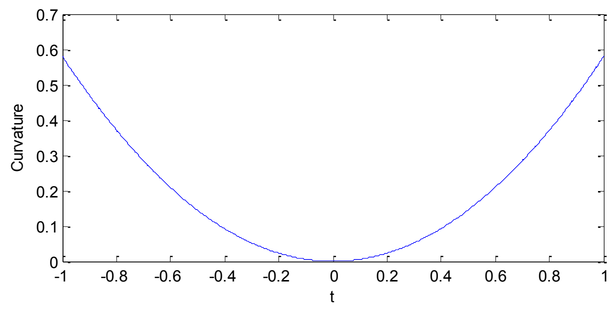


Figure 3.13: Progress of curvature when changing the parameter $t \in \langle -1; 1\rangle$ in (3.50).

The approximated vector field represented by (3.50) can be seen for different values of parameter t in Figure 3.14. It can be seen that the vector field has a different phase portrait for a different values t, however all of them have the same description using a linear approximation of the vector field around a critical point.

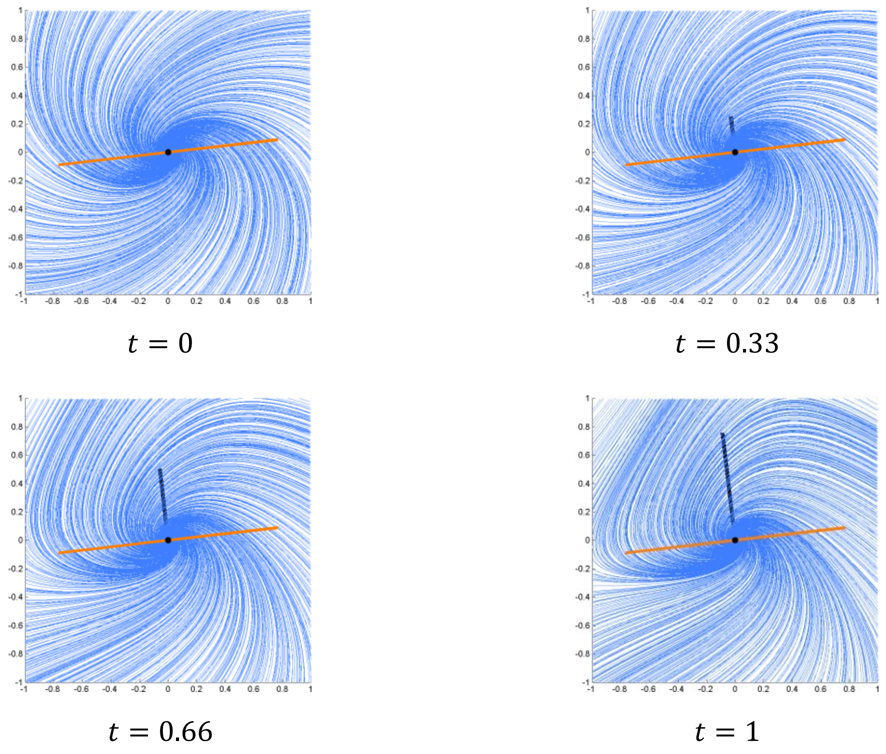


Figure 3.14: Vector fields and their curvatures for different parameters t in (3.50). The orange line visualizes the main axis of vector field, i.e. real part of the eigenvector obtained from the linear part of the approximation. The black line visualizes vector of the curvature of the main axis (note, that it is perpendicular to the main orange axis).

Real Vector Field

Calculation of vector field curvature was shown on synthetic datasets and can be calculated on real datasets as well. A wind dataset[1] was chosen and only a small part from this dataset with only one critical point was selected, (see Figure 3.15). For this critical point, which is a saddle point, the vector field curvature of both main axes was calculated. One curvature is equal to 0.05128 and the second one is 0.00496.

[1] US GFS global weather model. National Centers for Environmental Information (NCEI), https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs, [downloaded: 21.3.2016].

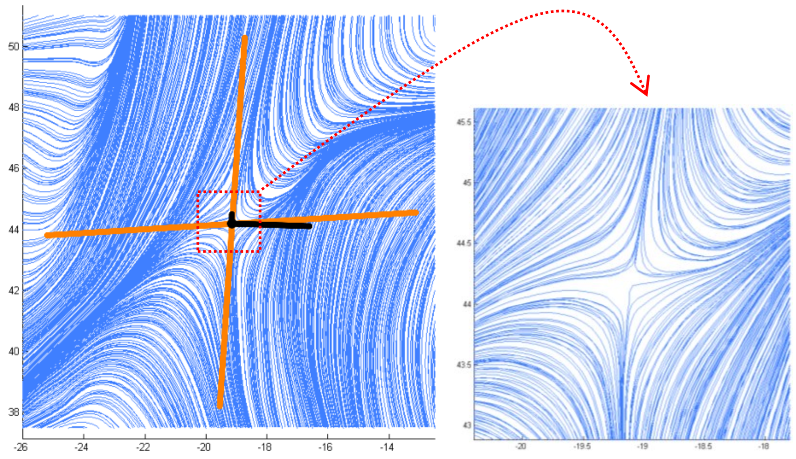


Figure 3.15: Vector field and its curvature. The orange lines visualize the main axes of the vector field obtained from the linear part of the approximation. The black lines visualize vectors of the curvature of the main axes (note that they are perpendicular to the main orange axes).

It can be seen that one main axis has a greater curvature than the other one, as this main axis has the shape of a quadratic curve, and the second axis has the shape of a linear or cubic curve (a cubic curve has zero curvature at its inflection point). The curvature vector of the main axis with greater curvature (the vertical main axis) is pointing to the right and thus this axis is curved to the right side as well.

3.4.3 Classification using description of conic sections

Each vector field can be approximated at a critical point with the approximation that uses the second order derivatives, i.e. Hessian matrix. One such example of approximated vector field around a critical point $\boldsymbol{x}_0 = [0, 0]^T$ can be

$$v_x = \frac{1}{2}[\Delta x \quad \Delta y \quad 1] \cdot \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 2 \\ 1 & 2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ 1 \end{bmatrix} \tag{3.52}$$

$$v_y = \frac{1}{2}[\Delta x \quad \Delta y \quad 1] \cdot \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 1.5 \\ -1 & 1.5 & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ 1 \end{bmatrix}. \tag{3.53}$$

Equation (3.52) represents for $v_x = 0$ a parabola and (3.53) for $v_y = 0$ a line. This approximated vector field can be seen in Figure 3.16.

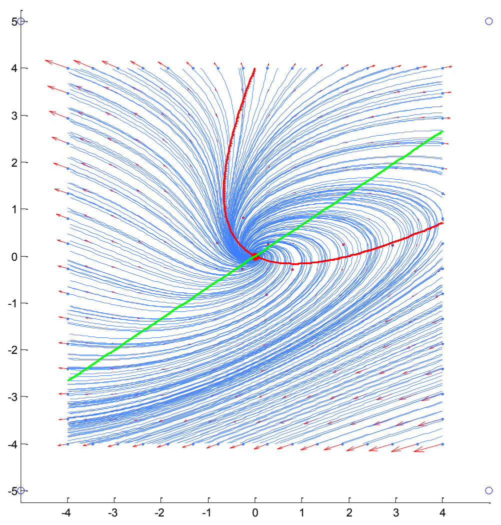


Figure 3.16: Vector field approximated as (3.52) and (3.53). The zero iso-lines are a line and a parabola.

Now, we showed conic sections that have only one intersection point at $[0,0]^T$. Two conic sections can have up to four intersections. Each intersection defines a critical point. Therefore, we can approximate a vector field around one critical point and some more critical points in the neighborhood will be included in this approximation.

Vector fields around a focus critical point can be for some real vector field approximated for example as

$$v_x = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}1 & -3 & 1\\ -3 & 1 & 2\\ 1 & 2 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \qquad v_y = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}0 & 0 & -1\\ 0 & 0 & 1.5\\ -1 & 1.5 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \quad (3.54)$$

$$v_x = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}-0.5 & 0.5 & 1\\ 0.5 & -0.5 & 2\\ 1 & 2 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \quad v_y = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}-1 & 1 & -1\\ 1 & -1 & 1.5\\ -1 & 1.5 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \quad (3.55)$$

This both approximations of vector fields describe behaviour around a focus critical point at $[0,0]^T$. Both of them contain one more critical point, which is a saddle critical point. These saddle critical points do not have to be real critical points of the approximated vector field, but they can be present in the vector field. Therefore, this approximation can give us some information about other possible critical points in the neighbourhood of approximated critical point $\boldsymbol{x}_0$. When locating all critical points in the vector field, we can use this information to increase the probability of finding all critical points.

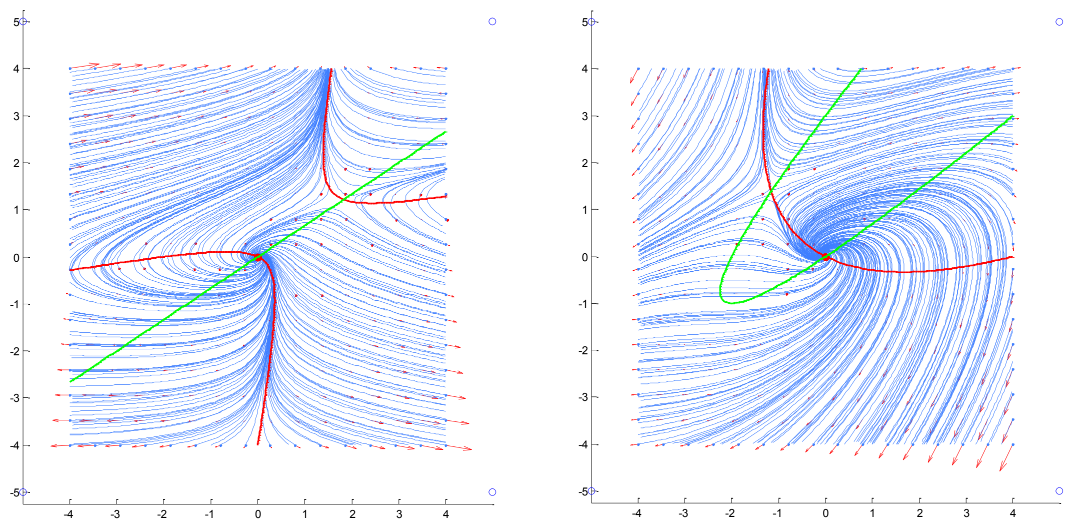
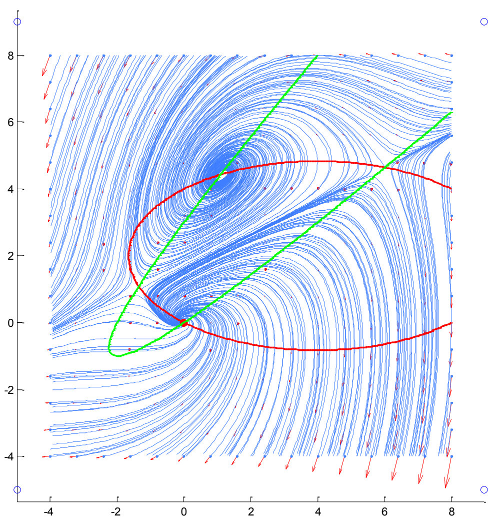


Figure 3.17: Vector field approximated as (3.54) (left) and (3.55) (right). The zero iso-lines are a line and a hyperbola (left), or two parabolas (right).

The maximal number of two conic sections intersection points is four. In the next example, we will show it. Let us have a vector field, which can be approximated at point $\boldsymbol{x}_0$ for example as

$$v_x = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}-0.25 & 0 & 1\\ 0 & -1 & 2\\ 1 & 2 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \quad v_y = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}-1 & 1 & -1\\ 1 & -1 & 1.5\\ -1 & 1.5 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \tag{3.56}$$

$$v_x = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 2\\ 1 & 2 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \quad v_y = \frac{1}{2}\begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix}^T \cdot \begin{bmatrix}1 & -1 & -1\\ -1 & 1.5 & 1.5\\ -1 & 1.5 & 0\end{bmatrix} \cdot \begin{bmatrix}\Delta x\\ \Delta y\\ 1\end{bmatrix} \tag{3.57}$$



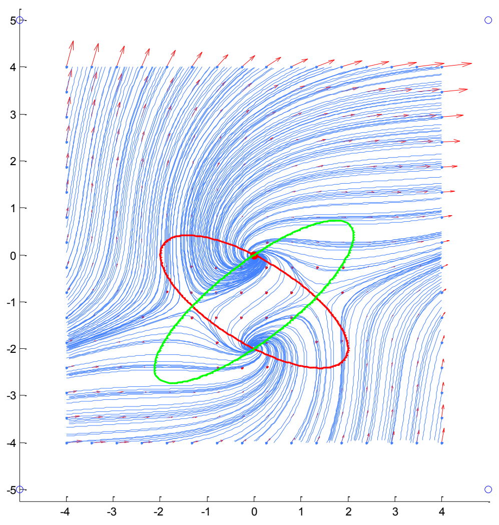


Figure 3.18: Vector field approximated as (3.56) (left) and (3.57) (right). The zero iso-lines are a parabola and an ellipse (left), or two ellipses (right).

These two approximations (3.56) and (3.57) of vector fields are visualized in Figure 3.18. It can be seen, that each approximation contains four critical points, i.e. one critical point where the vector field was approximated and three more critical points.

We showed the geometrical properties of vector field approximation using the second order derivatives, i.e. Hessian matrix. This approximation can be rewritten in a matrix form of a conic section formula. We presented, that approximation using Hessian matrix, rather than only Jacobian matrix, gives us more detailed description of vector field and can help us when locating critical points in a vector field.

3.5 Vector field topology

Vector field topology was introduced to the visualization community by Helman and Hesselink [Hel89]. They defined the concept of a topological skeleton consisting of critical points and connecting separatrices to segment the field into regions of topologically equivalent streamline behaviour. Algorithms to extract periodic orbits, completing this topological structure, were proposed in [The04], [Che07]. A good introduction to the concepts and algorithms of vector field topology is given in [Wei08].

We can think of flow topology in terms of surfaces (in case of $3D$ domains) or curves (in case of $2D$ domains) that divide the flow into separate regions. Two sets of surfaces or lines are of particular interest:

- Tangent surfaces that actually intersect the wall of a body where the flow attaches to or separates from that wall. Tangent curves on either side are deflected, moving along the surface of the body.
- Surfaces where tangent curves start arbitrarily close to each other can end up in substantially different regions.

Topological concepts are very powerful because, given the critical points in a vector field and the tangent curves or surfaces connecting them, you can infer the shape of other tangent curves and hence to some extent the structure of the entire vector field.

To improve the applicability of vector field topology, a variety of extensions like topology tracking, extraction of boundary topology, or extensions to $3D$ have been developed. As the topological skeleton of real world data sets is usually rather complex, a lot of work has been done towards simplification of topological skeletons of vector fields. Lodha et al. [Lod00], [Lod03] introduced a compression technique for $2D$ vector fields which prohibits strong changes of location and Jacobian matrix of the critical points. Theisel et al. [The03] presented an approach which guarantees that the topology of original and compressed vector field coincides both for critical points and for the connectivity of the separatrices. It is shown that even under these strong conditions high compression ratios for vector fields with complex topologies are achieved. [Agr15] subsample the velocity data and introduce a novel technique for informed selection of subsamples. Furthermore, they explore an adaptive system which exchanges the subsampling budget over parallel tasks, to ensure that subsampling occurs at the highest rate in the areas that need it most.

[Koch15] presented a vector field approximation for $2D$ vector fields that preserves their topology and significantly reduces the memory footprint. This approximation is based on a segmentation. The flow within each segmentation region is approximated by an affine linear function. The implementation is driven by following aims: the approximation preserves the original topology, the maximal approximation error is below a user-defined threshold in all regions, the number of regions is as small as possible and each point has the minimal approximation error. The generation of an optimal solution is computationally infeasible, so they provide a greedy strategy to efficiently compute a sensible segmentation that considers the presented aims. Finally, they use the region-wise affine linear approximation to compute a simplified grid for the vector field.

Simplifying vector fields via critical point cancellation has practical merit for interpreting the behaviours of complex vector fields such as turbulence. [Skr16] introduces the approach to directly cancel pairs or groups of 3D critical points in a hierarchical manner with a guaranteed minimum amount of perturbation based on their robustness, a quantitative measure of their stability. In addition, the algorithm does not require the extraction of the entire 3D topology, which contains non-trivial separation structures, and thus is computationally effective. Furthermore, the algorithm can remove critical points in any sub-region of the domain whose degree is zero and can handle complex boundary configurations. For more simplification algorithms see for example [Tri01], [Wei05].

Scalar field topology was developed almost independently from vector field topology. The main application areas in visualization include segmentation, transfer function design, and ridge extraction. Due to their robustness and stability, combinatorial extraction algorithms have been especially successful in this context [Gyu06], [Lew04]. To reduce the often very complex topological structure that is generated by these algorithms, a controlled simplification is introduced based on the mathematically well-founded concept of persistence in [Ede08]. Due to the simplicity and clarity of this simplification strategy, it has been widely adopted. Most of the above-mentioned extraction algorithms make use of Forman's work [For01] on a discrete scalar field topology for cell complexes. Rather than choosing a suitable class of continuous functions, a single number is assigned to each cell of the complex and all further steps are combinatorial.

4 Meshless interpolation and approximation

Interpolation and approximation are probably the most frequent operations used in computational techniques. Several techniques have been developed for data interpolation, but they expect some kind of data “ordering”, e.g. structured mesh, rectangular mesh, unstructured mesh etc. The typical example is a solution of partial differential equations (PDE) where derivatives are replaced by differences and rectangular or hexagonal meshes are used in the vast majority of cases. However in many engineering problems, data are not ordered and they are scattered in k-dimensional space, in general. Usually, in technical applications the scattered data are tessellated using triangulation but this approach is quite prohibitive for the case of k-dimensional data interpolation because of the computational cost.

4.1 Radial Basis Function

Radial basis function (RBF) is a technique for scattered data interpolation and approximation [Fas07], [Ska15]. RBF interpolation and approximation is computationally more expensive, because input data are not ordered and there is no known relation between them. However RBF has higher computational cost, it can be used for many applications, e.g. solution of partial differential equations, image reconstruction, neural networks, fuzzy systems, GIS systems, optics etc.

RBFs are the natural generalization of univariate polynomial splines to a multivariate setting. The main advantage of this type of approximation is that it works for arbitrary geometry with high dimensions and it does not require a mesh at all. A RBF is a function whose value depends only on the distance from some center point. Using distance functions, RBFs can be easily implemented to reconstruct a plane or surface using scattered data in $2D$, $3D$ or higher dimensional spaces. Due to the uses of the distance functions, the RBFs can be easily implemented to reconstruct the surface using scattered data in $2D$, $3D$ or higher dimensional spaces.

Application of RBF to the solution of the scattered data interpolation or approximation problem benefits from the fact that RBF is insensitive to the dimension k of the data space. Instead of dealing with a multivariate function, whose complexity will increase with the dimension k, it can be used the same univariate function φ for any choice of k.

RBF calculation can be used with many basis function. One such example is a well-known Gaussian radial basis function

$$\varphi(r) = e^{-(\varepsilon r)^2}, \tag{4.1}$$

where $r \in \mathbb{R}$. The shape parameter ε is related to the variance σ^2 of the normal distribution function by $\varepsilon^2 = 1/(2\sigma^2)$. If the Gaussian is composed with Euclidean distance function, one obtain for any fixed center point $\boldsymbol{x}_k$ a multivariate function

$$\Phi(\boldsymbol{x}) = e^{-(\varepsilon\|\boldsymbol{x}-\boldsymbol{x}_k\|)^2}\,. \tag{4.2}$$

The connection between (4.1) and (4.2) is given by

$$\Phi(\boldsymbol{x}) = \varphi(\|\boldsymbol{x}-\boldsymbol{x}_k\|)\,. \tag{4.3}$$

Radial function interpolants have the nice property of being invariant under all Euclidean transformations, i.e. translations, rotations and reflections. By this it means that it does not matter whether we first compute the RBF interpolation function and then apply a Euclidean transformation, or if we first transform all the data and then compute the radial function interpolants. This result of the fact that Euclidian transformations are characterized by orthogonal transformation matrices and are therefore 2-norm invariant.

Radial basis functions can be divided into two groups according their influence. First group are "global" RBF, for example:

$$\begin{array}{ll}
\text{Thin-Plate Spline (TPS)} & \varphi(r) = r^2 \log r \\
\text{Gauss function} & \varphi(r) = e^{-(\varepsilon r)^2} \\
\text{Inverse Quadric (IQ)} & \varphi(r) = \dfrac{1}{1+(\varepsilon r)^2} \\
\text{Inverse Multiquadric (IMQ)} & \varphi(r) = \dfrac{1}{\sqrt{1+(\varepsilon r)^2}} \\
\text{Multiquadric (MQ)} & \varphi(r) = \sqrt{1+(\varepsilon r)^2}
\end{array} \tag{4.4}$$

The "local" RBF were introduced by [Wen95] as Compactly Supported RBF (CSRBF) and satisfy the following condition

$$\varphi(r) = \begin{cases} (1-r)^q \cdot P(r) & 0 \le r \le 1 \\ 0 & r > 1 \end{cases}, \tag{4.5}$$

where $P(r)$ is a polynomial function and q is a parameter. Typical examples of CSRBF are

$$\begin{aligned}
\varphi_1(r) &= (1-r)_+ \\
\varphi_2(r) &= (1-r)_+^3(3r+1) \\
\varphi_3(r) &= (1-r)_+^5(8r^2+5r+1) \\
\varphi_4(r) &= (1-r)_+^2 \\
\varphi_5(r) &= (1-r)_+^4(4r+1) \\
\varphi_6(r) &= (1-r)_+^6(35r^2+18r+3) \\
\varphi_7(r) &= (1-r)_+^8(32r^3+25r^2+8r+1) \\
\varphi_8(r) &= (1-r)_+^3 \\
\varphi_9(r) &= (1-r)_+^3(5r+1) \\
\varphi_{10}(r) &= (1-r)_+^7(16r^2+7r+1)
\end{aligned} \tag{4.6}$$

Visualisation of CSRBF in (4.6) can be seen in Figure 4.1 and it should be noticed that the function value of all CSRBF for $r = 1$ is equal 0.

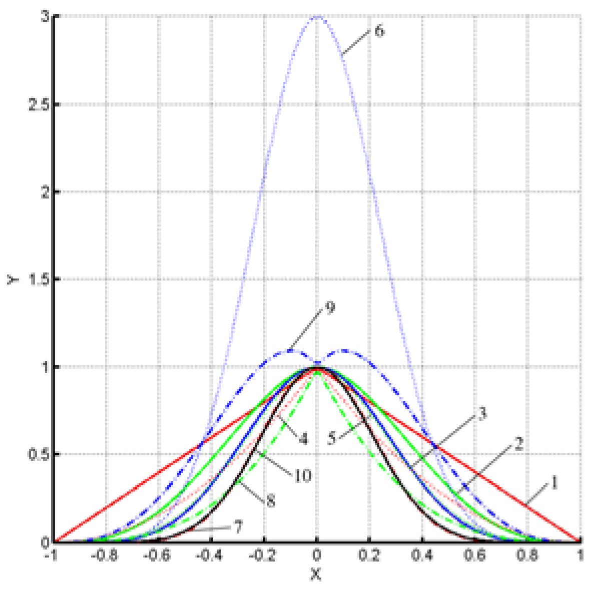


Figure 4.1: Examples of CSRBF from equation (4.6) (from [Uhl06]).

4.2 Radial Basis Function interpolation

The RBF interpolation was originally introduced by [Har71], is based on computing of the distance of two points in the k-dimensional space and is defined by a function

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} \lambda_j \, \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)\,, \tag{4.7}$$

where λ_j are weights of RBF, M is number of radial basis functions, i.e. number of interpolation points, and φ is the radial basis function. For a given dataset of points with associated values, i.e. $\{\boldsymbol{x}_i, h_i\}_1^M$, the following linear system of equations is obtained

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \qquad \text{for } \forall i \in \{1, \dots, M\}\,, \tag{4.8}$$

where λ_j are weights to be computed, see Figure 4.2 for visual interpretation of (4.7) or (4.8) for a 2½D function.

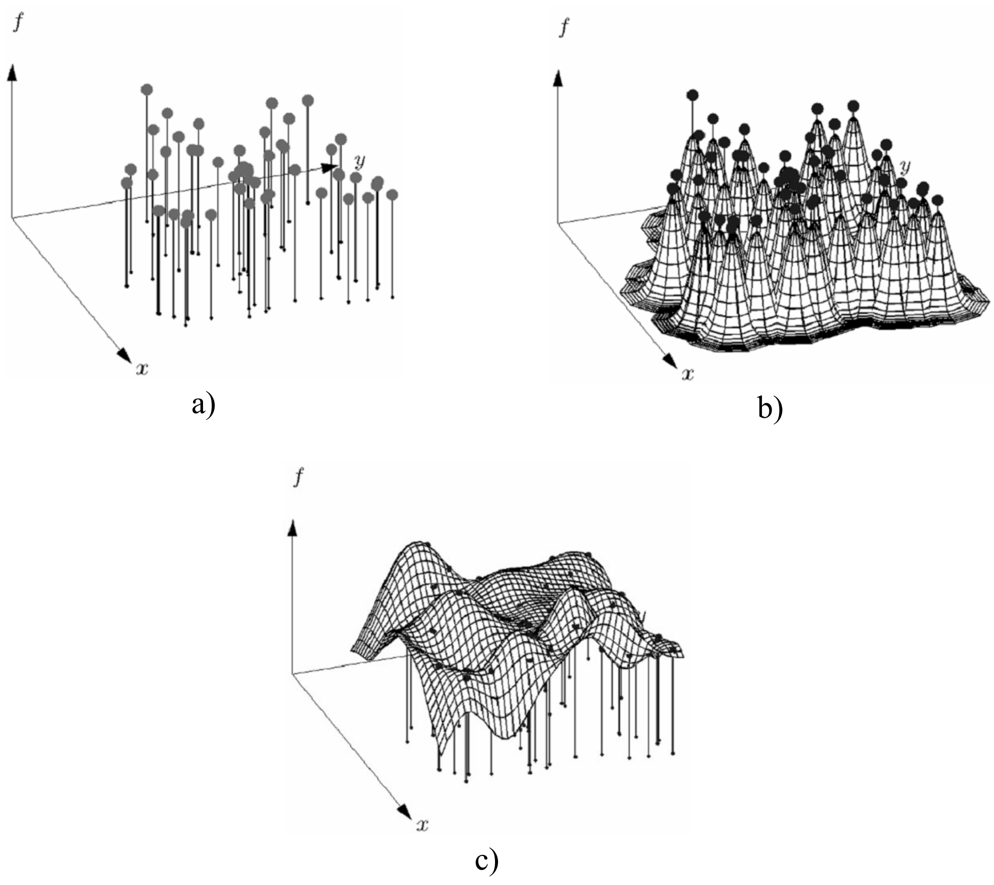


Figure 4.2: Data values $\{h_i\}_{i=1}^{M}$ (a), the RBF collocation functions (b), the resulting interpolant (c) (from [Fly09]).

Due to stability reasons [Wri03] added a polynomial $P_p(\boldsymbol{x})$ of a degree p to the form for RBF interpolation, i.e.

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|\right) + P_p(\boldsymbol{x}_i) \qquad \text{for } \forall i \in \{1, \dots, M\}\,, \tag{4.9}$$

where $P_p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_p x^p$ is a p degree polynomial function (for 1&½ dimensional case) with unknown coefficients $\boldsymbol{a} = \left[a_0, a_1, a_2, \dots, a_p\right]^T$. We can rewrite (4.9) as

$$\begin{bmatrix} \varphi(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|) & \cdots & \varphi(\|\boldsymbol{x}_1 - \boldsymbol{x}_M\|) & 1 & x_1 & \cdots & x_1^p \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \varphi(\|\boldsymbol{x}_M - \boldsymbol{x}_1\|) & \cdots & \varphi(\|\boldsymbol{x}_M - \boldsymbol{x}_M\|) & 1 & x_M & \cdots & x_M^p \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \\ a_0 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_M) \end{bmatrix}. \tag{4.10}$$

It can be seen from (4.10) that the system of linear equations has M equations with $(M+p+1)$ unknowns. For this reason [Wri03] formulated $(p+1)$ additional conditions to be fulfilled

$$\sum_{j=1}^{M} \lambda_j \left(x_j\right)^i = 0 \qquad \text{for } \forall i \in \{0, \dots, p\}\,. \tag{4.11}$$

Adding (4.11) to (4.10) we get the following system of linear equations

$$\begin{bmatrix} \varphi(\|\boldsymbol{x}_1-\boldsymbol{x}_1\|) & \cdots & \varphi(\|\boldsymbol{x}_1-\boldsymbol{x}_M\|) & 1 & x_1 & \cdots & x_1^p \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \varphi(\|\boldsymbol{x}_M-\boldsymbol{x}_1\|) & \cdots & \varphi(\|\boldsymbol{x}_M-\boldsymbol{x}_M\|) & 1 & x_M & \cdots & x_M^p \\ 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ x_1 & \cdots & x_M & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_1^p & \cdots & x_M^p & 0 & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \\ a_0 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_M) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{4.12}$$

System of linear equations (4.12) has $(M+p+1)$ equations with $(M+p+1)$ unknowns and can be rewritten using matrix notation as

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\,. \tag{4.13}$$

Matrix $\boldsymbol{B}$ is symmetrical, as $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|$, and thus the matrix $\boldsymbol{A}$ is symmetrical as well.

RBF interpolation can be done using "global" or "local" functions. When using "global" radial basis functions the matrix $\boldsymbol{B}$ will be full, but when using local radial basis functions the matrix $\boldsymbol{B}$ will be sparse, which can be beneficial when solving the system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$.

4.3 Radial Basis Function approximation

Radial basis function interpolation computes the system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is a square matrix. In some cases it is necessary to approximate the input data, i.e. for noisy data, oversampled data etc. When approximating the data, the RBF formulation is

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j\, \varphi\left(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|\right) + P_p(\boldsymbol{x}_i) \qquad \text{for } \forall i \in \{1, \dots, M\}\,, \tag{4.14}$$

where $\boldsymbol{\xi}_j$ are not given points, but can be freely defined as only coordinates are needed. This reference points can form any distribution of points, e.g. points can form a regular grid in $2D$, see Figure 4.3.

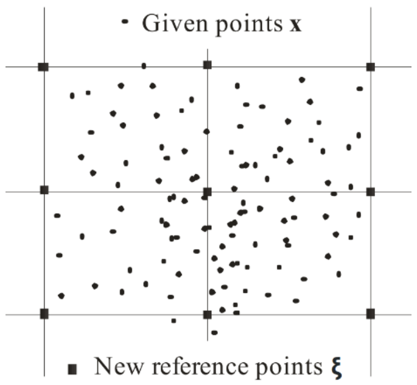


Figure 4.3: Example of reference points forming a regular grid in $2D$ (from [Ska13]).

The system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ is overdetermined, when the number of given points is larger than the number of new reference points. The approximation is done using a method of least square error. The error is computer using

$$r_i = f(\boldsymbol{x}_i) - \left(\sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\right\|\right) + P_p(\boldsymbol{x}_i) \right) \qquad \text{for } \forall i \in \{1, \dots, M\} \quad (4.15)$$

and using matrix notation

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax} \,. \quad (4.16)$$

The square of error $\boldsymbol{r}$ is calculated as

$$r^2 = (\boldsymbol{b} - \boldsymbol{Ax})^T(\boldsymbol{b} - \boldsymbol{Ax}) = \boldsymbol{b}^T\boldsymbol{b} - (\boldsymbol{Ax})^T(\boldsymbol{Ax}) - (\boldsymbol{Ax})^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{Ax} \,. \quad (4.17)$$

To find an extreme, the following condition must be valid

$$\frac{\partial r^2}{\partial \boldsymbol{x}} = 2\boldsymbol{A}^T\boldsymbol{Ax} - 2\boldsymbol{A}^T\boldsymbol{b} = 0 \,, \quad (4.18)$$

and thus the following system of linear equations has to be solved

$$\boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{A}^T\boldsymbol{b} \,. \quad (4.19)$$

The matrix $\boldsymbol{A}^T\boldsymbol{A}$ is squared and symmetric positively semi-definite matrix.

4.4 Vector field RBF interpolation

Vector fields are results of numerical simulations or data measuring process. This kind of vector field data has discrete representation, but an analytical formula describing the vector filed is much more useful. We will show our approach [Smo16a] how to interpolate a vector field using radial basis functions.

A very important future of a vector field are its critical points. The interpolation must preserve positions and types of all critical points. Thus, the RBF interpolation should interpolate the vector field at all positions of critical points to preserve their positions. To preserve their types, we should include few more points in the neighborhood of each critical point to the interpolation. The number of points in the neighborhood was experimentally chosen to be 4, as more points does not improve the interpolation in any significant way. Points in the neighborhood of a critical point $\boldsymbol{x}_0 = [x_0, y_0]^T$ are chosen using the following formula

$$\begin{bmatrix} P_x^{(k)} \\ P_y^{(k)} \end{bmatrix} = \begin{bmatrix} x_0 + r\sin\left(k\frac{\pi}{2}\right) \\ y_0 + r\cos\left(k\frac{\pi}{2}\right) \end{bmatrix}, \tag{4.20}$$

where $k \in \{0, 1, 2, 3\}$ and r is a small number depending on the distance of critical points, where the distance to the nearest critical point should be $\gg r$.

This set of critical points together with their neighborhood points can be interpolated using RBF (4.7), note that each component of vectors $\boldsymbol{v} = [v_x, v_y]^T$ is interpolated separately. This interpolation will preserve the location of critical points together with their types.

To get more accurate interpolation formula of a vector field at points $x \in [x_{\{min\}}, x_{\{max\}}] \times [y_{\{min\}}, y_{\{max\}}]$ we can include some more random points from this interval into the interpolation. The improvement of quality depending on the number of additionally included points will be shown in the following chapter.

4.4.1 Results using synthetic data

The results will be demonstrated on an analytical vector field, as we can measure the interpolation errors precisely. The analytical vector field, which we choose as an example, is described with the following equation

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} x\left(\frac{1}{2}x^2 + \frac{1}{2}\right) + y\left(-x + \left(\frac{1}{2}y - 1\right)y + \frac{1}{2}\right) \\ \frac{1}{2}x^2 y + x\left(-\frac{1}{2}y^2 + y - \frac{1}{2}\right) + \frac{1}{2}y - 1 \end{bmatrix}, \tag{4.21}$$

this vector field (4.21) has three critical points $\boldsymbol{x}_0$

$$\begin{aligned} &\text{source location:} && \boldsymbol{x}_0 = [-1, 1]^T \\ &\text{source location:} && \boldsymbol{x}_0 = [1, 1]^T \\ &\text{saddle location:} && \boldsymbol{x}_0 = [0.543689, 1.83929]^T . \end{aligned} \tag{4.22}$$

The vector field (4.21) will be interpolated and tested on interval $[-2, 2] \times [-1, 3]$, as all important features will be visible. The RBF function used for interpolation is a Gauss radial basis function and the shape parameter ε was experimentally selected as $\varepsilon = 1$.

Vector field (4.21) can be interpolated using 3 critical point positions and 12 more neighborhood points, i.e. 4 neighborhood points for each critical point. The neighborhood points are computed with (4.20) and the parameter $r = 0.1$. The v_x component of the vector field is interpolated with one RBF and the v_y component of the vector field is interpolated with one RBF as well. The phase portrait of original analytical vector field (4.21) is visualized in Figure 4.4a and the phase portrait of RBF interpolated vector field is visualized in Figure 4.4b. It can be seen, that both phase portraits look very similar and have the same vector field topology. Moreover, the critical points location is identical, as the average length of displacement error for all critical points is $7.0283 \cdot 10^{-8}$, which is only a numerical error of the critical points location algorithm.

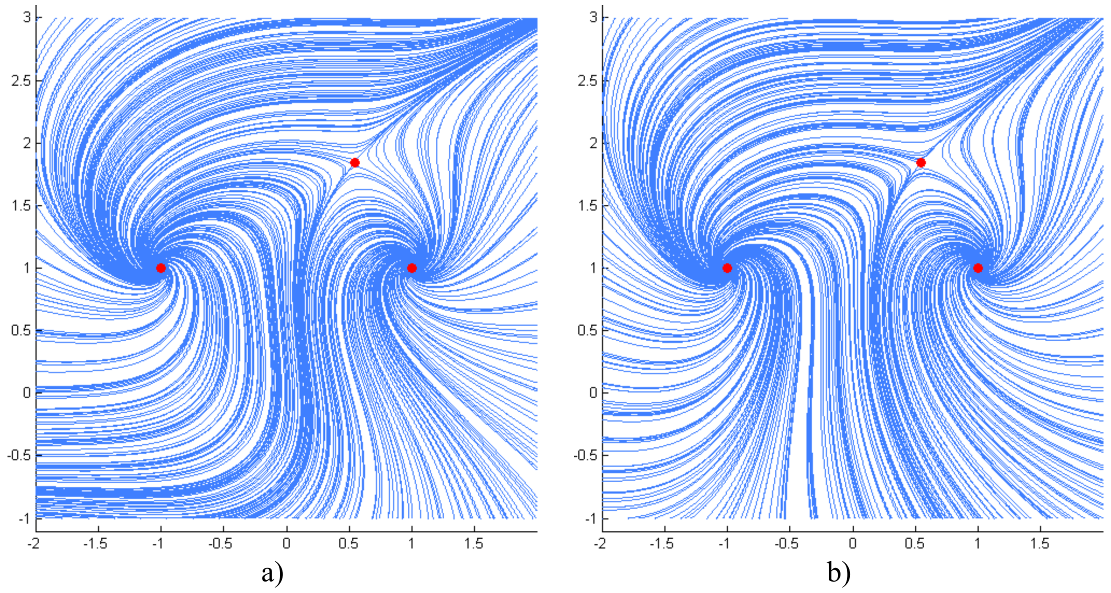


Figure 4.4: Phase portrait of the vector field (4.21) (a) and phase portrait of a RBF interpolation using only 15 reference points (3 critical points plus three times 4 neighbourhood points) (b).

We computed the interpolation error for v_x and v_y and visualized it in Figure 4.5. It can be seen that the interpolation error is getting higher as the distance from critical points increases. The average error of vector length at interval $[-2, 2] \times [-1, 3]$ is 1.7943 (the vector length varies from 0 to 12.6194) and the average error of vector angular displacement is 0.1966 $[rad]$.

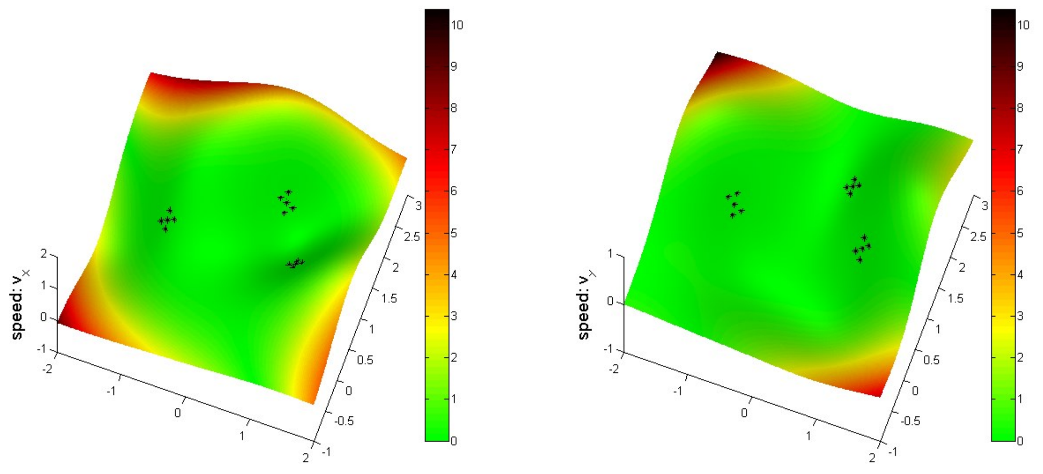


Figure 4.5: Interpolation error of RBF interpolation using only 15 reference points (3 critical points plus three times 4 neighbourhood points). Interpolation error of v_x (a) and interpolation error of v_y (b).

The vector field (4.21) was interpolated using 3 critical points locations plus three times 4 neighborhood points. We can include few more randomly distributed points into the interpolation to reduce the distance error from (4.21). We choose to generate additional 85 points from interval $[-2,2] \times [-1,3]$, so the interpolation of vector field will contain 10^2 points in total. This interpolation of vector field is visualized in a phase portrait, see Figure 4.6 and Figure 4.4a for comparison with original phase portrait.

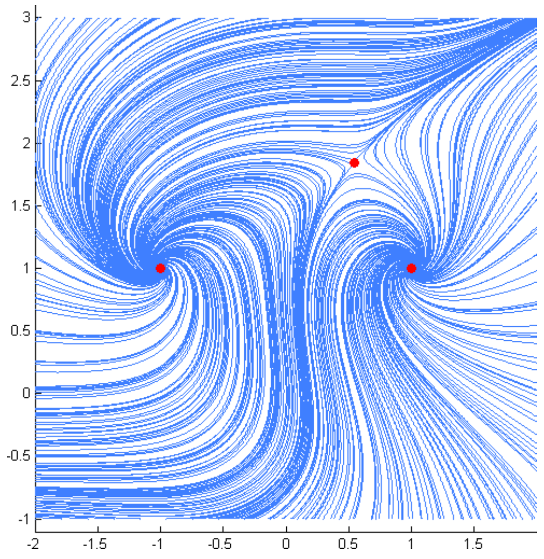
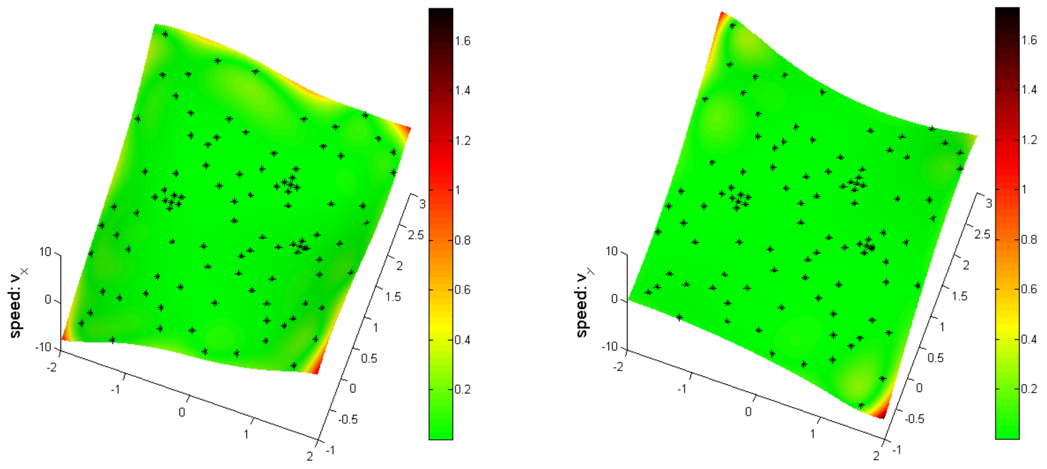


Figure 4.6: Phase portrait of a vector field RBF interpolation of (4.21) using 100 reference points (3 critical points plus three times 4 neighbourhood points plus 85 randomly distributed points).

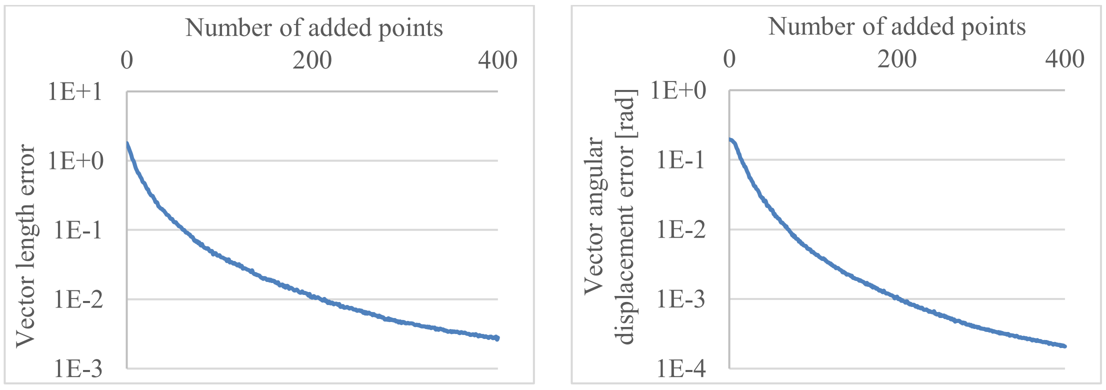
We computed the interpolation error for v_x and v_y and visualized it in Figure 4.7. It can be seen that the interpolation error is close to zero except for locations on the border. The average error of vector length at interval $[-2,2] \times [-1,3]$ is 0.0549 (note that the vector length varies from 0 to 12.6194) and the average error of vector angular displacement is 0.0065 $[rad]$.



a) b)

Figure 4.7: Interpolation error of RBF interpolation using 100 reference points (3 critical points plus three times 4 neighbourhood points plus 85 randomly distributed points). Interpolation error of v_x (a) and interpolation error of v_y (b).

The average vector length error and the average vector angular displacement error were measured for different number of interpolated points. A number of points k is used as added points for the RBF interpolation, thus the RBF interpolation uses $(k + 3 + 3 \cdot 4)$ points for interpolation of vector field, i.e. k randomly distributed points from interval $[-2, 2] \times [-1, 3]$ plus 3 critical points plus three times 4 neighborhood points. Number k was tested from 0 to 400 fifty times for each k with step $\Delta k = 1$ and results are visualized in Figure 4.8.



a) b)

Figure 4.8: Average errors of the RBF interpolation of vector field (4.21) using k added reference points, i.e. 3 critical points plus three times 4 neighbourhood points plus k randomly distributed points, where $k \in \{0, \dots, 400\}$. The vector field length error, note that the vector length varies from 0 to 12.6194 (a) and the vector field angular displacement error (b).

It can be seen that both errors in Figure 4.8 decrease with increasing number k of added points for the interpolation of vector field. According to the required accuracy of the interpolation, the user can select the minimal necessary number of added points and perform the interpolation according to the algorithm proposed.

We presented a new and easy to implement approach for the vector field interpolation using radial basis functions. In general, it can be used in any \$d-\$dimensional space, although the results were presented only for $2D$ vector field. The proposed RBF interpolation proved the ability to interpolate a vector field when preserving the location of critical points and the vector field topology as well.

The proposed approach offers not only analytical description of the discrete data of vector field, but also a significant data compression. This might be a significant feature for "progressive vector field visualization" approach.

4.4.2 Results using data from simulation

We tested our interpolation technique on data obtained from numerical simulation as well. This data were computed on an irregular tetrahedral mesh as Newtonian flow (see Figure 4.9)[1]. This dataset contains 50 052 positions, 274 498 tetrahedrons and 50 052 vectors at each time step of simulation representing the blood flow in a simple coronary bypass.

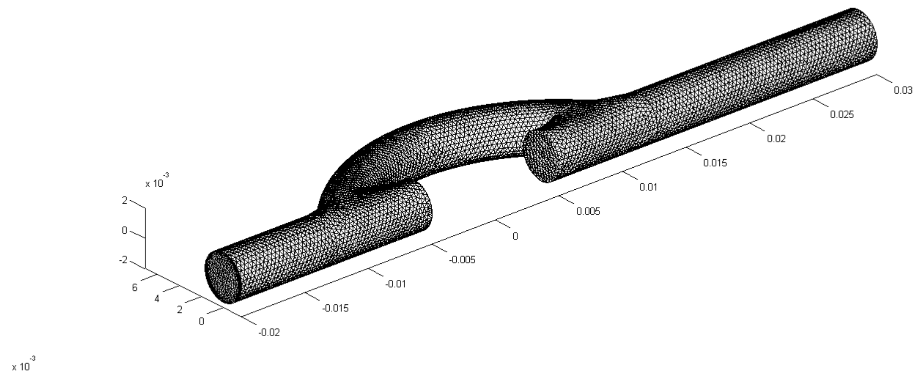


Figure 4.9: Phase portrait of a vector field RBF interpolation of (4.21) using 100 reference points (3 critical points plus three times 4 neighbourhood points plus 85 randomly distributed points).

For the illustration we selected a longitudinal cut of the dataset, which is exactly in the middle of the dataset, i.e. the z coordinate is equal 0. Note, that we are using only one cut of the dataset, thus the vector field and the new dataset becomes a two dimensional problem. For this cut we used a linear interpolation to compute the phase portrait of the vector field. The results can be seen in Figure 4.10 (right).

[1] Author of the dataset is Ing. Alena Jonášová, Ph.D. from KME ZČU in Pilsen.

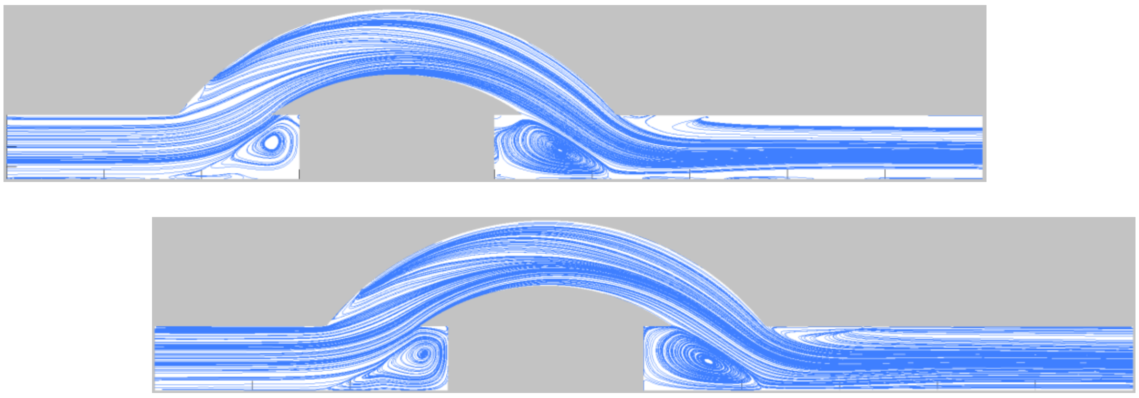
For the vector field RBF approximation we located the two critical points at the places of connection of the bypass. These two locations were used as points for RBF interpolation. Together with them were used 4 more surroundings points for each critical point. Using only these points is insufficient and thus we added additional 290 points for the RBF interpolation. These added points are located inside the new dataset, i.e. in the cut. For the RBF approximation we used

$$\varphi(r) = (1-r)_+^7(16r^2 + 7r + 1) \tag{4.23}$$

as the radial basis function. The results of our vector field RBF approximation method are in Figure 4.10 (left).

RBF approximation Original vector field

a) Simulation result



b) Simulation result

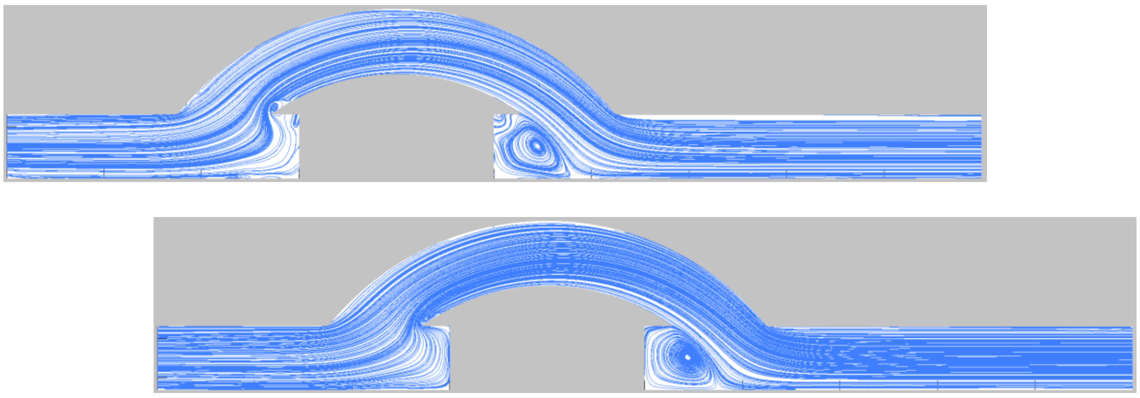


Figure 4.10: Phase portraits of vector fields as result of numerical simulation in different time steps. Phase portraits of vector fields RBF approximation (left) and phase portraits of vector fields created using linear interpolation on tetrahedral mesh (right). The RBF interpolation uses 300 reference points (2 critical points plus two times 4 neighbourhood points plus 290 randomly distributed points).

According to the results from Figure 4.10, it can be seen that the RBF approximation of a cut of vector field is very similar to the original vector field. However there are some issues on the border and especially on the corners of the dataset. To solve this problem we added more special points for the RBF interpolation on the edges and corners as well. The results of this vector field RBF approximation can be seen in Figure 4.11.

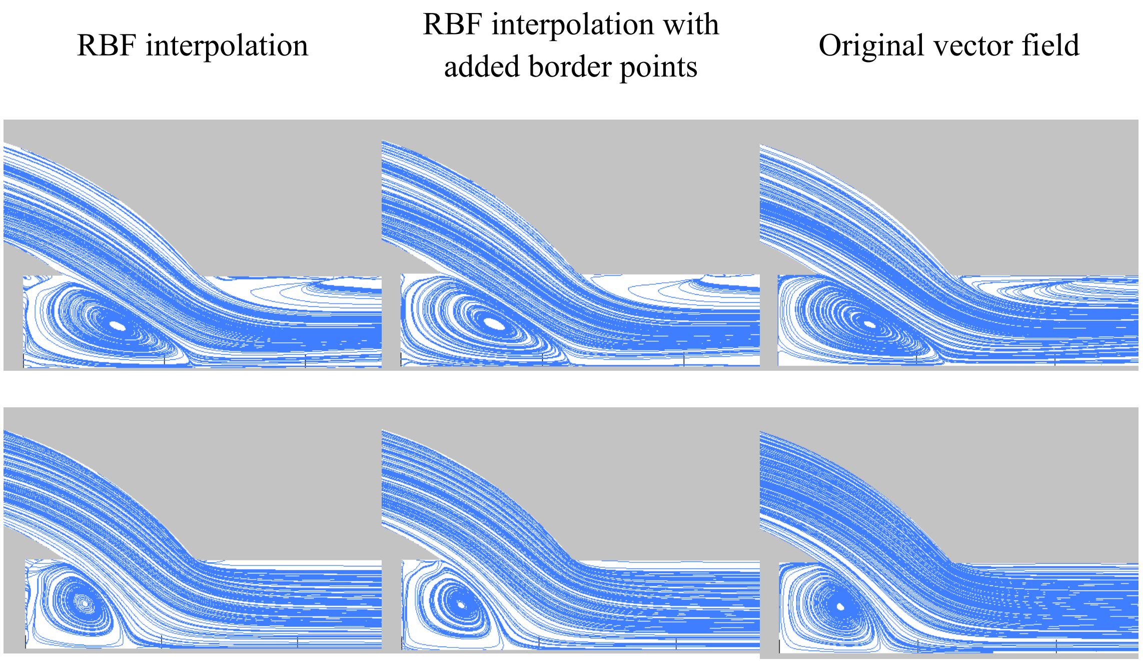


Figure 4.11: Phase portraits of vector fields as result of numerical simulation in different time steps. Phase portraits of vector fields RBF approximation (left), phase portraits of vector fields RBF approximation with added border points (middle) and phase portraits of vector fields created using linear interpolation on tetrahedral mesh (right). The RBF interpolation uses 300 reference points (2 critical points plus two times 4 neighbourhood points plus 290 randomly distributed points). The RBF interpolation with added border points uses 300 reference points (2 critical points plus two times 4 neighbourhood points plus 30 points distributed on the border plus 260 randomly distributed points).

According to the results in Figure 4.11, it can be seen that the RBF interpolation with added border points gives much better results than the RBF interpolation without extra added border points. Still, there are some problems with the RBF vector field approximation, especially close to the corners.

4.5 Vector field RBF interpolation on a sphere

The radial basis function interpolation can be computed on a sphere and has some advantages [Bax01], [Hub15]. We presents our approach published in [Smo16b]. There are any unphysical boundaries and there are no problems with interpolation on the poles, i.e. the sphere has no boundaries, and the vector field can be interpolated on the whole sphere surface at once compared to using only spherical coordinates and interpolation in $2D$. The other advantage is that there are no coordinate singularities and the maximal distance of any two points has an upper bound and the RBF interpolation does not need any mesh, i.e. triangulation, for interpolation.

The RBF interpolation interpolates scalar values on a sphere. However the vector field is not a scalar field, the RBF interpolation can be used for vector fields as well. For each component of the vector, we need to compute one RBF interpolation separately but it should be noted that the interpolation matrices for all component of the vector are the same.

The calculation of the distance r between two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ on a sphere can be computed as the Euclidian distance between this two points

$$\begin{aligned} r &= \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_{Euclidian} \\ &= \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \cdot (\boldsymbol{x}_1 - \boldsymbol{x}_2)}\,. \end{aligned} \tag{4.24}$$

In the case that both points lie on a unit sphere then $r \in \langle 0; 2\rangle$.

Or the distance can be computed as the shortest distance between two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ on the surface of a sphere, measured along the surface of the sphere. The distance is computed using

$$r = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_{spherical} = \cos^{-1}(\boldsymbol{n}_1 \cdot \boldsymbol{n}_2)\,, \tag{4.25}$$

where $r \in \langle 0; \pi\rangle$ and

$$\boldsymbol{n}_1 = \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|} \qquad \boldsymbol{n}_2 = \frac{\boldsymbol{x}_2}{\|\boldsymbol{x}_2\|} \tag{4.26}$$

The distance r in (4.25) is measured in radians. In the case that the sphere has radius equal to one, the computed distance in radians is equal to the distance measured along the surface of the sphere.

The RBF interpolation on a sphere is computed using the same formula as standard RBF. The only difference compared to standard equation for RBF interpolation is when computing the distance between two points as both mentioned approaches can be used.

4.5.1 Example of Vector Field on Sphere on Synthetic data

An example of a vector field on sphere can be described analytically. This analytical description must fulfil one criteria, which is that this function is continues all over the sphere. For this purpose we can use goniometric functions that have the period equal to 2π, i.e.

$$\sin\alpha = \sin(\alpha + k \cdot 2\pi) \qquad \cos\alpha = \cos(\alpha + k \cdot 2\pi)\,, \tag{4.27}$$

where k is an integer, i.e. $k \in \mathbb{Z}$.

The first example of a vector field on a sphere is described using the following equation

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sin 4\delta \\ \cos 4\theta \end{bmatrix} \qquad \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sin 3\delta + \cos 4\delta \cdot \cos 3\delta \\ \cos 4\theta - \sin 4\theta \cdot \sin 3\delta \end{bmatrix}. \tag{4.28}$$

where δ is an azimuth angle, i.e. $\delta \in (-\pi; \pi\rangle$ and θ is a zenith angle, i.e. $\theta \in \langle 0; \pi\rangle$. Data $[u, v]^T$ represents the direction vector on the surface of sphere at point $[P_x, P_y, P_z]^T$

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}^T = \begin{bmatrix} \sin\theta\cos\delta \\ \sin\theta\sin\delta \\ \cos\theta \end{bmatrix}^T \tag{4.29}$$

The vector fields (4.28) were discretized on uniformly distributed 10 000 points on a sphere and then interpolated using RBF on sphere with CSRBF with shape parameter equal to 1

$$\varphi(r) = (1 - r)_+^4(4r + 1)\,. \tag{4.30}$$

The interpolation, when using spherical distance (4.25) to compute the distance r for basis function $\varphi(r)$, can be seen in Figure 4.12. This visualization was created with ray-tracing and line integral convolution (LIC) on sphere.

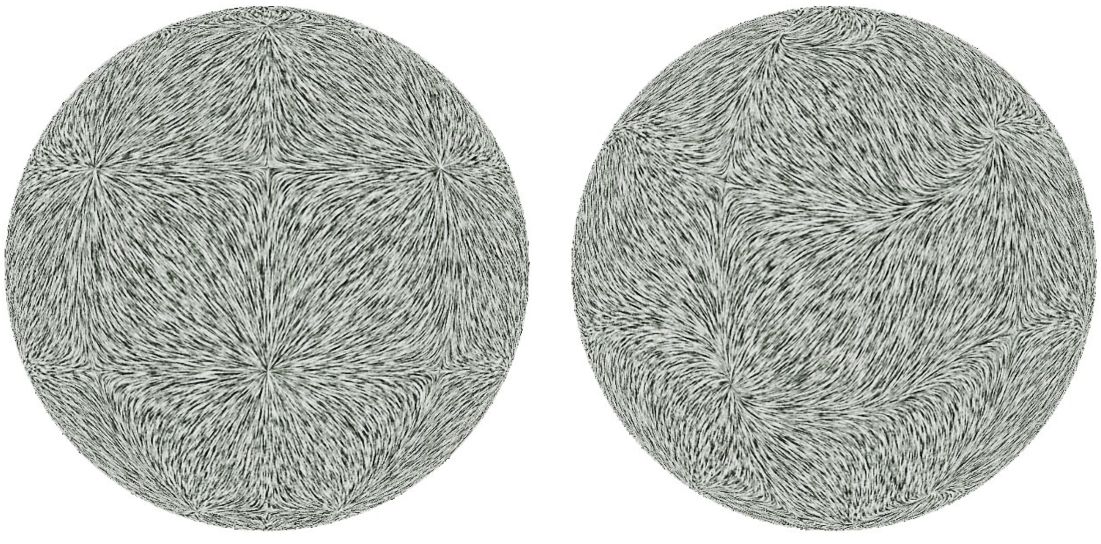


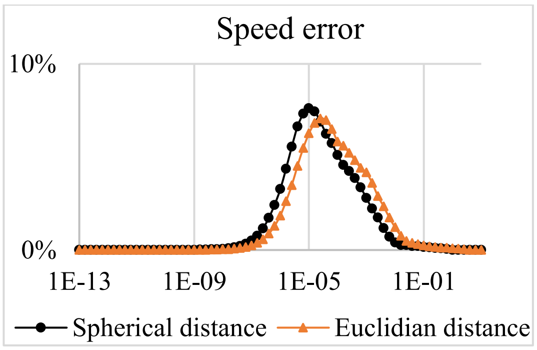
Figure 4.12: Visualization of examples of vector fields. All vector fields were interpolated using RBF and visualized as LIC images on sphere. Equation (4.28) left (a) and (4.28) right (b). In both images are clearly seen sources, resp. sinks, and saddles. Both images are visually identical to the ones with original analytical description.

To measure the quality of interpolation, we can compute the mean error of speed and the mean error of angular displacement of vectors. The mean errors were computed for 10^6 randomly generated positions on a sphere. The results for both equations (4.28) and both ways of calculating the distance between two points can be seen in Figure 4.12. Note that both vectors $[u, v]^T$ in (4.28) are computed in $[ms^{-1}]$.

Table 4.1: Errors of RBF interpolated vector fields (4.28) on a sphere for both ways of computing distance between two points.

		Speed error $[ms^{-1}]$	Angular displacement error $[rad]$
Euclidian distance	vector field (4.28) left	$2.452 \cdot 10^{-4}$	$4.233 \cdot 10^{-4}$
	vector field (4.28) right	$1.884 \cdot 10^{-3}$	$2.672 \cdot 10^{-3}$
Spherical distance	vector field (4.28) left	$1.686 \cdot 10^{-4}$	$3.074 \cdot 10^{-4}$
	vector field (4.28) right	$1.379 \cdot 10^{-3}$	$1.906 \cdot 10^{-3}$

It can be seen that the RBF interpolation when using spherical distance gives for both vector fields better results, i.e. more accurate speed and more accurate orientation at every location on the sphere in average, see Table 4.1. The RBF interpolation is less accurate for vector field (4.28) right) than for vector field (4.28) left). The reason is that the vector field (4.28) right) is significantly more complicated than (4.28) left). The distribution of speed errors and angular displacement errors is visualized in Figure 4.13. Histograms were created from 10^6 samples and data were grouped into 71 bins.



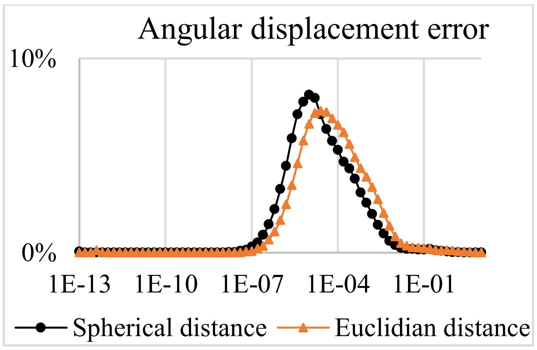


Figure 4.13: Histogram of speed error distribution (a) and displacement error distribution (b) for vector field (4.28) left.

The RBF interpolation performs slightly better interpolation results when using spherical distance (4.25) compared to the RBF with the Euclidian distance calculation (4.24). For this reason we use only the spherical distance calculation for all our tests.

We computed the RBF interpolation on a sphere of the original vector field (4.28) left using 10^3, $5 \cdot 10^3$ and 10^4 sampling points for different shape parameters in radial basis function

$$\varphi(r) = \begin{cases} (1 - \varepsilon r)_+^4 (4\varepsilon r + 1) & \varepsilon r \leq 1 \\ 0 & \varepsilon r > 1 \end{cases} \tag{4.31}$$

and measured the average vector length error and the average angular displacement error of interpolated vectors. The shape parameter ε cannot be less than $1/\pi$, as the CSRBF with $\varepsilon = 1/\pi$ covers the whole surface of a unit sphere and the CSRBF with shape parameter $\varepsilon > 1/\pi$ covers only a part of the sphere surface.

The results of the vector length error are visualized in Figure 4.14. It can be seen that the average error is almost identical for shape parameter $\varepsilon \in \langle 1/\pi\,;\ 4\rangle$ for $5 \cdot 10^3$ and 10^4 sampling points and for larger shape parameters the error increases. The vector length error for 10^3 sampling points is slightly higher than for $5 \cdot 10^3$ and 10^4 sampling points and starts distinctly increasing for shape parameter $\varepsilon > 2$.

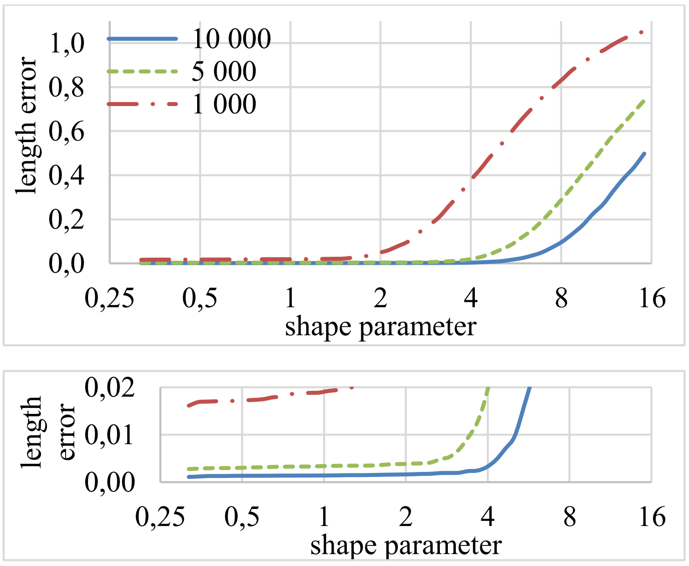


Figure 4.14: Average error of vector lengths of the RBF interpolation on a sphere for different shape parameters and different numbers of interpolated points.

The results of the average angular displacement error are visualized in Figure 4.15. The progress of the error is similar to Figure 4.14 and, thus, the quality of the vector field interpolation is almost identical for shape parameters $\varepsilon \in \langle 1/\pi\,;\ 4\rangle$ for $5 \cdot 10^3$ and 10^4 sampling points and for larger shape parameters the error increases. The angular displacement error for 10^3 sampling points is slightly higher than for $5 \cdot 10^3$ and 10^4 sampling points and starts distinctly increasing for shape parameter $\varepsilon > 2$.

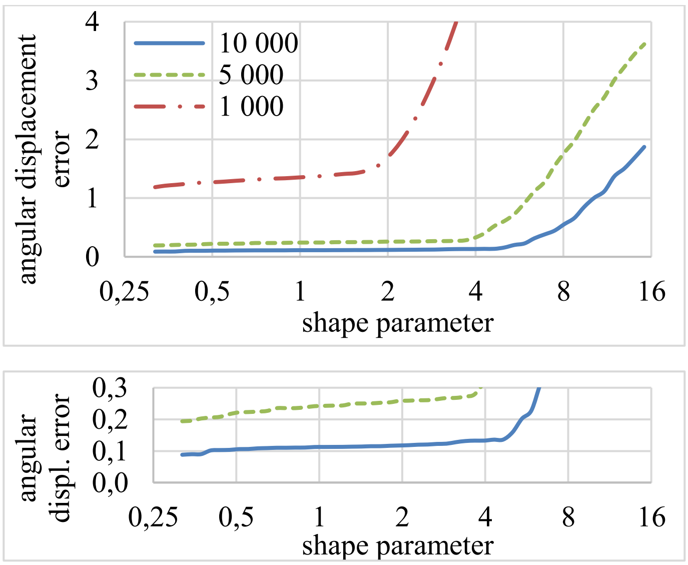


Figure 4.15: Average angular displacement error [°] of vectors of RBF interpolation on a sphere for different shape parameters and different numbers of interpolated points.

The CSRBF (4.31) is a "local" radial basis function, therefore, the RBF interpolation matrix is sparse. We varied the shape parameter and measured the occupancy of the interpolation matrix. The results can be seen in Figure 4.16. When the shape parameter is $\varepsilon > 2/\pi$ then more than half of the elements in the RBF interpolation matrix are equal zero.

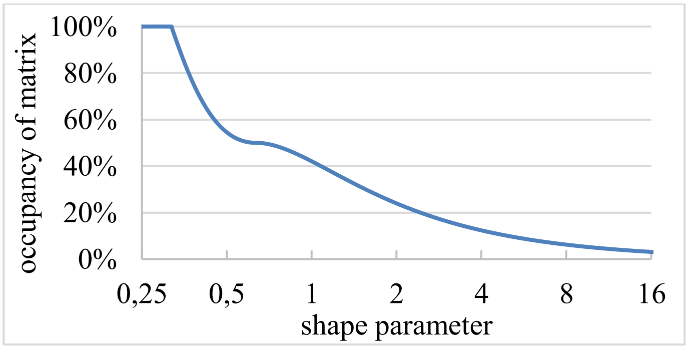


Figure 4.16: Occupancy of the interpolation matrix for the RBF interpolation on a unit sphere for different shape parameters.

The RBF interpolation matrix has different condition numbers for different shape parameters because the occupancy of the matrix changes for different shape parameters. The condition number of this matrix is visualized in Figure 4.17 and it can be seen that the matrix is better conditioned with increasing shape parameter. It is justified by the fact that the occupancy of the RBF interpolation matrix decreases for increasing shape parameter.

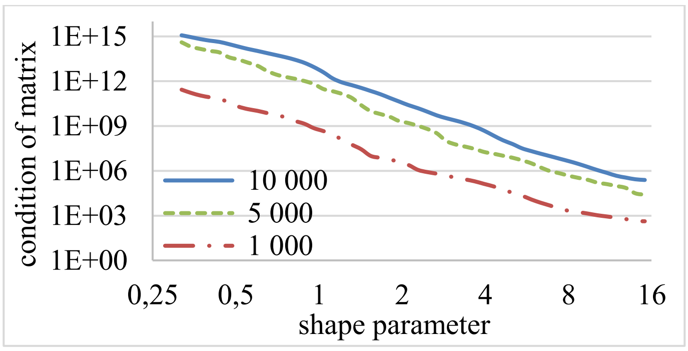


Figure 4.17: Condition of the RBF interpolation matrix for different shape parameters and different numbers of interpolated points.

Using the results from Figure 4.14-14, we can choose the best shape parameter to be $\varepsilon = 4$ for $5 \cdot 10^3$ and 10^4 sampling points and $\varepsilon = 2$ for 10^3 sampling points. For this parameters the interpolation errors are the smallest, the RBF interpolation matrix is sparse and has a rather small condition number. For $\varepsilon > 4$, resp. $\varepsilon > 2$, will increase both interpolation errors and for $\varepsilon < 4$, resp. $\varepsilon > 2$, will increase the occupancy and the condition number of RBF interpolation matrix.

The important property of the interpolated vector field is that for all shape parameters $\varepsilon < 4$ it preserves the type of all critical points in the vector field (4.28) left. And the location of all critical points in the interpolated vector field is almost identical to the locations of the critical points in the vector field (4.28) left. Thus the RBF interpolated vector field has the same topology as the vector field (4.28) left.

4.5.2 Real Example of Vector Field on Sphere on Experiment Data

Numerical forecasts can predict weather as well as wind velocity and direction. We used one such prediction of wind vector field for the whole word[1]. This data contains information about wind speed and wind direction every one degree in latitude and longitude. Therefore the resolution of numerically computed dataset is 360×180, which is 64 800 vectors in total.

We did some reduction of this dataset, as for the North or South Pole is needed only one vector and for locations near these two poles, we can reduce the computed vectors as well. After the reduction, we ended up with 62 742 vectors. This wind data were interpolated using RBF with CSRBF (4.30) and shape parameter $\varepsilon = 1$. The RBF interpolation was used to create the visualization of wind vector field on the sphere, see Figure 4.18.

[1] US GFS global weather model. National Centers for Environmental Information (NCEI), https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs, [downloaded: 21.3.2016].

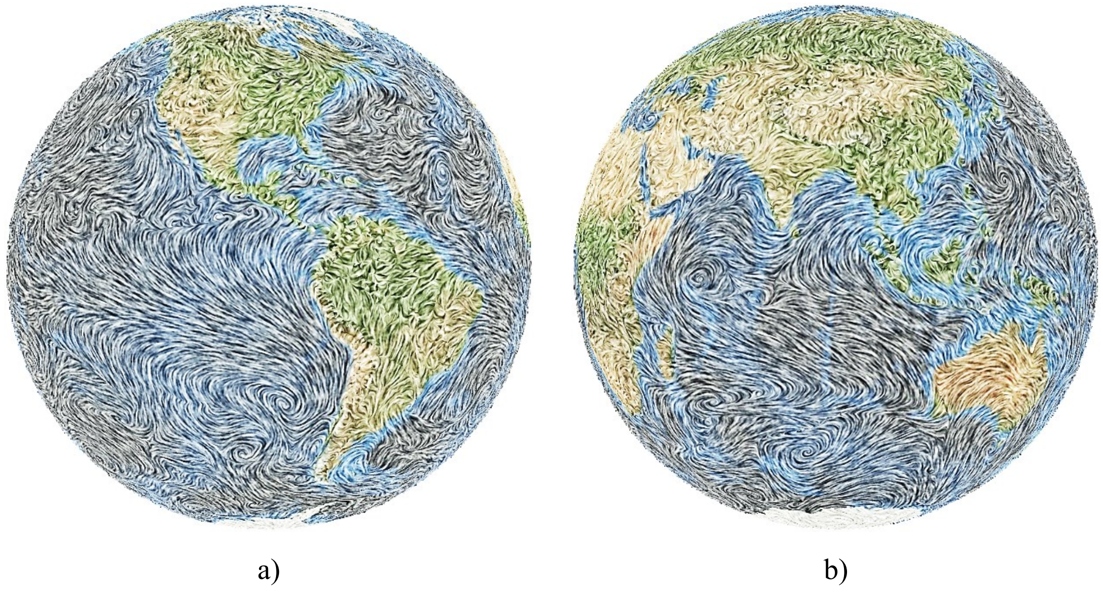


Figure 4.18: Visualization of RBF interpolated wind vector field from numerical simulation.

5 PDE and meshless technique

Meshless methods are uniquely simple, yet provide solution accuracies for certain classes of equations that rival those of finite elements and boundary elements without requiring the need for mesh connectivity. Ease in programming, no domain or surface discretization, no numerical integration, and similar formulations for $2D$, $3D$ and higher dimensional cases make these methods very attractive.

Although many numerical and analytical schemes exist for solving engineering problems, the meshless method is a particularly attractive method that is receiving attention in the engineering and scientific modelling communities. Finite difference (FDM), finite volume (FVM), and finite element (FEM) methods have been historically used to model a wide variety of engineering problems in complex geometries that may require extensive meshing. The meshless method is simple, accurate, and requires no meshing.

The need to accurately simulate various physical processes in complex geometries is important. Today, advances in numerical schemes and enhanced hardware have lead to many commercial codes that can solve complex stress-strain, heat transfer, fluid flow, and other problems. Recently, advances in the development and application of meshless techniques show they can be strong competitors to the more classical finite difference/volume and finite element approaches. Textbooks by Liu [Liu09] and Fasshauer [Fas07] discuss meshless methods, implementation, algorithms, and coding issues for stress-strain problems.

There exist several types of meshless methods. The more common techniques include kernel methods, moving least square method, meshless Petrov-Galerkin, partition of unity methods, smooth-particle hydrodynamics, and radial basis functions. Each technique has particular traits and advantages for specific classes of problems. Generally the simplest and easiest to implement is the radial basis function approach.

5.1 Derivation of Radial Basis Function

To compute PDEs using RBF meshless technique we need to know how to compute the derivation of radial basis function. A radial basis function of two variables, i.e.

$$\varphi(r) = \varphi(\|\boldsymbol{x}\|) = \varphi\left(\sqrt{x^2 + y^2}\right), \tag{5.1}$$

where $\boldsymbol{x} = [x, y]^T$. The chain rule implies

$$\begin{aligned}\frac{\partial}{\partial x}\varphi(\|\boldsymbol{x}\|) &= \frac{d}{dr}\varphi(r)\frac{\partial}{\partial x}r(x,y) \\ &= \frac{d}{dr}\varphi(r)\frac{x}{\sqrt{x^2+y^2}} \\ &= \frac{x}{r}\frac{d}{dr}\varphi(r)\end{aligned} \tag{5.2}$$

since

$$r = \|\boldsymbol{x}\| = \sqrt{x^2+y^2}\,. \tag{5.3}$$

Similarly

$$\frac{\partial}{\partial y}\varphi(\|\boldsymbol{x}\|) = \frac{y}{r}\frac{d}{dr}\varphi(r)\,. \tag{5.4}$$

The second order derivatives are given by

$$\begin{aligned}\frac{\partial^2}{\partial x^2}\varphi(\|\boldsymbol{x}\|) &= \frac{d^2}{dr^2}\varphi(r)\left(\frac{\partial}{\partial x}r(x,y)\right)^2 + \frac{d}{dr}\varphi(r)\frac{\partial^2}{\partial x^2}r(x,y) \\ &= \frac{x^2}{r^2}\frac{d^2}{dr^2}\varphi(r) + \frac{y^2}{r^3}\frac{d}{dr}\varphi(r)\,,\end{aligned} \tag{5.5}$$

as well as

$$\frac{\partial^2}{\partial y^2}\varphi(\|\boldsymbol{x}\|) = \frac{y^2}{r^2}\frac{d^2}{dr^2}\varphi(r) + \frac{x^2}{r^3}\frac{d}{dr}\varphi(r) \tag{5.6}$$

and

$$\frac{\partial^2}{\partial x\partial y}\varphi(\|\boldsymbol{x}\|) = \frac{xy}{r^2}\frac{d^2}{dr^2}\varphi(r) + \frac{xy}{r^3}\frac{d}{dr}\varphi(r)\,. \tag{5.7}$$

The derivatives of the basic function with respect to r, i.e. $\varphi(r)$, need to be replaced with the actual derivative. For example for globally supported Gaussian RBF

$$\varphi(r) = e^{-(\varepsilon r)^2} \tag{5.8}$$

is the derivative equal to

$$\frac{\partial}{\partial r}\varphi(r) = -2\varepsilon^2 r e^{-(\varepsilon r)^2} \tag{5.9}$$

$$\frac{\partial^2}{\partial r^2}\varphi(r) = 2\varepsilon^2 e^{-(\varepsilon r)^2}(2(\varepsilon r)^2 - 1) \tag{5.10}$$

and this function is C^∞ at the origin. For another example a compactly supported Wendland's $\varphi_{3,2}$ RBF

$$\varphi(r) = (1-\varepsilon r)_+^6(35(\varepsilon r)^2 + 18\varepsilon r + 3) \tag{5.11}$$

is the derivative equal to

$$\frac{\partial}{\partial r}\varphi(r) = -56\varepsilon^2 r(5\varepsilon r + 1)(1-\varepsilon r)_+^5 \tag{5.12}$$

$$\frac{\partial^2}{\partial r^2}\varphi(r) = 56\varepsilon^2(35(\varepsilon r)^2 - 4\varepsilon r - 1)(1-\varepsilon r)_+^4 \tag{5.13}$$

and this function is C^4 at the origin.

5.2 Kansa's collocation method

Kansa [Kan90a], [Kan90b] proposed a non-symmetric method for the solution of elliptic PDEs with radial basis functions. The collocation method is computed on some given domain $\Omega \subset \mathbb{R}^d$ and solves a linear elliptic partial differential equation of the form

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \text{ in } \Omega, \tag{5.14}$$

where $\mathcal{L}$ is a differential operator and we use Dirichlet boundary conditions

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \text{ on } \Omega. \tag{5.15}$$

Kansa's collocation method chooses to represent the approximate solution $\hat{u}$ by a radial basis function that is analogous to the one used for scattered data interpolation, i.e.

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j\, \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|), \tag{5.16}$$

where λ_j are coefficients of radial basis functions and $\boldsymbol{\xi}_j$ are centres of radial basis functions. The collocation matrix $\boldsymbol{A}$ that arises from equation (5.14) and the boundary condition (5.15) at the collocation points has the form

$$\boldsymbol{A} = \begin{bmatrix} \widetilde{\boldsymbol{A}}_{\mathcal{L}} \\ \widetilde{\boldsymbol{A}} \end{bmatrix}, \tag{5.17}$$

where two blocks $\tilde{A}_{\mathcal{L}}$ and $\tilde{A}$ are generated as following

$$\begin{aligned} (\widetilde{\boldsymbol{A}}_{\mathcal{L}})_{ij} &= \mathcal{L}\varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), \quad \boldsymbol{x}_i \in \zeta \\ (\widetilde{\boldsymbol{A}})_{ij} &= \varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), \quad \boldsymbol{x}_i \in \psi, \end{aligned} \tag{5.18}$$

where $\boldsymbol{x}_i \in \zeta$ are interior points and $\boldsymbol{x}_i \in \psi$ are boundary points. The problem is well-posed if the linear system $\boldsymbol{A\lambda} = \boldsymbol{h}$, with $\lambda = [\lambda_1, \dots, \lambda_N]^T$ and $\boldsymbol{h}$ a vector consisting of entries $f(\boldsymbol{x}_i)$ followed by $g(\boldsymbol{x}_i)$ has a unique solution.

The change of boundary conditions (5.15) changes only corresponding rows of the matrix $\boldsymbol{A}$ in (5.17) and the corresponding numbers in the right-hand side $\boldsymbol{h}$.

The above described method is rather a general description of a numerical method with no particular RBF in mind. Kansa proposed to use multiquadric radial basis function (5.19) in (5.16) and sometimes this method is called as multiquadric method.

$$\varphi(r) = \sqrt{1 + (\varepsilon r)^2} \tag{5.19}$$

Kansa proposed to use multiquadrics (5.19) with varying shape parameter ε_j. For a constant shape parameter ε the matrix $\boldsymbol{A}$ may be singular for some configuration of centres of radial basis function $\boldsymbol{\xi}_j$. This problem can appear as the matrix for collocation problem is composed of rows that are created from different functions, which depending on the differential operator $\mathcal{L}$, might not even be radial anymore.

Later Moridis and Kansa [Mor94] suggested how Kansa's collocation method can be used for other types of differential equation problems such as non-linear elliptic PDEs, systems of elliptic PDEs and time-dependent parabolic or hyperbolic PDEs.

5.3 Galerkin approximations

Galerkin method [Atl02] is a mathematical method used to obtain approximate solutions of partial differential equations which contain terms with odd order, like the following equation

$$\frac{du}{dx} + \frac{d^2u}{dx^2} = f(x)\,, \quad x \in (0, R) \tag{5.20}$$

with boundary conditions

$$\begin{aligned} u(0) &= u_0 \\ \left.\frac{du}{dx}\right|_{x=R} &= u'_R\,, \end{aligned} \tag{5.21}$$

where $u(x)$ is the unknown function to be approximated using Galerkin method. Multiplying equation (5.20) by a test function $v(x)$ and integrating by parts we obtain the weak formulation of (5.20)

$$\int_0^R \left(v\frac{du}{dx}\right) dx - \int_0^R \left(\frac{dv}{dx}\frac{du}{dx}\right) dx + \left[v\frac{du}{dx}\right]_0^R = \int_0^R f(x) v\, dx \qquad \forall v \in H_0^1(0, R)\,, \tag{5.22}$$

wher$H_0^1(0, R)$e is the Sobolev space

$$H_0^1(0, R) = \left\{v \in L^2(0, R) : \frac{dv}{dx} \in L^2(0, R) \text{ and } v(0) = v(R) = 0\right\}, \tag{5.23}$$

If u is regular (for example with two continuous derivatives) then problems (5.20) and (5.21) are equivalent. We can use (5.21) in order to define an approximation to u. We are going to construct polygonal approximations to u. With this purpose let us introduce a uniform partition of the domain $(0, R)$ into $N+1$ subintervals $(x_j\,, x_{j+1})$ with

$$x_j = \frac{jR}{N+1} \quad \text{for} \quad j = 0, \dots, N+1 \tag{5.24}$$

and consider the space V_N of polygonal functions vanishing at the boundary of $(0, R)$, i.e.,

$$V_N = \left\{ v \in C^0 \,:\, v|_{(x_j, x_{j+1})} \text{ is linear and } v(0) = v(R) = 0 \right\}, \tag{5.25}$$

where C_0 denotes the space of continuous functions.

Observe that, $\forall N$, V_N is a subspace of $H_0^1(0, R)$ and that V_N has finite dimension. Indeed, a polygonal function $v \in V_N$ is uniquely determined by its values at the finite number of points $x_1, \dots, x_N$.

We define the Galerkin approximation $u_N \in V_N$ to u by imposing (5.22) but only for functions $\in V_N$, i.e., $u_N \in V_N$ is such that:

$$\int_0^R \left(v \frac{du_N}{dx} \right) dx - \int_0^R \left(\frac{dv}{dx} \frac{du_N}{dx} \right) dx + \left[v \frac{du_N}{dx} \right]_0^R = \int_0^R f(x) v \, dx \qquad \forall v \in V_N \,. \tag{5.26}$$

We are going to see that there is a unique u_N satisfying (5.26) and moreover that it can be computed by solving a linear system of equations because V_N is finite dimensional. Given a basis ϕ_j of V_N, u_N can be written as

$$u_N = \sum_{j=1}^{N} U_j \phi_j \;, U_j \in \mathbb{R}\,. \tag{5.27}$$

Now, since any $v \in V_N$ is a linear combination of the ϕ_j, we can rewrite (5.26) as the following equation

$$\int_0^R \left(\phi_k \frac{du_N}{dx} \right) dx - \int_0^R \left(\frac{d\phi_k}{dx} \frac{du_N}{dx} \right) dx + \left[\phi_k \frac{du_N}{dx} \right]_0^R = \int_0^R f(x) \phi_k \, dx \quad \text{for } k = 1, \dots, N \tag{5.28}$$

and using (5.27) we can rewrite (5.28) as

$$\sum_{j=1}^{N} U_j \left(\int_0^R \left(\phi_k \frac{d\phi_j}{dx} - \left(\frac{d\phi_k}{dx} \frac{d\phi_j}{dx} \right) \right) dx + \left[\phi_k \frac{d\phi_j}{dx} \right]_0^R \right) = \int_0^R f(x) \phi_k \, dx \quad \text{for } k = 1, \dots, N \tag{5.29}$$

Therefore we can find $\boldsymbol{U} = [U_1, \dots, U_N]^T$ by solving the system of linear equations

$$\boldsymbol{AU} = \boldsymbol{F}\,, \tag{5.30}$$

where matrix $\boldsymbol{A}$ is a matrix of size $N \times N$ with elements

$$A_{kj} = \int_0^R \left(\phi_k \frac{d\phi_j}{dx} - \left(\frac{d\phi_k}{dx} \frac{d\phi_j}{dx} \right) \right) dx + \left[\phi_k \frac{d\phi_j}{dx} \right]_0^R \tag{5.31}$$

and vector $\boldsymbol{F}$ has elements

$$F_k = \int_0^R f(x) \phi_k \, dx \tag{5.32}$$

Solving the system of linear equations (5.30) we can then compute the final solution of function u_n using (5.27).

5.4 Hermite-based method

[Har90], [Wu92] and [Kan97] proposed a method for the solution of elliptic PDEs with radial basis functions. The collocation method is computed on some given domain $\Omega \subset \mathbb{R}^d$ and solves a linear elliptic partial differential equation of the form

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x})\,, \quad \boldsymbol{x} \text{ in } \Omega\,, \tag{5.33}$$

where $\mathcal{L}$ is a differential operator and we use Dirichlet boundary conditions

$$u(\boldsymbol{x}) = g(\boldsymbol{x})\,, \quad \boldsymbol{x} \text{ on } \Omega\,. \tag{5.34}$$

In order to be able to apply the results from generalized Hermite interpolation that will ensure the non-singularity of collocation matrix we can use the following expansion for the unknown function u

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N_\zeta} \lambda_j \mathcal{L}^\xi \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) \bigg|_{\xi = \xi_j} + \sum_{j=N_\zeta+1}^{N} \lambda_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|), \tag{5.35}$$

where λ_j are coefficients of radial basis functions, $\boldsymbol{\xi}_j$ are centres of radial basis functions, N_ζ denotes the number of nodes in the interior of Ω, and $\mathcal{L}^\xi$ is the differential operator used in the differential equation (5.33), but acting on φ viewed as a function of the second argument, i.e . $\mathcal{L}\varphi$ is equal to $\mathcal{L}^\xi\varphi$ up to a possible difference in sign ($|\mathcal{L}\varphi| = |\mathcal{L}^\xi\varphi|$).

After enforcing the collocation conditions

$$\begin{aligned} \mathcal{L}\hat{u}(\boldsymbol{x}_i) &= f(\boldsymbol{x}_i) \qquad \boldsymbol{x}_i \in \zeta \\ \hat{u}(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i) \qquad \boldsymbol{x}_i \in \psi\,, \end{aligned} \tag{5.36}$$

where $\boldsymbol{x}_i \in \zeta$ are interior points and $\boldsymbol{x}_i \in \psi$ are boundary points, we end up with collocation matrix $\boldsymbol{A}$ that is of the form

$$A = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^{\xi}} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^{\xi}} & \hat{A} \end{bmatrix}. \tag{5.37}$$

The four blocks in (5.37) are generated as follows

$$\begin{aligned} \left(\hat{A}_{\mathcal{L}\mathcal{L}^{\xi}}\right)_{ij} &= \mathcal{L}\mathcal{L}^{\xi}\varphi(\|\boldsymbol{x}-\boldsymbol{\xi}\|)\big|_{x=x_i,\xi=\xi_j} && \boldsymbol{x}_i, \boldsymbol{\xi}_j \in \zeta \\ \left(\hat{A}_{\mathcal{L}}\right)_{ij} &= \mathcal{L}\varphi\left(\|\boldsymbol{x}-\boldsymbol{\xi}_j\|\right)\big|_{x=x_i} && \boldsymbol{x}_i \in \zeta, \boldsymbol{\xi}_j \in \psi \\ \left(\hat{A}_{\mathcal{L}^{\xi}}\right)_{ij} &= \mathcal{L}^{\xi}\varphi(\|\boldsymbol{x}_i-\boldsymbol{\xi}\|)\big|_{\xi=\xi_j} && \boldsymbol{x}_i \in \psi, \boldsymbol{\xi}_j \in \zeta \\ \left(\hat{A}\right)_{ij} &= \varphi\left(\|\boldsymbol{x}_i-\boldsymbol{\xi}_j\|\right) && \boldsymbol{x}_i, \boldsymbol{\xi}_j \in \psi \end{aligned} \tag{5.38}$$

The matrix $\boldsymbol{A}$ is non-singular as long as φ is chosen appropriately and the collocation approach for $\hat{u}$ is well-posed. The matrix $\boldsymbol{A}$ is symmetrical and although $\boldsymbol{A}$ consist of four blocks, it still is the same size, namely $N \times N$, as the collocation matrix obtained for Kansa's approach.

5.5 Problems with PDE and meshless technique

When computing partial differential equations using meshless techniques, some problems can arise.

5.5.1 Hyper-viscosity

According to [For11], the RBF differentiation matrices for linear advection typically have some eigenvalues with positive real part, making time-integration unstable with standard ODE methods. A commonly used solution for this type of instability in finite difference methods is artificial viscosity, which acts to damp the growing eigenmodes and thus stabilizing the method. This is typically implemented by adding a viscous term to the right-hand side of the PDE

$$\frac{\partial u}{\partial t} = \mathcal{L}(u) + \varepsilon \Delta u\,, \tag{5.39}$$

where

$$\frac{\partial u}{\partial t} = \mathcal{L}(u) \tag{5.40}$$

is the original problem and ε is a scaling parameter. Optimally, the viscosity should leave the resolved, correctly advected modes intact. For high order methods which provide a larger range of resolved modes, a more selective damping is necessary. This suggests the use of a higher order viscosity operator, commonly known as hyper-viscosity.

Figure 5.1 shows the effect of hyper-viscosity for linear advection. In this example, an example of cosine bell is advected using the RBF finite difference method with and without added hyper-viscosity.

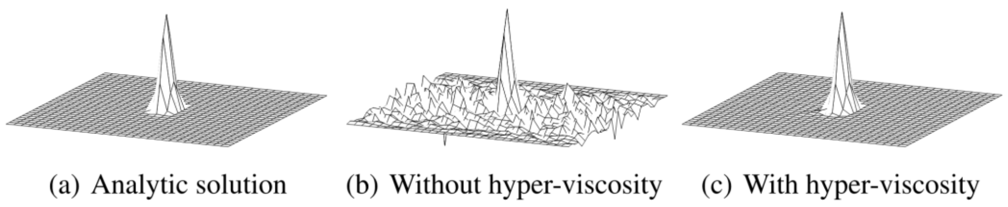


Figure 5.1: Example of the result of an advected cosine bell. (from [Leh12]).

5.5.2 Shape parameter of RBF

Many RBFs are defined by a constant called the shape parameter. The choice of basis function and shape parameter have a significant impact on the accuracy of an RBF method. Locating an optimal shape parameter is a difficult problem and a topic of current research.

Many RBFs, including all of the ones studied here, have a variable ε in their definitions. This variable ε is called the shape parameter. For the RBF definitions listed in Table 1, a smaller shape parameter corresponds to a "flatter" or "wider" basis function. The limit as $\varepsilon \rightarrow 0$ is often referred to as the "flat" limit, because $\varepsilon = 0$ corresponds to a constant basis function.

Figure 5.2 shows three Gaussian RBFs with different shape parameters. The RBFs are graphed with two-dimensional inputs on the same domain. Changing the shape parameter of an RBF alters the interpolant, can have a significant impact on the accuracy of the approximation and moreover the shape parameter has very high influence on the stability of PDE system that is solved using meshless techniques with RBF.

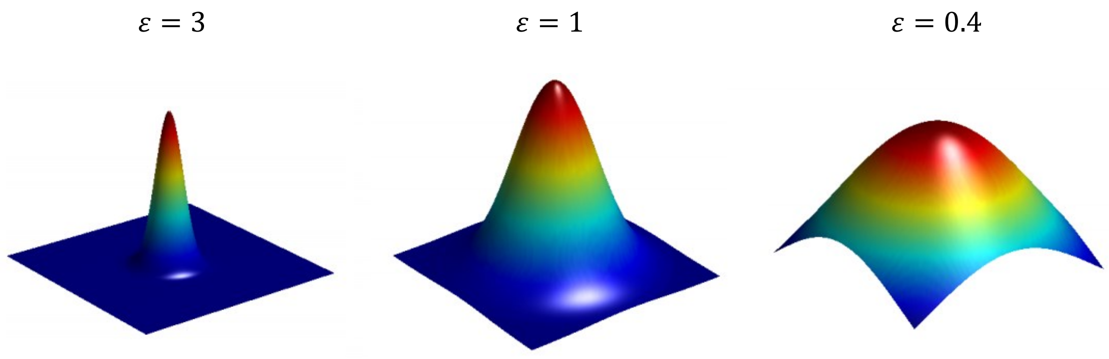


Figure 5.2: Gaussian RBFs with different shape parameters plotted on the same domain.

Most of the RBFs used to approximate the solution of partial differential equation contain a shape parameter ε which must be specified by the user. This random selection of ε is a disadvantage. A number of papers have been written on choosing optimal value of RBFs shape parameter. For example [Har71] suggested the use of shape parameter

$$\varepsilon = 0.815 \frac{\sum_{i=1}^{N} d_i}{N}, \tag{5.41}$$

where d_i is the distance from the data point x_i to its nearest neighbor. [Fra82] suggested to use

$$\varepsilon = 1.25 \frac{D}{\sqrt{N}}, \tag{5.42}$$

where D is the diameter of the minimal circle enclosing all data points. [Rip99] proposed an algorithm for choosing an optimal value of RBFs shape parameter. [Fas07a] suggested an algorithm for choosing optimal value of RBF shape parameter for iterated moving least squares approximation and for RBF pseudo-spectral methods for the solution of partial differential equations. Recently [Sch11] proposed another procedure for selecting good value of ε in RBF-interpolation. More recently [Bay11] proposed an algorithm for selecting an optimal value of multiquadric shape parameter ε in RBF-FD method.

6 Visualization of vector fields

One of the goals of scientific visualization is to display measurements of physical quantities so the underlying physical phenomena can be interpreted accurately, quickly, and without bias. Great care is taken in choosing where such measurements will be made so that inferences about the underlying phenomena will be correct. Many people have addressed, with qualitative or anecdotal advice, how best to design visualizations [Lai05] or how to compute them fast on GPU [Bur07].

Some examples of visualization methods can be seen in Figure 6.1. The shortcut GRID represents icons on a regular grid, i.e. Hedgehogs visualization technique. JIT represents icons on a jittered grid, LIT are icons using one layer of a visualization method that borrows concepts from oil painting [Kir99], LIC is shortcut for line integral convolution [Cab93], OSTR are image-guided streamlets, i.e. integral curves and GSTR are streamlets seeded on a regular grid.

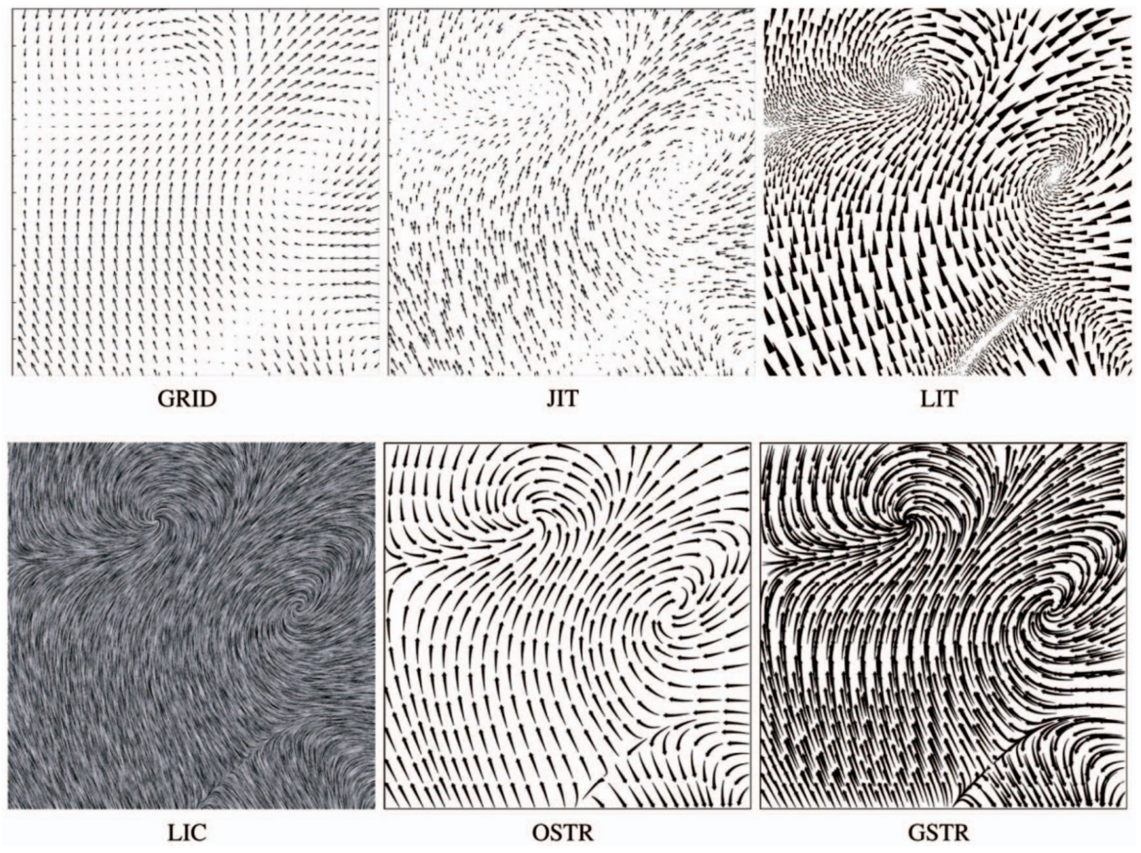


Figure 6.1: Example of some vector field visualization methods. (from [Lai05]).

Visualization techniques can be divided into two groups. One are global techniques and the second one are local techniques. Both of them will be described in the following chapters.

6.1 Global techniques

Global techniques of vector field visualization methods visualize the whole vector field in one picture. The user have to minimally interact with the visualization system, i.e. only very small interaction is required.

6.1.1 Hedgehogs

The less dimensions a dynamical system has, the easier visualization is. Techniques for the visualization of two-dimensional dynamical systems (or vector fields) already have quite a tradition in flow investigation. Hedgehog plots, also called arrow plots, usually show a large number of small arrows that indicate the flow direction at many (regularly spaced) points of the two-dimensional domain. Often arrows are normalized, so flow velocity is not encoded. This is, to prevent the display from overloading due to very long and overlapping arrows, see Figure 6.2 for an example.

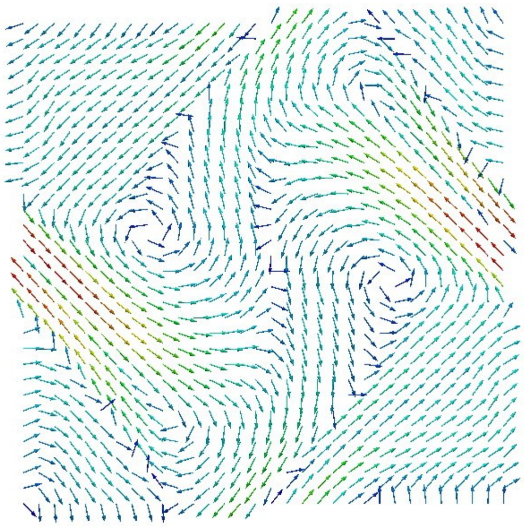


Figure 6.2: Example of vector flow visualized with hedgehogs method. Arrows show the direction of vector field and color represents the velocity.

6.1.2 Line Integral Convolution

Line Integral Convolution (LIC) is a powerful technique for generating striking images and animations from vector data. [Cab93] presented a powerful technique for imaging vector data called line integral convolution. This algorithm has been used as a general tool for visualizing vector fields. The method has rapidly found many application areas, ranging from computer arts to scientific visualization. Based upon locally filtering an input texture along a curved stream line segment in a vector field, it is able to depict directional information at high spatial resolutions.

Each data point in a flow field (apart from critical points) lies on a unique path or streamline. Ideally we can set the color or intensity of pixels in the output image such that pixels along a common path have a common color, while those on adjacent paths would have their own color. Line integral convolution is thus calculated using the formula

$$I(\boldsymbol{x}) = \int_{t_0-L}^{t_0+L} k(t-t_0) \cdot T(\boldsymbol{r}(t))dt\,, \tag{6.1}$$

where $I(\boldsymbol{x})$ is the output intensity of pixel at position $\boldsymbol{x}$, $T(\boldsymbol{x})$ is the intensity of noise at position $\boldsymbol{x}$, $\boldsymbol{r}(t)$ is the curve representing a streamline for parameter $t \in \langle t_0 - L, t_0 + L\rangle$ and $k(t)$ is the kernel of the filter. The kernel can be linear as the simplest solution or can have any other form. Using LIC we can easily understand the topology of the vector field, see Figure 6.3 and Figure 6.4.

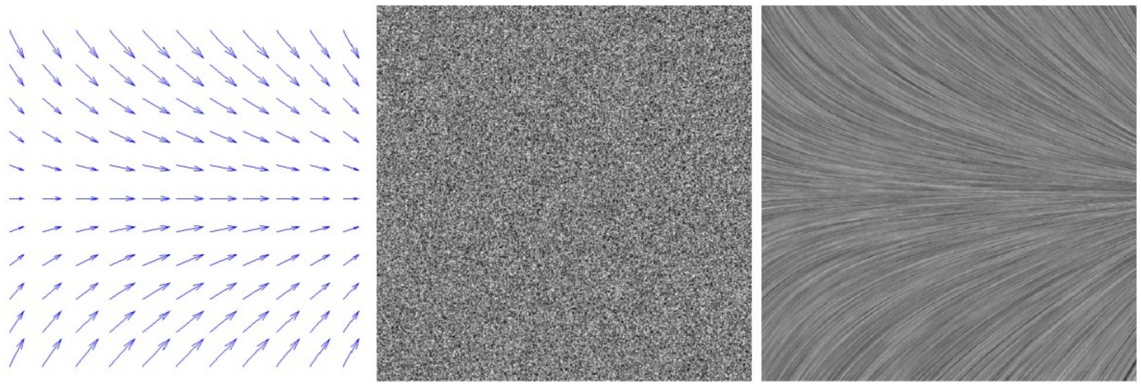


Figure 6.3: Hedgehog plot and noise texture as an input and LIC as an output of the Line integral convolution method.

The LIC algorithm is designed as a function which maps an input vector field and texture to a filtered version of the input texture. The dimension of the output texture is that of the vector field. Both the texture and the vector field can be pre-processed and combined with post processing on the output image, see Figure 6.4.

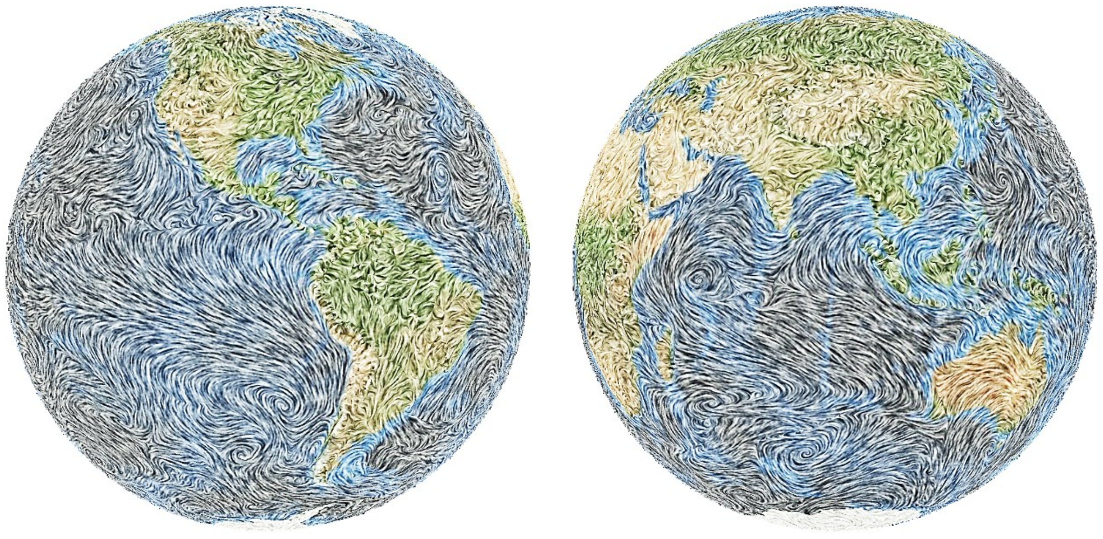


Figure 6.4: Example of RBF interpolated wind flow over the surface of earth, which is visualized with LIC method.

6.1.3 Method using concepts from painting

[Kir99] presents a new visualization method for $2D$ flows, which allows combining multiple data values in an image for simultaneous viewing. It utilizes concepts from oil painting, art, and design to examine problems within fluid mechanics. The method uses a combination of discrete and continuous visual elements arranged in multiple layers to visually represent the data. The representations are inspired by the brush strokes artists apply in layers to create an oil painting. Visualization displays commonly visualized quantities such as velocity and vorticity together with three additional mathematically derived quantities: the rate of strain tensor, the turbulent charge and turbulent current.

Example of vector field visualization can be seen in Figure 6.5. The velocity is represented by arrow direction, speed is represented by arrow area, vorticity is represented by underpainting color (blue for clockwise and yellow for counter-clockwise), rate of strain is represented by logarithms of ellipse radius, divergence is represented by ellipse area and shear is represented by ellipse eccentricity.

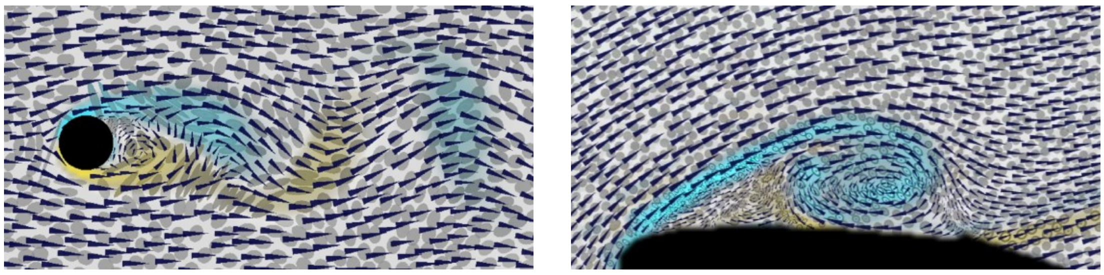


Figure 6.5: Visualization of simulated $2D$ flow past a cylinder at Reynolds number $= 100$ (left) and experimental $2D$ flow past an airfoil (right), (from [Kir99]).

6.1.4 Texture splats

A popular technique for volume rendering is known as splatting. It uses two textures, i.e. splats, for rendering the vector field. Figure 6.6 illustrates a series of such splats. For the splatting of vector splats, two additional calculations must be carried out. First, the vector field direction for each splat is determined and transformed to viewing coordinates. The projection of this vector is then used to determine a rotation matrix for the polygon splat. The splat is rendered by selecting a splat from the splat table in Figure 6.6 (left) as an intensity map, and for scalar fields, the same splat in Figure 6.6 (right) for an opacity map.

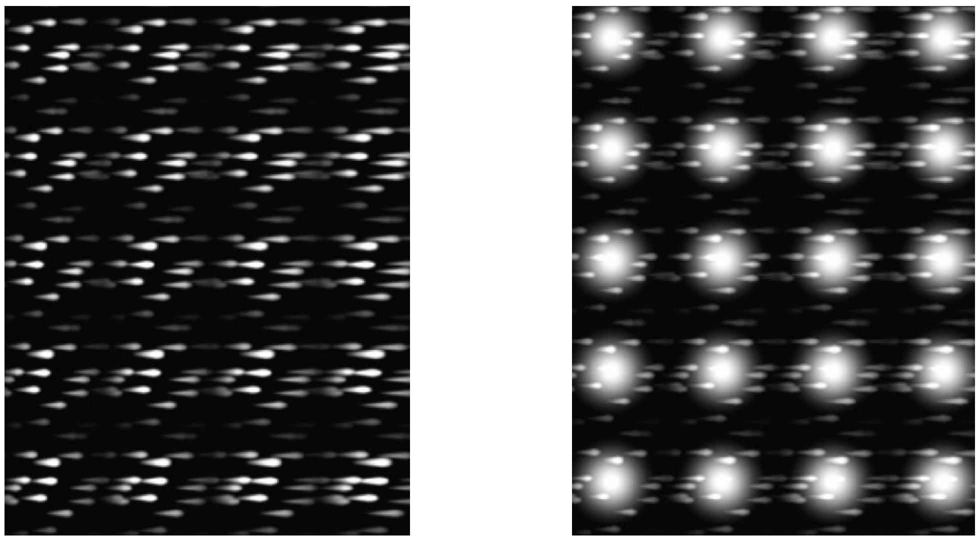


Figure 6.6: Portion of intensity table (left) and portion of opacity table (right) (from [Cra93]).

Example visualization of vector filed can be seen in Figure 6.7. It shows the wind velocities in different altitudes and the percent cloudiness on some part of world.

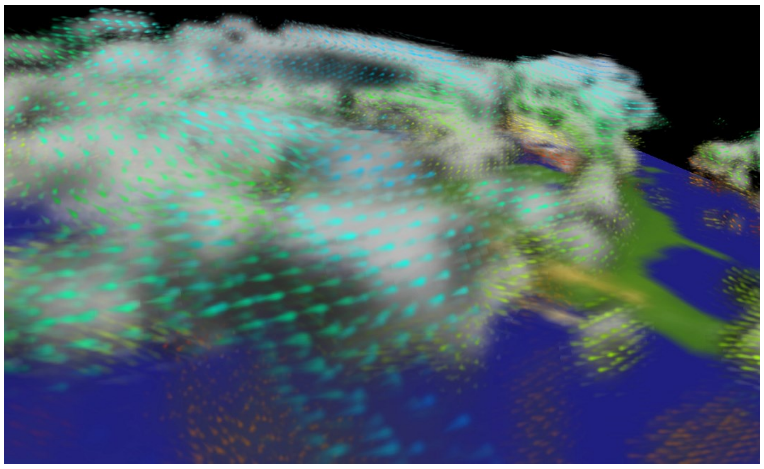


Figure 6.7: Visualization of percent cloudiness and wind velocities. The wind velocities are color coded by altitude (from [Cra93]).

6.1.5 Streamlets

Streamlets are generated by integrating the flow vectors for a very short time. Even though short, streamlets already communicate temporal evolution along the flow. Figure 6.8 illustrates an example of inspecting $2D$ flow field by several streamlets. This technique is easily extendable to $3D$, although perceptual problems may arise due to distortions resulting from the rendering projection. Thus, seeding becomes more important in $3D$.

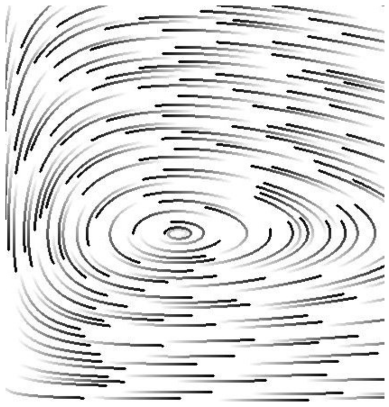


Figure 6.8: Example of vector field visualized by using streamlets.

[Lof98] uses a thread of streamlets along characteristic structures of $3D$ flow to gain selective, but importance-based seeding. The publication uses a probability distribution function assuring the streamlets to be distributed uniformly around a selected base trajectory. The function is designed so that with increasing distance from this trajectory, the distribution of streamlets fades out.

6.2 Local technique

Local techniques of vector field visualization methods visualize only some parts of the vector field. The user usually have to interact with the visualization system, i.e. select some important parts of vector field to perform the visualization.

6.2.1 Stream lines

Performing longer integration, in comparison to streamlets, results in obtaining streamlines. Streamlines are a family of curves that are instantaneously tangent to the velocity vector of the flow. These show the direction a massless fluid element will travel in at any point in time. Concerning the extension to $3D$ (see example in Figure 6.9), the same condition stands for streamlines as well as for streamlets, i.e. careful seeding is necessary. Otherwise, visual clutter can easily become a problem and the results might be difficult to interpret.

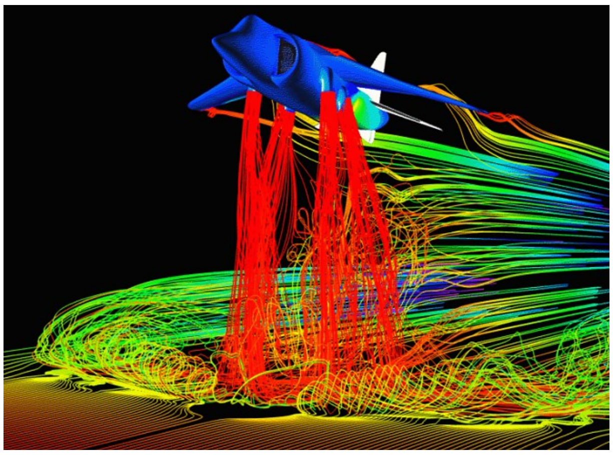


Figure 6.9: Example of vector field visualized by using streamlines.[1]

There are some important goals to consider in order to generate an effective streamline visualization. In particular, a good seeding strategy should have the following characteristics [Ver00].

The streamlines should not miss any interesting regions in the vector field. The interesting regions are those that we would like to study in the vector field, e.g. critical points, separation, and re-attachment lines. In addition, streamlines should cover the entire region of the field. Hence, even if the field is more or less uniform in a region, some streamlines should indicate the uniform nature of the flow in these regions. This goal is easier to achieve than other goals because one can always generate a lot of streamlines such that nothing important is missed. However, simply populating the field with more streamlines is not acceptable because some areas in the flow field, such as convergent regions, will force streamlines to cluster together, making it difficult to distinguish among

[1] image source: http://medspark.ms/Technical-Glossary-All.php

individual streamlines. More importantly, it defeats the characteristic of uniformity as described next.

The streamlines should be more or less uniformly distributed over the field. This is a more challenging goal to achieve because while we can control where to place the seeds, we do not know how the resulting streamlines will behave. Uniformity is directly related to the density of streamlines crossing a unit area of the flow field. Hence, density of streamlines is an important parameter.

It is desirable from the point of view of aesthetics that the streamlines show continuity in the flow. Hence, one would prefer fewer long streamlines over many short streamlines. The latter tend to give the impression of "choppiness" while the former tend to give an impression of smooth continuous flow. In general, given an arbitrary flow field, the longer the streamlines, the higher the likelihood that they will tend to crowd together in some areas and disperse in other areas, thereby making it difficult to meet both the uniformity and continuity criteria simultaneously. Therefore, this parameter needs to be balanced against the uniformity criterion.

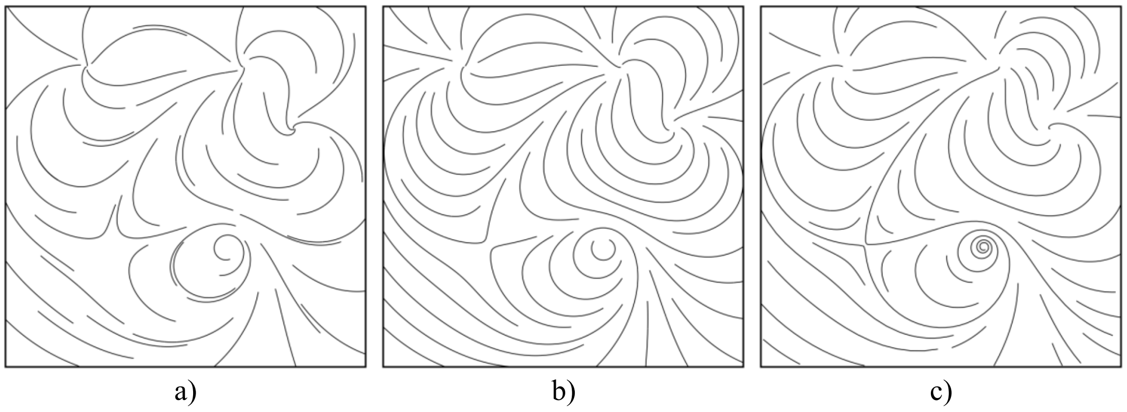


Figure 6.10: Effects of regular seeding (49 streamlines) (a), effects of image-guided seeding (47 streamlines) (b), effects of flow-guided seeding (47 streamlines) (c). Regular and image-guided seeding strategies may miss important flow features, especially when seeding is sparse. Both image-guided and flow-guided streamlines were generated such that the minimum separating distance of streamlines is 3% of the image width (from [Ver00]).

Since the flow pattern in the neighbourhood of a critical point is defined by the type of critical point, [Ver00] and [Ye05] proposed different seeding patterns for different types of critical points. They refer to these seeding patterns as templates. They are designed so that streamlines traced from them can effectively capture the local flow patterns around the critical points. Figure 6.11 illustrates the seeding templates for the three types of flow patterns: centre and spiral, source and sink, and saddle. Recall that the type of the critical point depends on the eigenvalues of the Jacobian matrix at the critical point. While the sign and magnitude of the eigenvalues determine the divergence or convergence rate of the flow, the related eigenvectors tell us the directions of the streamlines. Both the eigenvalues and eigenvectors are used as guides in developing the seeding templates.

Centre, spiral: place seeds along a straight line emanating from the critical point location. Figure 6.11 (a) shows the seed template for centre and spiral type of critical points.

Source, sink: place seeds along the perimeter of a circle around the critical point. Figure 6.11 (b) shows the seed template for this type of critical point.

Saddle: place seeds along the lines that bisect the principal eigen direction. Figure 6.11 (c) shows the seed template for saddles.

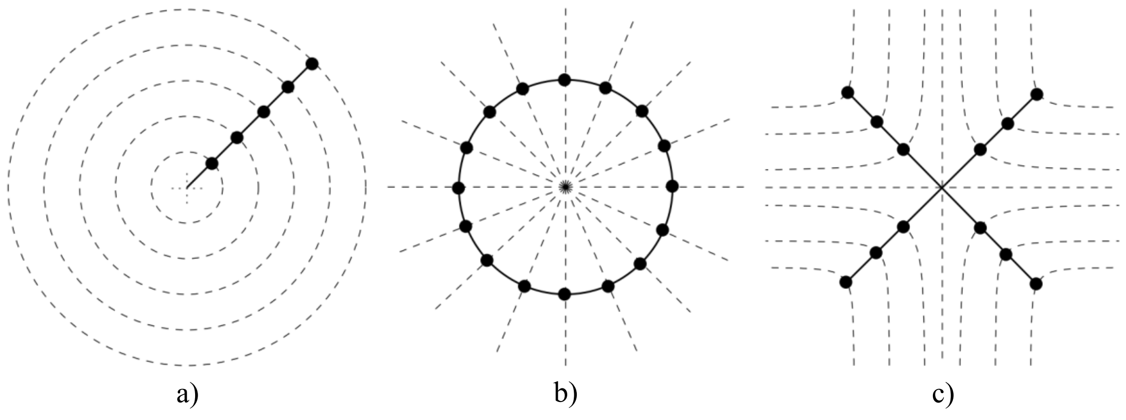


Figure 6.11: Seed templates for various critical points. The seeds are placed along the solid lines. The bold dots represent the seed template and the dashed lines are the streamlines traced using the seeds from the template. (a) centre, spiral; (b) source, sink; (c) saddle. (from [Ye05]).

6.2.2 Path lines

Pathlines are used to visualize the trajectory of a fluid particle as it advances with the passage of time. Pathlines, are therefore, history lines, an animation of all the fluid particles in the flow field.

Pathlines are the trajectories that individual fluid particles follow. These can be thought of as "recording" the path of a fluid element in the flow over a certain period. The direction that the path takes will be determined by the streamlines of the fluid at each moment in time. Pathlines are allowed to intersect themselves or other pathlines (except the starting and end points of the different pathlines, which need to be distinct).

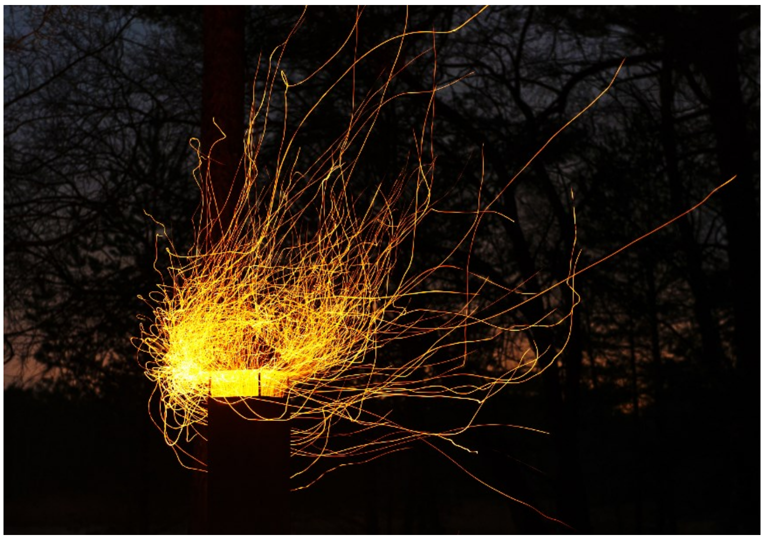


Figure 6.12: Visualization of pathlines. A long exposure photo of sparks from a campfire is used to illustratively show the pathlines for the flow of hot air.

6.2.3 Streak lines

Streaklines [Wei10] are locations of all the fluid particles that have passed continuously through a particular spatial point in the past. Particle injected into the fluid at a fixed point extends along a streakline. Streaklines are integral curves, produced by simulating continuous injection of particles into the flow field from constant location over certain period of time. Displaying streak lines serves, at the first place, for unsteady flow data visualization, for, in case of steady flows, streaklines coincide with streamlines and pathlines.

Streamline computation always exploits vector information from just a single instant of time. A trace obtained this way, therefore describes the trajectory of an imaginary massless particle moving through the flow field at infinite speed, which diverges from what users would usually expect. Such technique is called instantaneous. Streak lines, on the other hand, be long among time correlated methods, which progressively include information from consecutive temporal instants letting the integral curve develop in time.

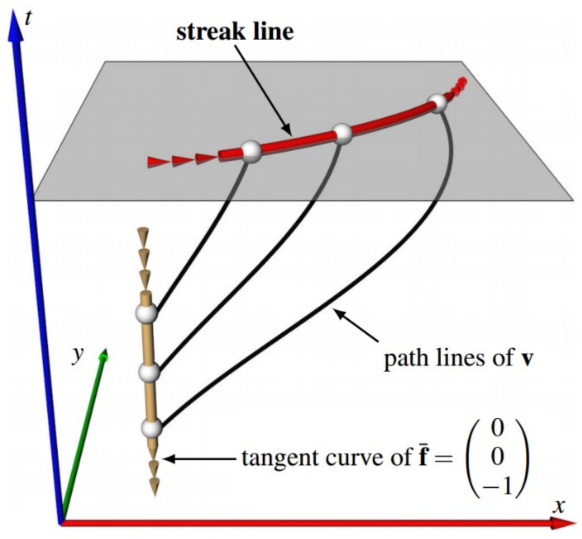

t
streak line
y
path lines of $\mathbf{v}$
tangent curve of $\bar{\mathbf{f}} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$
x

Figure 6.13: Visualization of streak line. Particles are injected from only one location but different times. We can see the difference between a streak line and a path line. (from [Wei12]).

7 Proposal of future research

In this work a study about the approximation and visualization of the vector fields has been presented. Although the broadness of the topic and the variety of existing techniques did not allow us to make this survey exhaustive but the main attitudes have been described and the most important aspects have been discussed in detail.

In the next text, we summarize some of the problems that arises in the flow visualization and that we want to investigate in our future research.

Advances in hardware are leading to more computational power and the ability to process larger and more complex simulations with faster computation times. Therefore, flow visualization algorithms must be able to handle this large amount of data and present the results ideally at interactive frame rates in order to be most useful in the analysis of simulation data. For this purpose can be used algorithms for vector field simplification. This algorithms reduce the size of the dataset and some of them even simplify the topological skeleton of the vector field. We want to use Radial basis functions for simplification of the vector field. When the RBF approximation will use appropriately placed positions of radial basis functions for approximation depending on the topology of the vector field, then this approximation will be a simplified description of the vector filed. Moreover, this approximation could be used for removing a noise from a vector field, as most measured vector fields contain a lot of noise. The RBF approximation of a vector field is an analytical description, which is much more useful than the standard discrete representation of a vector field.

One of the challenges specific to geometric flow visualization is the seeding strategy used to place the objects within the data domain. The position of the objects greatly affects the final visualization. Different features of the velocity field may be depicted depending on the final position and the spatial frequency of the objects in the data domain. It is critical that the resulting visualization captures the features of the velocity field, e.g. vortices, turbulence, sources, sinks and laminar flow, which the user is interested in. This aspect becomes an even greater challenge in the case of $3D$ vector fields where a balance of field coverage, occlusion and visual complexity must be maintained. Time-dependent data also raise a challenge because the visualization then depends on when objects are seeded. To create one such ideal visualization of the vector filed, we need to extract the most information about the vector field. Having a meshless approximation or interpolation of the vector field, we want to extract some more features of the vector field. This should be possible as the analytical description contains more information than the standard discrete representation.

Now, the solutions for topology-based unsteady flow visualization remain incomplete, compared with the level of research achieved for steady flows. Incremental extensions of methods that work well for steady flows are proven not to be able to fully capture the behaviour of time-varying vector flows. Therefore, new approaches and methods are needed and we would like to investigate our research in this area as well.

References

[Agr15] Agranovsky, A., Camp, D., Joy, K. I., Childs, H.: Subsampling-based compression and flow visualization. In IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics, 2015.

[Asi93] Asimov, D.: Notes on the topology of vector fields and flows. Technical report, NASA Ames Research Center, RNR-93-003, 1993.

[Ast10] Åström, K. J., Murray, R. M.: Feedback systems: an introduction for scientists and engineers, ISBN: 978-0-691-13576-2, Princeton university press, 2010.

[Atl02] Atluri, S. N., Shen, S.: The Meshless Local Petrov-Galerkin (MLPG) Method: A Simple & Less-costly Alternative to the Finite Element and Boundary Element Methods. CMES: Computer Modeling in Engineering & Sciences, Vol. 3, No. 1, pp. 11-52, 2002.

[Bar71] Bartsch, H. J.: Matematické vzorce, SNTL - Nakladatelstvo technické literatury, Nakladatelstvo Alfa, N.P., Bratislava, 3rd edition, 1971.

[Bax01] Baxter, B. J., Hubbert, S.: Radial basis functions for the sphere. Recent Progress in Multivariate Approximation, pp. 33-47, Birkhäuser Basel, 2001.

[Bay11] Bayona, V., Moscoso, M., Kindelan, M.: Optimal constant shape parameter for multiquadric based RBF-FD method. Journal of Computational Physics, Vol. 230, pp. 7384-7399, 2011.

[Ben66] Ben-Israel, A.: A Newton-Raphson method for the solution of systems of equations. Journal of Mathematical analysis and applications, Vol. 15, No. 2, pp. 243-252, Elsevier, 1966.

[Bha14] Bhatia, H., Gyulassy, A., Wang, H., Bremer, P. T., Pascucci, V.: Robust detection of singularities in vector fields. In Topological Methods in Data Analysis and Visualization III, pp. 3-18, Springer International Publishing, 2014.

[Bur07] Bürger, K., Schneider, J., Kondratieva, P., Krüger, J., Westermann, R.: Interactive Visual Exploration of Unsteady 3D Flows, EuroVis, pp. 251-258, 2007.

[Cab93] Cabral, B., Leedom, L. C.: Imaging vector fields using line integral convolution. Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 263-270, ACM, 1993.

[Che07] Chen, G., Mischaikow, K., Laramee, R., Pilarczyk, P., and Zhang, E.: Vector field editing and periodic orbit extraction using morse decomposition. IEEE Transactions on Visualization and Computer Graphics, Vol. 13, No. 4, pp. 769-785, IEEE, 2007.

[Cra93] Crawfis, R. A., Max, N.: Texture splats for 3D scalar and vector field visualization. IEEE Conference on Visualization'93, pp. 261-266, IEEE, 1993.

[Ede08] Edelsbrunner, H., Harer, J.: Persistent homology - a survey. Journal of Contemporary mathematics, Vol. 453, pp. 257-282, Providence, RI: American Mathematical Society, 2008

[Fas07] Fasshauer, G. E.: Meshfree approximation methods with MATLAB. World Scientific, ISBN: 978-981-270-633-1, 2007.

[Fas07a] Fasshauer, G. E., Zhang, J. G.: On choosing optimal shape parameters for RBF approximation. Numeric Algorithms, Vol. 45, pp. 345-368, 2007.

[Fly09] Flyer, N., Wright, G. B.: A radial basis function method for the shallow water equations on a sphere. In Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences (pp. rspa-2009). The Royal Society, 2009.

[For02] Forman, R.: A user's guide to discrete Morse theory. Séminaire Lotharingien de Combinatoire, Vol. 48, 35pp., 2002.

[For11] Fornberg, B., Lehto, E.: Stabilization of RBF-generated finite difference methods for convective PDEs. Journal of Computational Physics, Vol. 230, No. 6, pp. 2270-2285, 2011.

[For09] Forsberg, A. S., Chen, J., Laidlaw, D. H.: Comparing 3d vector field visualization methods: A user study. IEEE Transactions on Visualization and Computer Graphics, Vol. 15, No. 6, pp. 1219-1226, IEEE, 2009.

[Fra82] Franke, R.: Scattered data interpolation: tests of some methods. Mathematical Computing, Vol. 38, pp. 181-200, 1982.

[Gar92] Gardiner, J. D., Laub, A. J., Amato, J. J., Moler, C. B.: Solution of the Sylvester matrix equation AXB T+ CXD T= E. ACM Transactions on Mathematical Software (TOMS), Vol. 18, No. 2, pp. 223-231, 1992.

[Gjø04] Gjøystdal, K.: Classifying zeros in three dimensional vector fields. Master's thesis, The Norwegian University of Science and Technology, 2004.

[Glo91] Globus, A., Levit, C., Lasinski, T.: A tool for visualizing the topology of three-dimensional vector fields. The 2nd conference on Visualization'91, pp. 33-40, IEEE Computer Society Press, 1991.

[Gre92] Greene, J. M.: Locating three-dimensional roots by a bisection method. Journal of Computational Physics, Vol. 98, pp. 194-198, 1992.

[Gyu06] Gyulassy, A., Natarajan, V., Pascucci, V., Bremer, P.-T., and Hamann, B.: A topological approach to simplification of three dimensional scalar functions. IEEE Transactions on Visualization and Computer Graphics, Vol. 12, No. 4, pp. 474-484, 2006.

[Har71] Hardy, R. L.: Multiquadric equations of topography and other irregular surfaces. Journal of geophysical research, Vol. 76, No. 8, pp. 1905-1915, Wiley Online Library, 1971.

[Hel89] Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets, Vol. 22, No. 8, pp. 27-36, IEEE, 1989.

[Hel91] Helman, J., Hesselink, L.: Visualizing vector field topology in fluid flows, IEEE Computer Graphics and Applications, No. 3, pp. 36-46, IEEE, 1991.

[Hub15] Hubbert, S., Lê Gia, Q. T., Morton, T. M.: Spherical radial basis functions, theory and applications. ISBN: 978-3-319-17938-4, Springer, 2015.

[Kan90a] Kansa, E.J.: Multiquadrics - A scattered data approximation scheme with applications to computational fluid-dynamics - I surface approximations and partial derivative estimates. Computers & mathematics with applications, Vol. 19, No. 8, pp. 127-145, Elsevier, 1990.

[Kan90b] Kansa, E.J.: Multiquadrics - A scattered data approximation scheme with applications to computational fluid-dynamics - II solutions to parabolic, hyperbolic and elliptic partial differential equations. Computers & mathematics with applications, Vol. 19, No. 8, pp. 147-161, Elsevier, 1990.

[Kat97] Katok, A., Hasselblatt, B.: Introduction to the modern theory of dynamical systems, Vol. 54, Cambridge university press, 1997.

[Kir99] Kirby, R. M., Marmanis, H., Laidlaw, D. H.: Using concepts from painting. Proc.Visualization'99, pp. 333-540, IEEE, 1999.

[Koch15] Koch, S., Kasten, J., Wiebel, A., Scheuermann, G., Hlawitschka, M.: 2D Vector field approximation using linear neighborhoods. The Visual Computer, pp. 1-16, 2015.

[Lai05] Laidlaw, D. H., Kirby, R. M., Jackson, C. D., Davidson, J. S., Miller, T. S., Da Silva, M., Warren, W. H., Tarr, M. J.: Comparing 2D vector field visualization methods: A user study. IEEE Transactions on Visualization and Computer Graphics, Vol. 11, No. 1, pp. 59-70, 2005.

[Lar07] Laramee, R. S., Hauser, H., Zhao, L., Post, F. H.: Topology-based flow visualization, the state of the art. In Topology-based methods in visualization, pp. 1-19, Springer Berlin Heidelberg, 2007.

[Leh12] Lehto, E.: High Order Local Radial Basis Function Methods for Atmospheric Flow Simulations. Dissertation thesis, Uppsala, ISBN: 978-91-554-8421-7, 2012.

[Lew04] Lewiner, T., Lopes, H., and Tavares, G.: Applications of forman's discrete morse theory to topology visualization and mesh compression. IEEE Transactions on Visualization and Computer Graphics, Vol. 10, No. 5, pp. 499-508, 2004.

[Li06] Li, W. C., Vallet, B., Ray, N., Levy, B.: Representing higher-order singularities in vector fields on piecewise linear surfaces. IEEE Transactions on Visualization and Computer Graphics, Vol. 12, No. 5, pp. 1315-1322, IEEE, 2006.

[Liu09] Liu, G. R.: Meshfree methods: moving beyond the finite element method. Taylor & Francis, 2009.

[Lod00] Lodha, S. K., Renteria, J. C., Roskin, K. M.: Topology preserving compression of 2D vector fields. Visualization 2000. pp. 343-350, IEEE, 2000.

[Lod03] Lodha, S. K., Faaland, N. M., Renteria, J. C.: Topology preserving top-down compression of 2d vector fields using bintree and triangular quadtrees. IEEE Transactions on Visualization and Computer Graphics, Vol. 9, No. 4, pp. 433-442, 2003.

[Lof98] Loffelmann, H., Groller, E.: Enhancing the Visualization of Characteristic Structures in Dynamical Systems, Eurographics, pp. 35-46, 1998.

[Man02] Mann, S., Rockwood, A.: Computing singularities of 3D vector fields with geometric algebra, Proceedings of the conference on Visualization'02, pp. 283-290, IEEE, 2002.

[Mor94] Moridis, G., Kansa, E.J.: The Laplace transform multiquadric method highly accurate scheme for numerical solution of partial differential equations. Journal of Applied Science and Computing, Vol. 1, No. 2, pp. 375-407, 1994.

[Per87] Perry, A. E., Chong, M. S.: A description of eddying motions and flow patterns using critical-point concepts. Annual Review of Fluid Mechanics, Vol. 19, No. 1, pp. 125-155, 1987.

[Phi97] Philippou, P. A., Strickland, R. N.: Vector field analysis and synthesis using three dimensional phase portraits, Graph. Models Image Process., Vol. 59, No. 6, pp. 446-462, 1997.

[Pre92] Press, WTVWH, Teukolsky, S., Vetterling, W.: Numerical Recipes in C. Cambridge University Press, ISBN: 0-521-43108-5, pp. 379-383, 1992.

[Rip99] Rippa, S.: An algorithm for selecting a good value for the parameter c in radial basis function interpolation. Adv. Comput. Math., Vol. 11, pp. 193-210, 1999.

[Sal08] Salzbrunn, T., Janicke, H., Wischgoll, T., Scheuermann, G.: The state of the art in flow visualization: Partition-based techniques. In SimVis, H. Hauser, S. Straßburger, H. Theisel (Eds.). SCS Publishing House e.V. Magdeburg, Germany, pp. 75-92, 2008.

[Sch11] Scheuerer, M.: An alternative procedure for selecting a good value for the parameter c in RBF-interpolation, Adv. Comput. Math. Vol. 34, pp. 105-126, 2011.

[Sch79] Schagen, I. P.: Interpolation in Two Dimension – A New Technique. Journal of Applied Mathematics, Vol. 23, No. 1, pp. 53-59, IMA, 1979.

[Ska13] Skala, V.: Fast Interpolation and Approximation of Scattered Multidimensional and Dynamic Data Using Radial Basis Functions. WSEAS Transactions on Mathematics, Vol. 12, No. 5, pp. 501-511, WSEAS, 2013.

[Ska15] Skala,V.: Meshless Interpolations for Computer Graphics, Visualization and Games: An Informal Introduction, Tutorial, Eurographics 2015, doi: 10.2312/egt.20151046, Zurich, 2015.

[Skr16] Skraba, P., Rosen, P., Wang, B., Chen, G., Bhatia, H., & Pascucci, V.: Critical Point Cancellation in 3D Vector Fields: Robustness and Discussion. IEEE Transactions on Visualization and Computer Graphics, doi: 10.1109/TVCG.2016.2534538, 2016.

[Smo16a] Smolik, M., Skala, V.: Vector Field Interpolation with Radial Basis Functions, SIGRAD 2016, No. 3, pp. 15-21, ISBN: 978-91-7685-731-1, Linköping University Electronic Press, 2016.

[Smo16b] Smolik, M., Skala, V.: Vector Field RBF Interpolation on a Sphere, CGVCVIP 2016, pp. 352-356, ISBN: 978-989-8533-52-4, 2016.

[Smo16c] Smolik, M., Skala, V.: Classification of Critical Points Using a Second Order Derivative, IEEE Transactions on Visualization and Computer Graphics. [submitted: May 2016]

[The03] Theisel, H., Rössl, C., Seidel, H. P.: Compression of 2D vector fields under guaranteed topology preservation. Computer Graphics Forum, Vol. 22, No. 3, pp. 333-342, Blackwell Publishing, Inc, 2003.

[The04] Theisel, H., Weinkauf, T., Hege, H.-C., and Seidel, H.-P.: Grid-independent detection of closed stream lines in 2D vector fields. VMV 2004, Vol. 4, pp. 421-428, 2004.

[Tri01] Tricoche, X., Scheuermann, G., and Hagen, H.: Continuous topology simplification of planar vector fields. VIS 2001, IEEE Computer Society, pp. 159-166, 2001.

[Uhl06] Uhlir, K., Skala, V.: Radial basis function use for the restoration of damaged images, Computer vision and graphics. ISSN: 1381-6446, pp. 839-844. Springer, 2006.

[Ver00] Verma, V., Kao, D., Pang, A.: A flow-guided streamline seeding strategy. Visualization'00, pp. 163-170, IEEE Computer Society Press, 2000.

[Wei05] Weinkauf, T., Theisel, H., Shi, K., Hege, H.-C., and Seidel, H.-P.: Extracting higher order critical points and topological simplification of 3D vector fields. in Proc. IEEE Visualization 2005, Minneapolis, U.S.A., pp. 559-566, 2005.

[Wei08] Weinkauf, T.:Extraction of topological structures in 2D and 3D vector fields. Ph.D. dissertation, University Magdeburg, 2008.

[Wei10] Weinkauf, T., Theisel, H.: Streak lines as tangent curves of a derived vector field. IEEE Transactions on Visualization and Computer Graphics, Vol. 16, No. 6, pp. 1225-1234, 2010.

[Wei12] Weinkauf, T, Hege, H.-C., Theisel, H.: Advected Tangent Curves: A General Scheme for Characteristic Curves of Flow Fields. Computer Graphics Forum (Proc. Eurographics), Vol. 31, No. 2, 2012.

[Wen95] Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Advances in computational Mathematics, Vol. 4, No. 1, pp. 389-396, 1995.

[Wri03] Wright, G. B., Radial basis function interpolation: numerical and analytical developments. University of Colorado at Boulder, 2003.

[Ye05] Ye, X., Kao, D., Pang, A.: Strategy for seeding 3D streamlines. Visualization (VIS 05), pp. 471-478, IEEE, 2005.

Publications

Smolik, M., Skala, V.: Vector Field Interpolation with Radial Basis Functions, SIGRAD 2016, No. 3, pp. 15-21, ISBN: 978-91-7685-731-1, Linköping University Electronic Press, 2016.

Smolik, M., Skala, V.: Vector Field RBF Interpolation on a Sphere, CGVCVIP 2016, pp. 352-356, ISBN: 978-989-8533-52-4, 2016.

Smolik, M., Skala, V., Nedved. O.: A Comparative Study of LOWESS and RBF Approximations, ICCSA 2016, LNCS 9787, Vol.II, pp. 405-444, Springer, 2016.

Skala, V., Majdisova, Z., Smolik, M.: Space Subdivision to Speed-up Convex Hull Construction in E^3, Advances in Engineering Software (IF: 1,402), Vol. 91, pp. 12-22, ISSN: 0965-9978, 2016.

Skala, V., Smolik, M., Majdisova, Z.: Reducing the number of points on the convex hull calculation using the polar space subdivision in E2, SIBGRAPI2016, October 2016. [accepted]

Smolik, M., Skala, V.: A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E^2, ICIG 2015, Vol. 9217, pp. 394-404, ISSN: 0302-9743, 2015.

Smolik, M., Skala, V.: Highly Parallel Algorithm for Large Data In–Core and Out–Core Triangulation in E^2 and E^3, ICCS 2015, Vol. 51, pp. 2613-2622, ISSN: 1877-0509, Elsevier, 2015.

Skala, V., Smolik, M., Karlicek, L.: HS-Patch: A New Hermite Smart Bicubic Patch Modification, NAUN Journal: International Journal of Mathematics and Computers in Simulation, Vol. 8, pp. 292-299, ISSN: 1998-0159, 2014.

Smolik, M., Skala, V.: In-core and out-core memory fast parallel triangulation algorithm for large data sets in E^2 and E^3, SIGGRAPH 2014, No. 37, ISBN: 978-1-4503-2958-3, ACM, 2014.

Smolik, M., Skala, V.: Fast Parallel Triangulation Algorithm of Large Data Sets in E^2 and E^3 for In-Core and Out-Core Memory Processing, ICCSA 2014, Vol. 8580, pp. 301-314, ISBN: 978-3-319-09129-7, Springer, 2014.

Šmolík, M., Skala, V.: Paralelní triangulace v E^2 a E^3 na CPU a GPU, Technical report, No. DCSE/TR-2014-01, ZČU, Plzeň, May 2014.

Šmolík, M.: Metody triangulace v paralelním prostředí, Diploma thesis, ZČU, Plzeň, 2013.

A Project assignments, other activities

Research projects

- MSMT CR project LH12181
    - Development of Algorithms for Computer Graphics and CAD/CAM systems (2012-2015)
- MSMT CR project LG13047
    - EURO: Activities within Eurographics Association (2013-2015)
- SGS 2013-029 and SGS 2016-013
    - Advanced Computing and Information Systems.

A.1 Stays abroad

September 2012 - June 2013

- Erasmus stay at Mälardalen University, Västerås, Sweden

A.2 Conferences and talks

2016: SIGRAD 2016, Visby, Sweden

- Vector Field Interpolation with Radial Basis Functions

2016: CGVCVIP 2016, Funchal, Portugal

- Vector Field RBF Interpolation on a Sphere

2015: ICIG 2015, Tianjin, China

- A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2

2015: ICCS 2015, Reykjavík, Iceland

- Highly Parallel Algorithm for Large Data In-Core and Out-Core Triangulation in E2 and E3

2014: SIGGRAPH 2014: Vancouver, Canada

- In –Core and Out-Core Memory Fast Parallel Triangulation Algorithm for Large Data Sets in E2 and E3

2014: ICCSA 2014, Guimarães, Portugal

- Fast Parallel Triangulation Algorithm of Large Data Sets in E2 and E3 for In-Core and Out-Core Memory Processing

2013: Talk at ÚGN AVČR, Ostrava, Czech Republic

- Triangulation in E2 and E3 using CPU and GPU

A.3 Teaching activities

2015/2016

- Fundamentals of Computer Graphics (KIV/ZPG)

2014/2015

- Fundamentals of Computer Graphics (KIV(ZPG)
- Introduction to Computer Graphics (KIV/UPG)

2013/2014

- Fundamentals of Computer Graphics (KIV/ZPG)