# Flow Visualization via Partial Differential Equations

**4 authors**, including:

Alexandru Telea
University of Groningen
**225** PUBLICATIONS **4,014** CITATIONS

# Flow Visualization via Partial Differential Equations

Tobias Preusser[1], Martin Rumpf[2], and Alex Telea[3]

[1]  Center for Complex Systems and Visualization, University of Bremen,
     Universitätsallee 29, 28359 Bremen, Germany
     `tp@cevis.uni-bremen.de`
[2]  Institute for Numerical Simulation, Bonn University, Nussallee 15, 53115 Bonn, Germany
     `martin.rumpf@ins.uni-duisburg.de`
[3]  Department of Mathematics and Computer Science, Eindhoven University, Den Dolech 2,
     Eindhoven, Netherlands
     `alext@win.tue.nl`

**Summary.** The visualization of stationary and time-dependent flow is an important and challenging topic in scientific visualization. Its aim is to represent transport phenomena governed by vector fields in an intuitively understandable way. In this paper, we review the use of methods based on partial differential equations (PDEs) to post-process flow datasets for the purpose of visualization. This connects flow visualization with image processing and mathematical multi-scale models. We introduce the concepts of flow operators and scale-space and explain their use in modeling post processing methods for flow data. Based on this framework, we present several classes of PDE-based visualization methods: anisotropic linear diffusion for stationary flow; transport and diffusion for non-stationary flow; continuous clustering based on phase-separation; and an algebraic clustering of a matrix-encoded flow operator. We illustrate the presented classes of methods with results obtained from concrete flow applications, using datasets in 2D, flows on curved surfaces, and volumetric 3D fields.

## 1 Introduction

A great variety of different approaches for the visualization of vector field data has been presented in the past. The methodology ranges from simple discrete arrow plots applied to steady two-dimensional vector fields to advanced hardware-accelerated volumetric techniques for visualizing multivariate data for three-dimensional, unsteady flow problems and multi-scale feature detection and tracking techniques for complex time-dependent CFD problems.

The recent increase of the number of flow visualization techniques has been driven by two main factors. On one hand, the exponential growth in size of datasets produced by CFD simulations requires flow visualization methods to be able to display more data in shorter time. On the other hand, specific application fields, ranging from weather simulation, meteorology, and ground water flow, to automotive, aerodynamics, and machine design, have each their own particular requirements and questions to be answered regarding flow datasets. As a central and generally accepted

high-level goal, flow visualization should provide intuitively better receptible methods which give overall as well as detailed views on the flow patterns and behavior.

Given the above, several classifications of flow visualization methods have been recently proposed from different points of view. In their State of the Art report, Laramee et al. [20] have classified flow visualization into direct, dense texture-based, geometric, and feature-based methods, following a model of the flow data in discrete samples, continuous (dense) scalar fields, geometric integral primitives, and application-specific feature-based representations. A second overview of flow visualization methods is given by Weiskopf and Erlebacher [45]. Here, three classifications of flow visualization methods are proposed: based on the *visual primitive* used (points, curves, or features); based on the *density* of the produced image (sparse vs. dense, texture-based); and finally based on the data *structure* (2D, 2.5D, and 3D methods) and *discretization* (on various grid types). A more recent report of Laramee et al. [21] presents a comparison of major visualization techniques evaluated from the point of view of a specific application – the understanding of swirl and tumble flow data. Here, visualizations are classified into texture-based methods, clustering approaches, analyses of the vector field topology, and feature-tracking approaches.

In this paper, we give an overview of flow visualization methods based on partial differential equations (PDEs) [9, 12, 13]. These methods use a particular model of the flow data, as follows. The flow domain is seen as a subset of the continuous $\mathbb{R}^2$ or $\mathbb{R}^3$ space. The visualization process is described now in terms of a continuous, physical process, such as diffusion, advection, or phase separation. The particular type of PDE and its boundary conditions are used as instruments to model different visualization questions, such as: Which are the laminar, transient, and turbulent regions? How does the material density vary in time in the dataset? How does the flow look like on small spatial scales (flow details) as opposed to a global, coarse scale (flow overview)? Once the type and parameters of the proper PDE are established, the flow domain is discretized, usually by a finite-element or finite-difference scheme, and the underlying PDE is solved with appropriate solvers for the resulting equation or system of equations. Finally, the solution is displayed, thereby answering the initial set of visualization questions that target the flow data.

The above leads us to an outline of several characteristics of PDE-based flow visualization methods. For this, we shall use the terminology employed by the classifications presented in [20, 21, 45]. First, PDE-based flow visualizations are *dense* methods, by definition. Second, they work in all dimensions where flow visualization is of interest, i.e. 2D, 2.5D (curved surfaces embedded in 3D), and 3D. Third, they are applicable to both steady and unsteady (time-dependent) flow datasets. In terms of actual visual representation, PDE-based methods are naturally closely related to texture-based methods. Although the results of PDE-based methods can be displayed using also other techniques, such as slice planes, streamlines [12], and isosurfaces [9], their inherent continuous, dense nature makes them natural candidates for using 2D and 3D texture-based display techniques. In this sense, PDE-based methods can be seen as a front-end, that translate a given direct flow representation (vector field plus additional scalar quantities such as pressure or concentration) to a

second dense, usually scalar, representation, which is then visualized by a texture-driven back-end. On a high level, this translation, encoded in the PDE, enriches the data with application and question-specific semantics, in order to emphasize the specific aspects of the flow the user is interested in. Conversely, many texture-based visualization methods implemented using (programmable) graphics hardware have at their core a model based on an advection ordinary differential equation (ODE), as described further in Sect. 2. Finally, in terms of data discretization (grid type), all PDE-based methods can essentially be used on any grid, given a suitable underlying finite element discretization implemented on that grid (cf. Sect. 8).

From another point of view, PDE-based methods share many common aspects with multi-scale flow visualization methods. Overall, the main goal of such methods is to provide a multi-scale representation of the flow field, such that users can subsequently navigate between detailed, low-level views of the flow and global, overview pictures thereof. Several multi-scale methods exist in flow visualization [20, 45]. Clustering methods [14, 35] group similar flow dataset points together based on a task or application-dependent similarity measure, or correlation. Energy minimization techniques can be used to produce streamline visualizations at several levels of detail, represented by different streamline spatial densities [18, 19, 38]. PDE-based methods bring several powerful tools for defining the multi-scale, based on scale space theory (cf. Sect. 3), and are able to accommodate several scale notion definitions, ranging from continuous to discrete.

Given this close connection between PDE-based and texture-based flow visualization, we give first an overview of the texture-based methods in Sect. 2 followed by a brief introduction to the basic multi-scale methods in image processing, which motivate the approaches to be discussed here. Next, in Sect. 3 we review the connections to scale-space methodology from image processing. Together with the differential operator defined in Sect. 4, this leads to the presentation of the anisotropic diffusion method in Sect. 5. The extension of this model towards time-dependent flow fields is discussed in Sect. 6. In the remaining, we review clustering methods based on PDE techniques and we start with the model based on anisotropic phase separation in Sect. 7. A discussion of hierarchical multi-scale clustering using algebraic multigrid (AMG) methods follows in Sect. 9. Section 10 closes this article drawing conclusions.

## 2 Review of Texture Based Flow Visualization

*Texture-based flow visualization* is a notion generally used for those methods that output a full spatial coverage of the flow field to be described, in the region of interest chosen by the user. By full coverage, we mean that the discreteness of the output data is limited to the one inherent to the image, or texture primitives used in the visualization. Given this dense representation of the flow field, the texture image will mostly encode some continuous color, luminance, or transparency variation that conveys insight into the flow data. Often, the above continuous signal is naturally generated by physically-based methods. Texture-based methods differ, thus,

mainly in how, and what, they generate and encode in the output texture. We outline below the most important classes of texture-based methods, and refer for an in-depth overview to [20].

The spot noise method proposed by van Wijk [18] pioneered the use of textures in flow visualization. Elliptic splats of intensity based on a noise function and which are oriented along the field are blended together on 2D or 3D surface domains. The original first order approximation of the flow was improved by de Leeuw and van Wijk in [8] by using higher order polynomial deformations of the spots in areas of significant vorticity. By animating the intensity, spots appear to move along flow streamlines. Several applications of spot noise were presented in the context of smog prediction and turbulent flows [6], non-continuous flow visualization [22], and flow topology [7].

Line Integral Convolution (LIC) methods represent the second major class of texture-based visualizations. Introduced by Cabral and Leedom [2], LIC integrates the fundamental ODE describing streamlines forward and backward in time at every discrete domain point. White noise is convolved, using a Gaussian kernel, along these particle paths. The resulting value gives the intensity of the starting pixel. The resulting texture exhibits strong correlations along streamlines and weak correlation across, giving the perception of streamline-like filaments of varying intensity. Essentially, LIC is equivalent to a diffusion process along the vector field. Hege and Stalling [17] increased the performance of LIC by reusing portions of the convolution integral already computed on points along a given streamline. Forssell [11] proposed a similar method on surfaces, whereas Max et al. [24] approach flow visualization by texturing iso-surfaces. UFLIC [31] extended LIC to unsteady flows. Interrante and Grosch [16] generalized line integral convolution to 3D in terms of volume rendering of line filaments. Multivariate flow fields are visualized with LIC using a color mapping technique called color weaving [39]. OLIC [43] and its fast version FROLIC [29] add up- and downstream cues to the basic LIC by varying the intensity along the streamline. Finally, we mention 3D LIC, which uses texture-based volume rendering to compute and display LIC visualizations for 3D flow fields [27]. Again, the above is just a short overview of a wealth of existing LIC techniques. For a more detailed overview, see [20].

Yet another class of texture-based methods are the ones based on texture advection. Here, the visualization primitive is directly supported by the graphics unit or GPU. Consequently, the term texture in these methods often refers to the graphics hardware term. GPU-based methods are classified based on the primitive they advect, or warp (pixel or polygon) and the advection direction (forward or backward). Max and Becker [23] presented one of the first texture-advection methods using triangles. Image-based flow visualization (IBFV) proposes an injection of noise (stored as textures), advecting it by warping a polygon mesh, and blending the result for smooth visualization, with applications in 2D [41], curved surfaces [42], and 3D volumes [34]. Lagrangian–Eulerian Advection (LEA) is another such model, where particle positions are advected individually (Lagrangian step) and the color texture is updated in-place (Eulerian step) [17, 47]. Recently, the above (and other) frameworks, were united in UFAC (Unsteady flow advection-convolution), using an

implementation based on programmable GPUs [46]. Interestingly, the emergence of the "framework" for GPU-based methods as a collection of tightly-woven conceptual, modeling, and implementational aspects seems to be driven by the large importance of the implementational aspect in the whole process, in contrast to, e.g. LIC methods.

Especially for 3D velocity fields, particle tracing is a very popular tool. However, even relatively many seed particles released by the user can hardly cope with the complexity of 3D vector fields. Zöckler et al. [33] use pseudo randomly distributed, illuminated and transparent streamlines to give a denser and more receptible representation, which shows the overall structure and enhances important details.

Most notably every subclass of texture-based method seems to produce visualizations that carry an easily recognizable visual signature. For example, it is easy to tell spot-noise from LIC; the various IBFV and LEA methods have also a distinct visual appearance, probably due to the specific noise functions used; illuminated streamlines are also a class apart; reaction-diffusion methods create regular repetitive patterns which noise-injection methods cannot replicate. We believe that a perceptual classification of texture-based flow visualizations would bring valuable insight in the effectiveness (and limitations) of such methods and lead to a better understanding of flow data, although we are not aware of any such classification.

## 3 A Brief Introduction to Scale Space Methods in Image Processing

Textures used in various flow visualization approaches can be regarded as images and thus the type of flow post processing above discussed can be considered as image processing. In the last two decades powerful PDE based image processing methods for several fundamental tasks in imaging such as segmentation and de-noising have been introduced. In particular so called *scale space methods* introduce a natural scale of image representations. Most of the methods in flow post processing lack such a perspective of multiple scales. They have in common that the generation of a coarser scale requires a re-computation. For instance, if we ask for a finer or coarser scale of the line integral convolution patterns, the computation has to be restarted with a coarser initial image intensity. In case of spot noise larger spots have to be selected and their stretching along the field has to be increased. To motivate our PDE based approach, let us briefly review scale space methods based on anisotropic nonlinear diffusion in imaging.

Discrete diffusion type methods have been known for a long time. Perona and Malik [26] have introduced a continuous diffusion model which allows the de-noising of images together with the enhancement of edges. Alvarez et al. [1] have established a rigorous axiomatic theory of diffusive scale space methods. The recovery of lower dimensional structures in images is analyzed by Weickert [44], who introduced an anisotropic nonlinear diffusion method where the diffusion matrix depends on the so called structure tensor of the image.

In PDE-based scale-space methods of image processing we consider a function $u : \mathbb{R}_0^+ \times \Omega \to \mathbb{R}$ which solves the parabolic problem

$$\partial_t u - \mathcal{A}[u] = f(u) \quad \text{in } \mathbb{R}^+ \times \Omega ,$$
$$u(0, \cdot) = u_0 \qquad \text{on } \Omega , \tag{1}$$

for given initial density $u_0 : \Omega \to [0, 1]$. Here, the differential operator $\mathcal{A}[\cdot]$ is defined by

$$\mathcal{A}[u] := \text{div} \, (a(\nabla u_\epsilon)\nabla u)$$

and we prescribe Neumann boundary conditions $a(\nabla u_\epsilon)\nabla u \cdot \nu = 0$, where "$\cdot$" denotes the scalar product in $\mathbb{R}^+$ and $a : \mathbb{R}^n \to \mathbb{R}$ is the diffusion coefficient that controls the amount of spatial diffusion (blurring) in a given direction in $\mathbb{R}^n$. For the sake of robustness and well-posedness, a pre-smoothed version of the current density $u_\epsilon = \chi_\epsilon * u$ is used. In our setting we interpret the density as an image intensity, a scalar grey scale or a vector valued color. Thus, the solution family $u(\cdot)$ can be regarded as a family of images $\{u(t)\}_{t \in \mathbb{R}_0^+}$, where the time $t$ serves as a scale parameter. Let us remark that by the trivial choice $a = 1$ and $f(u) = 0$ we obtain the standard linear heat equation with its isotropic smoothing and coarsening effect.

In image de-noising, $u_0$ is a given noisy initial image and the goal is to remove this noise while keeping the important content of the given image. Thus, the diffusion is supposed to be controlled by the gradient of the image intensity. Large gradients mark edges in the image, which should be enhanced, whereas small gradients indicate areas of approximately equal intensity. For that purpose we prescribe a diffusion coefficient

$$a = g(\|\nabla u_\epsilon\|)$$

where $g : \mathbb{R}_0^+ \to \mathbb{R}^+$ is a monotone decreasing function with $\lim_{d \to \infty} g(d) = 0$ and $g(0) = \delta \in \mathbb{R}^+$, e. g. $g(d) = \frac{\delta}{1 + \|d\|^2}$ . A suitable choice for the pre-smoothing is Gaussian filtering or the convolution with the heat equation kernel. That is, we define $u_\epsilon = \tilde{u}(t = \epsilon^2/2)$ where $\tilde{u}$ is the solution of the heat equation with initial data $u$. Then $\epsilon$ is the variance of the corresponding Gaussian filter. The function $f$ may serve as a penalty which forces the scale of images to stay close to the initial image, e. g. choosing $f(u) = \gamma(u_0 - u)$ where $\gamma$ is a positive constant. Figure 1 gives an example of image smoothing and edge enhancement by nonlinear diffusion.



**Fig. 1.** The image on the *left* is successively smoothed by nonlinear diffusion. With increasing scale more and more fine-scale details vanish while the significant content is retained and enhanced
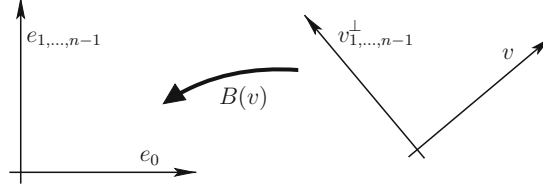
**Fig. 2.** The coordinate transformation $B(v)$

## 4  A Flow Aligned Differential Operator

Above, we have modeled an edge aligned operator $\mathcal{A}[\cdot]$, which enabled the feature sensitive de-noising of images. For the subsequent use, let us now define a streamline-aligned differential operator for flow fields. For a given vector field $v : \Omega \rightarrow \mathbb{R}^n$ we model linear diffusion in the direction of the vector field and a Perona–Malik type diffusion orthogonal to the field. Let us suppose that $v$ is continuous and $v \neq 0$ on $\Omega$. Then there exists a family of continuous orthogonal mappings $B(v) : \Omega \rightarrow SO(n)$ such that $B(v)v = \|v\|e_0$, where $\{e_i\}_{i=0,\dots,n-1}$ is the standard base in $\mathbb{R}^n$ (cf. Fig. 2).

We consider a diffusion tensor $a(v, \nabla u_\epsilon)$ which we define as

$$a(v, d) = B(v)^T \begin{pmatrix} \alpha(\|v\|) & \\ & g(\|d\|)\mathrm{Id}_{n-1} \end{pmatrix} B(v) \,,$$

where $\alpha : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ controls the linear diffusion in vector field direction, i. e. along streamlines, and the above introduced edge enhancing diffusion coefficient $g(\cdot)$ acts in the orthogonal directions ($\mathrm{Id}_{n-1}$ is the identity matrix in dimension $n-1$).

We may either choose a linear function $\alpha$ or, in case of, e.g. a velocity field which spatially varies over several orders of magnitude, we select a monotone function $\alpha$ (cf. Fig. 2) with $\alpha(0) > 0$ and $\lim_{s \rightarrow \infty} \alpha(s) = \alpha_{\max}$.

The differential operator based on this diffusion tensor is finally given by

$$\mathcal{A}[v, u] := \mathrm{div}\,(a(v, \nabla u_\epsilon)\nabla u) \,. \tag{2}$$

It encodes a strong coupling along the velocity field and in case of steep gradients in $u$ a weak coupling in directions perpendicular to the field.

## 5  Anisotropic Diffusion for Stationary Flow

We shall now make use of the differential operator defined in Sect. 4 to define a diffusion process, which generates texture patterns aligned to a flow field. These patterns will grow upstream and downstream, whereas the edges tangential to them are successively enhanced. Still there is some diffusion perpendicular to the field which supplies us for evolving time with a scale of progressively coarser representation of the flow field.
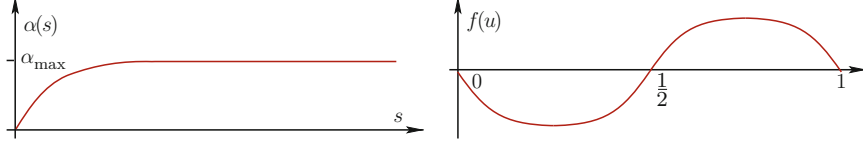
**Fig. 3.** *Left:* Graph of the velocity dependent linear diffusion coefficient $\alpha(\cdot)$. *Right:* Graph of the scalar contrast enhancing right-hand side $f(\cdot)$

In general it does not make sense to consider a specific initial image for such a diffusion process. As initial data $u_0$ we thus choose some random noise of an appropriate frequency range. If we run the evolution (1) for vanishing right-hand side $f$ the image contrast will unfortunately decrease due to the diffusion along streamlines. The asymptotic limit will turn out to be an averaged grey value. Therefore, we strengthen the image contrast during the evolution, selecting an appropriate continuous function $f : [0, 1] \rightarrow \mathbb{R}^+$ (cf. Fig. 3) with

(F1)  $f(0) = f(1) = 0$,
(F2)  $f < 0$ on $(0, 0.5)$, and $f > 0$ on $(0.5, 1)$.

Neglecting the diffusive term of the evolution at a first glance we realize that this right-hand side pushes values below the average value 0.5 towards the zero and accordingly values above 0.5 towards 1. A more detailed analysis of the contrast enhancement including the diffusive term is discussed in Sect. 6.1. However, well-known maximum principles ensure that the interval of grey values [0, 1] is not enlarged running the nonlinear diffusion. Here the property (F1) is of great importance.

Finally, we end up with the method of nonlinear anisotropic diffusion to visualize complex vector fields. Thereby we solve the nonlinear parabolic problem

$$\partial_t u - A[v, u] = f(u) \tag{3}$$

starting from some random initial data $u(0, \cdot) = u_0$ and obtain a scale of images representing the vector field in an intuitive way (cf. Fig. 4).

### 5.1 Enhancing the Resulting Texture

If we ask for point-wise asymptotic limits of the evolution, we expect an almost everywhere convergence to $u(\infty, \cdot) \in \{0, 1\}$ due to the choice of the contrast enhancing function $f$. Analytically, 0.5 is a third, but unstable fixed point of the dynamics, which, thus, will not turn out to be locally dominant numerically.

The space of asymptotic limits significantly influences the richness of the developing vector field aligned structures. We may ask how to further enrich the pattern which is settled by anisotropic diffusion. This turns out to be possible by increasing the set of asymptotic states. We no longer restrict the considerations to a scalar density $u$ but consider a vector valued $u : \Omega \rightarrow [0, 1]^2$ and a corresponding system of parabolic equations. The coupling is given by the nonlinear diffusion coefficient $g(\cdot)$ which now depends on the norm $\|\nabla u\|$ of the Jacobian of the vector
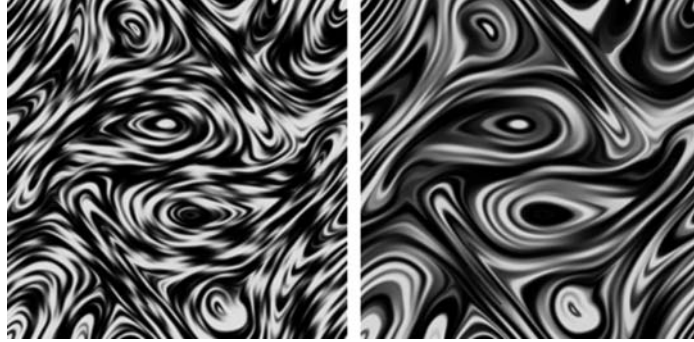
**Fig. 4.** A vector field from a 2D magneto-hydrodynamics simulation (MHD) is visualized by nonlinear diffusion. A discrete white noise is considered as initial data. We run the evolution on the *left* for a small and on the *right* for a large constant diffusion coefficient $\alpha$
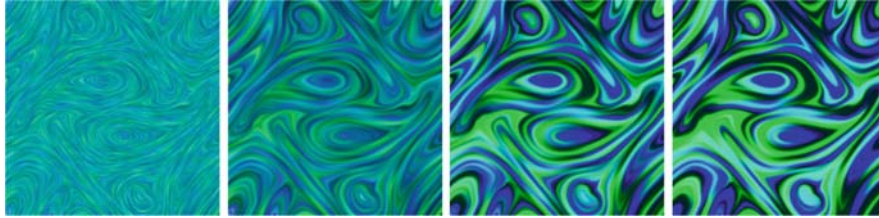


**Fig. 5.** Different snapshots from the multi-scale based on anisotropic diffusion are depicted for a 2D MHD simulation vector field (cf. Fig. 4). Here we consider a two-dimensional diffusion problem and interpret the resulting density as a color in a *blue/green* color space

valued density $\nabla u$ and the right-hand side $f(u)$. We define $f(u) = h(\|u\|)u$ with $h(s) = \tilde{f}(s)/s$ for $s \neq 0$, where $\tilde{f}$ is the old right-hand side from the scalar case, and $h(0) = 0$. Furthermore we select an initial density which is now a discrete white-noise with values in $B_1(0) \cap [0, 1]^2$. Thus, the contrast enhancing now pushes the point wise vector density $u$ either to the 0 or to some value on the sphere sector $S^1 \cap [0, 1]^2$. Again a straightforward application of the maximum principle ensures $u(t, x) \in B_1(0) \cap [0, 1]^2$ for all $t$ and $x \in \Omega$.

Figure 5 shows an example for the application of the vector valued anisotropic diffusion method applied to a 2D flow field from a MHD simulation. The field shows the dynamics of an electrically conducting fluid. The performed clustering outlines the so-called magnetic domains (thick clusters), i.e. domain regions where the fluid circulates, and current sheets (thin, closed clusters bordering the magnetic domains), which contain most of the current. Furthermore, Fig. 6 shows results of this method applied to several time steps of a convective flow field. An incompressible Bénard convection is simulated in a rectangular box with heating from below and cooling from above. The formation of convection rolls will lead to an exchange of temperature. We recognize that the presented method is able to nicely depict the global structure of the flow field, including its saddle points, vortices, and stagnation points on the boundary.
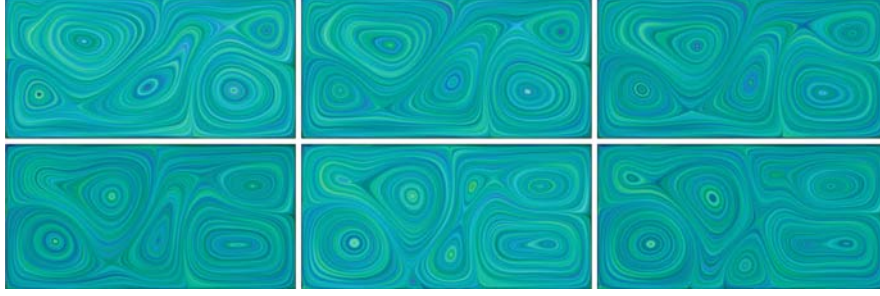
**Fig. 6.** Convective patterns in a 2D flow field are displayed and emphasized by the method of anisotropic nonlinear diffusion. The images show the velocity field of the flow at different time steps. Thereby the resulting alignment is with respect to streamlines of this time dependent flow

### 5.2 3D Flow Fields

The anisotropic nonlinear diffusion operator (2) has been formulated for arbitrary space dimension. It results in a scale of vector field aligned patterns which we then have to visualize. In 2D this has already been done in a straightforward manner in the above figures. In 3D we have somehow to break up the texture-volume and open up the view to inner regions. Otherwise we must confine ourselves with some pattern close to the boundary representing solely the shear flow.

Here we can benefit from the vector valued diffusion. Since for $m = 2$ the nontrivial asymptotic limits are in mean equally distributed on $S^1 \cap [0, 1]^2$, we can we reduce the image-content and focus on a ball shaped neighborhood $B_\delta(\omega)$ of a certain point $\omega \in S^1 \cap [0, 1]^2$. Now we can either use a volume rendering to visualize this type of sub-volumes or look at iso-surfaces of the function

$$\sigma(x) = \|u(x) - \omega\|^2.$$

Then the parameter $\delta^2$ allows us to depict the boundary of the pre-image of $B_\delta(\omega)$ with respect to the mapping $u$ (cf. Fig. 7).

### 5.3 Flow Fields on 2D Surfaces

So far we considered vector fields on domains which are subsets of the 2D or 3D Euclidean space. It is straight-forward to extend the methodology to tangential flow fields on surfaces, such as weather-map wind-fields over the earth, flow fields on stream-surfaces, or vector fields from differential geometry. We have to replace the Euclidean gradient $\nabla$ and the divergence operator div by their geometric counterparts $\nabla_\mathcal{M}$ and $\text{div}_\mathcal{M}$ respectively. Here the index $\mathcal{M}$ indicates that we are working with the tangential gradient and divergence on the surface or manifold $\mathcal{M}$. Proceeding as in Sect. 4 the differential operator describing the given flow field is given by

$$\mathcal{A}[u] := \text{div}_\mathcal{M}(a(\nabla_\mathcal{M} u_\epsilon)\nabla_\mathcal{M} u)$$
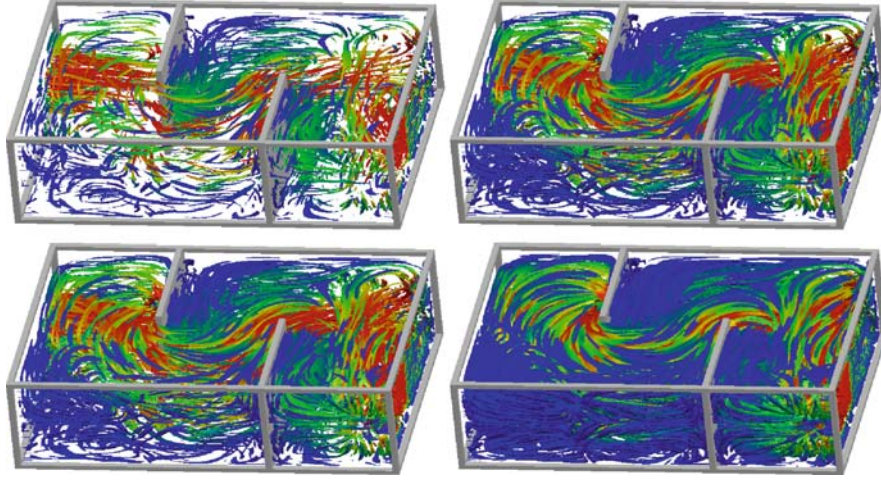
**Fig. 7.** The incompressible flow in a water basin with two interior walls and an inlet (on the *left* of the box) and an outlet (on the *right* of the box) is visualized by the anisotropic non-linear diffusion method. Iso-surfaces show the pre-image of $\partial B_\delta(\omega)$ under the vector valued mapping $u$ for some point $\omega$ on the sphere sector. From top left to bottom right the radius $\delta$ is successively increased. A color ramp *blue–green–red* indicates an increasing absolute value of the velocity. The diffusion is applied to initial data which is a relatively coarse grain random noise
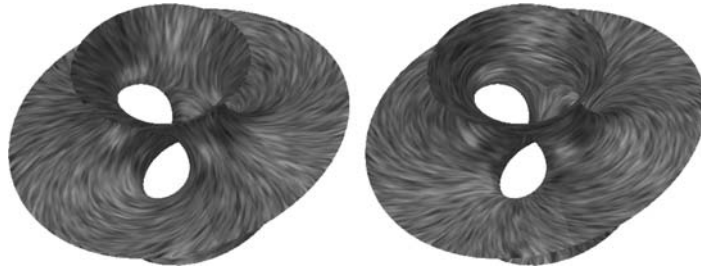


**Fig. 8.** The principal directions of curvature of a minimal surface are visualized using the anisotropic diffusion equation on surfaces

for $C^2$ functions $u$ on the manifold $\mathcal{M}$. As an illustration Fig. 8 shows the visualization of the principal directions of curvature on a minimal surface.

## 5.4 Flow Segmentation

The above applications already show the capability of the anisotropic nonlinear diffusion method to outline the flow structure not only locally. In particular for larger evolution times in the diffusion process the topological skeleton of a vector field becomes clearly visible. We will now investigate a possible flow segmentation by

means of the anisotropic diffusion. Let us restrict to the two-dimensional case of an incompressible flow with vanishing velocity $v$ at the domain boundary $\partial\Omega$. Then topological regions are separated by homoclinic, respectively heteroclinic orbits connecting critical points in the interior of the domain and stagnation points on the boundary. Critical points, by definition points with vanishing velocity $v = 0$, may either be saddle points or vortices. Furthermore we assume critical points to be non-degenerate, i. e. $\nabla v$ is regular. Saddle points are characterized by two real eigenvalues of $\nabla v$ with opposite sign, whereas at vortices we obtain complex conjugate eigenvalues with vanishing real part. Stagnation points on $\partial\Omega$ are similar to saddles. For details we refer to [15].

In each topological region there is a family of periodic orbits close to the heteroclinic, respectively homoclinic orbit. This observation gives reason for the following segmentation algorithm. At first, we search for critical points in $\Omega$ and stagnation points on $\partial\Omega$. We calculate the directions which separate the different topological regions. In case of saddle points these are the eigenvectors of $\nabla v$. Next, we successively place an initial spot in each of the sectors and perform an appropriate field aligned anisotropic diffusion.

Let us suppose that a single sector is spanned by vectors $\{w_+, w_-\}$ where the sign $\pm$ indicates incoming and outgoing direction. The method described by (3) would lead to a closed pattern along one of the above closed orbits for time $t$ large enough. To fill out the interior region we modify the diffusion by selecting an orientation for a "one sided" diffusion (cf. Fig. 9). That is, we select a unique normal $v^\perp$ to $v$ and consider the diffusion matrix

$$a(v, \nabla u_\epsilon) = B(v)^T \begin{pmatrix} \alpha & \\ & G((\nabla u_\epsilon \cdot v^\perp)_+) \end{pmatrix} B(v),$$

where $\alpha$ is a positive constant and $(s)_+ := \max\{s, 0\}$. Furthermore we consider a non negative, concave function $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ with $f(0), f(1) = 0$ as a source term in the diffusion equation. If the orientation of $\{w_+, w_-\}$ coincides with that of $\{v, v^\perp\}$, then linear diffusion in the direction towards the interior will fill up the complete topological region. A segmentation of multiple topological regions at the
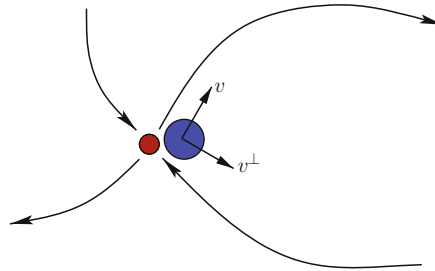


**Fig. 9.** A sketch of the four sectors at a critical point (indicated by *red disk*), the initial spot (*blue disk*) for the diffusion calculation and the oriented system $\{v, v^\perp\}$
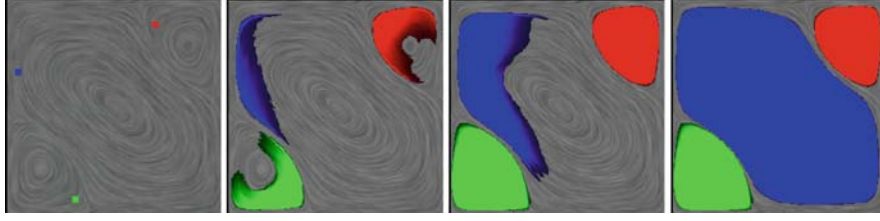
**Fig. 10.** Several time-steps from the nonlinear diffusion segmentation applied to a velocity field from a Bénard convection are shown. We have placed the seed-points as close as possible in terms of the grid size in the sectors spanned by the eigenvalues of the Jacobian $\nabla v$ of the velocity. Only to emphasize the evolution process a single grey-scale image from the diffusion calculation is underlying the sequence of segmentation time steps

same time is possible, if we carefully select the sectors where we release initial spots. Figure 10 shows different time steps of the segmentation applied to a convective incompressible flow.

## 6 Transport and Diffusion for Non-stationary Flow Fields

So far, the above anisotropic diffusion method generates streamline type patterns, which are aligned to trajectories of the vector field for a fixed given time. I. e. for a time-dependent vector field $v : \mathbb{R}^+ \times \Omega \to \mathbb{R}^d$ on a computational domain $\Omega \subset \mathbb{R}^d$ and $d = 2, 3$, we have been considering integral lines $\{x(s) \mid s \in \mathbb{R}\}$ with $\frac{d}{ds}x(s) = v(t, x(s))$ for a fixed time $t$. Thus, the method intuitively visualizes the vector field freezed at time $t$ but offers only very limited insight in the actual transport process governed by the underlying time-dependent flow field.

To ensure that our visualization actually displays this process we have to consider the true transport problem and its particle lines. Hence we take into account the particle motion obeying the equation $\frac{d}{dt}x(t) = v(t, x(t))$ and the induced transport of a given density $u(t, x)$. The fact that such a purely advected density $u$ stays constant along particle trajectories leads to the conservation law

$$\frac{D}{dt}u := \frac{d}{dt}u(t, x(t)) = \frac{\partial}{\partial t}u + \nabla u \cdot v = 0$$

which means a vanishing of the material derivative.

In addition to this conservation law, we have to incorporate a mechanism for the generation, the growth and enhancement of flow aligned patterns. Here we pick up the previous model (3) and consider a simultaneous anisotropic nonlinear diffusion process with linear diffusion along the particle line and sharpening in the perpendicular direction. Let us emphasize here that this diffusion process acts in forward *and* backward direction of the particle line. Thus, a careful control of the parameters is indispensable to avoid an artificial propagation in downwind direction with the accompanying visual impression of a wrong velocity. In the next section we will discuss in detail a suitable balance of parameters.

Altogether our basic *transport diffusion* model for time-dependent vector fields looks as follows: On the computational domain $\Omega \subset \mathbb{R}^d$ we consider for a given vector field $v : \mathbb{R}^+ \times \Omega \to \mathbb{R}^d$ the boundary and initial value problem:

$$\partial_t u + \nabla u \cdot v - \mathcal{A}[v, u] = f(u) \qquad \text{in } \mathbb{R}^+ \times \Omega,$$
$$(A\nabla u) \cdot v = 0 \qquad \text{on } \mathbb{R}^+ \times \partial\Omega,$$
$$u(0, \cdot) = u_0 \qquad \text{in } \Omega,$$

where $\mathcal{A}[v, u]$ is the diffusion tensor (2) already known from the anisotropic diffusion for steady flow fields. The initial data $u_0$ is again assumed to be a white noise of appropriate frequency. Still the role of the right-hand side $f$ is to ensure contrast enhancement. Consequently we apply functions $f$ which fulfill the properties (F1), (F2) mentioned previously.

The new model generates and stretches patterns along the flow field and transports them simultaneously. The resulting motion texture is characterized by a dense coverage of the domain with streakline type patterns, which do not have a fixed injection point but move in time with the fluid (cf. Fig. 11). The method is applicable in any dimension, in particular on 3D domains and 2D surfaces as for the static flow case before, although we have not performed such computations yet.
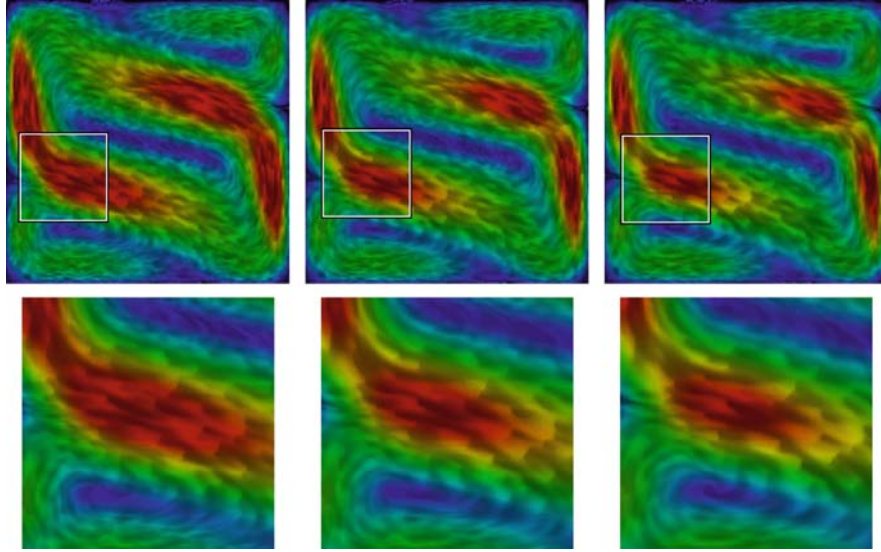


**Fig. 11.** Three successive time-steps of the transport diffusion process generating directed patterns of a Bénard convection (cf. Sect. 6.1). The additional coloring indicates the speed of the flow field. *Red colors* indicate high velocity, whereas *blue colors* indicate low velocity. To emphasize the transport of patterns we have magnified the marked sections of the images in the *lower row*

## 6.1 Balancing Parameters

In general, transport and diffusion are contrary processes. Our goal in mind – the generation and transport of patterns which simultaneously diffuse along the flows – there has to be a careful weighting of the parameters that steer the transport and the diffusion respectively. Otherwise the diffusion may overrun the transport, resulting in a process that is rather diffusion than transport with some pattern generating diffusion.

Let us suppose the temporal resolution of the given vector-field data is of size $\tau$. It is well known that the solution of the heat equation at a time $t$ corresponds to the convolution of the initial data with a Gaussian kernel of variance $\sqrt{2t}$. Since the diffusion tensor $a(v)$ invokes linear diffusion with a coefficient $\alpha(\|v(x)\|)$ in the direction of the velocity $v(x)$ for every $x \in \Omega$, we consider the corresponding variance

$$\mathcal{D}(\alpha(x)) := \sqrt{2\tau\alpha(x)}$$

to be a measure for the diffusion within the transport diffusion process for the time $\tau$. Of course a measure for the corresponding expected transport distance is

$$\mathcal{T}(x) := \tau \|v(x)\| .$$

Typically $\mathcal{T}(\cdot)$ is more or less fixed, since $\tau$ is in general prescribed by the underlying CFD data. Thus, we would like to adjust $\alpha$ locally such that $\mathcal{D}$ is balanced with $\mathcal{T}$. To this end we introduce a balancing parameter $\eta \in \mathbb{R}^+$ and consider the balancing condition

$$\mathcal{D}(\alpha(x)) = \eta\mathcal{T}(x).$$

Roughly speaking we then have the following relations:

| | |
|---|---|
| $\eta \ll 1$ | Transport dominates the model, |
| $\eta = 1$ | Transport $\approx$ Diffusion, |
| $\eta \gg 1$ | Diffusion dominates the model. |

Hence, choosing $\eta < 1$ fixed, and solving the balance condition for $\alpha(x)$, we get a suitable diffusion coefficient

$$\alpha(\|v\|)(x) = \frac{\eta^2 \|v(x)\|^2 \tau}{2}$$

as a function on the domain $\Omega$ which instead of the one defined in Sect. 4 is inserted into the diffusion tensor $a(v, \nabla u_\epsilon)$ of our transport diffusion model.

Let us furthermore study the amplification of certain frequencies of the initial image due to the right-hand side of our model. Our focus will be on the influence of the shape of $f$ on the contrast enhancing property of the model. To this end let us consider a much simpler setting of a high frequency initial data given by

$$u_0(x) = \frac{1}{2}\left[\sin\left(\frac{x}{\epsilon}\right) + 1\right]$$

and restrict ourselves to a simple diffusion equation along a (one-dimensional) streamline, which is given by

$$\partial_t u - \alpha \Delta u = f(u) \qquad \text{in } [0, 1].$$

We consider the linearization of $f$ around $\frac{1}{2}$

$$\tilde{f}(u) = \gamma \left( u - \frac{1}{2} \right),$$

where $\gamma$ is the slope of the original $f$ at $1/2$. Now let us take into account the ansatz $u(t) = b(t) \left( u_0(x) - \frac{1}{2} \right) + \frac{1}{2}$ for the evolution of a one-dimensional image-density. Inserting the ansatz into the linear diffusion-equation we obtain

$$\frac{1}{2} \left[ b' + \left( \frac{\alpha}{\epsilon^2} - \gamma \right) b \right] \sin \left( \frac{x}{\epsilon} \right) = 0$$

and so

$$b(t) = \exp \left[ \left( \gamma - \frac{\alpha}{\epsilon^2} \right) t \right].$$

This means that frequencies above $\sqrt{\alpha/\gamma}$ are damped, whereas frequencies below this threshold are amplified. Given an upper threshold $1/\epsilon$ for the frequencies which we want to amplify, we choose

$$\gamma = \frac{\alpha}{\epsilon^2}.$$

Finally, we construct our nonlinear right-hand side $f(\cdot)$ in such a way that the slope at $1/2$ equals $\gamma$.

## 6.2 A Blending Strategy for Long-Term Animation

With the anisotropic diffusion model for steady flow fields we have generated a whole scale of representations. Here, the scale was identified with the time $t$ of the evolution process. But as proposed in the last section, the scale parameter is now coupled to the actual transport process in our transport diffusion model. In particular for long-time visualization purposes this coupling leads to unsatisfactory results. Because due to the nature of our model, we are unable to freeze the scale and solely consider the evolution of suitable patterns at that specific scale in time, which would be the optimum process.

   The solution we propose here is a compromise based on the blending of different results from the transport diffusion evolution started at successive time-points. First, we select a suitable interval for the scale parameter $[s_0, s_1]$ with $s_1 > s_0 > 0$ around our preferred multi-scale resolution for the resulting images. Based on a smooth blending function $\psi : \mathbb{R} \to [0, 1]$ having support in $(-1, 1)$ and such that

$$\psi(t) = \psi(-t) = 1 - \psi(1 - t),$$
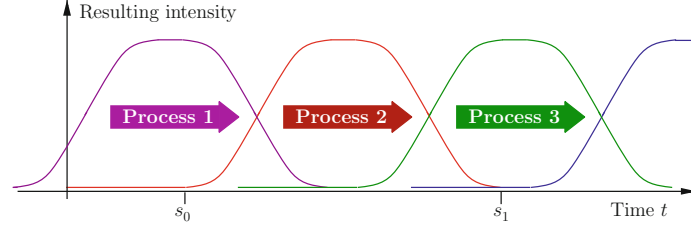$$\psi(0) = 1,$$

**Fig. 12.** The weighting factors in the blending operation together with the overlapping scale/time intervals of the considered transport diffusion processes are shown in a diagram over time

we can construct a partition of unity $\{\psi_i\}$ on the real line $\mathbb{R}$. That is, we define $\psi_i(t) = \psi(\frac{2t-(i+1)(s_0+s_1)}{s_1-s_0})$. Now, for all $i = 0, 1, \ldots$ we separately solve the above transport diffusion problem for different starting times $t_i = i(\frac{s_1-s_0}{2})$ always considering some white noise of a fixed frequency range as initial data and denoting the resulting solution by $u_i$. For negative time we suppose a suitable extrapolation of the velocity field to be given. Finally, applying blending of at least two different solutions we compute

$$u(t, x) = \sum_i \psi_i(t)u_i(t, x).$$

This intensity function is well defined for arbitrary times and characterized by the initially prescribed scale parameter interval.

We use this construction for an animation of the flow over a certain time interval (cf. Fig. 12 for a graph of the blending functions). Such an animation involves all solutions $u_i$ for which the blending function $\psi_i$ has a non vanishing overlap with the given time interval. Other constructions of a partition of unity and corresponding blending functions are near at hand and especially multiple overlaps can be considered which requires the blending of more than two intensity functions at the same time. We emphasize that the application of this blending technique does not introduce any inaccuracy, because for any time $t$ the resulting image $u(t, x)$ consists of images $u_i(t, x)$ showing streaklines at time $t$ and at slightly varying scale.

## 7 Continuous Clustering via Anisotropic Phase Separation

So far, we have discussed the generation of flow aligned multi scale textures. Let us now look the hierarchical clustering of flow data, ranging from small clusters showing strong local coherence of the flow to large global cluster sets gathering large flow patterns. As before, we will discuss this task in the framework of continuous models. Before we detail the application of such a model for the actual flow clustering let us review the underling physical model for the coarsening of structures in metal alloys, which goes back to Cahn and Hilliard [5].

## 7.1 The Cahn–Hilliard Model

The Cahn–Hilliard model was introduced to describe phase separation and coarsening in binary alloys. Phase separation occurs when a uniform mixture of the alloy is quenched below a certain critical temperature underneath which the uniform mixture becomes unstable. As a result a micro-structure of two spatially separated phases with different concentrations develops. In later stages of the evolution on a much slower time scale than that of the initial phase separation the structures become coarser: either by merging of particles or by growing of bigger particles at the cost of smaller ones. This coarsening can be understood as a clustering, where the system mainly tries to decrease the surface energy of the particles which leads to coarser and coarser structures during the evolution. In the basic Cahn–Hilliard model this surface energy is isotropic. There are no preferred directions of the interfaces. Hence the particles tend to be ball shaped (cf. Fig. 13).

In the following paragraph we briefly outline the basic concept of the Cahn–Hilliard model. For more details we refer to the review papers by Elliott [10] and Novick-Cohen [25]. The model is based on a Ginzburg–Landau free energy which is a functional in terms of the concentration difference $u$ of the two material components. The Ginzburg–Landau free energy $E$ is defined to be

$$E(u) := \int_{\Omega} \left\{ \Psi(u) + \frac{\gamma}{2} |\nabla u|^2 \right\},$$

where $\Omega$ is a bounded domain. The first term $\Psi(u)$ is the chemical energy density and typically has a double-well form. In this paper we take

$$\Psi(u) = \frac{1}{4} \left( u^2 - \beta^2 \right)^2$$

with a constant $\beta \in (0, 1]$ (cf. Fig. 14). We note that the system is locally in one of the two phases if the value of $u$ is close to one of the two minima $\pm\beta$ of $\Psi$.

Now, the diffusion equation for the concentration $u$ is given by

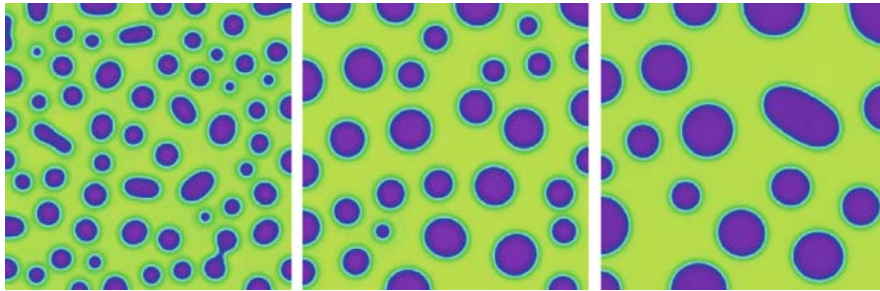$$\frac{\partial u}{\partial t} - \Delta w = 0$$



**Fig. 13.** Three time-steps of the original Cahn–Hilliard phase separation
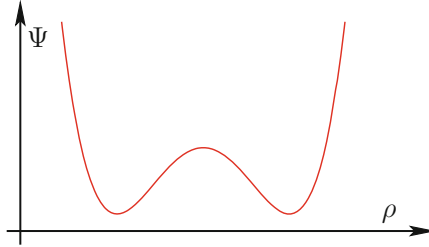
**Fig. 14.** Chemical energy as function of concentration

on $\mathbb{R}^+ \times \Omega$, where $w$ is the local chemical potential difference, which is given as the variational derivative of $E$ with respect to $u$

$$w = \frac{\delta E}{\delta u} = -\gamma \Delta u + \Psi'(u).$$

As boundary and initial conditions we request $\partial_\nu w = \partial_\nu u = 0$, where $\nu$ is the outer normal on $\partial \Omega$, and $u(0, \cdot) = u_0(\cdot)$ for some initial concentration distribution $u_0$.

Starting with a random perturbation of a constant state $\bar{u}_0$, which has a values in the unstable concave part of $\Psi$, we observe the following: In the beginning the chemical energy decreases rapidly whereas the gradient energy increases. This is due to the fact that during phase separation $u$ attains values which are at large portions of the domain close to the minima of the chemical energy $\Psi$. Since regions of different phase are separated by transition zones with large gradients of $u$, the gradient energy increases during phase separation. In the second stage of the evolution – the actual clustering – when the structures become coarser, the total amount of transition zones decreases. Correspondingly the amount of gradient energy becomes smaller again.

### 7.2 Anisotropic Interface Energy

Let us now turn to the clustering model for flow data. We introduce a cluster mapping $u : \mathbb{R}_0^+ \times \Omega \to \mathbb{R}$ which will be the solution of an appropriate evolution problem. Thereby, time will again serve as the scale parameter leading from fine cluster granularity to successively coarser clusters. For fixed scale $t$ our definition of the set of clusters $\mathcal{C}(t)$ is

$$\mathcal{C}(t) = \{x \mid u(t, x) \geq 0\}.$$

This set splits up into the connected components of $\mathcal{C}(t)$

$$\mathcal{C}(t) = \bigcup_i \mathcal{C}_i(t).$$

The evolution problem steering the clusters via the quantity $u$ should satisfy the following properties:

- The number of clusters generically decreases in time.
- The shape of the cluster components strongly corresponds to correlations in the data field.
- The volume fraction covered by $\mathcal{C}(t)$ is approximately constant in $t$, i. e. $\frac{|\mathcal{C}(t)|}{|\Omega|} \approx \Theta$ for $\Theta \in (0, 1)$.

These conditions motivate us to pick up the physical Cahn–Hilliard model with the double-well separation potential $\Psi(u)$, a separation energy $E_s = \int_\Omega e_s(u)$ and energy density $e_s(u) = \Psi(u)$. Among all $u$ with $\int_\Omega u = \bar{u}_0 = const.$ the energy $E_s$ attains its minimum if $u$ has the values $\pm\beta$ only. This leads to a binary decomposition of the domain into two parts, where one part corresponds to $\{x \mid u(x) = \beta\}$. However this set can have many connected components and may even be very unstructured. Furthermore there is no mechanism which enforces a successive coarsening and thus a true multi-scale of clusters.

We remedy this behavior by introducing a term which penalizes the occurrence of many disconnected cluster components with high interfacial area. To this end we choose a gradient energy $E_\partial = \int_\Omega e_\partial$ with local energy density $e_\partial$ that penalizes rapid spatial variations of $u$. In order to have flexibility to choose an anisotropic and inhomogeneous gradient energy, an appropriate definition of an interfacial energy density is given by

$$e_\partial(\nabla u) = \frac{\gamma}{2}a\nabla u \cdot \nabla u,$$

where $\gamma$ is a scaling coefficient and $a \in \mathbb{R}^{n \times n}$ is some symmetric positive definite matrix that may depend on the space variable and other quantities involved.

In the following we will refer to the set $\partial\{x \mid u(x) = 0\}$ as the interface. The orientation of the interface can be described by its normal which, in the case that $\nabla u \neq 0$, is given by the normalized gradient

$$v = \frac{\nabla u}{\|\nabla u\|}.$$

For $a = \text{Id}$ all gradients of $u$ and hence, all interfaces are penalized equally independent of their orientation. But with respect to our clustering intention we consider an anisotropic energy density which strongly depends on the orientation of the local interface and thereby on the direction of $\nabla u$.

For a given static vector-field $v : \Omega \to \mathbb{R}^n$ a natural clustering should emphasize the coherence along the induced streamlines. Thus, interfaces aligned across streamlines have to be penalized significantly by the gradient energy whereas interfaces oriented along streamlines are tolerated. We choose the diffusion tensor similar to the ones used above (cf. Sect. 4)

$$a(v) := B(v)^T \begin{pmatrix} 1 & 0 \\ 0 & \alpha(\|v\|)\text{Id}_{n-1} \end{pmatrix} B(v)$$

Since interfaces that cross streamlines shall have larger energy we choose a positive function $\alpha$ with $\alpha \leq 1$.

Altogether we have defined the energy

$$E(u) := \int_\Omega \left\{ \Psi(u) + \frac{\gamma}{2} a \nabla u \cdot \nabla u \right\}$$

and proceed as for the basic Cahn–Hilliard model. We define the first variation of the energy and arrive at the potential

$$w = \Psi'(u) - \gamma \mathcal{A}[v, u],$$

where $\mathcal{A} = \mathrm{div}(a(v)\nabla u)$ is defined as in the previous sections.

Let us continue as before and assume that the evolution of the cluster mapping $u$ is governed by diffusion where the corresponding flux linearly depends on the negative gradient of the first variation of energy. Thus, we choose $\partial_t u - \Delta w = 0$ as above and end up with the following fourth order differential equation:

$$\partial_t u - \Delta \left( \Psi'(u) - \gamma \mathcal{A}[v, u] \right) = 0$$

with boundary conditions $\partial_\nu u = \partial_\nu w = 0$ and prescribed initial data $u(0, \cdot) = u_0(\cdot)$. After an initial short period of phase separation it is mainly the interfacial energy contribution which is successively reduced.

As in the texture generation approaches it does not make sense to consider certain initial data, if no a priori information on the clustering is known. Consequently we choose a constant value $\bar{u}_0$ plus some random perturbations as initial data $u_0$. The constant $\bar{u}_0$ depends on the volume fraction $\Theta$ of the domain which shall be covered by the clusters, i.e. by the sets $\{x \mid u(t, x) \geq 0\}$. Therefore, we choose $\bar{u}_0 = \Theta\beta - (1 - \Theta)\beta$.

During the evolution very rapidly cluster patterns will grow without any pre-scribed location and orientation. This is in order to decrease $E_s = \int_\Omega \Psi(u)$ which forces the solution to obtain values close to $\pm\beta$ in most of the domain $\Omega$. After this starting phase the clusters orient themselves in an anisotropic way to decrease the amount of the anisotropic gradient energy $E_\partial$. In addition they become coarser and coarser due to the fact that smaller particles shrink and larger ones grow. In particular one observes that a large particle being surrounded by smaller ones grows to the expense of the smaller ones. This implies that as time evolves locally only the main features of the clusters will be kept (Fig. 15).

Finally we obtain a scale $u(t, \cdot)$ of cluster mappings and induced cluster sets $\mathcal{C}(t)$. They represent a successively coarser representation of simulation data and continuously enhances coherences in the underlying simulation dataset, where the cluster set $\mathcal{C}(t)$ will cover a volume of approximate size $\Theta|\Omega|$.
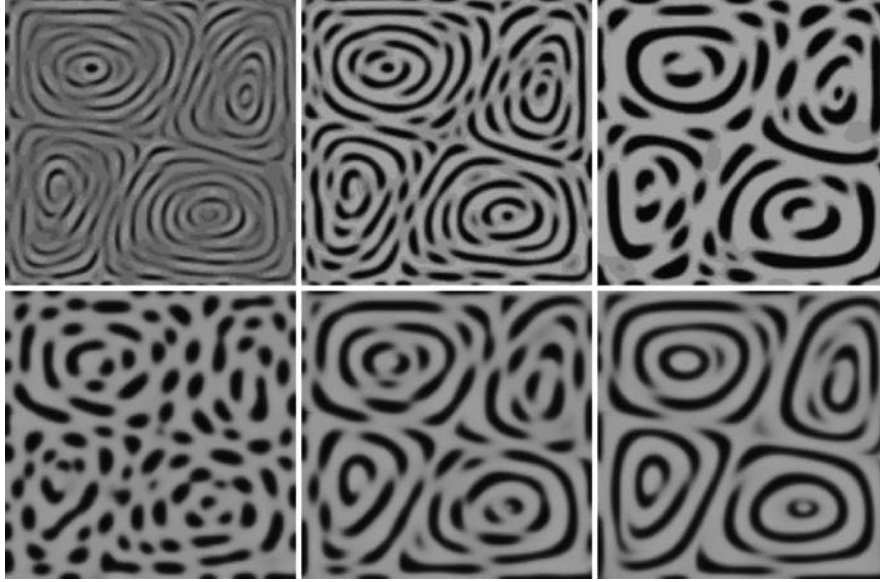
**Fig. 15.** *Top:* Successive stages of the continuous clustering of a Bénard convective flow field. *Bottom:* Effect of increasing anisotropy $\alpha$. The computations are based on a grid of resolution $257^2$

## 8 Remarks on the Finite Element Implementation

So far, we have not yet discussed discretization in space and time of the above introduced continuous time-dependent partial differential equations. Hence, we deal with the variational formulation of the different PDE problems introduced above. We propose to consider a finite element discretization in space and a semi-implicit backward Euler scheme in time.

The semi-implicit temporal discretization means that the nonlinear diffusion tensor $\mathcal{A}$ and the nonlinear term on the right-hand side $f(u)$ as well as the derivatives of the non convex potential $\psi$ are evaluated at the old time-steps.

For the spatial discretization we can restrict the numerical considerations to regular hexahedral grids in 2D and 3D. On these grids we have bilinear, respectively trilinear finite element spaces. However below we will use triangular elements as well. In any case we can base numerical integration on the lumped-masses product $(\cdot, \cdot)^h$ [36] for the $L^2$ product and a midpoint quadrature rule for the bilinear form $(A\nabla \cdot, \nabla \cdot)$.

Finally, in each step of the discrete evolution we have to solve a single system of linear equations for the vector of nodal values for the density function $u$ and the chemical potential $w$, respectively. In case we need pre-smoothed data, we consider a single discrete, implicit time-step for the computation of the heat equation with the density being smoothed as initial data.

# 9 Clustering Based on Hierarchical Decomposition of a Differential Operator

In the previous section we have discussed a continuous physical model for the clustering of flow fields. Instead of involving methods adapted from continuum mechanics, we might ask for a direct hierarchical decomposition of the differential operators $\mathcal{A}$ from Sect. 4, which represent diffusion processes strongly aligned to the flow field.

## 9.1 Review of Algebraic Multigrid

The idea we develop in this section uses algebraic multigrid (AMG) methodology to decompose the corresponding discrete operator. AMG methods were first introduced in the early 1980s [2, 3, 28] for the solution of discrete linear systems $AU = F$ of equations coming from the discretization of linear differential equations $\mathcal{A}[u] = f$ on domains $\Omega$ with suitable boundary conditions. We refer to [37] for a detailed introduction. Thereby $U$ is supposed to be a finite element approximation of the continuous solution $u$ and $A$ the finite element stiffness matrix corresponding to $\mathcal{A}$. Finally, $F$ is the corresponding discrete right-hand side.

The development of AMG was led by the idea to mimic classical (geometric) multigrid methods for applications where a hierarchy of nested meshes is either not available at all, or cannot reflect particular properties such as strength of diffusion of the discretized operator appropriately on coarse grid levels.

Consequently one has to work with the matrix $A$ and its algebraic structure. The general procedure is sketched in Fig. 16. AMG tries to coarsen this matrix independently from any underlying fine grid discretization, where $n$ is the number of degrees of freedom. It computes a sequence of prolongation matrices $P^l$ which encodes how coarse scale ($l$) basis functions are combined using the basis functions on the finer scale ($l - 1$). This induces a sequence of corresponding matrices $A^l$, defined by the so-called Galerkin projection $A^l := R^l A^{l-1} P^l$, where the restriction $R^l$ is given
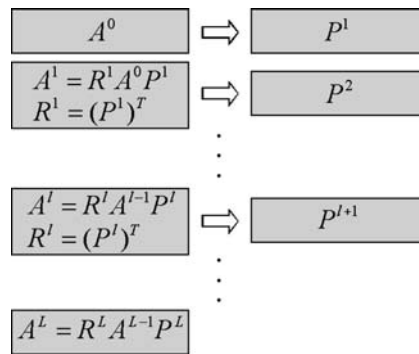


**Fig. 16.** General AMG construction. From the fine scale matrix $A^0$ input, AMG computes prolongations $P^l$, restrictions $R^l$, and coarse scale matrices $A^l$ on successively coarser scales $l = 1, \ldots, L$

as the transpose of the prolongation $P^l$ ($R^l := (P^l)^T$). The prolongation matrices $\{P^l\}_{l=1,\ldots,L}$ are computed using information from the matrix $A^{l-1}$ on the previous level $l-1$ only. The sequence of prolongation matrices allows for the construction of a problem-dependent basis $\{\Psi^{l,i}\}$. One constructs a coarser basis $\{\Psi^{l,i}\}$ which captures the appropriate features relevant for the approximation of the corresponding continuous problem.

## 9.2 AMG for Flow Field Clustering

The theory and design of efficient AMG tools is rather involved. However, we emphasize that our flowing clustering requires just basic AMG capabilities. We perform no specific tuning of the AMG for flow clustering. Let us apply the method to the concrete discrete finite element matrix $A$ of the differential operator $\mathcal{A}$. This stiffness-matrix $A$ can be regarded as a description of the structure of the flow field $v$, because as in the operator $\mathcal{A}$ the flow alignment is encoded in this matrix. Indeed, the matrix simultaneously represents dominant flow patterns as well as successively finer, more detailed flow structures.

With the AMG we find a tool which is able to represent flow patterns in a hierarchical multi-scale fashion. AMG delivers a set of descriptions of the flow-induced coupling in terms of matrices $A^l$ for $l = 0, \ldots, L$, ranging from detailed ($A^0 = A$) to very coarse ($A^L$).

Let us illustrate how AMG works using two simple examples. Consider the flow fields $v_1(x) = (-1, 1)$ and $v_2(x) = (1, 1)$ on the square domain $\Omega = [-1; 1]^2 \subset \mathbb{R}^2$. We can define a simple diffusion tensor

$$a(v) = B^T \begin{pmatrix} \|v\| + \epsilon & 0 \\ 0 & \epsilon \end{pmatrix} B := B^T \begin{pmatrix} \sqrt{2} + 0.001 & 0 \\ 0 & 0.001 \end{pmatrix} B,$$

where $B$ is a rotation of $\mp 45$ degrees and we chose a small diffusion value $\epsilon = 0.001$ orthogonal to the direction of the vector field $v$. We need a non-zero diffusion value in this direction too. Indeed, the diffusion value couples neighbor domain points (or element nodes, in the discretized version). Having a non-zero diffusion across the field $v$ to be clustered ensures that we shall obtain thick clusters on coarse scales, as described later. Having a relatively large diffusion along the field $v$ ensures that these clusters will be much larger in the direction of the field than across, which is the desired result. The choice of the $\epsilon$ value is not important as long as it stays a few orders of magnitude smaller than the average value of $\|v\|$. We then consider the corresponding differential operator $\mathcal{A}[u] = \text{div}(a\nabla u)$ and apply the AMG method to the matrix which results from the discretization of $\mathcal{A}$ on a regular triangulation. Figure 17 shows the coupling strengths encoded in the matrices $A^l$ for the first three finest levels $l = 0, 1, 2$, for the fields $v_1 = (1, 1)$ and $v_2 = (-1, 1)$, using a blue-to-red colormap. For the same fields, Fig. 18 shows selected basis functions on the four coarsest decomposition levels.

For the actual flow field clustering application we consider the differential operator $\mathcal{A}[u] = \text{div}(a(v)\nabla u)$ where the diffusion tensor is
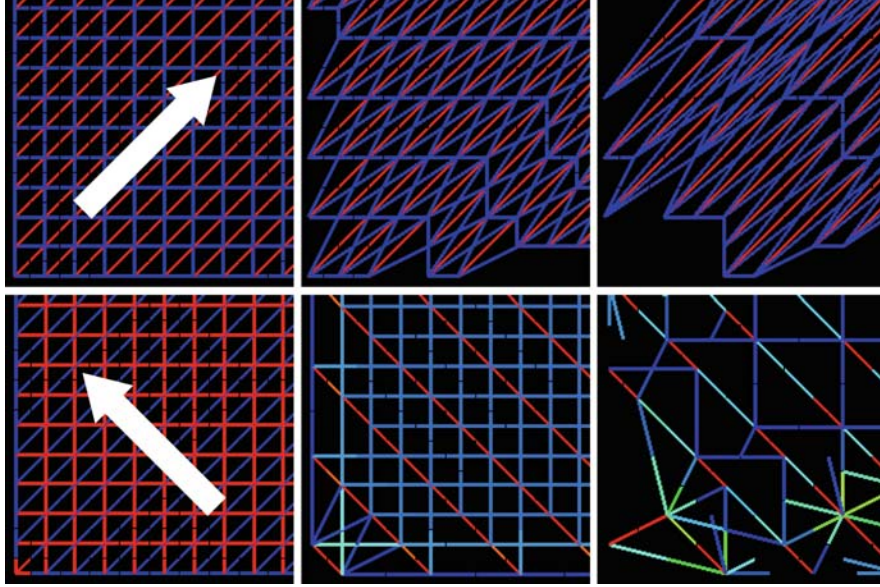
**Fig. 17.** Color-coded coupling strength (zoomed in) on the computational grid. Three finest levels (*left to right*) are shown for the fields $v_1 = (-1, 1)$ (*bottom row*) and $v_2 = (1, 1)$ (*top row*). The *white arrows* show the field direction
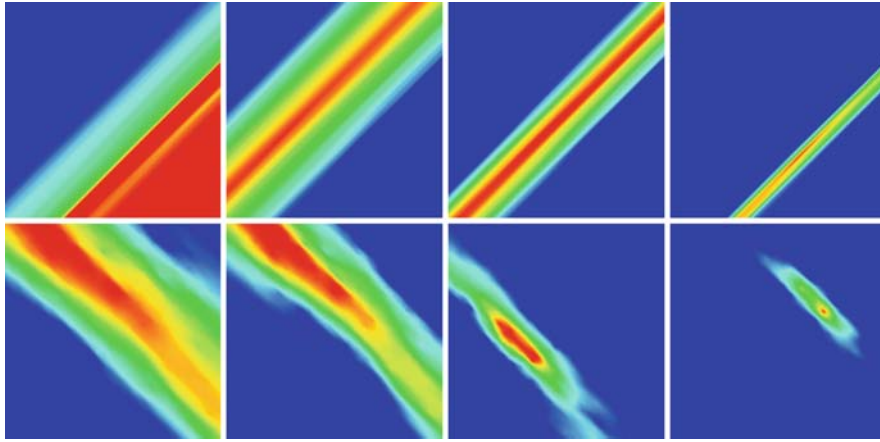


**Fig. 18.** For the two vector fields $v_1 = (-1, 1)$ (*bottom row*) and $v_2 = (1, 1)$ (*top row*) basis functions on the four coarsest levels are shown. Obviously the basis functions are clearly aligned to the flow field (cf. Fig. 17)

$$a(v) := B(v)^T \begin{pmatrix} \alpha(\|v\|) & 0 \\ 0 & \mathrm{Id}_{n-1} \end{pmatrix} B(v)$$

and $B(v)$ is the same rotation as above.

When we apply the AMG algorithm to the matrix $A \in \mathbb{R}^{n,n}$ corresponding to the above differential operator, we obtain a sequence of prolongation matrices $P^l \in \mathbb{R}^{n_{l-1},n_l}$ as output, where $n_l$ for $l = 0, \ldots, L$ are the decreasing numbers of remaining unknowns and $n_0 = n$. The entries in each column $i = 1, \ldots, n_l$ of $P^l$ give the coefficients of the linear combination of the finer basis functions $\Phi^{l-1,j}$ for $j = 1, \ldots, n_{l-1}$ corresponding to the coarser basis function $\Phi^{l,i}$ on level $l$. In other words, each matrix $A^l$ delivered by the AMG, starting with the initial, finest one $A^0 = A$ down to the coarsest one $A^L$, approximates the fine grid operator using the (matrix-dependent) basis $\{\Phi^{l,i}\}_{i=1,\ldots,n_l}$:

$$A_{ij}^l = A\overline{\Phi^{l,i}} \cdot \overline{\Phi^{l,j}} = \int_\Omega a(v) \nabla \Phi^{l,i} \cdot \nabla \Phi^{l,j} ,$$

where $\overline{\Phi^{l,i}}$ is the nodal vector corresponding to the function $\Phi^{l,i}$, i. e. denoting the initial basis functions with $\Theta^j$ we have $\Phi^{l,i} = \sum_{j=1,\ldots,n}(\overline{\Phi^{l,i}})_j \Theta^j$.

Hence, the following simple recursive recipe can be used to calculate the multi-scale of basis functions $\Phi^{l,i}$

$$\Phi^{l,i} := \sum_{j=1,\ldots,n_{l-1}} P_{ji}^l \Phi^{l-1,j} \quad \forall i = 1, \ldots, n; \; l = 1, \ldots, L$$

$$\Phi^{0,i} := \Theta^i \quad \forall i_1, \ldots, n$$

Figure 18 already indicates that the shapes of the basis functions clearly show the strength of the local coupling. The AMG method clusters vertices along a streamline already on a fine scale, since they are strongly coupled. Vertices not aligned to the flow are clustered on coarser scales, since their coupling is relatively weaker.

### 9.3 From Basis Functions to Clusters

But as usual with finite elements, the supports of basis functions on a given scale are overlapping. Therefore, we need to derive a multi-scale of domain decompositions from the set of basis functions to partition the domain into disjoint clusters. Such a domain decomposition

$$\mathcal{D}^l := \{\mathcal{D}^{l,i}\}_{i=1,\ldots,n_l}$$

can easily be defined for every $l = 0, \ldots, L$ as follows:

$$\mathcal{D}^{l,i} := \{x \in \Omega \mid \Psi^{l,i}(x) \geq \Psi^{l,j}(x) \, \forall j = 1, \ldots, n_l\}$$

In other words, a domain $\mathcal{D}^{l,i}$ on level $l$ is the set of points where the basis $\Phi^{l,i}$ is dominant on that level.

Now, several observations can be made:

- The domains on different scales need not be *strictly* spatially nested – the supports of the shape functions are, but the decomposition arising from the maximum property is not. However, the domains are clearly aligned to the flow field.

- All domains on a given level $l$ have comparable sizes and the average domain size is reduced by a factor, roughly equal to 2, from level $l$ to level $l + 1$. These properties are inherited from the bottom-up coarsening scheme used by the AMG method.
- The clustering of the field $v_1 = (-1, 1)$ (Fig. 17 top row) is perfectly aligned with the field (cf. the basis functions in Fig. 18 top row). However, the clustering of the field $v_2 = (1, 1)$, although very similar, is less regular (Figs. 17 and 18 bottom row). This is the unavoidable impact of the underlying operator discretization (which is here a mesh containing triangles). Since $v_2$ is perpendicular to the initial mesh edges, this is the worst-case scenario. However, even in this case, the constructed domains are still very much aligned with the field.
- The supports of the basis functions, respectively the induced domains on a given level, do not have *exactly* the same size (area), since AMG cannot evaluate (integrate) the *mass* of the basis functions. Indeed it does not employ any *geometric* nodal information, but only a matrix of coupling strengths. However these restrictions cause no practical problems for visualizing real-world datasets.

Finally we can show the color-coded domains and in addition velocity-colored curved arrow icons (cf. [12, 35]). For every domain $\mathcal{D}^{l,i}$, we draw one such icon, using a streamline seeded at the point where the corresponding basis is maximum. Figure 19 shows the decomposition of a magneto-hydrodynamics (MHD) flow dataset.

## 9.4 Clustering 3D Flow Fields

Our method works identically for 3D (volumetric) vector fields. The only difference is the use of tetrahedral, instead of triangular, meshes. However, direct visualization of a color-coded domain decomposition, as in the 2D case, is not effective due to the volumetric occlusion. Hence, we use a few post-processing steps. For every domain $\mathcal{D}^{l,i}$ on a given level $l$, we construct a closed triangle mesh that bounds $\mathcal{D}^{l,i}$. Next, we smooth these meshes using, e.g. a Laplacian filter or a windowed sinc filter [30]. As a result, the meshes become slightly smaller, which allows us to better separate them visually. Next, we implement an interactive navigation scheme in which domains $\mathcal{D}^{l,i}$ can be made half or completely transparent by a mouse click. Users can interactively "carve" into the flow volume to, e.g. remove uninteresting areas and bringing inner flow structures into sight, see Fig. 20. Alternatively, we can visualize the flow at a given level of detail using the same colored arrow glyphs as in the 2D case. Figure 20 shows the first three coarsest decomposition levels of a 3D helix flow and of a 3D laminar flow with $v = (1, 1, 1)$ respectively. We use the interactive technique sketched above to remove the outer domains and to expose the more interesting inner flow structure. The remaining smoothed domains are shown in the top row of Fig. 20 for the helix flow and in the bottom trow of Fig. 20 for the laminar flow (compare the latter with the 2D field in Fig. 17). In the center row of Fig. 20 the same domains as in the top row are shown, but this time half transparent and equipped with an arrow icon.
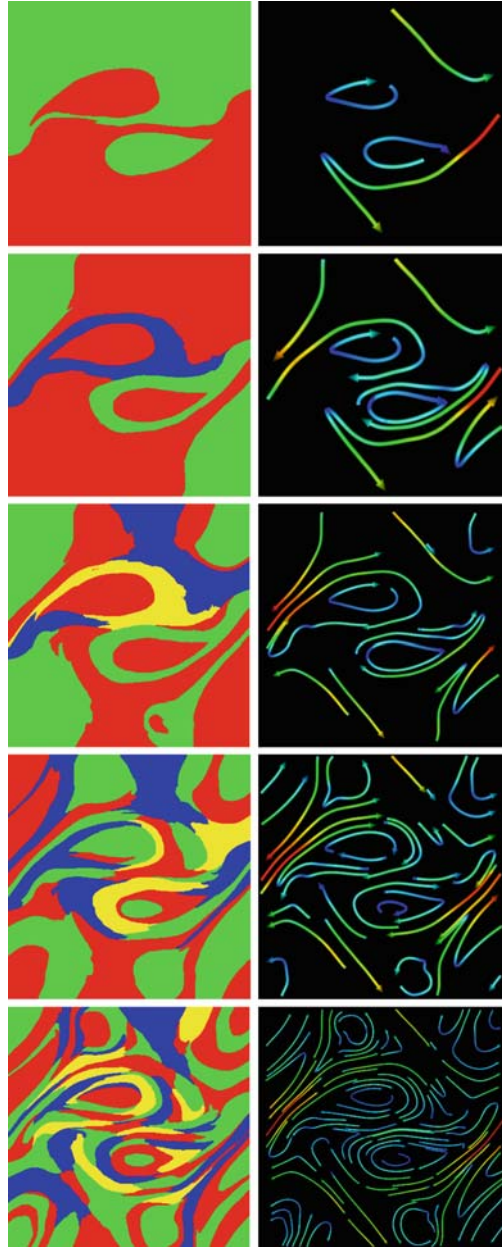
**Fig. 19.** For the vector field from a magneto-hydrodynamics simulation (MHD) shown in Figs. 4 and 5 the hierarchical decomposition is shown. *From top to bottom* the clusters on the five coarsest levels are indicated with a color-coding (*left*). The origins of the cluster-corresponding basis functions serve as the starting point for the integration of trajectories (*right*)
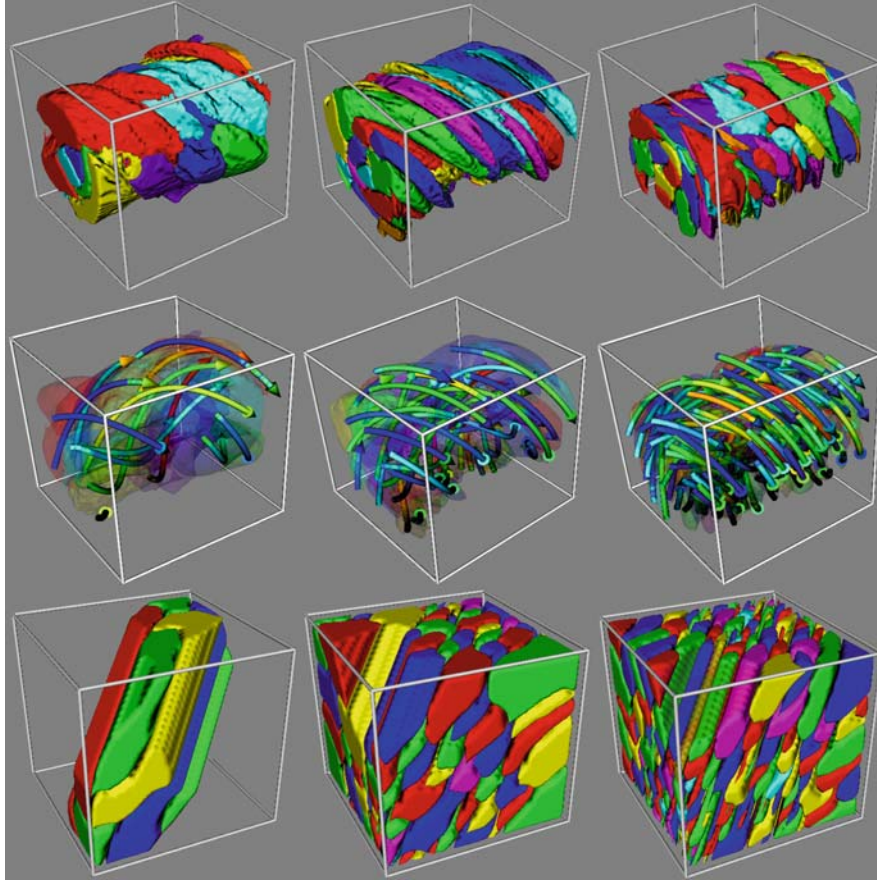
**Fig. 20.** Helix flow, selected domains (*top row*), half-transparent domains with *arrow icons* (*middle row*). Diagonal flow, selected domains (*bottom row*)

Finally, we consider the incompressible flow in a water basin with two interior walls, an inlet and an outlet – the same dataset as shown in Fig. 7. Figure 21 shows several multi-scale levels, visualized with curved arrow icons. These images show that our method scheme works in 3D just as well as in 2D.

### 9.5  Clustering Vector Fields on 2D Surfaces

For the clustering approach we considered vector fields on Euclidean domains so far. Since we have seen in Sect. 5.3 that we can extend to differential operator towards surfaces, we can apply the same generalization to the AMG clustering as well. Again we replace the Euclidian gradient and divergence operators by their geometric counter parts and apply the AMG to a discretization of

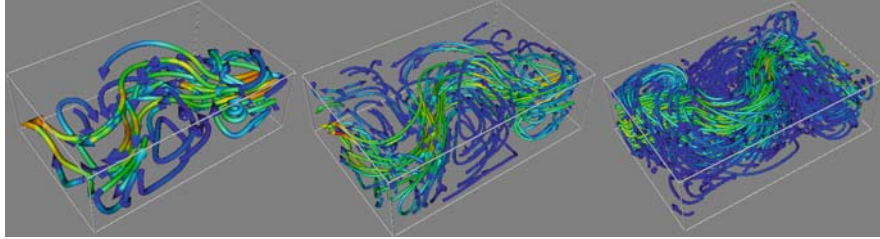$$\mathcal{A}[u] := -\text{div}_{\mathcal{M}}(A\nabla_{\mathcal{M}}u).$$

**Fig. 21.** For the water basin dataset (cf. Fig. 7) we show the three coarsest levels of the hierarchical decomposition
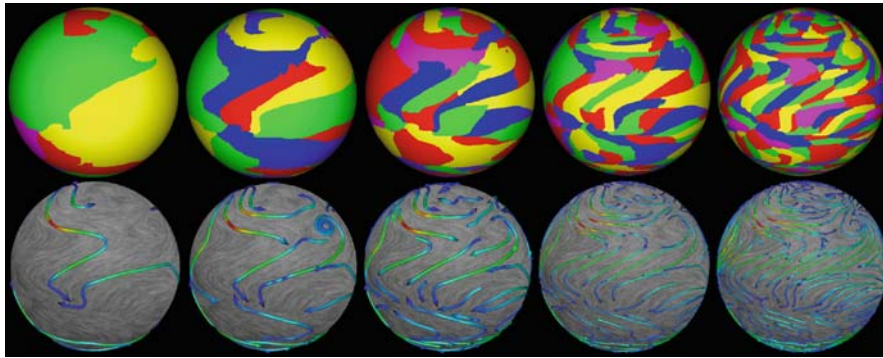


**Fig. 22.** Climate dataset decomposition, five coarsest levels (*left to right*). Domains (*top row*) and flow texture overlaid with *curved arrow icons* (*bottom row*)

The finite element discretization is now completely analogous to the above Euclidean case. In fact, we use exactly the same code for all our applications. We approximate the surface $\mathcal{M}$ by a triangulation $\mathcal{M}_h$ and compute in the same way as on flat domains the stiffness matrix $A$ corresponding to the operator $\mathcal{A}$.

As an illustration, we show the multi-scale decomposition of the average wind stress field on the surface of the Earth in Fig. 22 (the dataset is taken from [42]). The flow texture in the bottom row was produced with the IBFV method described in [42].

## 10 Conclusions

We have presented an overview of flow field visualization methods using partial differential equations (PDEs). These methods cover a broad area between pure simulation of physical processes based on such equations, and pure post-processing of such simulation data obtained by other techniques. PDE-based visualization methods have a number of strong advantages. First, they are dense methods that produce visualizations where every point of the considered spatial domain is represented. This naturally associates them with, and brings them close to, texture-based visualization

methods. Second, PDE-based visualizations can naturally target any spatial dimension, e.g. from 2D to time-dependent 3D datasets, in a uniform modeling and computational framework. Third, one can use existing, well-proven and well understood numerical techniques to solve the underlying PDE discretizations, yielding an overall robust approach to data visualization. Fourth, many such methods have a natural support of the multiscale notion, being able to capture and represent flow data details at different spatial scales. Often, time serves as the parameter controlling the scale. However, probably the most attractive aspect of PDE-based visualizations is their ability to model a wide range of phenomena and data enhancement operations, ranging from simple filtering to sophisticated multiscale feature-preserving decomposition techniques, in a well-founded mathematical way, that leads to novel and insightful visualizations.

# References

1. L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations of image processing. *Arch. Ration. Mech. Anal.*, 123 (3):199–257, 1993.
2. A. Brandt. Algebraic multigrid theory: the symmetric case. In *Preliminary Procs. of the Intl. Multigrid Conf.*, Copper Mountain, Colorado, April 1983.
3. A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*. Cambridge University Press, Cambridge, 1984.
4. B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 263–272, Aug. 1993.
5. J. Cahn and J. Hilliard. Free energy of a non-uniform system I. Interfacial free energy. *J. Chem. Phys.*, 28:258–267, 1958.
6. W. de Leeuw. Divide and conquer spot noise. In *Proceedings of Supercomputing 97 (CD-ROM)*. ACM SIGARCH and IEEE, 1997.
7. W. de Leeuw and R. van Liere. Multi-level topology for flow visualization. *Comput. Graph.*, 24(3):325–331, 2000.
8. W. C. de Leeuw and J. J. van Wijk. Enhanced spot noise for vector field visualization. In *Proceedings Visualization '95*, 1995.
9. U. Diewald, T. Preusser, and M. Rumpf. Anisotropic diffusion in vector field visualization on euclidean domains and surfaces. *IEEE Trans. Vis. Comput. Graph.*, 6(2):139–149, 2000.
10. C. M. Elliott. The Cahn–Hilliard model for the kinetics of phase separation. *Num. Math.*, pages 35–73, 1988.
11. L. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In *Proceedings IEEE Visualization '94*, pages 240–246, 1994.
12. H. Garcke, T. Preusser, M. Rumpf, A. Telea, U. Weikard, and J. J. van Wijk. A phase field model for continuous clustering on vector fields. *IEEE TVCG*, 7(3):230–241, 2000.
13. M. Griebel, A. Schweitzer, T. Preusser, M. Rumpf, and A. Telea. Flow field clustering via algebraic multigrid. In *Proceedings of IEEE Visualization 2004*, 2004.
14. B. Heckel, G. Weber, B. Hamann, and K. I. Joy. Construction of vector field hierarchies. *Proceedings IEEE Visualization '99*, pages 19–25, 1999.

15. J. L. Helman and L. Hesselink. Visualizing Vector Field Topology in Fluid Flows. *IEEE Comput. Graph. Appl.*, 11(3):36–46, 1991.

16. V. Interrante and C. Grosch. Stragegies for effectively visualizing 3D flow with volume LIC. In *Proceedings IEEE Visualization '97*, pages 285–292, 1997.

17. B. Jobard, G. Erlebacher, and Y. M. Hussaini. Lagrangian-eulerian advection for unsteady flow visualization. In *Proceedings of IEEE Visualization '01*, San Diego, CA, October 2001.

18. B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In W. Lefer, M. Grave (eds.), *Visualization in Scientific Computing '97*, 1997.

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

20. R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Comput. Graph. Forum*, 23(2):203–221, 2004.

21. R. S. Laramee, D. Weiskopf, J. Schneider, and H. Hauser. Investigating swirl and tumble flow with a comparison of visualization techniques. In *Proceedings of IEEE Visualization '04*, pages 51–58, 2004.

22. H. Löffelmann, T. Ku$\nu$cera, and E. Gröller. Visualizing Poincaré maps together with the underlying flow. In *Mathematical Visualization*, pages 315–328. Springer, Berlin, 1998.

23. N. Max and B. Becker. Flow visualization using moving textures. In *Proceedings of the ICASE/LaRC Symposium on Visualizing Time-Varying Data*, pages 77–87, 1995.

24. N. Max, R. Crawfis, and C. Grant. Visualizing 3D Velocity Fields Near Contour Surface. In *Proceedings IEEE Visualization '94*, pages 248–254, 1994.

25. A. Novick-Cohen. The Cahn–Hilliard equation: mathematical and modelling perspectives. *Adv. Math. Sci. Appl.*, 8:965–985, 1998.

26. P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:629–630, 1990.

27. C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive exploration of volume line integral convolution based on 3D texture mapping. In *Proceedings IEEE Visualization '99*, pages 233–240, 1999.

28. J. W. Ruge and K. Stüben. Efficient Solution of Finite Difference and Finite Element Equations by Algebraic Multigrid. In D. J. Paddon and H. Holstein, editors, *Multigrid Methods for Integral and Differntial Equations*, The Institute of Mathematics and its Applications Conference Series. Clarendon, Oxford, 1985.

29. R. Wegenkittl and E. Gröller. Fast oriented line integral convolution for vector field visualization via the internet. In *Proceedings IEEE Visualization '97*, pages 309–316, 1997.

30. W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, New Jersey, 1995.

31. H. W. Shen and D. L. Kao. UFLIC: a line integral convolution algorithm for visualizing unsteady flows. In *Proceedings IEEE Visualization '97*, pages 317–323, 1997.

32. D. Stalling and H.-C. Hege. Fast and resolution independent line integral convolution. In *SIGGRAPH 95 Conference Proceedings*, pages 249–256. ACM SIGGRAPH, Addison-Wesley, New York, 1995.

33. D. Stalling, M. Zöckler, and H.-C. Hege. Fast display of illuminated field lines. *IEEE Trans. Vis. Comput. Graph.*, 3(2), 1997. ISSN 1077-2626.

34. A. Telea and J. J. van Wijk. 3D IBFV: Hardware-accelerated 3d flow visualization. *Proceedings IEEE Visualization '03*, pages 223–240, 2003.

35. A. C. Telea and J. J. van Wijk. Simplified representation of vector fields. *Proceedings IEEE Visualization '99*, pages 35–42, 1999.

36. V. Thomée. *Galerkin – Finite Element Methods for Parabolic Problems*. Springer, Berlin, 1984.
37. U. Trottenberg, C. W. Osterlee, and A. Schüller. *Multigrid*, Appendix A: An Introduction to Algebraic Multigrid by K. Stüben, pages 413–532. Academic, San Diego, 2001.
38. G. Turk and D. Banks. Image-guided streamline placement. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 1996.
39. T. Urness, V. Interrante, I. Marusic, E. Longmire, and B. Ganapathisubramani. Effectively visualizing multi-valued flow data using color and texture. In *Proceedings IEEE Visualization '03*, pages 115–122, 2003.
40. J. J. van Wijk. Spot noise-texture synthesis for data visualization. In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 309–318, July 1991.
41. J. J. van Wijk. Image based flow visualization. *Computer Graphics (Proc. SIGGRAPH '02), ACM, pp. 263–279*, 2001.
42. J. J. van Wijk. Image based flow visualization for curved surfaces. *Proceedings IEEE Visualization '03*, pages 123–130, 2003.
43. R. Wegenkittl, E. Gröller, and W. Purgathofer. Animating flow fields: Rendering of oriented line integral convolution. In *CA '97: Proceedings of the Computer Animation*, pages 15–21, 1997.
44. J. Weickert. *Anisotropic diffusion in image processing*. Teubner, Stuttgart, 1998.
45. D. Weiskopf and G. Erlebacher. Flow visualization overview. In *Handbook of Visualization*, pages 261–278. Elsevier, Amsterdam, 2005.
46. D. Weiskopf, G. Erlebacher, and T. Ertl. A texture-based framework for spacetime-coherent visualization of time-dependent vector fields. In *Proceedings IEEE Visualization '03*, pages 107–114, 2003.
47. D. Weiskopf, G. Erlebacher, M. Hopf, and T. Ertl. Hardware-accelerated lagrangian-eulerian texture advection for 2D flow visualizations. In *Proceedings of the Vision Modeling and Visualization Conference 2002 (VMV-02)*, pages 439–446, 2002.