Scientific Visualisation in Pollutant Transport Modelling

Szécsi László
s8707sze@hszk.bme.hu

Computer Graphics Research Group
Budapest University of Technology and Economics
Budapest / Hungary

Abstract

This paper examines the applicability of various visualisation methods for displaying results of finite element flow simulations. After a brief overview of transport phenomena, their modelling equations and finite element modelling, the special options and requirements of visualisation are discussed. Thereafter, various possible methods are evaluated for every type of data calculated. Finally, rendering techniques and additional effects are explained.

KEYWORDS: finite element, mesh generation, illuminated streamlines, fake Phong method.

1 Introduction

Modelling transport phenomena, especially the transport of pollutants, grew more and more important as the aspects of environmental protection and sustainable development became imperative. Industrial decision making relies on the ability to forecast environmental impacts, and that is accomplished by modelling and simulation. Impact assessment is usually a major task involving expensive surveying and laboratory measurements, therefore much effort is given to make the most use of the acquired data, including vast computer capacities and scientific work on simulation software. Graphic display of the results is also involved, as it is beneficial for both the expert staff and the decision-makers to have a visual perception of the facts. It helps find the errors, gives a qualitative description, and makes the results easier to understand.

In this paper we will present the algorithms and techniques developed for a specific expert system for environmental decision making. That research project is led by Prof. Miklós Bulla at the University of Applied Sciences in Győr, Hungary. The images will also be taken from that application. Consequently, we will focus on soil and groundwater pollution, that is, groundwater flow and solute transport modelling, and the finite element way of solution. Visualising the discrete results characterising the finite element approach involves both limitations and possibilities. We will show how more or less known display methods can be applied.

2 Modelling transport phenomena

Generally, the objective is to calculate three-dimensional fields of future hydraulic pressure, groundwater flow velocity, contaminant concentrations and contaminant transport fluxes. The input data for the simulation involves physical and chemical features of the interacting materials, such as soil porosity, hydraulic conductivity, pollutant decay rate, adsorption function, viscosity, soil and liquid compressibility. Specifying mass sources and sinks as marginal conditions and starting values as initial conditions are also necessary. Summing that up, if we are able to measure the current groundwater and contaminant distribution, to survey the soil layer geometry, and the contaminant sources are also known, we should be able to predict future pollutant distribution. That makes it possible to tell how far the pollution will advance in given time, or what concentration it will reach at a specified location.

2.1 Fundamental balance statement

Continuous physical systems are mostly modelled using partial differential equations, and that also applies to flow and contaminant transport phenomena. Let us consider an infinitesimal volume, and formulate the law of mass conservation for it [1].

$$\frac{\partial \rho \Psi}{\partial t} = -\nabla \cdot (\rho \Psi v) - \nabla \cdot j + \rho f$$

Ψ may be any balance quantity, in our case hydraulic potential h or contaminant concentration C. ρ is density, v is media velocity, j is the mass flux and f covers any additional mass sinks or sources. The equation does mean nothing more than that the change of the mass within the volume equals the mass moving into it minus the mass moving out, modified by the yield of any internal source or sink. The actual mass fluxes can be calculated using famous laws of flow and transport.

2.2 Equations for groundwater flow

In the case of groundwater flow, the pressure differences drive water towards lower potential. We assume that the flow is slow enough to neglect inertial terms. Darcy's law describes exactly that situation[1].

$$\epsilon v = -\underline{K}(\nabla h - \Theta \xi)$$

$\underline{K}$ is the conductivity tensor, ξ is the direction of gravity and Θ is a soil parameter called tortuosity. How much water is stored in the volume depends on the pressure there, as the liquid and solid material may both be compressed. Therefore, the change of mass depends on the change of pressure. The relation is described by the storativity S. Substituting those into the basic balance equation[1]:

$$L(h) = S\frac{\partial h}{\partial t} - \nabla \cdot (\underline{K} \cdot (\nabla h + \Theta \xi)) - \epsilon Q_\rho = 0$$

ϵQ_ρ represents additional sinks or sources.

2.3 Equations for pollutant transport

The case of pollutant transport is somewhat more complicated, as there are more phenomena to consider. The basic type of pollutant movement is advection, meaning that the pollutant moves along with the water. Molecular diffusion and macro-

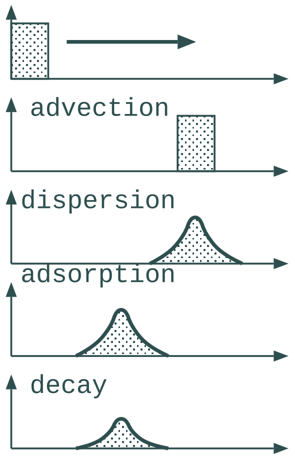

Figure 1: Transport Phenomena

scopic soil structure artifacts making the flow non-unidirectional cause that the pollution tends to expand in space. That is the phenomenon called dispersion. Adsorption means that the soil is able to gather pollutants, removing them from the solute. This is a two-way process, where the ratio of solved and adsorbed material depends on the concentration as defined by the sorption isoterms. The material removed form the solute is represented in the balance equation. In addition, pollutants may decay as time passes. This happens both to solved and adsorbed material, however, the rate of decay in the two phases is significantly different, and the decay in the solid phase may be neglected, staying on the safe side. The final equation containing all factors of transport looks like[1]:

$$\Re_d \frac{\partial C}{\partial t} + \epsilon v \nabla \cdot C + \nabla \cdot (\epsilon D \cdot \nabla C) + (\epsilon Q_\rho + \Re \vartheta) C + Q_C = 0$$

The first term stands for adsorption, the $\Re_d$ derivative retardation is similar to the storativity S in the flow equation. The second term represents advection, the third is for dispersion, with D dispersion tensor calculated from molecular diffusion, longitudinal and transversal diffusion coefficients. The last two terms cover contaminant mass change from pollutant sources, polluted water entering or leaving the volume and decay, with decay coefficient ϑ and solved material ratio $\Re$.

3 The finite element method

Both equations explained above are differential equations with respect to time and space. In order to solve these with numerical methods, we have to select a series of time instances and a finite set of variables to describe an approximate solution.

In other words, both temporal and spatial discretisation is necessary. There is no difficulty in the one-dimensional time domain, it is sufficient to introduce an appropriate time step. However, the segmentation of the usually non-homogenous three-dimensional space is a problem of great concern.

The finite difference, finite volume and finite element methods differ in the way the values of the selected computational variables are related to the approximate continuous solution. In case of the most general finite element method, the space is split up into small volumes, called elements. Within each element, the values are approximated as a linear combination of weighting or shape functions, where the coefficients are the computational variables [1].

If we substitute this approximate solution back into the flow and contaminant transport differential equations, the balance will not be identically equal to zero any more. What we get is the error function, a continuous function containing the computational variables. From this point, the task is to find those values for the variables that result in minimal error in some sense. Using the Galerkin method this can be done by solving a linear system of equations for every time step. However, the details exceed the scope of this paper.

3.1 The finite element mesh

In order to understand what the finite element mesh is, we have to have a look at the shape functions and the direct meaning of computational values. The construction of the approximate solution can also be considered an interpolation, where the variables represent values at given points of the space, and the shape functions are the weighting functions. In this aspect, every variable has a point associated to it at the maximum of the weighting function, called a node in the finite element mesh. During the calculation, it is crucial to identify the nodes the values of which contribute to the value at a given point. The set of points that share the same influent nodes is called an element. Using the usual linear shape functions, elements will be convex polyhedra with nodes at the finite element nodes. Finding the most appropriate nodes and elements, the process called mesh generation, actually means selection of the shape functions. Therefore, it influences the composition of the approximate solution and thus, the accuracy of the computation[3].

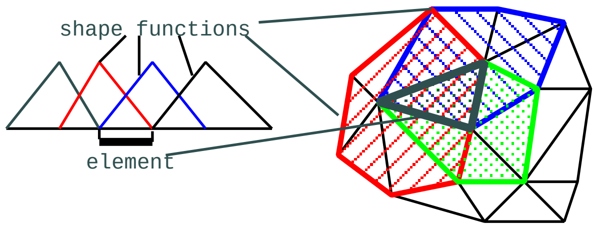

Figure 2: Shape functions and elements in one and two dimensions

A mesh is composed of non-overlapping elements that cover all of the examined space without holes. Because the shape functions should be as symmetrical as possible, the touching elements should not have significantly different size or shape. However, as the computation is more accurate where the elements are smaller, it should be possible to locally control the accuracy via a gradual change in element size. It is also important that the elements fit the bounding surface, or the surfaces separating homogeneous volumes, because elements are treated as a unit in the calculations and handling inhomogeneous elements is therefore complicated. It is difficult to meet all these requirements with a regular mesh in which the connections between nodes and elements follow some simple rule. As there is no advantage taken of mesh structure in the finite element method, irregular meshes are generally used. Generation algorithms either aim to create a hexahedral element mesh easy to comprehend, or make use of the flexibility of Delaunay meshes composed solely of tetrahedra.

Generally, what we are left to work with after the solution of the equations is a mesh of nodes and elements, with known connections, and time series of various data associated with the nodes or the elements. Some of them are scalar values like saturation or concentration, but there also are vectors like flow velocities. Moreover, values calculated at discrete nodes are related to the approximate continuous solution we may also wish to display. Further on in this paper, we will make use of that structure to visualise the results.

4 Platform issues

The mesh generation and computation packages are usually written in pure standard Fortran, C, or as in the case of the implementation that we used for research, in C++. They are not dependent on operating system or hardware, and they are actually run on a large scale of computers. Visualisation, and computer graphics in general, however, strongly relies on hardware. Although it is not a necessity to do the calculations and the display on the same computer, it was a priority to maintain platform independence in both modules. Therefore, OpenGL was used to render everything displayed, including the graphic user interface elements.

5 Basic visualisation methods

First of all, it is important to be able to display the mesh itself. It gives a good idea of the whole simulation process and makes node density, element shape quality or even mesh errors visible. It is also possible to relate other displayed data to the mesh. Secondly, the user should be able to check and edit the spatially referenced input data, e.g. local parameters as hydraulic conductivity or porosity. A step further is the visualisation of discrete scalar values that change over time. That includes the boundary conditions and the calculated hydraulic pressure and contaminant concentration values. This also covers the initial data, of course. However, the results may also be interpreted as the representation of a continuous three-dimensional field of scalar values. This is the level where we actually draw something resembling the physical process modelled. Finally, the display of a vector field requires us to force the most information into the two dimensions of the screen.

5.1 Depth perception

The basic problem with displaying three-dimensional data is the dimension we do not have on the screen, namely depth. Firstly, the viewer has to be able to get an idea how far a displayed object is, or at least, which of two objects is further. In order to achieve that, perspective projection, depth cueing, colour coding of depth and, most efficiently, lighting can be used. Secondly, objects that would take up the same area of the screen should all be visible at the same time. To solve this, objects should be made transparent, either by colour blending, or by avoiding solid opaque objects.

5.2 Direct display of mesh-related data

It is quite simple to display the mesh itself. It is just as hopeless as it is unnecessary to achieve a complete comprehension of an irregular, perhaps even tetrahedral mesh. However, it is possible and sufficient to render the nodes as points or small objects to show the node density distribution, element edges as lines to give an impression of the structure, or element faces as polygons to show the bounding surface. All of the methods to help depth perception described above can be used. As all of the discrete data is associated to mesh components, the display of the mesh is the basic tool to show simple data like discrete local parameters or concentration values. These may be coded as the colour, size or width of the displayed objects.

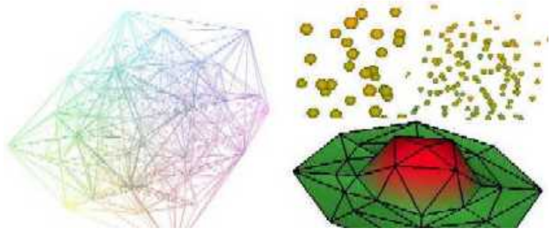

Figure 3: Mesh nodes (poor depth perception), mesh edges (overstuffed image), and a bounding surface made up of mesh polygons

5.3 Visualisation as a tool for data analysis

Obviously, the simple method does not decrease the amount of data channelled to the screen. It is quite possible that the numerous less important nodes make it impossible to observe intriguing results in the centre of the area. The objective of more sophisticated techniques is to provide the user with tools to answer his specific questions. In this way, visualisation is a powerful method of data analysis.

The most straightforward idea is to display the mesh and thus the associated data only partially. At the same time, it is inconvenient to be restricted by the discrete nature of the elements, and questions the user asks usually concern the continuous result, not values at specified points. Theoretically, we should use the shape functions of the finite element approximation to obtain the values, but it is sufficient to use linear interpolation. In fact, the two are exactly the same in the case of the generally used first-order weighting functions.

5.4 Cross-section

The easy way to simplify the three-dimensional problem is to select a plane to investigate. In order to display data along such a chosen cross-section, the points where the plane intersects the edges of the elements should be determined. The values there are given by the interpolation of nodal values along the edges. Then, for every element, the intersected polygon can be drawn. If the colours at the vertices represent the values on a linear scale, the linear interpolation of the colour performed by OpenGL when rendering the polygon will make the image match the real approximate solution.

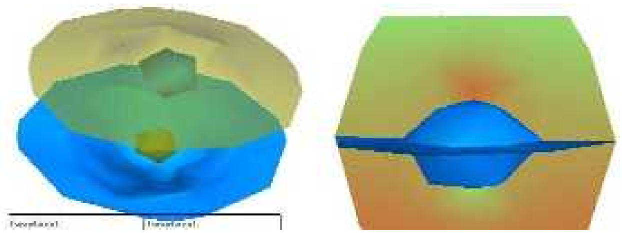

Figure 4: Isosurface and cross-section image displaying hydraulic pressures around a lens of confining material

5.5 Isosurface

The user may also be interested in finding points that fulfil a certain condition, e.g. what volume is fully saturated, or where confining rock layers are. Such a query would split up space into two volumes that can be shown by only drawing the surface separating them. Such a surface can always be considered an isosurface of a value, as it shows answers to questions like where contaminant concentration over a given value is, or where the hydraulic potential exactly zero is. For an isosurface it is obviously insufficient to know the values only at the nodes, we have to use interpolation similar to the cross-section case. Via interpolation it is easy to find where the surface would intersect the edges, and display the polygons making up the surface in every element. Colour coding of the value naturally has little relevance here, as long as it is not the value some different variable. It is useful, for instance, to show saturation values along a layer boundary, which can also be considered to be an isosurface, namely the isosurface of the layer identifiers assigned to the nodes.

6 Volume rendering

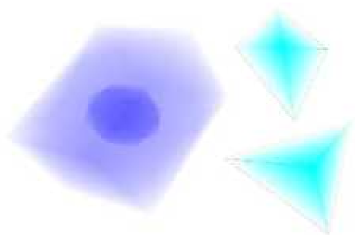

Figure 5: Volume visualisation using transparent elements

Although drawing well-selected surfaces is extremely useful, there are also various techniques to render the volume of objects. These make extensive use of transparency to give an impression of value distribution. Values are coded as the opacity of the media, resulting in an X-ray like image. Usually, volumes are rendered with the use of particle systems or volume tracing. However, in case of finite element data, these time-consuming methods may be substituted with a simpler one if we render elements one by one. Assuming that the value we are to display is constant in each element, a single more-or-less transparent simplex is not extremely hard to draw. Within the polygonal segments formed by overlapping front and back facets the thickness of the material, and so its opacity changes linearly. These polygons should be rendered beginning form the furthest element, with appropriate opacity values assigned to the vertices. Naturally, these values, and the vertices themselves depend on the viewing angle, and so the polygons have to be recalculated for every new camera vector. Unfortunately, the method fails as soon as the value within an element is not constant, because the opacity does not change linearly then.

7 Visualising vector fields

The above methods are all well applicable to scalar data, but fail to display vector fields. It is possible, and sometimes useful to display only the magnitude of the vectors, but that gives no idea about the direction of the flow or the path of material transport. One approach would be to colour code directions, but reading that needs an experienced eye, and in most cases we have to trade in the display of the magnitude in exchange. The solution here is the same as before, we have to select a set of points that represent the entire flow.

7.1 Hedgehog plot

Firstly, we may use a three-dimensional extension of the hedgehog plot, that is, we place arrows pointing towards the direction of the flow. However, the perception of depth is worse in case of separate arrows than in case of surfaces, and the image may easily be overloaded.

A similar technique is the use of streamlets or streamlines. However, using lighting, transparency and a good selection of starting points it is possible to improve depth perception. It is primarily the lighting of the streamlines that makes this technique applicable in three dimensions. However, a large number of streamlets are usually needed in order to follow flow around divergence nodes and to give a surface-like impression necessary for a good perception of depth. This means that modelling streamlines with thin tubes made up of polygons does not allow for rendering fast enough. Therefore, streamlets should be drawn as lines, but unfortunately, graphic libraries do not directly support shading lines.

7.2 Using textures for shading

The solution that yields a result equivalent to the thin tube approach is actually a variant of fake Phong shading applied to lines. The fake Phong method uses the hardware-accelerated texturisation for shading. The idea is to let the graphics library and the hardware transform and linearly interpolate some adequate values, and store the corresponding pre-calculated light intensities in a texture. The texture transformation matrix mechanism allows us to calculate up to three dot products and use them as texture co-ordinates. The simplified shading equation for local lighting is:

$$L = k_a + k_d \vec{L} \cdot \vec{N} + k_s (\vec{R} \cdot \vec{V})^n$$

k_a, k_d, and k_s are constant coefficients, characterising the material. $\vec{L}$ is the light vector, $\vec{V}$ is the view vector, $\vec{N}$ is the surface normal and L is light intensity or radiance. The reflection vector $\vec{R}$ may be expressed as:

$$\vec{R} = 2\vec{N}(\vec{L} \cdot \vec{N}) - \vec{L}$$

That way the radiance is expressed as a function of $\vec{L} \cdot \vec{N}$ and $\vec{V} \cdot \vec{N}$:

$$L = k_a + k_d \vec{L} \cdot \vec{N} + k_s \Big((2\vec{N}(\vec{L} \cdot \vec{N}) - \vec{L}) \cdot \vec{V} \Big)^n = k_a + k_d \vec{L} \cdot \vec{N} + k_s \Big(2(\vec{V} \cdot \vec{N})(\vec{L} \cdot \vec{N}) - \vec{L} \cdot \vec{V} \Big)^n$$

It is possible to use these two values as texture co-ordinates, however, the approximation will not be perfect. Although the interpolation of the dot products is equivalent to the interpolation of the normal vector, there is no possibility to normalise the interpolated vector for every pixel. This method is usually called flat Phong shading. If the normal vector is set as the vertex texture co-ordinate for every vertex, the dot products are evaluated and scaled into the [0,1] domain by the texture transformation itself, provided the transformation matrix is set up as follows:

$$M_{text} = \frac{1}{2} \begin{pmatrix} L & 1 \\ V & 1 \\ \bar{0} & 0 \\ \bar{0} & 2 \end{pmatrix}$$

Unfortunately, it is not possible to completely eliminate the normalisation problem without multiple three-dimensional textures. However, a different approximation is obtained if we do not interpolate the normal itself, but two other vectors perpendicular to each other and the normal, and applying the Pythagorean theorem.

If both diffuse and specular lighting are needed, this still requires a huge three-dimensional texture. The method is called fake Phong shading [2], and although it tends to create exaggerated lights in darker areas, it is more accurate than flat Phong shading near the more important highlighted spots.

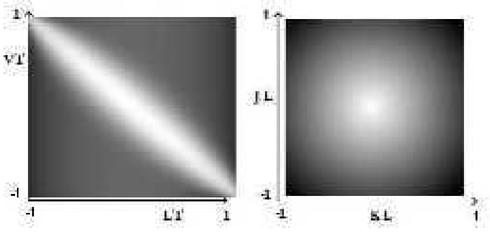

Figure 6: Textures used for illuminated lines[5] and in fake Phong shading[2]. J and K vectors are perpendicular to the surface normal.

7.3 Illuminated streamlines

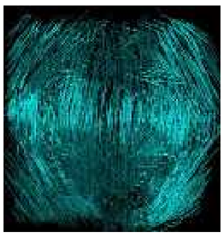

Figure 7: Illuminated streamlines

How the shading method described above can be applied to lines was described by Zöckler [5]. In the case of lines the normal vector is not unique, and it is of course not sufficient to select an arbitrary one. If the line was a thin tube, intensities for all the possible normal vectors could be visible. It is of course the most intense value that catches the eye, and therefore we have to choose the normal vector corresponding to that value. That vector is the normal coplanar to the light vector $\vec{L}$ and the tangent $\vec{T}$. Luckily, $\vec{L} \cdot \vec{N}$ and $\vec{R} \cdot \vec{N}$ can be expressed using $\vec{L} \cdot \vec{T}$ and $\vec{V} \cdot \vec{T}$:

$$\vec{L} \cdot \vec{N} = \sqrt{1 - (\vec{L} \cdot \vec{T})^2}$$

$$\vec{R} \cdot \vec{N} = (\vec{L} \cdot \vec{T})(\vec{V} \cdot \vec{T}) - \sqrt{1 - (\vec{L} \cdot \vec{T})^2}\sqrt{1 - (\vec{V} \cdot \vec{T})^2}$$

Furthermore, if the tangential is interpolated instead of the normal, the inaccuracy of the interpolation becomes less significant. The $\vec{L} \cdot \vec{T}$ and $\vec{V} \cdot \vec{T}$ products will contain some error when the light and the streamline, or the eye vector and the streamline are nearly parallel. This means, that the line should either be dark or barely visible at all. The trick is actually not different from that of fake Phong shading, the dot product is not calculated for the normal, but for a perpendicular vector. However, because of the clever selection of the normal, the three-dimensional texture map is not necessary.

If the texture vector associated with the vertices is the tangent vector, the dot products are evaluated via the transformation. The luminance map is filled up according to the radiance formula:

$$L(LT, VT) = k_a + k_d\sqrt{1 - (\vec{L}\cdot\vec{T})^2} + k_s\left((\vec{L}\cdot\vec{T})(\vec{V}\cdot\vec{T}) - \sqrt{1 - (\vec{L}\cdot\vec{T})^2}\sqrt{1 - (\vec{V}\cdot\vec{T})^2}\right)^n$$

7.4 Streamline seeding

It is vital to choose the starting points of streamlines well. The lines should be evenly distributed on the screen, and it is not enough to spread the seed points evenly to achieve that. That is because streamlet length may be different and there may be divergence and convergence areas influencing line density. It is usually necessary to apply sophisticated flow-guided seeding algorithms. However, in case of the finite element data, it is possible to count the number of intersecting streamlines for every element to determine if any more lines in that region are necessary. Furthermore, the density may be controlled locally in every element using data like flow velocity.

7.5 Stream surfaces and stream volumes

A stream surface is similar to a streamline, but it does not start from a point, but from a line segment. The line segment moving with the flow will sweep along a surface depicting the flow well. As of now, this feature is not included in the application, due to the fact that there was no real need to display complex flows. Because of the Darcy assumption, there are no whirls or other higher-order phenomena. How stream surfaces can be used at best efficiency was described by Löffelman [4].

Stream volumes show what volume the material travelling with the flow would occupy. They are quite expensive to calculate; it usually includes tracing several particles and casting a surface around them. However, it is easy to notice that the pollutant transport calculation, which is quite expensive indeed, already have done the job. There is no real need for stream volumes, because adding a theoretical pollutant source and displaying the isosurface of the concentration does exactly the same.

8 Future aspects

There are two ways of further development. Firstly, the presently available methods should be improved, and secondly, new features should be added to match the future improvements of the flow simulation engine.

It will be important to find the most convenient parameter set for the visualisation methods. This will include setting up clipping planes or displaying only elements that meet a certain condition.

More sophisticated streamline seeding methods will be tested to achieve a better depiction of the flow. These will concentrate on eliminating streamlines that do not tell much about the global characteristics of the flow but make the screen overstuffed.

It is one of the most intriguing questions how the volume visualisation method described above could be extended to elements of non-uniform values. It is possible that the problem will be solved using texture mapping, a bit similarly to the illuminated streamlines.

References

[1] H.-J. G. Diersch. Discrete feature modeling of flow, mass and heat transport processes by using feflow. Technical report, WASY Institute for Water Resources Planning and Systems Research Ltd., Berlin, 1999. http://www.wasy.de/eng/prodinfo/feflow/white_papers/frac/frac2.html.

[2] H. Elias. Fast phong shading. Technical report, 1995. http://freespace.virgin.net/hugo.elias/graphics/x_polyp2.htm.

[3] M. Filipiak. Mesh generation. Technical report, Edinburgh Parallel Computing Centre, The University of Edinburgh, 1996. http://www.epcc.ed.ac.uk/epcc-tec/documents/tw-meshgen/MeshGeneration.book_1.html.

[4] H. Löffelmann. *Visualizing Local Properties and Characteristic Structures of Dynamical Systems.* PhD thesis, Vienna University of Technology, http://www.cg.tuwien.ac.at/˜helwig/diss, 1998.

[5] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualisation of 3d–vector fields using illuminated streamlines. In *IEEE Visualization '96 Proceedings*, pages 107–113, 1996. http://www.zib.de/Visual/papers/islVis96/index.html.