3D Loop Detection and Visualization in Vector Fields

Thomas Wischgoll[1] and Gerik Scheuermann[1]

Computer Science Department, University of Kaiserslautern, Germany
{*wischgol,scheuer*}*@informatik.uni-kl.de*

Summary. Visualization has developed a tendency to use mathematical analysis to obtain and present important data properties. In three-dimensional fluid flows, engineers are interested in several important features. One type are recirculation zones where the fluid stays for a long time. This plays a key role in combustion problems since recirculation allows a completion of chemical reactions which usually have a smaller time scale than fluid dynamics. Strong indicators for such recirculation zones are looping streamlines in a steady vector field or in the time steps of unsteady data. The article presents a method for the detection of such loops by analyzing streamlines approaching them.

1 Introduction

Many problems in natural science and engineering involve vector fields. Fluid flows, electric and magnetic fields are nearly everywhere, so measurements and simulations of vector fields still increase dramatically. As with other data, too, analysis is much slower and needs improvement. Mathematical methods together with visualization can provide help in this situation. In most cases, the scientist or engineer is interested in integral curves of the vector field like streamlines in fluid flows or magnetic field lines. The qualitative nature of these curves can be studied with topological methods developed originally for dynamical systems. Especially in the area of fluid mechanics, topological analysis and visualization have been used with success [3], [6], [11], [15].

This article concentrates on a specific topological class of integral curves, namely loops, also called limit cycles. These integral curves are closed, so that a particle traveling along the curve loops around forever. Their importance stems from the fact that quite often neighboring integral curves either tend toward the loop or come from there (i. e. tend toward the loop after reversing the direction of time). This is a well-established result from dynamical systems theory [4], [8]. It may be noted that loops may also behave like saddles in this three-dimensional case, but then they do not indicate recirculation and therefore this case is less relevant in our context.

Several publications have dealt with related topics. Hepting et al. [7] study invariant tori in four-dimensional dynamical systems by using suitable projections into three dimensions to enable detailed visual analysis of the tori. Wegenkittel et al. [18] present visualization techniques for known features of dynamical systems. Bürkle et al. [1] use a numerical algorithm developed by

some of the coauthors [2] to visualize the behavior of more complicated dynamical systems. In the numerical literature, we can find several algorithms for the calculation of closed curves in dynamical systems [10], [17], but these algorithms are tailored to deal with smooth dynamical systems where a closed form solution is given.

Since visualization deals in many cases with piecewise linear, bilinear or trilinear data, we present an algorithm tailored to this situation that can be integrated into standard integral curve computation algorithms. While computing an integral curve, we track the visited cells checking for repetition. If we find a revisited cell, we check if the integral curve stays in the same cell cycle forever. For this, we look at the boundary of the cell cycle in question and check if the integral curve can cross the boundary. We have proposed a similar algorithm for the two dimensional case earlier [19]. In contrast to this two dimensional case where it is sufficient to find integral curves bounding the region where the integral curve can go, we have to work with integral surfaces in this article. We use a simplified version of Hultquist's algorithm [9] to construct these surfaces.

2 Mathematical Background

This section gives the necessary theoretical background and the terms for our algorithm. We restrict our consideration in this article to steady, linearly interpolated vector fields defined on a tetrahedral grid

$$v : \mathbb{R}^3 \supset D \to \mathbb{R}^3, \quad (x, y, z) \mapsto v(x, y, z).$$

D is assumed to be bounded. This is the situation for many experimental or simulated vector fields that have to be visualized. We are interested in the behavior of integral curves

$$c_a : \mathbb{R} \to \mathbb{R}^3, \qquad t \mapsto c_a(t)$$

with the properties

$$\begin{aligned} c_a(0) &= a \\ \frac{\partial c_a}{\partial t}(t) &= v(c_a(t)) . \end{aligned}$$

For Lipschitz continuous vector fields, one can prove the existence and uniqueness of integral curves c_a through any point $a \in D$, see [8], [12]. The actual computation of integral curves is usually done by numerical algorithms like Euler methods, Runge-Kutta-Fehlberg methods or Predictor/Corrector methods [16].

The topological analysis of vector fields considers the asymptotic behavior of integral curves. The α-limit set of an integral curve c is defined by $\{p \in$

$\mathbb{R}^3 | \exists (t_n)_{n=0}^{\infty} \subset \mathbb{R}, t_n \to -\infty, \lim_{n\to\infty} c(t_n) \to p\}$. The ω-limit set of an integral curve c is defined by $\{p \in \mathbb{R}^3 | \exists (t_n)_{n=0}^{\infty} \subset \mathbb{R}, t_n \to \infty, \lim_{n\to\infty} c(t_n) \to p\}$. If the α- or ω-limit set of an integral curve consists of only one point, this point is a critical point or a point at the boundary ∂D. (It is usually assumed that the integral curve stays at the boundary point forever.)

The most common case of a α- or ω-limit set in a vector field containing more than one inner point of the domain is a loop or limit cycle [8]. This is an integral curve c_a, so that there is a $t_0 \in \mathbb{R}$ with

$$c_a(t + nt_0) = c_a(t) \quad \forall\, n \in \mathbb{N}\,.$$

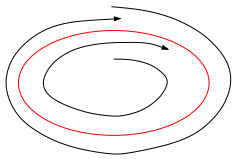

Fig. 1. A loop may attract integral curves in its neighborhood.

Figure 1 shows a typical example. Such a loop is called structurally stable if, after small changes, the vector field still contains a loop.

3 Loop Detection

The principal to detect loops in a three dimensional vector field is similar to the two dimensional case [19]. But there are some significant differences. To avoid confusion we start with a short notation:

Notation 3.1 (Actually investigated streamline)
We use the term actually investigated streamline *to describe the streamline we check if it runs into a loop.*

To reduce computational cost we first only integrate the streamline using a Runge-Kutta-method of fourth order with an adaptive stepsize control. Every cell that is crossed by the streamline is stored during the computation. If a streamline approaches a loop it has to reenter the same cell again. This results in a *cell cycle*:

Definition 3.2 (Cell cycle)
Let s be a streamline in a given vector field V. Further, let G be a set of cells representing an arbitrary tetrahedral grid without any holes. Let $C \subset G$ be a finite sequence $c_0, \ldots, c_n$ of neighboring cells where each cell is crossed by the streamline s in order and $c_0 = c_n$. If s crosses every cell in C in this order again while continuing, C is called a cell cycle.

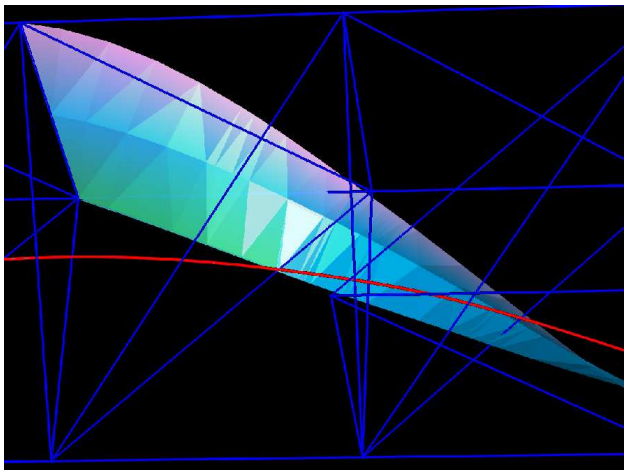

Fig. 2. Backward integrated surface.

This cell cycle identifies a region where we want to see if it can be left by the streamline. To check this, we have to consider every backward integrated streamline starting at an arbitrary point on a face of the boundary of the cell cycle. Looking at the edges of a face we can see directly that it is not sufficient to just integrate streamlines backward. Figure 2 shows an example. We integrated backward a streamsurface starting at an edge of the cell cycle. The streamlines starting at the vertices of that edge leave the cell cycle earlier than the complete surface. So it may be possible that a part of the streamsurface stays inside the cell cycle although the backward integrated streamlines starting at the vertices leave it. Consequently, we have to find another definition for exits than in the two dimensional case.

Definition 3.3 (Potential Exit Edges)
Let C be a cell cycle in a given tetrahedral grid G as in Definition 3.2. Then we call every edge at the boundary of the cell cycle a potential exit edge. *Analogue to the two dimensional case we define a line on a boundary face where the vector field is tangential to the face as a* potential exit edge.

Due to the fact that we use linear interpolation inside the tetrahedrons we can show that there will be at least a straight line on the face where the vector field is tangential to the face or the whole face is tangential to the vector field. An isolated point on the face where the vector field is tangential to the face cannot occur.

When dealing with edges as exits we have to compute a streamsurface instead of streamlines to consider every point on an exit edge. This leads us to the following notation.

Notation 3.4 (Backward integrated streamsurface)
We use the term backward integrated streamsurface *to describe the streamsurface we integrate by taking the negative vectors of the vector field starting at a potential exit edge in order to validate this exit edge.*

Analogue to the definition in the 2D case we define *real exit edges*.

Definition 3.5 (Real exit edge)
Let E be a potential exit edge of a given cell cycle C as in definition 3.3. If the backward integrated streamsurface does not ***completely*** *leave the cell cycle after one full turn through C then this edge is called a* real exit edge.

For the backward integrated streamsurface we use a simplified version of the streamsurface algorithm introduced by Hultquist [9]. Since we do not need a triangulation of the surface we only have to process the integration step of that algorithm. Initially we start the backward integration at the vertices of the edge. If the distance between these two backward integrations is greater than a special error limit we start a new backward integration in between. This continues with the two neighboring integration processes until we created an approximation of the streamsurface that respects the given error limit.

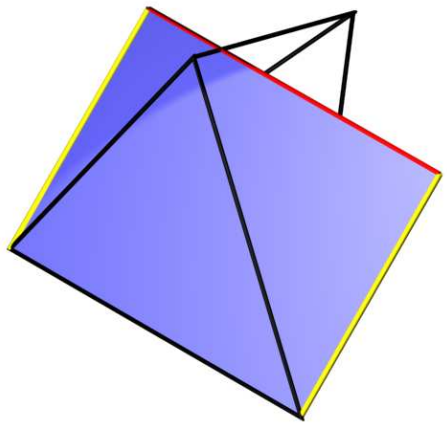

Fig. 3. Backward integration in one cell.

The integration stops if the whole streamsurface leaves the cell cycle or if we completed one full turn through the cell cycle. But to construct the surface properly we may have to continue a backward integration process across the boundary of the cell cycle. This is due to the fact that some part of the streamsurface is still inside the cell but the backward integrated streamline already left it. Figure 3 shows a simplified example. Both streamlines - shown

as yellow lines - leave the cell, in fact they leave right after they started. But the integration process must be continued until the whole surface created by these two streamlines leaves the cell. This is marked by the red line at the end of the streamsurface.

With these definitions and motivations we can formulate the main theorem for our algorithm:

Theorem 3.6
Let C be a cell cycle as in definition 3.3 with no singularity inside and E the set of potential exit edges. If there is no real exit edge among the potential exit edges E or there are no potential exit edges at all then there exists a loop inside the cell cycle.

Proof:
Let C be a cell cycle with no real exit edges. Every backward integrated streamsurface leaves the cell cycle C completely. It is obvious that we cannot leave the cell cycle if every backward integration starting at an arbitrary point on a face of the boundary of the cell cycle C leaves the cell cycle. So we have to prove that the actually integrated streamline cannot leave the cell cycle C.

We look at each face of the boundary of the cell cycle C. Let Q be an arbitrary point on a face F of the boundary of the cell cycle C. Let us assume that the backward integrated streamline starting at Q converges to the actually investigated streamline. We have to show that this is a contradiction. We have two different cases:

1.: The edges of face F are exit edges and there is no point on F where the vector field is tangential to F.
From a topological point of view the streamsurfaces starting at all edges of F build a tube that leaves the cell cycle. Since the backward integrated streamline starting at Q converges to the actually investigated streamline it does not leave the cell cycle. Consequently, it has to cross the tube built by the streamsurfaces. But streamlines cannot cross each other and therefore a streamline cannot cross a streamsurface.

2.: There is a potential exit edge e on the face F that is not a part of the boundary of F.
Obviously, the potential exit edge e divides the face F into two parts. In one part there is outflow out of the cell cycle C while at the other part there is inflow into C. We do not need to consider the part with outflow any further because every backward integrated streamline starting at a point of that part immediately leaves the cell cycle C.
The backward integrated surface starting at the potential exit edge e and parts of the backward integrated streamsurfaces starting at the boundary edges of the face F build a tube again from a topological point of view. Consequently, the backward integrated streamline starting at Q has to leave the cell cycle C.

We have shown that the backward integrated streamline starting at the point Q has to leave the cell cycle also. Since there is no backward integrated streamline converging to the actually investigated streamline at all the streamline will never leave the cell cycle. ❑

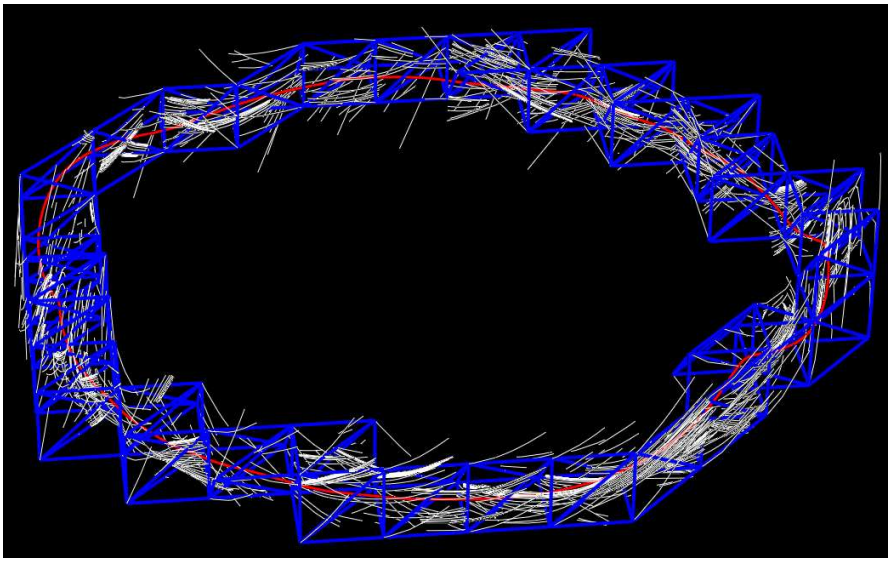

Fig. 4. Loop including cell cycle and backward integrations.

With theorem 3.6 we are able to describe our algorithm in detail. It is quite similar to the two dimensional case and mainly consists of three different states:

❶ streamline integration: identifying one cell change after the other, check at each cell if we reached a cell cycle.
❷ checking for exits: going backwards through the crossed cells and looking for potential exit edges.
❸ validating exit: integrating backwards a curve from potential exit through the whole cell cycle.

Figure 4 shows an example of our backward integration step. There, also the loop is drawn in red and the cell cycle is shown in blue. Every backward integrated streamsurface leaves the cell cycle. According to theorem 3.6, there exists a loop inside this cell cycle. Then we can find the exact location by continuing the integration process of the streamline that we actually investigate until the difference between two successive turns is small enough. This numerical criterion is sufficient in this case since we have shown that the streamline will never leave the cell cycle.

4 Results

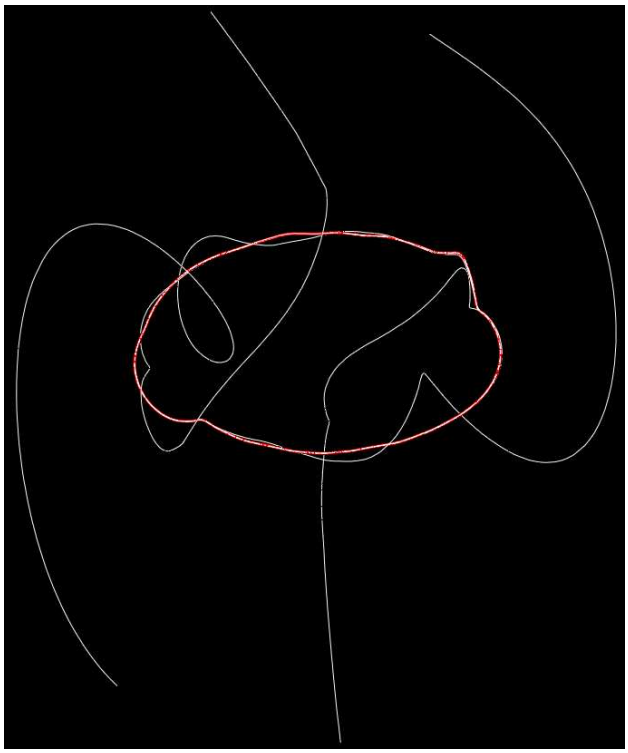

Fig. 5. Loop including some streamlines.

To test our implementation we created a synthetic dataset which includes one loop. We first produced a two dimensional vector field which is symmetrical with respect to the y-axis. Additionally, all vectors residing at the y-axis where zero. Then we rotated it around the y-axis and distorted it a little bit to get a three dimensional flow. Figure 5 shows the result. The loop is colored red. To visualize a little bit of the surrounding flow several streamlines are drawn. Obviously, every streamline is attracted by the loop. After a short time, while the streamline spirals around the loop, it completely merges into it. We can see in this example that the loop in this three dimensional flow acts like a sink.

Figure 6 shows the same loop with two streamsurfaces. The streamsurfaces are – just like the streamlines – attracted by the loop. The streamsurface gets smaller and smaller while it spirals around the loop. After a few turns around the closed streamline it is only slightly wider than a streamline and finally

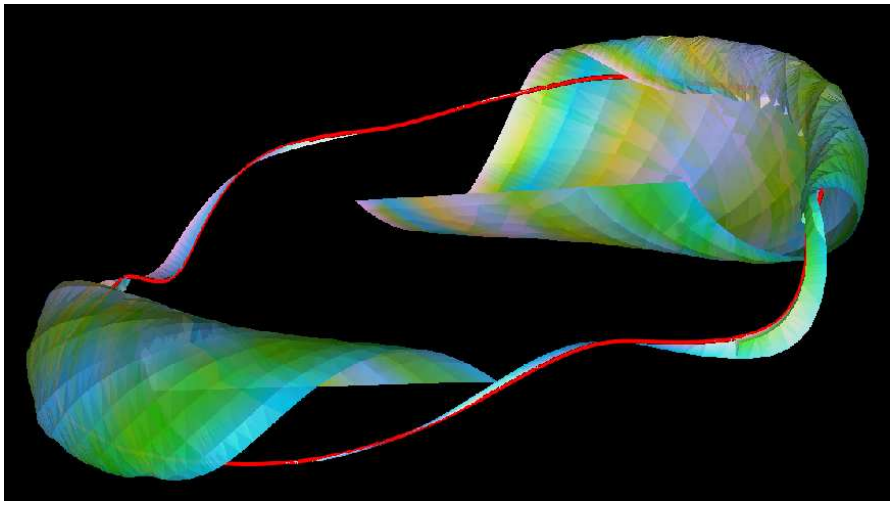

Fig. 6. Loop in a 3D vector field with streamsurfaces

it totally merges with the loop. We used a rather arbitrary color scheme for the surface to enhance the three dimensional impression.

5 Acknowledgments

This research was supported by the DFG project "Visualisierung nichtlinearer Vektorfeldtopologie". Further, we like to thank Tom Bobach, Holger Burbach, Stefan Clauss, Jan Frey, Christoph Garth, Martin Öhler, Max Langbein, Aragorn Rockstroh, René Schätzl and Xavier Tricoche for their programming efforts and Inga Scheler for helping with some of the figures. The continuous support of all members of the computer graphics and visualization team in Kaiserslautern gives us a productive working environment.

References

1. D. Bürkle, M. Dellnitz, O. Junge, M. Rumpf, and M. Spielberg. Visualizing complicated dynamics. In A. Varshney, C. M. Wittenbrink, and H. Hagen, editors, *IEEE Visualization '99 Late Breaking Hot Topics*, pp. 33 – 36, San Francisco, 1999.
2. M. Dellnitz and O. Junge. On the Approximation of Complicated Dynamical Behavior. *SIAM Journal on Numerical Analysis*, 36(2), pp. 491 – 515, 1999.
3. A. Globus, C. Levit, and T. Lasinski. A Tool for Visualizing the Topology of Three-Dimensional Vector Fields. In G. M. Nielson and L. Rosenblum, editors, *IEEE Visualization '91*, pp. 33 – 40, San Diego, 1991.

4. J. Guckenheimer and P. Holmes. *Dynamical Systems and Bifurcation of Vector Fields*. Springer, New York, 1983.
5. R. Haimes. Using residence time for the extraction of recirculation regions. *AIAA Paper 99-3291*, 1999.
6. J. L. Helman and L. Hesselink. Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications*, 11(3), pp. 36–46, May 1991.
7. D. H. Hepting, G. Derks, D. Edoh, and R. R. D. Qualitative analysis of invariant tori in a dynamical system. In G. M. Nielson and D. Silver, editors, *IEEE Visualization '95*, pp. 342 – 345, Atlanta, GA, 1995.
8. M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems and Linear Algebra*. Academic Press, New York, 1974.
9. J. P. M. Hultquist. Constructing stream surface in steady 3d vector fields. In *Proceedings IEEE Visualization 1992*, pp. 171–177. IEEE Computer Society Press, Los Alamitos CA, 1992.
10. M. Jean. Sur la méthode des sections pour la recherche de certaines solutions presque périodiques de syst'emes forces periodiquement. *International Journal on Non-Linear Mechanics*, 15, pp. 367 – 376, 1980.
11. D. N. Kenwright. Automatic Detection of Open and Closed Separation and Attachment Lines. In D. Ebert, H. Rushmeier, and H. Hagen, editors, *IEEE Visualization '98*, pp. 151–158, Research Triangle Park, NC, 1998.
12. S. Lang. *Differential and Riemannian Manifolds*. Springer, New York, third edition, 1995.
13. K. Museth, A. Barr, and M. W. Lo. Semi-immersive space mission design and visualization: Case study of the "terrestrial planet finder" mission. In *Proceedings IEEE Visualization 2001*, pp. 501–504. IEEE Computer Society Press, Los Alamitos CA, 2001.
14. G. M. Nielson, H. Hagen, and H. Müller, editors. *Scientific Visualization, Overviews, Methodologies, and Techniques*. IEEE Computer Society, Los Alamitos, CA, USA, 1997.
15. G. Scheuermann, B. Hamann, K. I. Joy, and W. Kollmann. Visualizing local Vetor Field Topology. *Journal of Electronic Imaging*, 9(4), 2000.
16. J. Stoer and R. Bulirsch. *Numerische Mathematik 2*. Springer, Berlin, 3 edition, 1990.
17. M. van Veldhuizen. A New Algorithm for the Numerical Approximation of an Invariant Curve. *SIAM Journal on Scientific and Statistical Computing*, 8(6), pp. 951 – 962, 1987.
18. R. Wegenkittel, H. Löffelmann, and E. Gröller. Visualizing the Behavior of Higher Dimensional Dynamical Systems. In R. Yagel and H. Hagen, editors, *IEEE Visualization '97*, pp. 119 – 125, Phoenix, AZ, 1997.
19. T. Wischgoll and G. Scheuermann. Detection and Visualization of Closed Streamlines in Planar Flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2), 2001.
20. P. C. Wong, H. Foote, R. Leung, E. Jurrus, D. Adams, and J. Thomas. Vector fields simplification – a case study of visualizing climate modeling and simulation data sets. In *Proceedings IEEE Visualization 2000*, pp. 485–488. IEEE Computer Society Press, Los Alamitos CA, 2000.