Eurographics/ IEEE-VGTC Symposium on Visualization 2010
G. Melançon, T. Munzner, and D. Weiskopf
(Guest Editors)

Volume 29 (2010), Number 3

Topology Aware Stream Surfaces

Dominic Schneider [1], Wieland Reich [1], Alexander Wiebel [2] and Gerik Scheuermann [1]

[1] University of Leipzig
[2] Max Planck Institute for Human Cognitive and Brain Sciences

Abstract

We present an algorithm that allows stream surfaces to recognize and adapt to vector field topology. Standard stream surface algorithms either refine the surface uncontrolled near critical points which slows down the computation considerably and may lead to a poor surface approximation. Alternatively, the concerned region is omitted from the stream surface by severing it into two parts thus generating an incomplete stream surface. Our algorithm utilizes topological information to provide a fast, accurate, and complete triangulation of the stream surface near critical points. The required topological information is calculated in a preprocessing step. We compare our algorithm against the standard approach both visually and in performance.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: flow visualization, stream surface, topology—Line and curve generation

1. Introduction

Efficient analysis of flow fields in general relies on visualization. An intuitive and well understood visualization technique are stream surfaces. An ideal stream surface is a two-dimensional continuum of streamlines starting from a well defined space curve. It can be seen as a representation of the area through which virtual particles released into a steady flow from the seed curve pass. Numerous variants of methods to compute stream surfaces can be found in the flow visualization literature (see Sec. 2). However, as we will show they either tend to bog down or produce incomplete surfaces if they intersect the 2D stable manifold of a saddle point.

A critical point or singularity is a point in the domain where the vector field describing the flow becomes zero. A saddle point is a critical point with a certain eigenvalue configuration of its Jacobian matrix at that particular point. In a dataset with intricate flow the intersection of 2D stable manifolds with stream surfaces is very likely to be encountered.

The aim of this work is to provide an algorithm for computing stream surfaces which correctly detects, handles and if necessary incorporates critical points when the stream surface approaches them. Our work was partly motivated by a discussion in a paper by Peikert and Sadlo [PS07] that brought up the term of *topology aware* stream surfaces. Further related work can be found in the following section.

2. Related Work

In this section, we want to discuss relevant previous work which can be divided into three major categories discussed in the following subsections. Firstly, we recall the basics of vector field topology, then we review literature on stream surfaces, and finally we describe topologically relevant stream surfaces.

2.1. Vector Field Topology

Dynamical systems research and analysis [HSD03, GH83] motivated the usage of topological methods in visualization. Thus, topological methods for vector fields focus on the so-called invariant sets, i.e. critical points, periodic orbits and the like. They were introduced as a visualization technique by Helman and Hesselink [HH91, HH89]. Globus *et al.* [GLL91] extended the display of critical points with local topological information such as eigenvectors to provide more insight into the behavior of the flow around these critical points.

Figure 1 shows the two types of a 3D non-degenerate (Jacobian has full rank) saddle point. Both saddle points have attracting behavior in the grey plane and repelling behavior elsewhere (repelling and attracting behavior can also be vice versa). The right image shows a focus or spiral saddle, that has rotating behavior around the 1D unstable manifold,

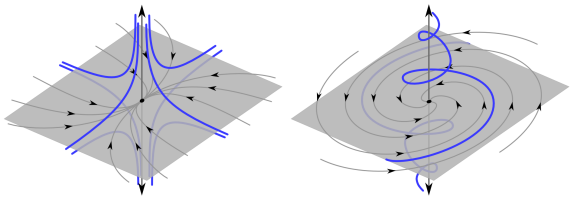

Figure 1: *Illustration of different types of 3D vector field saddles. Node saddle (left) and spiral saddle (right).*

whereas the left image shows a so-called node saddle that exhibits no rotation.

The sole depiction of critical points with local topological information such as eigenvectors [GLL91] does not yield a picture of the whole vector field topology. Another important element of vector field topology are the so-called separatrices.

In 3D a separatrix (separation surface, separating 2D manifold) is a uniquely defined stream surface which is locally tangential to the plane spanned by two eigenvectors of a saddle point (grey plane in Fig. 1). This stream surface is also called 2D stable manifold if the eigenvalues associated with the two eigenvectors are both negative. If they are both positive it is called 2D unstable manifold. For the sake of not confusing the reader we will generally restrict ourselves, without loss of generality, to saddle points with one positive and two negative eigenvalues exhibiting a 2D stable manifold. This matter will be discussed in more detail in section 2.3.

2.2. Stream Surfaces

In this section, we will review literature related to the construction of stream surfaces and their approximation. One of the first efficient algorithms for stream surface computation in visualization was described by Hultquist [Hul]. In his algorithm, like in almost any other stream surface algorithm, the seed curve is discretized into a finite number of points from which stream lines are started. Typically [GTS*04], the surface is constructed by advancing a front of virtual particles representing stream lines. Thus, the advancing front is a polyline consisting of segments connecting the endpoints of neighboring stream lines.

In order to deal with diverging flow, often a distance based refinement scheme is used: If the distance between two stream lines exceeds a predefined threshold, a new stream line is started in the middle of the segment between the current end points of adjacent stream lines. The current ribbon is split into two. This refinement scheme accounts for time line stretching in regions with diverging flow, but does not perform very well in regions with intricate flow.

Garth *et al.* [GTS*04,GKT*08] presented two approaches improving on Hultquist's work and showed how to obtain surfaces with higher accuracy. In the first work, the improvements are achieved by using a higher order integration scheme combined with arc-length parameterization. In the latter, a curve refinement scheme is used to approximate time lines yielding accurate path surfaces in large time-dependent vector fields. Using Hermite interpolation for the refinement process, Schneider *et al.* [SWS09] achieved fourth order accuracy yielding smooth C^1-continuous stream surfaces.

Scheuermann *et al.* [SBH*01] presented an algorithm exploiting the existence of an analytic solution to the stream surface problem for a tetrahedral cell with linear interpolation. For a grid consisting of tetrahedral cells the stream surface is then computed on a per-cell basis. The restriction to tetrahedral cells represents a serious limitation since the algorithm is not applicable to model computations based on other interpolation schemes.

A completely different approach was taken by Schafhitzel *et al.* [STWE07]. Their method essentially performs a GPU-accelerated splatting of a massive amount of advected particles to achieve the impression of a surface in the vector field. Therefore, this algorithm stays mostly unaffected of vector field topology with the drawback of not generating an explicit mesh.

Recently, a work fully employing the segmenting property of stream surfaces was published by Obermaier *et al.* [OKHBH09]. The stream surfaces in their application do not originate from saddle points but rather from so-called separation lines where the flow separates from boundary walls. This method constructs a *watertight* segmentation of the complete vector field using stream surfaces.

A number of other papers have used stream lines, stream surfaces and their variations for illustrating topologically interesting regions of flows (e.g. [LMGP97, SZH97]. Additional references about work on stream surfaces can be found in a survey by McLoughlin *et al.* [MLP*10].

None of the above mentioned algorithms is "topology aware", i.e. recognizes critical points explicitly. Yet critical points itself can be subject to seeding stream surface. In the following section we discuss these stream surfaces related to critical points.

2.3. Topology Related Stream Surfaces

As mentioned above, stream surfaces play a major role in three-dimensional vector field topology as 2D (un)stable manifolds of 3D saddle points. Like streamlines in 2D they segment the flow into regions with common origin and destination. An example where the separation surface has been used to illustrate the flow behavior in the vicinity of critical points can be found in [KOD*05]. Peikert and Sadlo [PS09] presented an algorithm for computing the separating 2D manifold of a saddle by using stream surface techniques.

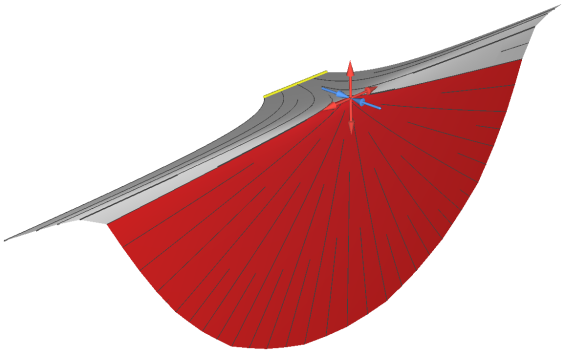

Figure 2: *Stream surface in linear vector field with highly diverging stream lines where the angle criterion (130 degrees) fails because the surface does not run into the saddle. The red part of the surface would have been left out if the angle criterion were to be used.*

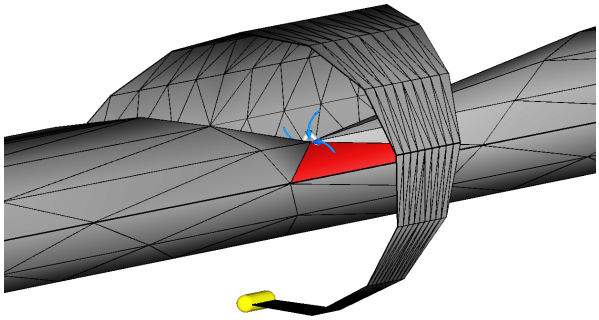

Figure 3: *Stalling's algorithm applied to a spiral saddle. When the angle between flow vectors exceeds 150 degrees the saddle is incorporated leading to visual artifacts such as the red triangle.*

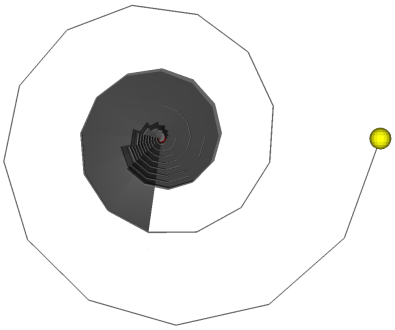

Figure 4: *Figure 3 viewed from the side. The approximation of the spiral by the red triangle in Fig. 3 leads to helix like structures in the stream surface.*

Their algorithm robustly handles the generation of a suitable start curve which is a difficult problem in its own right. However, their approach is not concerned with stream surfaces intersecting separatrices, which is the topic of this work.

In general, displaying all separatrices of a 3D flow is problematic. While in 2D all separatrices can be observed simultaneously, the display of the separatrices in 3D strongly suffers from occlusion. To handle this problem Theisel *et al.* [TWHS] suggested to display only the so-called saddle connectors which are the intersection curves of two separating 2D manifolds.

3. Stream Surfaces and Vector Field Topology

In this section, we want to raise the awareness for the interplay between stream surfaces and vector field topology. Stream surface approximation algorithms, especially Hultquist like approaches, have to deal with diverging flow i.e. diverging stream lines. Naturally, refinement schemes take care of an adequate resolution by inserting new stream lines and terminating existing ones where necessary. The problem with this method is that if the stream lines diverge exponentially a distance based refinement scheme inserts exponentially many new stream lines and stream surface computation eventually bogs down. This kind of behavior is observable near topologically interesting structures, especially near critical points of saddle type. In the following we will discuss this matter in detail.

Let us consider a critical point of saddle type with two eigenvalues with negative real part (i.e. attracting nature) and one eigenvalue with positive real part (i.e. repelling nature, see Fig. 1). Now let us consider further a stream surface in a 3D domain intersecting the 2D stable manifold (separatrix) of that saddle point. Due to the fact that the separatrix is stable the stream surface will be drawn closer to the critical point. Yet this holds only for one stream line of the stream surface. Namely, the one lying in the 2D stable manifold, i.e. the stream line where both surfaces intersect. The other stream lines of the stream surface are first attracted to the saddle point but then repelled in direction of the 1D unstable manifold corresponding to the eigenvalue with positive real part. Thus, the widely used distance based refinement scheme refines the stream surface exponentially. This leads to an enormous computational overhead the closer the stream surface is drawn to the saddle, i.e. the further it is integrated. Therefore, it is common practice to measure the angle of the flow vectors of the vector field between neighboring stream lines at the front of the stream surface during integration. If it exceeds a certain threshold the front is severed into two parts which are then treated separately by the algorithm. Unfortunately, this leaves a gap in the surface between the two stream lines. Moreover, the angle criterion is not a sufficient means to detect the intersection with a 2D stable manifold (see Fig. 2).

The first approach, to the best of our knowledge, trying to incorporate critical points into the stream surface has been described by Stalling in his PhD thesis [Sta98]. He presented an algorithm (see Sec. 2.3) to incorporate node saddles and sinks into the stream surface. Since our work is an extension of this approach we describe it in detail in the following.

Firstly, a global list of all critical points along with their eigenvalues and eigenvectors is computed. Stalling describes different strategies to handle sinks and saddles. The presence of the former is detected by the stream line integration scheme itself. If the sink is only a small distance away the position of this sink is added as the last stream line position. Afterwards this stream line is terminated. The resulting triangulation thus contains no cracks or wholes (see Fig. 5 left).

Saddles are detected by means of the angle criterion. If the angle exceeds a prescribed threshold (e.g. 150 degrees) a saddle is assumed to lie in close vicinity. From the three different eigenvectors of the saddle the one most parallel to the quadrilateral, i.e. the front curve, indicating the saddle is chosen. Using the direction of this vector, two new stream lines are inserted above and below the critical point (see Fig. 5 right). This way the saddle is incorporated into the stream surface and it is split into two fronts.

While the above mentioned approach works in 2D we will explain in the following why it cannot be extended reliably to 3D. Namely, because the angle criterion is neither a reliable nor a sufficient means to detect the intersection of the stream surface with the separatrix of a saddle point. As a simple counterexample, consider a stream surface very close to the 1D stable manifold of a node saddle with two repelling and one attracting eigenvalue. During integration, depending on the distance to the 1D manifold, the stream surface gets arbitrarily close to the saddle point. Near to the saddle point the stream surface is subject to highly diverging flow (see Fig. 2). Thus, the angle criterion can rightly indicate a saddle yet the decision to incorporate it would be wrong in this case. The same example can be constructed with the stream surface started or getting close to the 2D stable manifold of a saddle point.

Due to the mentioned shortcomings we will improve on Stalling's work and present an algorithm in the next section that works reliably in 3D. Moreover, we extend it by providing an algorithm to handle focus (spiraling) saddles. This is necessary as focus saddles are the major type of critical points in 3D flow datasets.

Applying Stalling's algorithm for a node saddle to a focus saddle leads to the approximation of a spiral by a line (i.e. triangle). This yields visual artifacts (see Fig. 3 and 4) and an approximation error of unknown size. However, we cannot prevent similar artifacts from occurring, but with our new algorithm we can control their size and shift it to a region where this kind of approximation can be safely made in terms of approximation error.

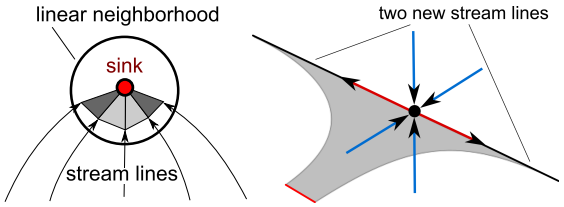

Figure 5: ***Left:*** *Incorporation of a sink into a stream surface.* ***Right:*** *Depiction of a node saddle where two new stream lines are inserted along the repelling eigenvector.*

4. The Topology-Aware Stream Surface

In this section, we describe the algorithm that detects whether a stream surface intersects the separating 2D manifold of a saddle point and the incorporation of the saddle point into the surface. We omit some technical details which will follow in Sec. 4.3.

The incorporation of a critical point into the stream surface is only correct in two cases: Firstly, the stream surface has been computed for $t \to \infty$. This is because for time based integration the stream surface reaches the saddle only in the limit, i.e. for $t \to \infty$. Secondly, the stream surface is parameterized by arc-length and not by time because an arc-length parameterized stream surface reaches the saddle after a finite arc-length. Yet there is a case where incorporation of a critical point into the surface is reasonable even for finite time based integration. Namely, when the distance between the stream surface front and the critical point is close to the prescribed numerical precision.

For our algorithm we need the concept of a linear neighborhood around a critical point x_c. We define the *linear neighborhood* $U_L(x_c)$ around a critical point $x_c \in \mathbb{R}^3$ as the region for which a linear approximation of the vector field holds within a certain bound $C_L \in \mathbb{R}, C_L > 0$:

$$U_L(x_c) = \left\{ y \in \mathbb{R}^3 \;\middle|\; \frac{\|\mathbf{v}(y) - J(x_c)\cdot(y-x_c)\|}{\|\mathbf{v}(y)\|} < C_L \right\}$$

where J denotes the Jacobian.

4.1. The Sink

To incorporate a sink into the stream surface we extend the algorithm by Stalling (see Sec. 3) with one additional step. Namely, we integrate the stream lines which have signaled to close in on a sink until they reach the linear neighborhood (see Fig. 5 left) of that sink. Once the stream line has reached the linear neighborhood the connection to the sink is made. We render this additional step necessary since it now ensures that the stream lines really run into the sink.

4.2. The Saddle

In order to incorporate a saddle into a stream surface, we need to detect the intersection of the stream surface with the 2D stable manifold of the saddle point. To achieve this, we continue integrating the stream surface until its front enters, i.e. intersects, the linear neighborhood of the saddle point. Within the linear neighborhood one can easily determine whether the front is intersected by the 2D stable manifold or not since it is now a plane spanned by the two attracting eigenvectors. Another advantage is that the stream surface and especially stream lines can be described analytically (see [NJ99]).

Depending on the eigenvalues of the Jacobian matrix, we need to distinguish two sub-types of saddle points as they are treated fundamentally different by our algorithm. These sub-types are discussed in the following subsections.

4.2.1. The Node Saddle

If all eigenvalues are real valued the saddle is of node type. As already described in Stalling's thesis [Sta98] there are two new stream lines inserted starting a little distance away from the saddle point (see Fig. 5 right). One tracer in positive and one in negative direction of the eigenvector associated with the repelling (i.e. positive) eigenvalue. The distance is determined by the linear neighborhood of the saddle point. These new streamlines are then used for further triangulation of the surface. The triangulation then includes the saddle itself.

4.2.2. The Focus Saddle

If the two negative eigenvalues occur as a conjugate-complex pair $\mu \pm \lambda \mathbf{i}$ the saddle is of focus (spiral) type. Stream surfaces intersecting the separatrix of this saddle type typically suffer from visual artifacts (see Fig. 11) generated by the exponential refinement prominently taking place close to the saddle.

Therefore, we propose to handle refinement differently once the stream surface front has entered the linear neighborhood of a focus saddle and intersects its separating 2D manifold. Namely, we insert two new streamlines on the front segment. One stream line on either side of the separating 2D manifold. Now the problem consists of finding a good starting point on the front segment for the new stream lines. This is crucial, because if they are placed too far off the separating manifold the two inserted stream lines will diverge too quickly. On the other hand, if they are placed too close to the separating manifold they stay near it for an unknown amount of time. It would be desirable to insert the streamlines such that they separate from the manifold in a defined way after a predefined amount of time or arc-length. Additionally, the seeding strategy needs to take into account a non-orthogonal eigenvector basis. That is the angle between separating 2D manifold and the real eigenvector can take on arbitrary values in the interval $(0, \frac{\pi}{2})$. The effect of non-orthogonal eigenvectors is that streamlines seeded at the same distance from the separating manifold can have substantially different rates of departing from the separating 2D manifold.

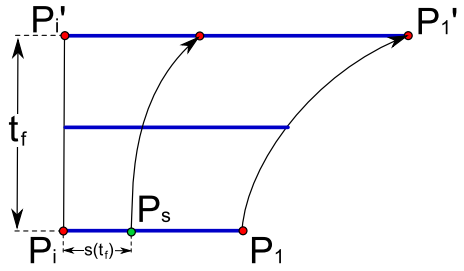

Figure 6: *Illustration of the algorithm for finding a seeding position P_s of a stream line inserted near a focus saddle.*

The seeding strategy presented in the following fulfills the above criteria. It uses the formulas provided by Nielson (see [NJ99] case 4) to anticipate stream line behavior and adjust the seeding for each of the two inserted streamlines individually. (P_1, P_2) denotes the front segment and P_i the intersection point with the 2D stable manifold on that segment. Without loss of generality, we describe our seeding strategy in the following for the segment (P_i, P_1) since it works analogous for (P_i, P_2).

For the seeding strategy we introduce a forecast time $t_f \in \mathbb{R}$ and compute the analytic stream lines ϕ for t_f from the point P_1 and P_i using Nielson's formulas. From the analytic stream lines we obtain two new locations in space: P_i' and P_1'. Now the stretching of the segment can be calculated to obtain a parameter s along the segment (P_i, P_1):

$$s(t_f) = \frac{\|P_1 - P_i\|}{\|P_1' - P_i'\|} \quad s \in \mathbb{R}$$

The parameter s is used to linearly interpolate a seeding location P_s for the new stream line on the front segment (see Fig. 6):

$$P_s = (1-s)P_i + sP_1$$

The seeding of new stream lines ends for time based integration when the distance between the intersection point and the critical point $dist(P_i, P_{C_i})$ falls below a prescribed threshold. Within this region around the critical point it is assumed to be safe to replace the spiral by a triangle in terms of approximation error. While this threshold controls the approximation error, the exact dependency of the approximation error on this threshold is part of future work. For arc-length integration the seeding ends when the remaining arc-length along the spiral into the saddle falls below the threshold.

4.3. Algorithmic Details

In this section, we give an overview of the algorithm steps and provide algorithmic details which have been omitted in the former description. An overview of the algorithm in pseudo code is given in Algorithm 1.

4.3.1. The Linear Neighborhood

In a preprocessing step, along with the critical points, the linear neighborhood of each critical point is computed. In order to determine the intersection point between separatrix plane and front segment, one only needs the linear neighborhood restricted to the separatrix plane. Therefore, one can compute the radius of the linear neighborhood using an ellipse lying in the plane and determine its radius. This is accomplished by making the two repelling eigenvectors the main axes of an ellipse. The ellipse is then discretized and every discretized position is checked for linearity. The linearity test consists of checking whether the relative error induced by the linearization around the critical point x_c falls below a prescribed threshold C_L. The relative error consists of the absolute error between the linearized field and the real vector $\mathbf{v}$ of the field divided by the magnitude of $\mathbf{v}$:

$$\frac{\|\mathbf{v}(x) - J \cdot (x - x_c)\|}{\|\mathbf{v}(x)\|} < C_L \tag{1}$$

If one of the positions exceeds the prescribed threshold for linearity the radius is halved and the algorithm is executed recursively until it succeeds. This way, one finds an approximation for the maximal radius with linear flow behavior in the separatrix plane.

A modified version of the above algorithm using a ray instead of the ellipse is used to determine the interval of the linear neighborhood along the 1D unstable manifold. The interval boundaries provide starting positions for the two new stream lines inserted along the 1D unstable manifold of the critical point (one on either side of the separatrix).

4.3.2. Practical Considerations

In order to speed up the algorithm, we use the angle criterion as a cheap means to indicate a *possible* intersection of a front segment with a separating 2D manifold of a saddle. We found that using an angle of 90 degrees yields good results. If that angle is exceeded a simple test is performed whether the reporting front segment intersects the linear neighborhood of a critical point. This being the case, the two end points are checked whether they lie on different sides of the linear separatrix plane. This finally decides if the stream surface is intersected by the 2D stable manifold. If an intersection is found the algorithm needs to discriminate between the focus and node saddle case to choose the appropriate algorithm. If any of the above tests fail the stream surface is integrated further.

The end points of the segment (P_1, P_2) can lie outside the linear neighborhood making the estimate of the separation rate less accurate. However, the algorithm robustly handles these inaccuracies. Since only the starting location P_s is affected, suboptimal choices will be corrected in future refinement steps resulting in a few additional stream lines at the worst. Alternatively, the intersection points between segment and linear neighborhood might be utilized.

Algorithm 1 Short overview of the algorithm in pseudo code

Require: Computed saddle points P_{C_i}

```
for all front segments (P1, P2) do
  Integrate segment
  if angle(v(P1), v(P2)) ≥ 90° then
    if (P1, P2) in linear neighborhood of P_Ci then
      if (P1, P2) intersected by 2D stable manifold (intersection point denoted P_i) then
        if Node Saddle then
          Insert two streamlines in direction of the repelling eigenvector
          Severe the front
        else
          if dist(P_i, PC_i) < thresh then
            Insert two streamlines in direction of the repelling eigenvector
            Severe the front
          else
            Insert two new stream lines on the segment (P1, P2) on either side of the separating 2D manifold
          end if
        end if
      end if
    end if
  end if
end for
```

The linearity threshold defines the size of the linear neighborhood and thus the distance at which the algorithm can start its work. The problem with increasing the threshold is the increasing uncertainty of the intersection point P_i between the segment and the 2D stable manifold. Thus, overzealous increase might lead to seeding locations not separated by the 2D stable manifold. We used a threshold of 0.1 for all examples;

Inside the linear neighborhood, the forecast time t_f controls roughly how many time line approximations are leaped before refinement occurs again. It can be tuned to fit the threshold for distance-based refinement in case of arc-length integration. For time based integration it can be chosen $\frac{\pi}{2}$. This way, roughly every quarter rotation new stream lines are seeded.

Using the described algorithm, mild exponential refinement still occurs in the case of focus saddles, yet its occurrence is limited especially in the case of arc-length parameterization.

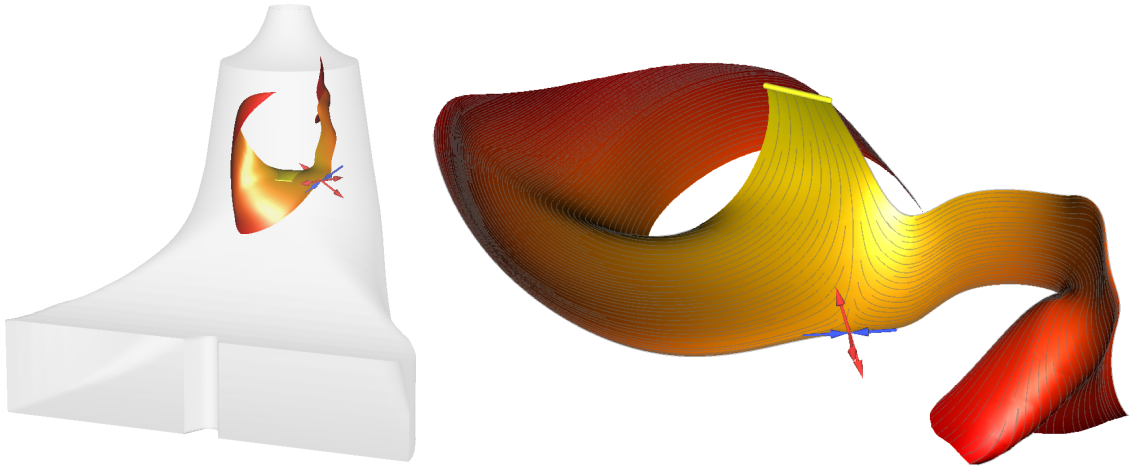

Figure 7: *Negative time stream surface in Draft Tube dataset* ***Left:*** *Location of the surface in the dataset.* ***Right:*** *Overview of the topology-aware stream surface running into a node saddle. Close ups of the saddle region can be seen below in Fig. 8.*

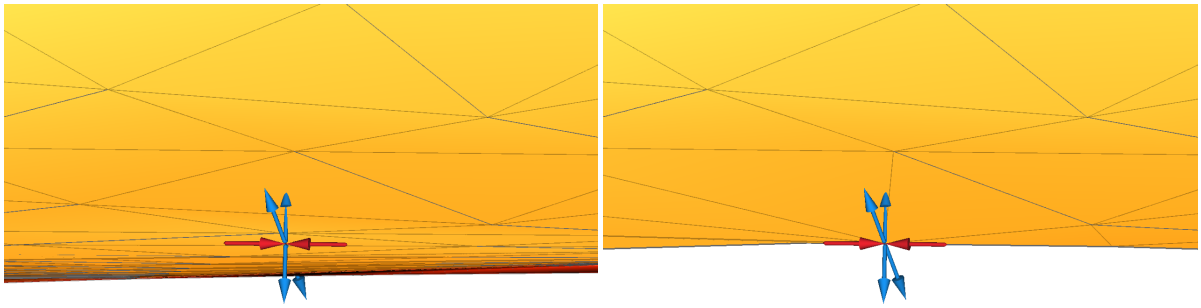

Figure 8: *Close ups of the saddle region showing the triangulation.* ***Left:*** *Stream surface generated by the standard algorithm continuously refining the stream surface near the saddle. The visual artifacts from the refining process are clearly visible.* ***Right****: Stream surface computed with our algorithm incorporating the saddle cleanly into the surface.*

5. Results

We applied our algorithm to a number of complex realistic datasets and were able to prove its utility and robustness. We provide images and a discussion for two of the datasets in the following.

5.1. Francis Draft Tube

The Draft Tube dataset is a CFD simulation with about one million arbitrarily shaped hexahedron cells. The dataset is very turbulent resulting in a large number of critical points. We chose a stream surface intersecting the 2D unstable manifold of a node saddle (see Fig. 7). The surface is therefore integrated backwards in time and parameterized by arc-length. The stream surface is basically split into two parts by the saddle point. Beyond the saddle point the two parts of the surface move largely independently. Figure 8 shows close ups of the saddle point and visualizes the difference between a standard stream surface and our algorithm.

Figure 9 shows the efficiency gain from our algorithm. Especially arc-length parameterization benefits from it since the arc-length of the stream line running into the saddle point is finite. Thus, parameterization is easily maintained after incorporation of the saddle point.

Integr. Length	Incorps. / Splits	Stream Lines Topo Aware	Stream Lines Standard
0.1	0	110	110
0.2	1	343	393
0.3	1	430	680
0.4	1	490	941
0.5	1	588	1239

Figure 9: *Comparison between standard and topology aware algorithm for the Draft Tube dataset. As can be seen, our algorithm produces considerably fewer stream lines.*

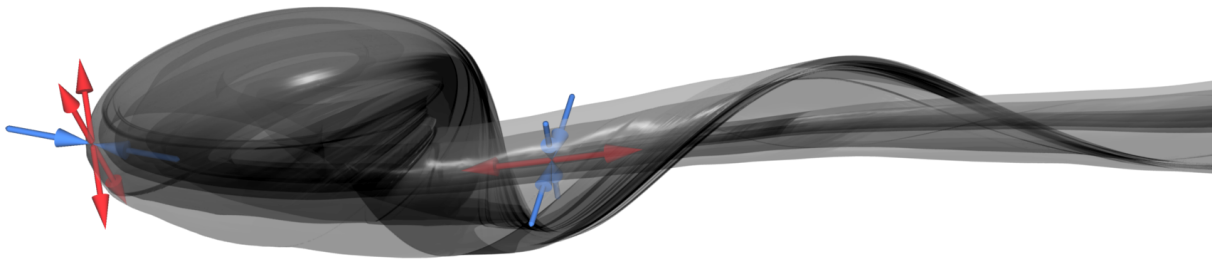

Figure 10: *High quality rendering of the 2D unstable manifold stream surface of the left breakdown bubble in the Delta Wing dataset running into the opposing focus saddle point. Red eigenvectors indicate repelling behavior whereas blue eigenvectors indicate attracting behavior as communicated by the arrow directions.*

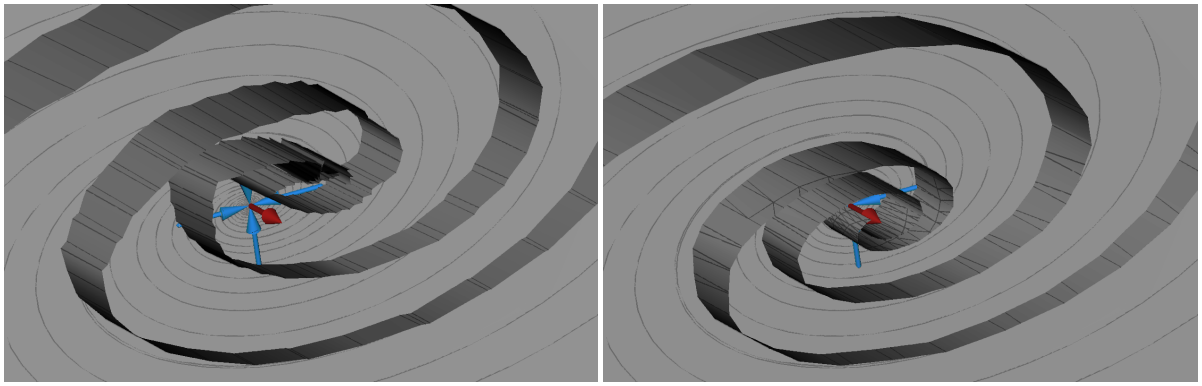

Figure 11: *Close ups of Fig. 10. Both surfaces consist of a similar amount of streamlines (approx. 8300) and have the same integration time of 0.035.* ***Left:*** *Close up of the surface generated by the standard algorithm showing artifacts of the refinement process and the resulting broken surface approximation.* ***Right****: Close up of the surface computed with our topology aware algorithm with* $t_f = \frac{\pi}{2}$.

5.2. The Delta Wing

This dataset stems from an unsteady vortex breakdown study around a delta wing configuration. The CFD simulation consists of a total of 1000 time steps. For our example we chose time step 650 with fully developed vortex breakdown bubbles. The data is given on an unstructured grid with about 12 million cells (tetrahedra and prisms).

Figure 10 shows an overview of the left breakdown bubble configuration with two spiraling saddle points. The 2D unstable manifold of the first saddle intersects the 2D stable manifold of the second saddle multiple times due to folding of the stream surface. This is an excellent example for our algorithm because the second saddle needs to be incorporated multiple times. Now the algorithm is actually capable of counting the number of incorporations (see Fig. 12). Furthermore, it can provide the exact parameter of the event at least for arc-length parameterization. Moreover, our algorithm generates a superior triangulation near the saddle for both arc-length and time parameterization. In Fig. 11 one can see that the closer the stream surface of the standard algorithm gets to the focus saddle the worse the surface approximation becomes. This is, because the fixed distance refinement strategy is too crude for the small region around the saddle. In contrast, our algorithm maintains a good surface approximation due to adapting seeding positions to the flow.

Arc Length	Incorps. / Splits	Stream Lines Topo Aware	Stream Lines Standard
0.2	1	427	472
0.3	2	911	1315
0.4	4	1802	2641
0.5	10	3889	5864

Figure 12: *Comparison between standard and topology aware arc-length integration for the Delta Wing dataset.*

6. Conclusion

A method for robustly and efficiently handling the special conditions near critical points while maintaining a correct stream surface approximation has been presented. Especially for the very intricate case of focus saddles.

The computation times for the presented examples lie in the order of tens of seconds. Depending on the integration length our approach was up to 30% faster.

The introduction of the linear neighborhood enables the algorithm to make definite assumptions about the flow behavior around the critical point and about how to incorporate it into the stream surface. Therefore, the algorithm is capable of reliably detecting the intersection of the stream surface with the 2D (un)stable manifold of a saddle point.

For arc-length parameterization our algorithm produces considerably fewer stream lines than standard implementations. Stream surfaces parameterized by time intersecting a 2D (un)stable manifold of a focus saddle benefit less from these savings. Still both parameterizations benefit from the superior triangulation producing a correct stream surface approximation in contrast to standard algorithms. The algorithm takes only two additional parameters: the maximum (starting) radius to determine the linear neighborhood of a critical point and the linearity threshold.

One limitation of this approach is that the algorithm needs to integrate until a stream surface front segment intersects the linear neighborhood of a critical point. Depending on the linearity threshold this might happen very late and after a considerable amount of refinement. However, this ties in with the rest of the stream surface quality settings.

Acknowledgments

We wish to thank Markus Rütten from German Aerospace Center (DLR) in Göttingen for the Delta Wing dataset, and Ronald Peikert and VA Tech for the Draft Tube dataset. This work was partially supported by DFG grant SCHE 663/3-8.

References

[GH83] GUCKENHEIMER J., HOLMES P.: *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, vol. 42 of *Applied Mathematical Sciences*. Springer, 1983. Corrected seventh printing, 2002. 1

[GKT*08] GARTH C., KRISHNAN H., TRICOCHE X., BOBACH T., JOY K. I.: Generation of accurate integral surfaces in time-dependent vector fields. In *Proceedings of IEEE Visualization '08* (October 2008). 2

[GLL91] GLOBUS A., LEVIT C., LASINSKI T.: A tool for visualizing the topology of three-dimensional vector fields. In *Proceedings of IEEE Visualization '91* (October 1991), pp. 33 – 40. 1, 2

[GTS*04] GARTH C., TRICOCHE X., SALZBRUNN T., BOBACH T., SCHEUERMANN G.: Surface techniques for vortex visualization. In *Data Visualization 2004 - Eurographics/IEEE TCVG Symp. on Vis. Proc.* (Konstanz, Germany, May 2004), Eurographics Association, pp. 155 – 164. 2

[HH89] HELMAN J. L., HESSELINK L.: Representation and Display of Vector Field Topology in Fluid Flow Data Sets. *IEEE Computer 22*, 8 (August 1989), 27 – 36. 1

[HH91] HELMAN J. L., HESSELINK L.: Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications 11*, 3 (May 1991), 36 – 46. 1

[HSD03] HIRSCH M. W., SMALE S., DEVANEY R. L.: *Differential Equations, Dynamical Systems, and an Introduction to Chaos*, 2nd ed., vol. 60 of *Pure and Applied Mathematics*. Academic Press, October 2003. 1

[Hul] HULTQUIST J. P. M.: Constructing Stream Surfaces in Steady 3D Vector Fields. In *Proc. of IEEE Visualization '92*. 2

[KOD*05] KRAUSKOPF B., OSINGA H. M., DOEDEL E. J., HENDERSON M. E., GUCKENHEIMER J., VLADIMIRSKY A., DELLNITZ M., JUNGE O.: A survey of methods for computing (un) stable manifolds of vector fields. *International Journal of Bifurcation and Chaos (IJBC) 15*, 3 (March 2005), 763–791. 2

[LMGP97] LÖFFELMANN H., MROZ L., GRÖLLER E., PURGATHOFER W.: Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems. *The Visual Computer 13*, 8 (1997), 359–369. 2

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. In *Computer Graphics Forum* (April 2010), Pauly M., Greiner G., (Eds.). 2

[NJ99] NIELSON G. M., JUNG I.-H.: Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE TVCG 5*, 4 (1999), 360–372. 5

[OKHBH09] OBERMAIER H., KUHNERT J., HERING-BERTRAM M., HAGEN H.: Stream volume segmentation of grid-less flow simulation. In *Topo. Methods in Data Anal. and Visualization*, Pascucci V., Tricoche X., Hagen H., (Eds.). 2009. 2

[PS07] PEIKERT R., SADLO F.: Topology-guided visualization of constrained vector fields. In *Topology-Based Methods in Visualization* (2007), Helwig Hauser H. H., Theisel H., (Eds.), Springer-Verlag, pp. 21–34. 1

[PS09] PEIKERT R., SADLO F.: Topologically relevant stream surfaces for flow visualization. In *Proc. Spring Conference on Computer Graphics* (April 2009), Hauser H., (Ed.). 2

[SBH*01] SCHEUERMANN G., BOBACH T., HAGEN H., MAHROUS K., HAMANN B., JOY K. I., KOLLMANN W.: A tetrahedra-based stream surface algorithm. In *Proceedings of IEEE Visualization '01* (Los Alamitos, CA, 2001), IEEE Computer Society Press, pp. 151 – 158. 2

[Sta98] STALLING D.: *Fast Texture-Based Algorithms for Vector Field Visualization*. PhD thesis, Freie Universität Berlin, 1998. 4, 5

[STWE07] SCHAFHITZEL T., TEJADA E., WEISKOPF D., ERTL T.: Point-based stream surfaces and path surfaces. In *Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 289–296. 2

[SWS09] SCHNEIDER D., WIEBEL A., SCHEUERMANN G.: Smooth stream surfaces of fourth order precision. *Computer Graphics Forum (EuroVis '09) 28*, 3 (2009), 871–878. 2

[SZH97] STALLING D., ZÖCKLER M., HEGE H.-C.: Fast display of illuminated field lines. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 118–128. 2

[TWHS] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3d vector fields. In *Proceedings of IEEE Visualization '03*, pp. 225 – 232. 3