Eurographics/ IEEE-VGTC Symposium on Visualization 2009
H.-C. Hege, I. Hotz, and T. Munzner
(Guest Editors)

Volume 28 (2009), Number 3

Smooth Stream Surfaces of Fourth Order Precision

Dominic Schneider [1], Alexander Wiebel [1] and Gerik Scheuermann [1]

[1] University of Leipzig

Abstract

We introduce a novel technique for the construction of smooth stream surfaces of 4th order precision. While common stream surface techniques use linear interpolation for generating seed points for new streamlines in the refinement phase, we use Hermite interpolation. The derivatives needed for Hermite interpolation are obtained by integration along the streamlines. This yields stream surfaces of 4th order precision. Additionally, we analyse the accuracy of the well known Hultquist approach and our new algorithm and proof that Hultquist's method is exact for linear vector fields. We compare both methods using the well known distance based and a novel error based refinement strategy. Our resulting surface is C^1-continuous, enabling improved rendering among other benefits.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: flow visualisation, error analysis, stream surface—

1. Introduction

The analysis of flow data often relies on flow visualisation. The most common technique in this context are integral lines and in particular stream lines. They are an intuitive and well understood technique in flow visualisation. Stream lines are tangent to the velocity vector of steady flow, meaning that they represent the path of particles through the flow field. Despite their illustrative and intuitive power, they can suffer from visual complexity and clutter, mainly because of missing depth cues. Stream surfaces are the natural generalisation of stream lines as they represent a continuum of stream lines started from a predefined curve. Thus, they are as intuitive as stream lines but possess the visual power of an actual surface, enhancing depth perception greatly.

However, there is an asymmetry in existing stream surface computation schemes. Namely, stream surfaces are computed using fourth and higher order stream line integration schemes in time, but for interpolation between stream lines only linear interpolation is used. As an analysis will show, linear interpolation is C^0 continuous and the interpolation error is only of order two. Therefore, we present a novel algorithm which uses Hermite interpolation. Hermite interpolation has an interpolation error of order four and is C^1-continuous alleviating the mentioned asymmetry. More precisely, we compute bicubic patches forming the stream surface. Thus, the resulting surface is smooth (C^1-continuous) and of fourth order precision.

2. Previous Work

An efficient algorithm for stream surface computation in visualisation was first described by Hultquist [Hul92]. In order to seed stream lines, the start curve is discretised into a finite number of points from which stream lines are started. The surface is constructed by advancing a front of positions approximating a line on the surface. The front is a polyline consisting of linear segments connecting the ends of neighbouring stream lines. Together with the stream lines, these segments form ribbons which are advanced recursively by considering the two diagonals in a quad, formed by the current and the next point on the two integral lines of the ribbon. The shorter of the two diagonals is chosen in order to avoid thin, long triangles. Adjacent ribbons are advanced following a recursive scheme. The integral lines are advanced one fixed step at a time. The recursion stops once all integral lines have reached their defined integration length.

To be able to deal with diverging flow, Hultquist used a distance based refinement scheme. If the distance between two integral lines exceeds a predefined threshold, a new integral line is started in the middle of the segment between the current end points of adjacent integral lines. The current ribbon is split into two. Moreover, if the angle between the tangents of two neighbouring integral lines exceeds a given threshold, this ribbon is not advanced any further and thus terminated. The stream surface is split into two parts which

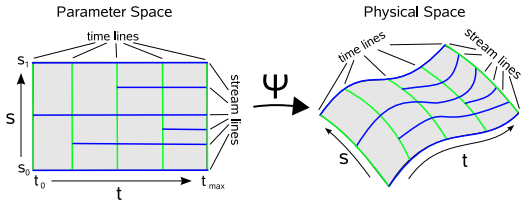


Figure 1: *Parameterisation of stream surfaces.*

are advanced separately. This is to prevent exponential overhead near critical points or obstacles.

This refinement scheme accounts for the stretching of the stream surface in regions with flow divergence, but does not perform very well in regions with intricate flow. In particular folding, twisting, or shearing of the surface yield problems. Garth et al. [GTS*04] presented an approach improving on Hultquist's work and showed how to obtain surfaces with higher accuracy, especially in regions with complex flow structures by using higher order integration schemes combined with arc length parameterisation. Recently, Garth et al. [GKT*08] presented a scheme on computing accurate path surfaces in large time-varying datasets by separating integral surface approximation from generation of a graphical representation.

Scheuermann et al. [SBH*01] presented a method for stream surface construction that exploits the fact that, in a tetrahedron, the stream surface is given in closed form assuming linear interpolation throughout the cell. Thus the stream surface can be computed analytically on a per tetrahedron basis. While being elegant, obviously the resolution of the resulting surface is tied very strictly to the resolution of the underlying grid.

In contrast to the methods mentioned above which compute stream surfaces explicitly, van Wijk [vW93] presented an algorithm to compute stream surfaces implicitly by defining a scalar function on the boundary of the grid. In a second step, the scalar values are advected from the boundary through the flow domain. Then it is possible to construct stream surfaces by means of standard isosurface extraction algorithms. Unfortunately, van Wijk's approach is not able to extract all possible stream surfaces of a flow which prevents it from being generally applicable.

A technique presented by Schafhitzel [STWE07], relies on the integration of particles on the GPU, which are rendered using a splatting method. Despite the fast surface integration, a problem arising from such an approach is that no explicit mesh is produced. It would have to be reconstructed from a set of points. Stream surfaces can also be used for a topological segmentation of the vector field [MBS*04] dividing it into regions of similar flow behaviour.

3. Stream Surfaces

This section introduces some theoretical aspects about stream surfaces: Let v be a Lipschitz continuous vector field defined over a domain $\Omega \subset \mathbb{R}^3$, then a *stream line* or *integral line* $\phi(t;t_0,x_0)$ is a solution to the ordinary differential equation:

$$\frac{d}{dt}\phi(t;\,t_0,x_0) = v(\phi(t;\,t_0,x_0)) \quad \phi(t_0;\,t_0,x_0) = x_0$$

From the definition we see that stream lines are tangent to the vector field at every point of the stream line.

A *stream surface* $\psi : [t_0,t_{max}] \times [0,1] \mapsto \mathbb{R}^3$ can be described as a 2D parametric surface embedded in a 3D flow. A natural parameterisation is to use a parameter $t \in [t_0,t_{max}]$ along every stream line parameterising the surface in time and a parameter $s \in [0,1]$ which parameterises the stream lines according to their starting point positions along a space curve $c(s)$ in Ω:

$$\psi(t,s) = \phi(t;\,t_0,c(s)) \quad \psi(t_0,s) = c(s)$$

where curves of constant s are called stream lines whereas curves of constant t are called *time lines* (see Fig. 1).

A stream surface approximation is constructed by seeding stream lines from the space curve c which is given as a polyline connecting a finite set of points. In our approach, the exact stream lines ϕ are numerically approximated by stream lines $\hat{\phi}$ using the DoPri5 [PD81] integration scheme. Similar to [GKT*08] time lines are approximated at equally spaced time values $t_n = n \cdot \Delta t$ covering the interval $[t_0,t_{max}]$ with $n \in \mathbb{N}$. The user needs to specify the Δt parameter to determine the spacing. This is necessary to compute a well defined bicubic patch bounded by time and stream lines. The approximated stream surface is denoted by $\hat{\psi}$.

4. C^1-continuous Stream Surfaces

In the following, we present a novel method for the construction of a fourth order C^1-continuous stream surface, meaning that the interpolation error of the surface is of order four. This is achieved by using Hermite interpolation in t *and* in s-direction, generating bicubic patches.

4.1. Hermite Interpolation

The cubic Hermite interpolation on the interval $[s_i,s_{i+1}]$ with $r = \frac{s-s_i}{s_{i+1}-s_i}$ for a fixed time t is defined as follows:

$$\begin{aligned} H(r) &= H_0(r)\psi(t,s_i) + H_1(r)\psi(t,s_{i+1}) \\ &+ (s_{i+1}-s_i)\left(H_2(r)\frac{d}{ds}\psi(t,s_i) + H_3(r)\frac{d}{ds}\psi(t,s_{i+1})\right) \end{aligned}$$

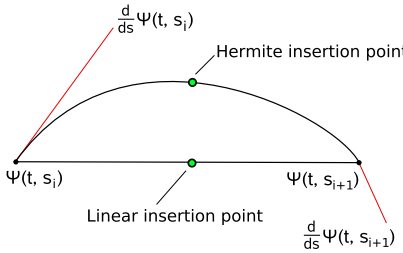


Figure 2: *Hermite interpolation.*

with the following Hermite polynomials

$$\begin{aligned} H_0(r) &= 2r^3 - 3r^2 + 1 \\ H_1(r) &= -2r^3 + 3r^2 \\ H_2(r) &= r^3 - 2r^2 + r \\ H_3(r) &= r^3 - r^2 \end{aligned}$$

Figure 2 shows a schematic drawing of the Hermite interpolation between two points $\psi(t, s_i)$ and $\psi(t, s_{i+1})$ along a time line (constant t) on the stream surface. The points are located on two neighbouring streamlines.

4.2. Approximation of the Covariant Derivative

As we have seen in the previous section, Hermite interpolation along stream and time lines requires the covariant derivatives with respect to t and s. Therefore, we describe the calculation of these derivatives in the following: The derivative with respect to t can be obtained by evaluating the vector field at the desired positions, e.g. on the stream line. Thus, Hermite interpolation along stream lines is always possible. More effort needs to be spent in order to obtain the covariant derivatives with respect to s. The general idea is to define the derivatives on the well known seed curve and then integrate them along every stream line with the same numerical scheme that is used for stream line integration. We will refer to this process in the following as *derivative integration*.

First, for derivative integration any numerical integration scheme based on a Runge-Kutta formula, represented by Θ, can serve as a basis:

$$y_{n+1} = y_n + h_n \Theta(t_n, h_n, y_n) \tag{1}$$

where y_n denotes $\hat{\phi}(t_n;\ t_0, x_0)$ to avoid notational clutter. We use formula (1) to obtain a numerical scheme integrating the derivative by differentiating it with respect to s, yielding

$$\frac{d}{ds} y_{n+1} = \frac{d}{ds} y_n + h_n \frac{d}{ds} \Theta(x_n, h_n, y_n). \tag{2}$$

Since the above description is very abstract, we give a practical example in the following, where Θ represents a two-stage Runge-Kutta method:

$$\begin{aligned} k_1 &= v(t_n,\ y_n) \\ k_2 &= v\left(t_n + \frac{2}{3}h,\ y_n + h\frac{2}{3}k_1\right) \\ y_{n+1} &= y_n + \frac{1}{4}h\,(k_1 + 3k_2) \\ \Theta(t_n, h_n, y_n) &= \frac{1}{4}\,(k_1 + 3k_2)\,. \end{aligned}$$

As mentioned, differentiating this formula with $\frac{d}{ds}$ yields equation (2). By propagating the differential further into the Runge-Kutta formula and applying the chain rule, we obtain:

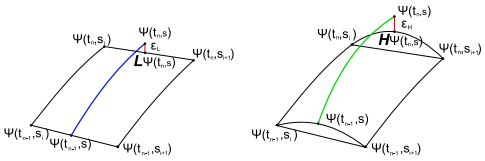
$$\begin{aligned} \frac{d}{ds}k_1 &= \nabla v\,(t_n,\ y_n)\,\frac{d}{ds}y_n \\ \frac{d}{ds}k_2 &= \nabla v\left(t_n + \frac{2}{3}h,\ y_n + h\frac{2}{3}k_1\right)\left(\frac{d}{ds}y_n + h\frac{2}{3}\frac{d}{ds}k_1\right) \\ \frac{d}{ds}y_{n+1} &= \frac{d}{ds}y_n + h\frac{1}{4}\left(\frac{d}{ds}k_1 + 3\frac{d}{ds}k_2\right) \end{aligned}$$

where ∇ denotes the gradient operator. ∇v is the matrix of the partial derivatives, also known as Jacobian matrix. A more detailed example of a three stage Runge-Kutta method can be found in the supplemental material on the conference DVD. By comparing the original Runge-Kutta formulas with the ones we obtained for the covariant derivative we find that the Jacobian must be evaluated at exactly the same positions as the Runge-Kutta scheme. This means that the two methods are firmly tied together.

4.3. Coarsening and Refinement

In order to refine or coarsen a stream surface, stream lines need to be inserted or terminated. For a C^0-continuous stream surface, this process has been described in the work of Hultquist [Hul92]. In the case of a C^1-continuous stream surface, the algorithm needs some modification. For refinement, the algorithm remains basically the same. If the necessity of refinement is detected by the refinement scheme, a new stream line is inserted at an interpolated position using Hermite interpolation (see Fig. 2). Therefore, the transition of the derivative at constant s from time t_{n-1} to t_n is continuous and the surface remains C^1-continous.

The case of coarsening is a bit more complicated because the end point of the terminated stream line does not lie on the Hermite interpolation curve if only the left and right neighbouring stream line is considered. Thus, the order of precision drops locally since an interpolation point is removed from the front. Globally the front curve is still of fourth order. This is not a special problem of the algoithm but a general phenomenon of Hultquist like algorithms caused by coarsening. However, to maintain C^1 continuity the derivative must be linearly interpolated from the integration endpoint of the terminated stream line at time t_{n-1} to the corresponding point on the Hermite interpolation curve at time t_n.



(a) Linear (b) Hermite

Figure 3: *Interpolation Schemes.*

4.4. Error Analysis

Stream surface generation algorithms, among them Hultquist's method, so far are assuming linear interpolation between integral curves, whereas our method uses Hermite interpolation. In the following, we analyse the local interpolation error of linear and Hermite interpolation between two time lines to obtain the order of accuracy in s-direction. As a result, we show that linear interpolation is exact for linear vector fields and Hermite interpolation for cubic vector fields.

4.4.1. Analysis of Hultquist's Method

The linear interpolation error can be expressed as the difference between the correct solution $\psi(t_n,s)$ and the linear interpolated solution $L\psi(t_n,s)$:

$$\varepsilon_L(t_n,s) = \psi(t_n,s) - L\psi(t_n,s) \tag{3}$$

Let $s_i < s < s_{i+1}$, then equation (3) can be expressed as follows:

$$\varepsilon_L(t_n,s) = \psi(t_n,s) - \frac{s_{i+1}-s}{s_{i+1}-s_i}\psi(t_n,s_i) - \frac{s-s_i}{s_{i+1}-s_i}\psi(t_n,s_{i+1}) \tag{4}$$

In order to analyse the linear local interpolation error as it is transported from one time line $\psi(t_{n-1},s)$ to the next $\psi(t_n,s)$, we need to reformulate equation (4), which is written in terms of time t_n, with terms of time t_{n-1}. Therefore, we apply Taylor expansion in time around t_{n-1} to all terms and additionally in space to the terms belonging to linear interpolation. By doing so we rephrase equation (4) in terms of time t_{n-1}. Since we are analysing the local error, a precondition is that the local error at time t_{n-1} is zero, i.e. $\varepsilon_L(t_{n-1}) = 0$ (see Fig. 3(a)). Comparing the terms of the Taylor expansions we find that the linear approximation error of the surface in s-direction is of order $O(\Delta s^2)$, meaning that Hultquist's algorithm interpolates correctly in linear vector fields. A more detailed derivation can be found in the supplemental material.

4.4.2. Analysis of Our New Algorithm

In order to analyse the local interpolation error of our new algorithm, again first the correct solution $\psi(t_n,s)$ is compared, in this case, to Hermite interpolation $H\psi(t_n,s)$:

$$\varepsilon_H(t_n,s) = \psi(t_n,s) - H\psi(t_n,s)$$

which, explicitly written, yields

$$\varepsilon_H(t_n,s) = \psi(t_n,s) - H_0(r)\psi(t_n,s_i) - H_1(r)\psi(t_n,s_{i+1}) - (s_{i+1}-s_i)\left(H_2(r)\frac{d}{ds}\psi(t_n,s_i) + H_3(r)\frac{d}{ds}\psi(t_n,s_{i+1})\right)$$

Again, the local error ε_H at time t_{n-1} is zero. Now the term $\psi(t_n,s)$ is expanded in t around t_{n-1} up to fourth order. The remaining terms are first expanded in time around t_{n-1} and then in space around s (see Fig. 3(b)), both up to fourth order. The resulting terms of up to third order vanish, meaning that the cubic Hermite approximation error is of order $O(\Delta s^4)$. Thus, this type of interpolation is exact for cubic vector fields. A more detailed derivation can be found in the supplemental material.

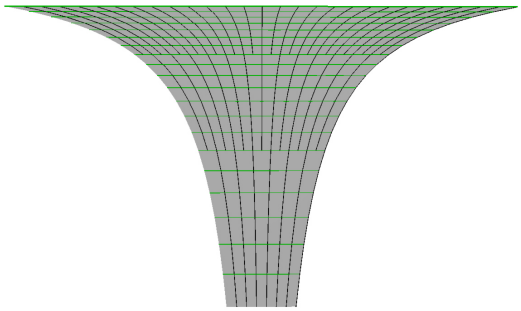


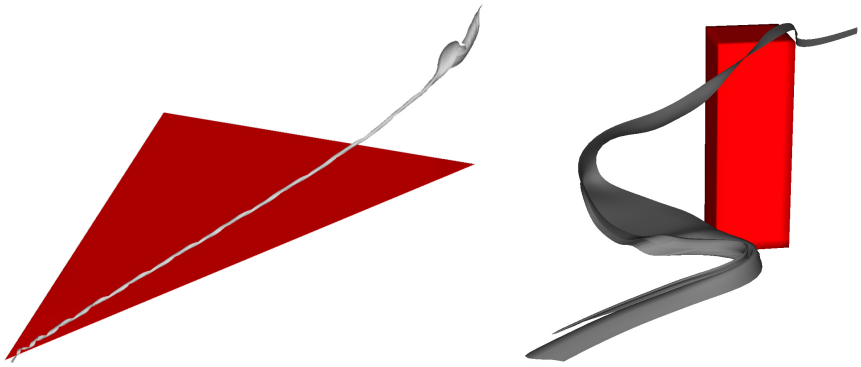
Figure 4: *Refinement of stream surface near a linear saddle point using distance based refinement.*

5. Refinement Strategies

In order to obtain a good approximation, an acceptable sampling density of the front must be maintained. Therefore, stream surface construction algorithms provide adaptivity towards refinement and coarsening. In the following sections, we analyse the distance based and a novel error based refinement scheme which allows to prescribe a local error bound that is maintained through stream surface construction. We describe the algorithms for inserting and removing stream lines, i.e. points on the front, for each scheme.

5.1. Distance Based Strategies

Distance based refinement strategies split a ribbon and seed a new integral curve if the distance at time t_n between two neighbouring stream lines, with parameter values s_i and s_{i+1}, exceeds a given threshold. Thus, the front is refined when it is stretched in s-direction, which is the case in the presence of diverging flow. The seeded stream line is inserted at time t_n at parameter value $s = \frac{1}{2}(s_i + s_{i+1})$. The position is interpolated using either linear interpolation, which is the classical method, or by using Hermite interpolation which is proposed in this work.



(a) Delta Wing (b) Cuboid

Figure 5: *Overview of computed stream surfaces for the numerical experiments*

In order to coarsen the stream surface, three neighbouring stream lines, i.e. three neighbouring points on the front, are considered. If the sum of the distance between any two of them falls below a second prescribed distance threshold, the stream line in the middle is removed. The threshold for coarsening should of course be lower than the threshold for refinement.

Despite their popularity and robustness, distance based refinement strategies have some shortcomings. The approximation error of the front is measured only in an indirect way by means of a geometric property of the surface (distance between stream lines). Thus, the surface might be refined in regions where the approximation error is low, e.g. in the presence of a linear saddle point (see Fig. 4) where the distance based refinement causes exponential overhead. On the contrary, it cannot account for e.g. folding or twisting, which leads to undersampling in such areas. However, we seek to resolve this issue by introducing *error based refinement strategies* in the next section.

5.2. Error Based Strategies

Error based refinement strategies split a ribbon when the local interpolation error exceeds a given bound. This error is estimated directly by seeding a new short stream line from a position interpolated between neighbouring stream lines at time t_{n-1} integrating it to time t_n. The estimated error is the difference between the interpolation of the position at time t_n and the integration endpoint of the inserted stream line (see Fig. 3(a) and 3(b)):

$$\varepsilon(t_n, s) = \hat{\phi}(t_n;\, t_{n-1}, \hat{F}(t_{n-1}, s)) - \hat{F}(t_n, s)$$

where $\hat{F}$ is a piecewise interpolant, interpolating between two neighbouring stream lines. This means that the front is only refined in regions where the underlying flow is of higher order than the piecewise interpolant. For example, error based strategies with linear interpolation do not refine the front in a linear vector field, since linear interpolation is exact for linear vector fields (see Sec. 4.4.1).

For coarsening, again three neighbouring stream lines, i.e. three neighbouring points on the front, are considered. If the local interpolation error for two abutting segments on the front falls below a second prescribed error bound the stream line connecting both segments is removed forming one segment.

However, error based strategies of this type basically double the cost for stream surface computation. For every ribbon a short streamline from time t_{n-1} to time t_n needs to be calculated for every time step t_n. In return, they allow to specify an error bound for the local error of the surface. Thus, error based strategies refine the surface only when it is necessary, i.e. where the error bound is exceeded.

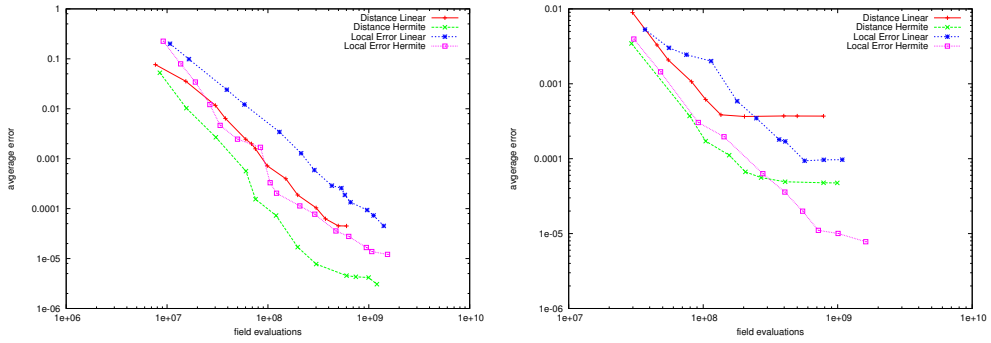
6. Results

In this section, we describe the datasets we have used for conducting our numerical experiments, discuss the performance of the different refinement schemes and present the improved rendering that has become possible by Hermite interpolation along time lines.

6.1. Datasets

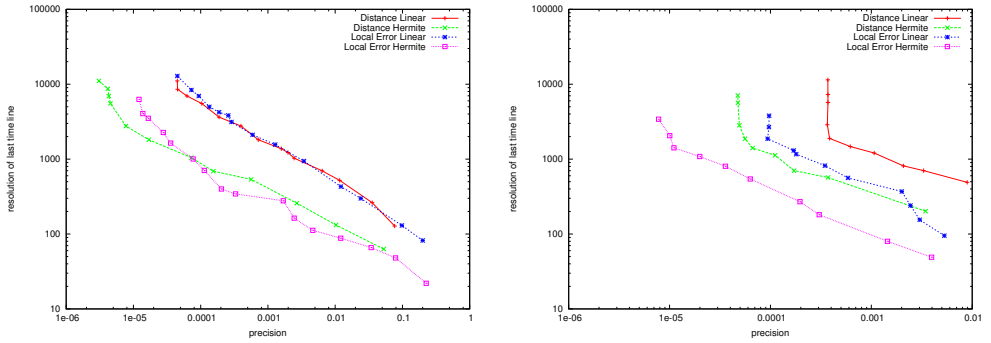
To compare the different stream surface techniques we applied them to the following application datasets.

Delta Wing This dataset is a study of an unsteady vortex breakdown above a delta wing given on an unstructured grid. The angle of attack increases during the simulation. The grid consists of around 3 million vertices and 11 million cells. We have chosen time step 65 out of a total of 86 time steps to run our numerical experiment. The vortex breakdown bubbles are fully developed at this time. We started integration in front of the delta wing where the surface is drawn into one of



(a) Cuboid (b) Delta Wing

Figure 6: *Numerical comparison of accuracy and field evaluations.*



(a) Cuboid (b) Delta Wing

Figure 7: *The diagrams show the resolution of the last time line for a given precision.*

the main vortices and curls up against the breakdown bubble later on (see Fig. 5(a)).

Cuboid The Cuboid dataset stems from an unsteady numerical simulation of fluid flow around a cuboid. The simulation was carried out with the NaSt3DGP† flow solver. The data is given on a rectilinear $100 \times 100 \times 100$ grid. We computed a stream surface starting near one of the corners of the cuboid for our numerical experiments (see Fig. 5(b)).

6.2. Results of the numerical experiments

In this section, we present the results of numerical experiments with the datasets introduced in the previous section. The refinement schemes are compared on the basis of the number of function evaluations of the underlying vector field as this is the computationally most expensive operation in the stream surface integration algorithm.

In order to evaluate our new algorithm and the different refinement schemes, we conducted a number of numerical experiments with different datasets. For each dataset we chose a start curve $c(s)$ and a time interval $[t_0,\ t_N]$ for which the stream surfaces are computed. First, a highly resolved stream surface, representing the ground truth, is computed. For the ground truth surface a large number of M stream lines, equidistantly seeded on $c(s)$, are integrated. We compare the last time line $\hat{\psi}(s,\ t_N)$ of each computed stream surface with the last time line of the ground truth surface $\hat{\psi}_{GT}(s,\ t_N)$ using the following error measure:

$$e(t_N) = \frac{1}{M} \sum_{k=1}^{M} \|\hat{\psi}(s_k,\ t_N) - \hat{\psi}_{GT}(s_k,\ t_N)\|^2$$

Despite the graph for each refinement scheme being displayed in one diagram for each dataset, distance and error based methods should not be compared directly, since their refinement behaviour is fundamentally different. The issue preventing direct comparison is that for distance based meth-

† NaSt3DGP was developed by the research group in the Division of Scientific Computing and Numerical Simulation at the University of Bonn. It is essentially based on the code described in a book by Griebel et al. [GDN98].

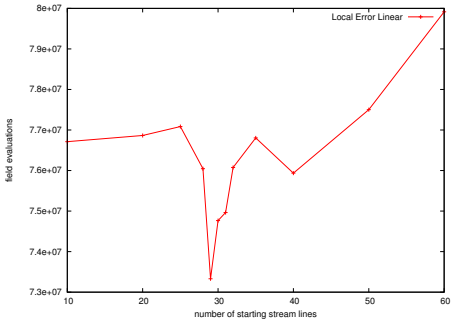


Figure 8: *The diagram shows the number of starting stream lines versus the number of field evaluations for a constant local error bound of 0.00002.*

ods the start curve $c(s)$ (see Sec. 3) needs to be refined to a certain level according to the distance threshold (see Sec. 5.1). Otherwise, refinement will occur in the first steps until the distance between points on the front falls below this threshold. Thus, the initial start curve discretisation depends on the distance threshold.

In contrast, error based methods do not exhibit this behaviour. The initial discretisation of the start curve can be chosen freely because refinement occurs on the basis of the local interpolation error. On the other hand, the initial start curve discretisation, i.e. the number of starting stream lines, is a crucial parameter because the dependency between the number of starting stream lines and the computational effort to compute the stream surface is non-monotonic. There exists a start curve discretisation, for a fixed error bound, for which the computational effort is minimal (see Fig. 8). This is, because the starting stream lines are the most accurate ones, meaning the more starting stream lines are used, the more accurate the resulting surface will be. This applies to distance based schemes as well. On the other hand, the more starting stream lines there are, the less refinement is needed. Thus, the issue whether the error based scheme is more efficient than the distance based scheme is a question of finding the optimal start curve discretisation, which is yet an unsolved task and part of future work. Therefore, the number of starting stream lines for the error based methods has been arbitrarily chosen to be eight to underline the incomparability.

From Figures 6(a), 6(b) and 9 it can be seen that refinement schemes using Hermite interpolation are always significantly more accurate for a given computational effort. Both refinement schemes benefit from the more accurate interpolation.

Despite the non-smooth surface of the delta wing and the Cuboid with their sharp edges and pointy corners, resulting in a very volatile Jacobian, the algorithms using Hermite interpolation perform quite well. The delta wing dataset poses an additional hurdle, because it is given on an unstructured grid, where the Jacobian is discontinuous at cell boundaries of a tetrahedron.

One confinement of the algorithm is, that the Δt-parameter must be chosen small enough such that refinement can occur frequently enough, which applies to both, distance and error based methods. This problem has been addressed by Garth et al. [GKT*08] by examining the number of integration steps the numerical scheme takes.

A disadvantage of the error based scheme, as presented in section 5.2, is the usually higher computational effort in comparison to distance based methods if a non-optimal start curve discretisation is used (see Fig. 8). On the contrary, Figure 7(b) and 7(a) show that for a fixed precision and type of interpolation, error based methods mostly insert fewer stream lines, i.e. the resolution of the last time line $\hat{\psi}(s, t_N)$ is lower, than distance based methods. This is especially apparent for the delta wing dataset where the stream surface undergoes a large amount of twisting (see Fig. 6(b)).

However, the value of error based methods as such, is not solely a matter of computation time (see Fig. 6 and 7). They rather allow to specify a local error bound in s-direction, which is maintained through the stream surface computation. Moreover, since the local error is guiding refinement, a sparse initial discretisation of the start curve has little influence on the final discretisation of the front. However, it does have an influence on the accuracy, i.e. the average error. Namely, the finer the initial discretisation, the more accurate the surface will be.

6.3. Improved rendering

Naturally, we employ Hermite interpolation not only for inserting new streamlines in the refinement process but also when rendering the surfaces. The derivatives computed along the streamlines allow us to get a smooth approximation of the stream lines and time lines on the surface by means of the Hermite interpolation. Both smooth curves build up a bicubic patch providing the information we need to produce renderings superior to those possible with the standard approach (see e.g. Fig. 10).

Additionally, the smoothness of the surface allows to compute continuously varying normals over the whole surface. This is important for advanced shading, e.g. ray tracing, if high quality images are needed.

7. Conclusion

We have presented a new algorithm for constructing stream surfaces of fourth order precision in every direction by computing bicubic patches producing a smooth C^1-continuous surface. The smooth surface allows for a much better rendering improving the visual quality of the surface. Furthermore, the smoothness enables methods requiring continuous

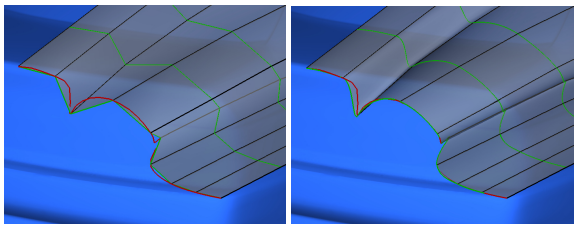


Figure 9: *Comparison of accuracy of stream surfaces in car dataset constructed using 10 stream lines. The left image shows the surface for the Hultquist like method, the right image shows the surface constructed using the new Hermite interpolation based technique. The green lines represent time lines of the approximated surfaces. The red line is a time line from a high quality surface representing the* ground truth *for comparison. The comparison shows the greatly improved accuracy of the new method.*

derivatives on the surface. FTLE fields [GWT*08] on stream surfaces indicating converging or diverging behaviour are only one idea in this direction. Finally, we have presented an error based refinement scheme which represents a first step towards an error controlled stream surface construction.

Although we have shown the usefulness and feasibility of computing stream surfaces using Hermite interpolation, there is much room for further work on the topic. First, we plan to increase the performance of the error based refinement scheme. Second, we want to extend the error based refinement scheme to work on the basis of one global error bound for the whole surface.

Acknowledgements

We wish to thank Markus Rütten from German Aerospace Center (DLR) in Göttingen for providing the delta wing data set and BMW for providing the BMW dataset. This work was partially supported by DFG grant DFG SCHE 663/3-8.

References

[GDN98] GRIEBEL M., DORNSEIFER T., NEUNHOEFFER T.: *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, Philadelphia, 1998. 6

[GKT*08] GARTH C., KRISHNAN H., TRICOCHE X., BOBACH T., JOY K. I.: Generation of accurate integral surfaces in time-dependent vector fields. In *VIS '08: Proceedings of the conference on Visualization '08* (October 2008). 2, 7

[GTS*04] GARTH C., TRICOCHE X., SALZBRUNN T., BOBACH T., SCHEUERMANN G.: Surface techniques for vortex visualization. In *Proceedings of Joint Eurographics - IEEE TCVG Symposium on Visualization* (May 2004), pp. 155–164. 2

[GWT*08] GARTH C., WIEBEL A., TRICOCHE X., JOY K., SCHEUERMANN G.: Lagrangian visualization of flow-embedded surface structures. *Computer Graphics Forum 27*, 3 (May 2008), 1007–1014. 8

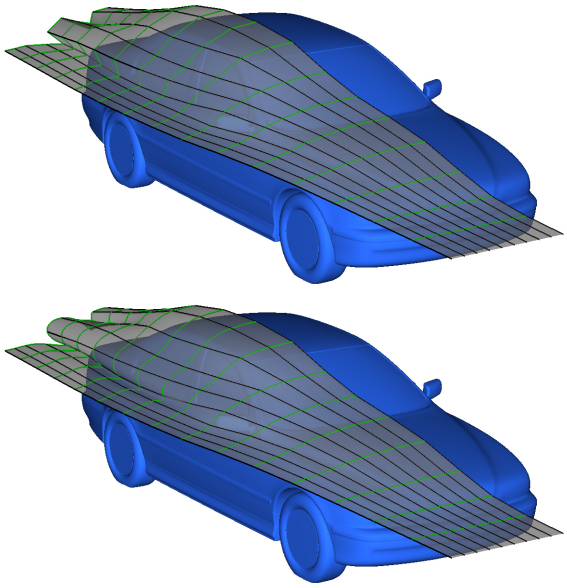


Figure 10: *Stream surfaces in car dataset constructed using 10 stream lines. The upper image shows a surface obtained with linear interpolation. The lower image shows a surface obtained with Hermite interpolation. The difference in the visual quality is striking.*

[Hul92] HULTQUIST J. P. M.: Constructing stream surfaces in steady 3d vector fields. In *VIS '92: Proceedings of the 3rd conference on Visualization '92* (Los Alamitos, CA, USA, 1992), IEEE Computer Society Press, pp. 171–178. 1, 3

[MBS*04] MAHROUS K., BENNETT J., SCHEUERMANN G., HAMANN B., JOY K. I.: Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004), 198–205. 2

[PD81] PRINCE P. J., DORMAND J. R.: High order embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics 7(1)* (1981). 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 *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 151–158. 2

[STWE07] SCHAFHITZEL T., TEJADA E., WEISKOPF D., ERTL T.: Point-based stream surfaces and path surfaces. In *GI '07: Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 289–296. 2

[vW93] VAN WIJK J. J.: Implicit stream surfaces. In *VIS '93: Proceedings of the 4th conference on Visualization '93* (1993), pp. 245–252. 2