

One-Sided Smoothness-Increasing Accuracy-Conserving Filtering for Enhanced Streamline Integration through Discontinuous Fields

David Walfisch · Jennifer K. Ryan · Robert M. Kirby · Robert Haimes

Received: 23 May 2008 / Revised: 8 July 2008 / Accepted: 16 July 2008 / Published online: 13 August 2008
© Springer Science+Business Media, LLC 2008. This article is published with open access at Springerlink.com

Abstract The discontinuous Galerkin (DG) method continues to maintain heightened levels of interest within the simulation community because of the discretization flexibility it provides. One of the fundamental properties of the DG methodology and arguably its most powerful property is the ability to combine high-order discretizations on an inter-element level while allowing discontinuities between elements. This flexibility, however, generates a plethora of difficulties when one attempts to use DG fields for feature extraction and visualization, as most post-processing schemes are not designed for handling explicitly discontinuous fields. This work introduces a new method of applying smoothness-increasing, accuracy-conserving filtering on discontinuous Galerkin vector fields for the purpose of enhancing streamline integration. The filtering discussed in this paper enhances the smoothness of the field and eliminates the discontinuity between elements, thus resulting in more accurate streamlines. Furthermore, as a means of minimizing the computational cost of the method, the filtering is done in a one-dimensional manner along the streamline.

Keywords High-order methods · Discontinuous Galerkin · Filtering · Accuracy enhancement · Streamline integration · Adaptive error control

D. Walfisch · R. Haimes
Department of Aeronautics & Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA

D. Walfisch
e-mail: walfisch@mit.edu

R. Haimes
e-mail: haimes@mit.edu

J.K. Ryan (✉)
Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: J.K.Ryan@tudelft.nl

R.M. Kirby
School of Computing, Univ. of Utah, Salt Lake City, UT, USA
e-mail: kirby@sci.utah.edu

1 Introduction

The discontinuous Galerkin (DG) methods provide a high-order extension of the finite volume method in much the same way as high-order or spectral/*hp* elements [20, 29] extended standard finite elements. The DG methodology allows for a dual path to convergence through both elemental ‘h’ and polynomial ‘p’ refinement, making it highly desirable for computational problems which require resolution fidelity. In the overview of the development of the discontinuous Galerkin method, Cockburn *et al.* [11] trace the developments of DG and provide a succinct discussion of the merits of this extension to finite volumes.

The primary mathematical advantage of DG is that unlike classic continuous Galerkin FEM, which seeks approximations which are piecewise continuous, the DG methodology only requires functions which are L_2 integrable. Much like FEM, DG uses the variational form, however, instead of constraining the solution to be continuous across element interfaces, the DG method merely requires weak constraints on the fluxes between elements. This feature provides a discretization flexibility which is difficult to match with conventional continuous Galerkin methods.

Generating data from a DG simulation, however, is never the end of the story. The simulation science pipeline consists initially of modeling and simulation, then requires evaluation through post-processing which allows for information to be extracted. This post-processed information relates back to the scientific or engineering question of interest. One such means of evaluation is visualization.

Visualizing vector fields has always been considered challenging. There are several visualization techniques that try to address this challenge, such as: particle tracing [22, 31], icon placement [30, 33] or texture [23, 32] based methods. A commonly used technique is streamline visualization. The bias toward using streamlines is in part explained by studies which show streamlines to be effective visual representations for elucidating the salient features of the vector fields [21]. Furthermore, streamlines as a visual representation are appealing because they are applicable for both two-dimensional and three-dimensional fields [35].

The most common means of integrating streamlines is to employ one of the many available numerical ordinary differential equation (ODE) solvers on the vector field. In this case, a new error is added to the simulation error budget, namely, the errors arising from the choice of technique used for post-processing simulation data. Darmofal and Haimes [14], in an attempt to quantify post-processing errors which occur in streamline computations of finite element fields, analyzed different types of time integrators (multi-step, multi-stage and hybrid schemes) for their efficiency, convergence rate and computational cost in stream-, streak- and path-line calculations. They also showed that an improperly chosen time-step can result in non-physical streamlines and incorrect critical points even in continuous fields.

DG discretizations provide an additional challenge to the integration process—element-wise discontinuities violate the smoothness assumptions upon which most of these integration schemes are based. If no manipulation of the field is accomplished, accurate streamlines through DG fields require adaptive time-stepping which effectively integrates up to and over discontinuities in a controlled way [16]. This process tends to be very expensive as many field evaluations are required to effectively “find” and integrate over the discontinuity.

One alternative strategy is to seek filtering techniques which enhance the smoothness of the field without destroying its formal accuracy. Such smoothness-increasing accuracy-conserving (SIAC) filters have been proposed in previous work [28], where they have been shown to ameliorate the difficulties that arise in streamline integration. Because of the nature of the study in [28], two limitations existed: (1) results were only demonstrated for structured quadrilateral meshes and (2) the entire DG field was post-processed. Applying the filter in

this form to realistic data can be challenging in that structured quadrilateral meshes are not optimal and post-processing the entire field can be computationally expensive. Thus, we are employing a strategy for filtering DG data which addresses these two specific limitations.

The goal of this work is to extend the previous ideas presented in [28] by demonstrating an alternative SIAC filtering algorithm that can be applied *during* streamline integration through datasets generated on unstructured discontinuous Galerkin discretizations. The technique uses a one-dimensional convolution kernel to introduce continuity between elements. The idea is to increase smoothness while not introducing additional error in the solution. Furthermore, we work in arc-length coordinates so that we are able to have a one-dimensional implementation that is the same regardless of the dimension of the simulation data. This in turn will aid in accomplishing the goals of visualization of data over more complex geometries while still improving the smoothness of the field, not compromise the accuracy of the data, and provide for a reasonable computational cost.

The paper is organized as follows. In Sect. 2 we discuss the discontinuous Galerkin method for systems of hyperbolic equations as well as presenting background on smoothness-increasing accuracy-conserving filters, including using one-sided filtering. In Sect. 3 we show the practical issues in applying this SIAC filter to multi-dimensional data, which encompasses the choice of characteristic length for scaling the convolution kernel, integration of the convolution kernel, as well as time-stepping techniques for obtaining the streamline. We also provide a discussion of implementation details needed to understand the application of the filters and demonstration of the efficacy of the filters prior to streamline integration. We complete this by providing applications to simulation data in Sect. 4 that demonstrate the effectiveness of this SIAC filter. A summary of our findings and a discussion of future work are presented in Sect. 5.

2 Background

2.1 The Discontinuous Galerkin Method

The discontinuous Galerkin (DG) method for hyperbolic equations has been discussed extensively by both Cockburn *et al.* [4–6, 8, 9] and others [2, 7]. For the purposes of this paper, we are interested in systems of equations of the form

$$\mathbf{u}_t + \nabla \cdot \mathbf{F}(\mathbf{u}) = 0 \quad \text{in } (0, T) \times \Omega. \tag{1}$$

To create a DG approximation, we first construct a tessellation of the domain Ω , $\mathcal{T}(\Omega) = \tilde{\Omega}$, and an approximation space, \mathbf{V}_h , consisting of piecewise polynomials of degree less than or equal to k on each element of $K \in \tilde{\Omega}$, where $k + 1$ is the order of accuracy of the approximation. To obtain the approximation, we multiply (1) by a test function $\mathbf{v} \in \mathbf{V}_h$ and integrate by parts to obtain the variational formulation:

$$\frac{d}{dt} \int_K \mathbf{u}(\mathbf{x}, t) \mathbf{v} d\mathbf{x} + \sum_{e \in \partial K} \int_e \mathbf{F}(\mathbf{u}) \cdot \mathbf{n}_{e,K} \mathbf{v} d\Gamma - \int_K \mathbf{F}(\mathbf{u}) \cdot \nabla \mathbf{v} d\mathbf{x} = 0, \tag{2}$$

where $\mathbf{n}_{e,K}$ denotes the outward unit normal to edge e . The numerical scheme is then given by

$$\frac{d}{dt} \int_K \mathbf{u}(\mathbf{x}, t) \mathbf{v} d\mathbf{x} + \sum_{e \in \partial K} \int_e \mathbf{h}(\mathbf{u}(\mathbf{x}^-, t), \mathbf{u}(\mathbf{x}^+, t)) \mathbf{v} d\Gamma - \int_K \mathbf{F}(\mathbf{u}) \cdot \nabla \mathbf{v} d\mathbf{x} = 0 \tag{3}$$

for all test functions $\mathbf{v} \in \mathbf{V}_h$ where $\mathbf{h}(\mathbf{u}(\mathbf{x}^-, t), \mathbf{u}(\mathbf{x}^+, t))$ represents the numerical flux.

The key points to retain from this brief DG introduction are the following: 1) DG requires a valid tessellation of the domain of interest to be constructed, but there are no restrictions on the types of elements (e.g. triangles or quadrilaterals in 2D) or combination of elements, only the normal constraints found in the finite element and finite volume literature on element quality; 2) the approximation space is piecewise discontinuous with bounded jumps. It is precisely these two components of DG which make post-processing of DG simulation data challenging as most current post-processing techniques assume that the underlying field on which they are operating is continuous. This leads us naturally to the question of whether one can make a DG field continuous (without destroying the formal accuracy of the results), which is the issue to be addressed in the next section.

2.2 Smoothness-Increasing Accuracy-Conserving Filters

There have been many filtering techniques developed that aim to smooth out discontinuous fields and create a continuous function from a discrete solution. A group of these methods uses B-splines for the filtering. An overview of B-spline filtering techniques can be found in [25]. Hou and Andrews [19] use linear combination of cubic B-splines for image processing. Their technique is very similar to the one used in this work. Mitchell and Netravali [24] also use piecewise cubic filters for image reconstruction. Similar to the work mentioned above, we also implement a technique based on B-splines; however, the B-splines are chosen based on the polynomial order of the discontinuous Galerkin solution. This smoothness-increasing accuracy-conserving filter is outlined below as well as in the series of papers [10, 12, 13, 26–28].

2.2.1 One-Sided Filtering

In the one-dimensional application of the smoothness-increasing accuracy-conserving filter to streamlines, we concentrate on the implementation of the one-sided filter. This one-sided convolution kernel was introduced by Ryan and Shu [26] and displays only minor differences from the symmetric kernel:

$$K^{2(k+1),k+1}(x) = \sum_{\gamma=-2k-1}^{-1} c_{\gamma}^{2(k+1),k+1} \psi^{(k+1)}(x - \gamma).$$

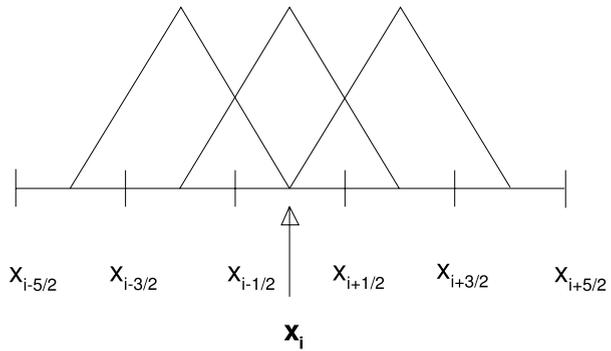
Note that the support of the B-splines are simply shifted to one side of the current element being filtered in a way which is analogous to one-sided finite difference stencils. Additionally, the coefficient calculation for $c_{\gamma}^{2(k+1),k+1}$, like the support of the B-splines, is also non-symmetric and is given by

$$\begin{bmatrix} \int \psi^{(k+1)}(x - y - 2k - 1)y^0 dy & \dots & \int \psi^{(k+1)}(x - y - k)y^0 dy \\ \vdots & \ddots & \vdots \\ \int \psi^{(k+1)}(x - y - 2k - 1)y^k dy & \dots & \int \psi^{(k+1)}(x - y - k)y^k dy \end{bmatrix} \begin{bmatrix} c_{-2k-1} \\ \vdots \\ c_{-k} \end{bmatrix} = \begin{bmatrix} x^0 \\ \vdots \\ x^k \end{bmatrix}.$$

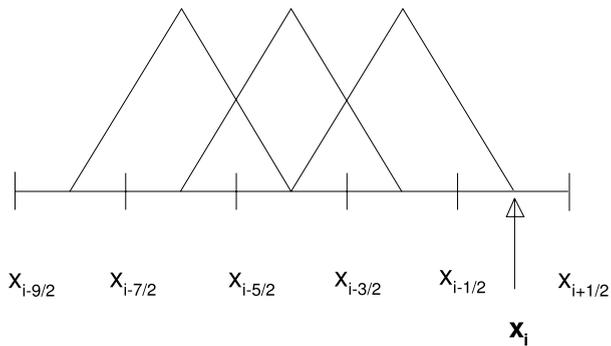
For the piecewise linear case, we would have

$$\begin{bmatrix} 1 & 1 & 1 \\ x + 3 & x + 2 & x + 1 \\ x^2 + 6x + \frac{55}{6} & x^2 + 4x + \frac{25}{6} & x^2 - 2x + \frac{7}{6} \end{bmatrix} \begin{bmatrix} c_{-3} \\ c_{-2} \\ c_{-1} \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}.$$

Fig. 1 B-splines used in the convolution kernels for the $k = 1$ case. The actual kernel is a linear combination of these B-splines with the emphasis being on the current element being filtered, that is, we want to filter the element containing x_i



(a) B-splines for the symmetric kernel



(b) B-splines for the fully one-sided kernel

We emphasize that the entire one-sided kernel has its support contained to one side of the point of interest, and therefore the limits of integration in the convolution change. Therefore, the solution is not needed from the other side of the point being filtered. This is an important feature since the future of the streamline is not known when the filtering scheme is employed.

It has also been demonstrated that the one-sided technique improves accuracy from order $k + 1$ to $2k + 1$ for uniform meshes [26]. When the underlying mesh becomes non-uniform, the method loses its accuracy enhancing characteristic in the sense of order of convergence, but keeps an accuracy conserving behavior [13]. This is important, as when we consider the multi-dimensional approximation, we can no longer guarantee that the mesh is uniform. This is because we apply the filter in a one-dimensional manner along the streamline. Even if the data is given over a uniform quadrilateral mesh, the streamline may cut through a corner of an element. However, it is the smoothness increasing part of the filter that is important for this application.

Figure 1 shows an example of the B-splines used in the symmetric kernel as well as the fully one-sided kernel for $k = 1$. The x -axis is scaled by the characteristic length, and x_i corresponds to the location of the point that is being post-processed. The y -axis is scaled by the kernel coefficients.

3 One-Sided Filtering for Streamline Calculation

We now address adapting the aforementioned one-sided filtering techniques to one-dimensional streamlining from multi-dimensional data. We first discuss the broader picture of streamlining and then the options for streamline calculation.

Consider a computational domain Ω over which a tessellation $\tilde{\Omega} = \mathcal{T}(\Omega)$ exists consisting of triangles and quadrilaterals in two-dimensions or tetrahedra, hexahedra, prisms and pyramids in three-dimensions. This work is concerned with piecewise smooth fields built over $\tilde{\Omega}$. That is, assume that we take two geometrically adjacent elements $K_1, K_2 \in \tilde{\Omega}$, the functions $f_1 : K_1 \rightarrow \mathbb{R}$ and $f_2 : K_2 \rightarrow \mathbb{R}$ are smooth but do not have cross-element continuity. To illustrate the idea, we consider a two-dimensional example. Assume

$$\mathbf{v} = \begin{bmatrix} u(x, y) \\ v(x, y) \end{bmatrix}.$$

For each element $K \in \tilde{\Omega}$, $\mathbf{v} : K \rightarrow \mathbb{R}^2$ is well-defined. However, if \mathbf{v} is evaluated over $\tilde{\Omega}$ as it is for the discontinuous Galerkin approximation, then vector field would be uniquely defined on the open set of elements only; it would be multiply defined on the traces of the elements (i.e. on the edges of the elements in two-dimensions or on the faces of the elements in three-dimensions). This means that for element interfaces there is more than one choice for \mathbf{v} . It is exactly this inter-element (non-physical) discontinuity that we seek to smooth, without destroying the accuracy of our DG approximation.

The idea is the following: We are interested in defining streamlines of the vector field \mathbf{v} . The directional derivative is

$$\frac{d}{d\xi} = \mathbf{v} \cdot \nabla.$$

This derivative allows us to define a streamline \mathbf{r} of the vector field \mathbf{v} as being curves in which

$$\frac{d\mathbf{r}}{d\xi} = \mathbf{v}$$

holds for any point on the curve. Note that $\mathbf{v} = \mathbf{v}(\mathbf{r})$.

In order to trace this stream line curve, we use a time-stepping algorithms such as Euler Forward. If we define an initial starting point \mathbf{r}^0 and a spacing $\Delta\xi$, the Euler-Forward approximation of the curve is generated by iterating

$$\frac{\mathbf{r}^{n+1} - \mathbf{r}^n}{\Delta\xi} = \mathbf{v}(\mathbf{r}^n).$$

for $n = 1, \dots, N$. In this expression, the vector field \mathbf{v} is being evaluated along the streamline given by samples of \mathbf{r} . The higher the order of approximation used for the derivative term $\frac{d\mathbf{r}}{d\xi}$, the more smoothness of the field that is required. The difficulty lies in the fact that if the vector field is given by a DG vector field, element-wise smoothness is not available.

To ameliorate the discontinuity problem, Steffen *et al.* used the SIAC filter to increase the smoothness of the field without compromising accuracy [28]. The streamline calculation was performed by filtering the entire multi-dimensional vector field \mathbf{v} using the B-spline convolution kernels to produce a new field $\tilde{\mathbf{v}}$ which is smooth across interfaces. Once this is done, it was possible to employ a wide variety of different ODE integration algorithms.

Although this technique is effective in the sense of handling discontinuities, it required filtering the entire field, which can be very expensive, and it assumed only simplified geometry (a uniform quadrilateral mesh). In general, this approach has these additional issues:

- Mesh Generation. In 2-D it is possible to automatically partition a bounded region into quadrilaterals where the post-processing convolution kernels can be applied. Automatically blocking complex shaped volumes into hexahedral regions (in 3-D) is an unsolved problem.
- Maintaining boundary values. The one-sided kernel needs to be applied when the region of interest touches a boundary. The current form of this kernel does not preserve values at the bounds of the domain. Therefore there is no mechanism to ensure that resultant streamlines do not pierce walls when the viewed solution has flow tangency (for Euler calculations) or zero velocity enforced at solid boundaries used in viscous simulations.

We therefore seek an alternative that will allow for non-uniformity in the mesh and will not require filtering of the entire field. The idea is the following: assume that one is given a collection of points $\{\dots, \mathbf{r}^{n-2}, \mathbf{r}^{n-1}, \mathbf{r}^n\}$ along a streamline. Additionally, assume that one has filtered the vector field at those points, that is, $\{\dots, \tilde{\mathbf{v}}^{n-2}, \tilde{\mathbf{v}}^{n-1}, \tilde{\mathbf{v}}^n\}$. To find the next position along the streamline, implement the backward differential formula and obtain \mathbf{r}^{n+1} . It is important to note that it will be necessary for the streamline integration step to be chosen such that the new position, \mathbf{r}^{n+1} , is within the convolution filter support of the points used in the backward differentiation. This guarantees that the influence of element boundaries are properly used. Thus, the filtering is done as we proceed forward in time for the streamline calculation. Details of this alternative method are outlined below.

In order to apply the filter introduced in Sect. 2.2.1, it is necessary to slightly modify the kernel and the process of evaluation of the data. As mentioned previously, we no longer assume uniformity of elements. Recall that the previous kernel was scaled by the uniform element size. Therefore, we must investigate a characteristic length for the scaling of the B-splines. Additionally, it is necessary to address the integration of the convolution kernel for filtering the discontinuous Galerkin solution. Lastly, since this is a streamline calculation, we must consider the proper time integration method that will take into account our filtering method. To introduce the incorporation of all these ideas we develop a model problem. Once we have discussed the necessary components of forming the SIAC filter, we test the validity of the SIAC filter on our model problem.

3.1 Characteristic Model Problem

Given a curve through a discontinuous Galerkin data set, the approximation naturally contains oscillations. For purposes of illustration, we have taken a multi-dimensional function consisting of smooth oscillating circles and projected it to a discontinuous Galerkin basis (see Fig. 2). The idea for using a one-sided streamline integrator is this: If we take a one-dimensional cut of this approximation, it will still contain oscillations on the lower dimensional manifold, as shown in Fig. 4 (left). Given the oscillatory nature of the approximation, we wish to filter the solution. It is desired that the filter increases the smoothness of the approximation and conserve the accuracy of the projection.

In the constructed model problem, a two dimensional vector field was created from

$$\begin{bmatrix} u(r, \theta) \\ v(r, \theta) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \cos(20 \cdot \theta) \cos(\theta) - r \sin(\theta) \\ \frac{1}{2} \sin(20 \cdot \theta) \sin(\theta) - r \cos(\theta) \end{bmatrix}. \quad (4)$$

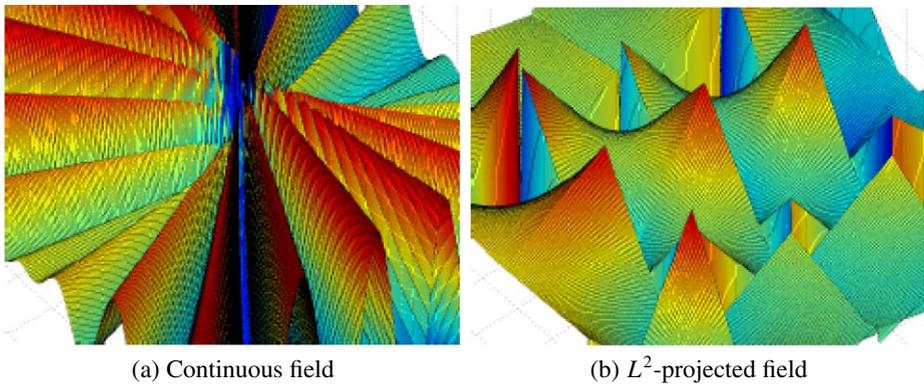
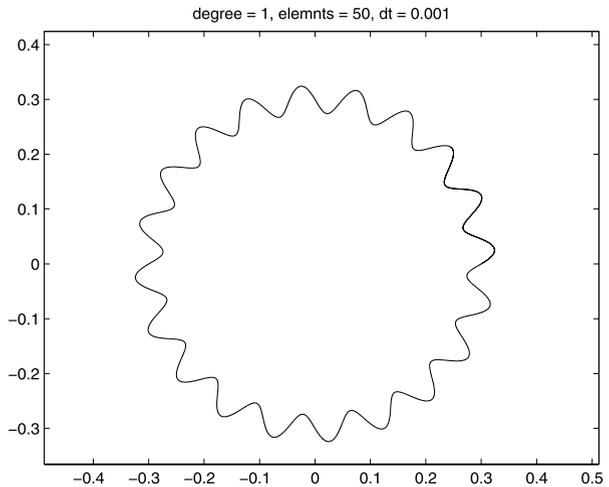


Fig. 2 Multi-dimensional field containing smooth oscillating circles. The two fields show the difference between continuous data and the data types that are obtained by the discontinuous Galerkin method. The latter being illustrated by using an L^2 -projection of the exact solution onto a DG basis

Fig. 3 A sample streamline of oscillating circles over the continuous field with a starting location of (0, 0.3). This will be our model problem. It is desirable that the filter reproduce this information from the discontinuous data



This has streamlines which are also oscillating circles. Figure 3 shows a sample streamline that was calculated on the original continuous field with a fourth-order Runge-Kutta (RK4) method and a time-step $\Delta t = 0.001$. It can be seen that there are a total of 20 cycles of oscillation around one loop.

The field was then projected onto a uniform mesh with the L^2 projection as a first-order polynomial to make it discontinuous. This replicates the behavior of a numerical approximation generated by a discontinuous Galerkin solver. In Fig. 4, plots of the exact u values (x -velocity) along the streamline in arc-length coordinates (x -axis) are given as well as the projected u values in arc-length coordinates. It can be seen that the projected solution is neither continuous nor smooth.

Once we have developed our filtering algorithm, we will test it on this problem in order to verify that it indeed can replicate the exact streamline.

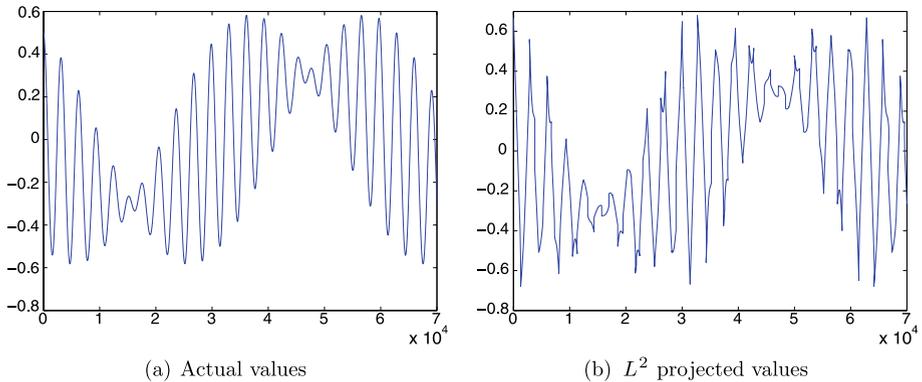


Fig. 4 x -velocity along the streamline (x -axis is in 10^{-4} arc-length coordinates). The *left figure* represents the solution of the continuous field and the *right figure* represents the discontinuous Galerkin field

3.2 Characteristic Length

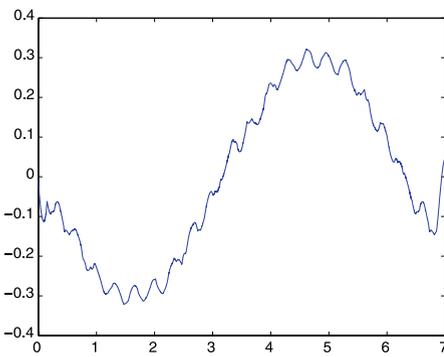
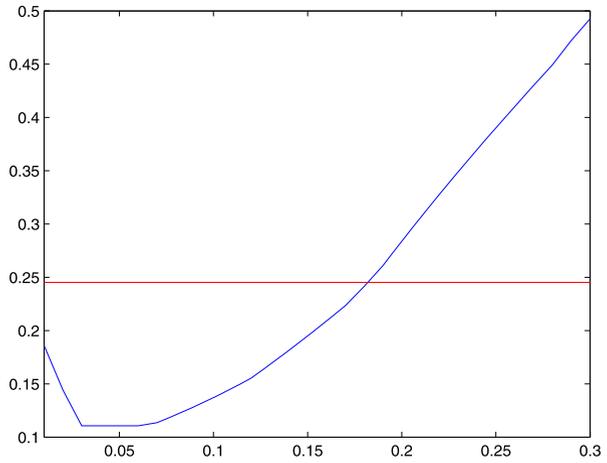
The characteristic length is the scaling factor of the convolution kernel. For uniform meshes it is the element size. However, in this method we have created a one-dimensional streamline using multi-dimensional data, and consequently the information obtained from the discontinuous Galerkin solution is on a nonuniform mesh. Therefore, it is necessary to explore the ideal characteristic length one would need to use for filtering over complex geometries.

In order to analyze the ideal characteristic length, numerical experiments were performed on the model problem with the one-sided filter applied to this solution along the streamline. The approximated streamline was sampled with a time step size of $\Delta t = 10^{-7}$ with the integration performed using Jacobi-Gauss quadrature rule with $k + 1$ quadrature points along the real oversampled streamline. We take such an extreme time step to allow for the real coordinates of every quadrature point to be known exactly thus allowing us to closely examine the characteristic length. The filtered solution was calculated with characteristic length ranging from 0.014 to 0.3 in arc-length coordinates. The maximum error between the real and the filtered solution is shown in Fig. 5. The horizontal line is the maximum error between the projected and the real solution.

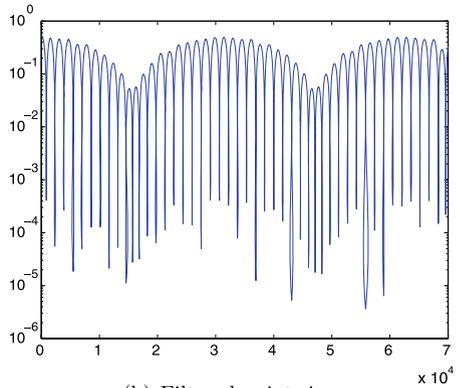
The maximum error improves during the filtering for characteristic lengths smaller than 0.18. The reason for this is that if the kernel support is too large then high frequencies of the solution are filtered out. At the same time if the characteristic length is too small, the support is not large enough to reduce the unwanted oscillations. Figure 6 shows the two extreme cases, one with characteristic length of 0.3, the other with 0.05.

Next, the same test was run on similar vector fields, with a smaller number of oscillation cycles around the circle. It was observed that when the characteristic length was set to the largest element size, the filtered maximum error was always lower than the non-filtered maximum error. Table 1 summarizes the results by showing the maximum error for the non-filtered (projected) and the filtered solution, with the maximum element size as the characteristic length. Figure 7 shows the filtered u values and the pointwise error when the maximum element size was used as the characteristic length for the convolution kernel. It can be seen that the solution is continuous and significantly smoother than the original projected solution (Fig. 4b). Based on these results, the experiments used in this paper have the characteristic length equal to the length of the largest element size.

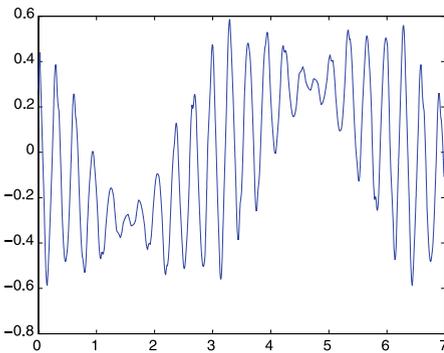
Fig. 5 Maximum error in the first-component (y-axis) versus characteristic length in arc-length coordinates (x-axis). The horizontal line is the maximum error between the projected and the real solution. The maximum error improves with filtering for characteristic lengths smaller than 0.18



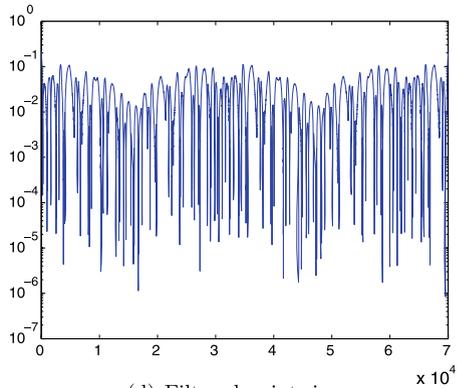
(a) Filtered first-component value



(b) Filtered pointwise error $\times 10^4$



(c) Filtered first-component value



(d) Filtered pointwise error $\times 10^4$

Fig. 6 Results with a large characteristic length of 0.3 (top) and the smaller characteristic length of 0.05 (bottom). For large characteristic lengths, the high frequencies are filtered out. For small characteristic lengths, oscillations are still present

Table 1 Filtered error when the characteristic length is set to the maximum element size. The filtered maximum error is always lower than the non-filtered

Number of cycles	Maximum element size	Maximum filtered error	Maximum projected error
2	0.154	0.068	0.104
4	0.244	0.018	0.019
6	0.179	0.014	0.031
8	0.108	0.021	0.052
10	0.151	0.048	0.063
12	0.126	0.051	0.105
14	0.124	0.078	0.151
16	0.158	0.159	0.191
18	0.138	0.136	0.172
20	0.101	0.137	0.249

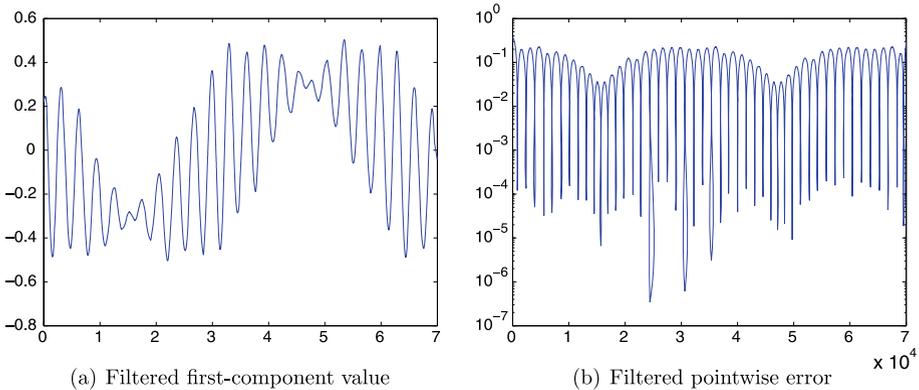


Fig. 7 Results with characteristic length set to maximal element size of 0.14. The filtered maximum error is always lower than the non-filtered

3.3 Integration of the Filtering Kernel

The convolution step of the filter requires evaluation of an integral,

$$u^*(x) = \frac{1}{h} \int_{-\infty}^{\infty} K_h \left(\frac{y-x}{h} \right) u_h(y) dy.$$

For this integral the Jacobi-Gauss quadrature rule is used to obtain the exact answer. As the integrand is discontinuous, each continuous subregion is numerically integrated according to the appropriate quadrature rule.

For example, when the filtered solution needs to be calculated at point y as in Fig. 8, the integrand is a polynomial of degree $2k$ within each continuous region, but there are discontinuities at the boundaries. For the numerical integration to give the exact solution it requires $k + 1$ quadrature points per segment. The integrand is then evaluated at these locations, multiplied by the appropriate weights, and the result is the summation.

Additionally, since we are operating on a lower dimensional manifold, we use arc-length coordinates. These coordinates of the quadrature points have to be converted to physical

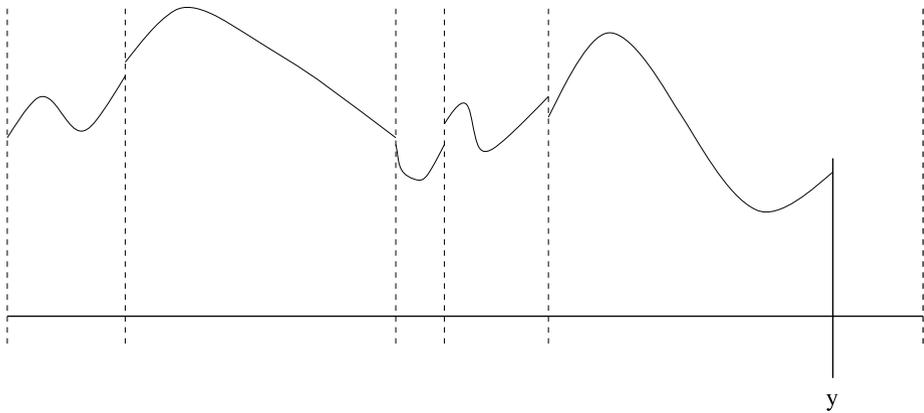


Fig. 8 Discontinuous integrand as seen by a streamline

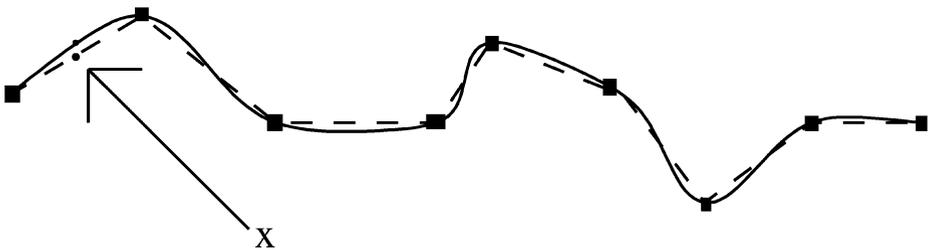


Fig. 9 The *continuous line* represents a real, higher-order streamline. The *dashed line* is created from linear line segments

coordinates in order to obtain the solution data. Since the streamline is calculated with a time integrator, it is built up and visualized from linear line segments, while in reality it is a higher order curve. However, the quadrature points of the integral do not necessarily coincide with the locations that the time integrator calculates. To get the real coordinates of a quadrature point that is not one of the actual sample points, an interpolation is done between the two end points of the linear segment. Once the coordinates are found, the real high-order interpolant is used as the solution at the point. Figure 9 shows how there can be a difference between the real coordinates of a quadrature point if it is calculated on linear segments or higher-order curves. The squares represent the locations the time-integrator calculated, x is a quadrature point along the streamline.

Alternatively, Hermite interpolation could also be used. In Table 2 the maximum errors are compared for five different time steps for both linear polynomial interpolation and Hermite interpolation. In this table, the errors are compared for the case of 20 cycles of oscillation with the maximum element size 0.101 as characteristic length. For cases with fewer oscillations, the difference between hermite and linear interpolation is smaller as one can expect since the frequency decreases. This table gives us insight into the fact that although the Hermite interpolation is always better than the linear polynomial interpolation, other errors are clearly dominating. Therefore we can conclude that for a small enough time steps, the linear interpolation is sufficient since the impact of the Hermite interpolation diminishes with decreasing time step.

Table 2 Maximum filtered error for linear polynomial interpolation and Hermite interpolation for five different time steps

Time step	Maximum filtered error	
	Linear	Hermite
10^{-2}	0.21017	0.19863
10^{-3}	0.17634	0.17101
10^{-4}	0.15303	0.15186
10^{-5}	0.14371	0.14291
10^{-7}	0.13693	0.13691

3.4 Time Integration Along a Streamline

Time-stepping represents the second major numerical approximation used in this approach. There are a wide variety of time-integration techniques available for streamline calculations, a few are presented in [1, 34]. These streamline time-stepping techniques are normally broken down into multi-stage and multi-step methods. For example, the generic form of a multi-step scheme is the following:

$$\sum_{t=0}^s \alpha_t x^{n+1-t} = k \sum_{t=0}^s \beta_t u(x^{n+1-t}, t^{n+1-t}),$$

where s represents the number of steps, for explicit schemes $\beta_0 = 0$. The main disadvantage of higher order accurate multi-step methods ($s > 1$) is that they require $s - 1$ startup steps that are lower order accurate. Thus additional error may be introduced into the time-integration due to initialization choices. Computationally these schemes tend not to be as expensive as multi-stage methods since only one velocity field needs to be evaluated per iteration; the previous positions can be stored and reused.

The other family of methods is multi-step methods. Even though some families of multi-stage methods can be written down in a generic form (for example Strong Stability Preserving Runge-Kutta methods [17, 18]), there is no general descriptive formula for all multi-stage methods. The most commonly used multi-stage techniques are from the Runge-Kutta family. One of the main concerns with these schemes is that they require the calculation of the velocity at intermediate steps which can be computationally expensive and result in longer calculation times. In order to calculate the velocity, the underlying element in the grid that contains the intermediate stage needs to be found. This process is the most computationally expensive part of the streamline calculator. In addition, for cases when the solution is only known at sample points, intermediate velocity calculations can cause difficulty. This difficulty is usually solved by the use of a Newton-Raphson scheme to get an interpolated value. Introducing this extra numerical algorithm for interpolation produces additional error in the streamlining process. However, if the time integrator has access to the entire polynomial solution within the time integration domain and not just the visualized one, the interpolation phase can be avoided and the order of accuracy is not compromised.

There are also methods from both the multi-step and multi-stage families that use adaptive time-steps (e.g. predictor-corrector, RK-4/5 [1]) in order to bound the local (per step) error. In a discontinuous Galerkin setting, the jump in the solution at element boundaries can appear as “errors” to the time-integrator (in the sense that they show up as places in need of local refinement). The local error depends on the size of the jump and the size of the time step, Δt . In order to introduce the minimal amount of error at large jumps in the solution, the timestep must be small. As Δt becomes smaller, more iterations are needed and getting past

a single element boundary becomes computationally expensive. As a result, if one wishes to achieve a relatively accurate streamline with these methods, the process can become rather time consuming due to the many discontinuities the streamline encounters in its path [16]. Additionally, these methods still rely on the velocities being calculated at locations different than the sample points.

In this work we assume that we are dealing with data that is discontinuous at element interfaces. This places restrictions on the type of time integration techniques that can be used. In order to use implicit schemes, the solution needs to have smooth first derivatives. For this work, we assume that we only know information about the solution itself. Furthermore, we also want to implement this filter with an explicit method that demonstrates a high order of convergence to reduce the number of time-steps needed for the calculation. The most commonly used method is RK4 or RK4/5. But, the implementation of these techniques requires sampling the field from locations that are not along the streamline. To see this, we can examine the RK4 formula shown below where steps 3 and 4 do not satisfy the requirement of being along the streamline.

$$\begin{aligned}
 k_1 &= hf(t_n, y_n) \\
 k_2 &= hf\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_1\right) \\
 k_3 &= hf\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}k_2\right) \\
 k_4 &= hf(t_n + h, y_n + k_3) \\
 y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4.
 \end{aligned}$$

Since the convolution proposed in this work is done along the streamline, calculating the filtered solution for intermediate stages that are not on the path is inconsistent and may lead to unquantifiable errors. Thus we choose to use a second order Runge-Kutta method to reduce these errors. RK2 uses one intermediate stage, which is a half time-step in the future along the streamline.

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

Using this method requires two filtering steps per iteration since the solution has to be calculated at y_n and $y_n + \frac{h}{2}k_1$.

3.5 Validation of One-Dimensional Filter for Multi-Dimensional Data

Now that the foundation has been laid for using the smoothness-increasing accuracy-conserving filter for multi-dimensional streamline calculation, we can link the components together to produce an algorithm and test the validity on the model problem consisting of oscillating circles outlined in Sect. 3.1. This model problem will aid us in predicting whether this approach is feasible for more realistic examples as those given in Sect. 4, where the exact solution is unknown.

The algorithm that we developed requires evaluation using filtered elements and then integrating in time. We can apply this in a multi-dimensional manner by using arc-length coordinates. More specifically, we seek to obtain the filtered solution \mathbf{v}^{n+1} . This is done in the following way:

Step 1: Assume we have a collection of points obtained from the discontinuous Galerkin field $\{\dots, \mathbf{r}^{n-2}, \mathbf{r}^{n-1}, \mathbf{r}^n\}$ along a streamline.

Step 2: Based on the sample history, create a local parameterization of the curve (*i.e.* parameterized approximation of the streamline). Do this either:

- (a) Using the derived streamline position values to fit polynomials; or
- (b) Using both position and tangent information to fit Hermite polynomials or more generally splines.

We note that this is now the domain over which we integrate the filtering kernel.

Step 3: Obtain the filtered collection of points along the curve, $\{\dots, \tilde{\mathbf{v}}^{n-2}, \tilde{\mathbf{v}}^{n-1}, \tilde{\mathbf{v}}^n\}$, using the one-sided SIAC filter.

$$u^*(x) = \frac{1}{L} \sum_{\gamma=-2k-1}^{-1} c_{\gamma}^{2(k+1),k+1} \sum_{j=-k'}^0 \int_{I_{i+j}} \psi^{(k+1)} \left(\frac{y-x}{L} - \gamma \right) u_{h,i}(y) dy. \quad (5)$$

Step 4: March forward in time to obtain \mathbf{r}^{n+1} . We emphasize again that this new position will be within the support of the filter.

We note that in this paper, the model problem consists of piecewise linear data and linear polynomial interpolation is used for step (2). Generally, the filter requires $2k' + 1$ pieces of information including the current element, where $k' = \lceil \frac{3k+1}{2} \rceil$. Thus for the linear filter five pieces of information are required. To begin the streamline integration we assume that we start at the fifth piece of information.

To test the accuracy of the smoothness-increasing accuracy-conserving filter on one-dimensional streamline calculation from multi-dimensional data, we calculated the streamlines of our model problem. These results are presented in Fig. 10. Without filtering, the calculated streamlines do not remain consistent with the solution and spiral into the circle. With filtering, the calculated streamlines are closer to the actual solution.

4 Applications to Simulation Data

To demonstrate the SIAC filtering method to real world data, streamlining is applied to the results from aerodynamic simulations using MIT’s *Project X* DG solver [15]. Below we test the one-dimensional SIAC filter on two cases of inviscid flow, and one of viscous flow.

4.1 Two-Dimensional Transonic Airfoil

In this example, the streamlines are calculated for a NACA 0012 airfoil with an angle of attack $\alpha = 1.128$ and Mach Number $Ma = 0.8$. The solution uses a fifth order polynomial in each element that is built up from 1836 triangles. The geometry of the airfoil is represented by cubic triangle sides, otherwise all other triangle segments are linear.

Streamline results are shown for a two-dimensional inviscid (Euler Equations) transonic airfoil simulation. Figure 11 depicts two streamlines with the same starting position. Note that in the figure the geometry is drawn as linear but it is cubic in the simulation. Both of the

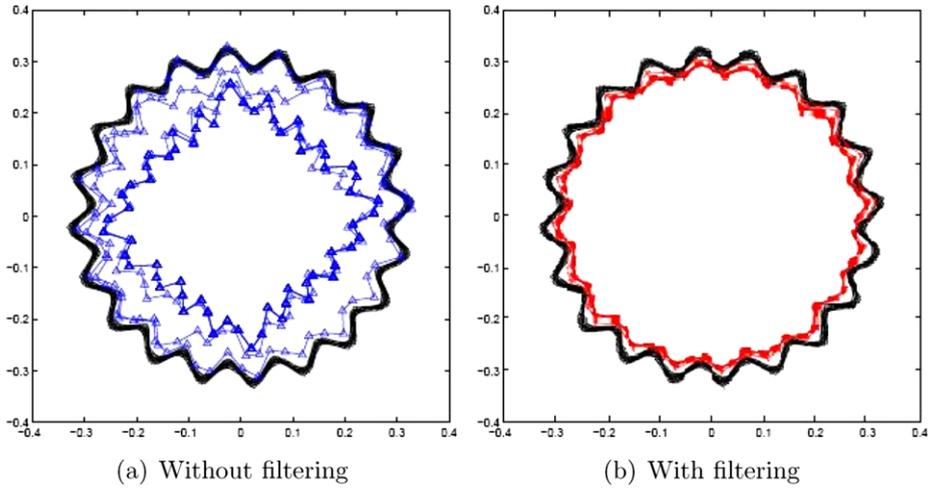
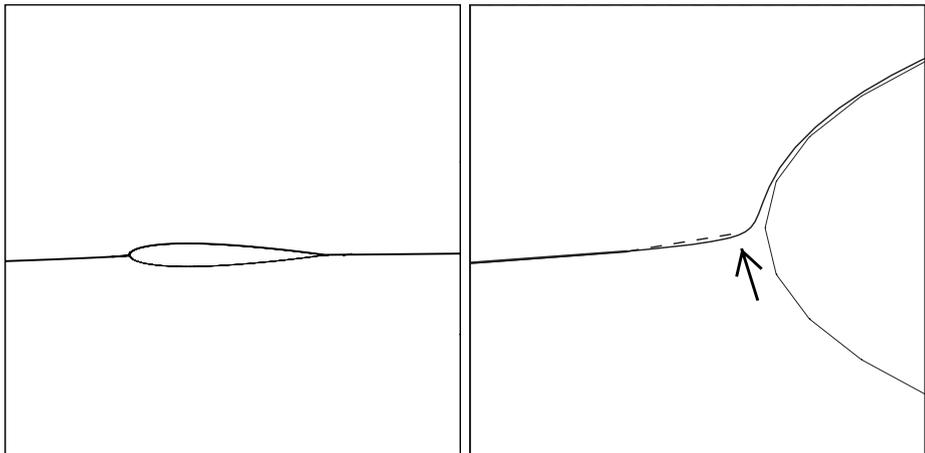


Fig. 10 Without filtering (*left*), the calculated streamlines spiral into the circle. With filtering (*right*), the calculated streamlines are closer to the actual solution. The *thin line* represents the real solution and *bold line* represents the calculated streamline



(a) Streamline close to the airfoil (b) Zoomed view of the streamlines

Fig. 11 Streamline results on a 2D transonic simulation. *Dashed line* is a streamline without post-processing; *solid line* denotes the streamline with post-processing. The streamline stops just short of the airfoil, as emphasized by the *arrow*

streamlines shown were traced using a second-order Runge-Kutta method with $\Delta t = 0.01$; the dashed line was calculated without filtering and stops close to the airfoil boundary as indicated by the arrow. This problem can be caused by the following two considerations:

- *Element jumps*: The velocity direction indicates that the flow is leaving a cell at a particular face. Its neighboring element (the one sharing the face) can also specify that the flow

is leaving the element at that face (acting like a sink). This situation can easily occur in regions close to stagnation or where the flow is under-resolved.

- *Time-step/Numerical errors*: The integrator targets the next position outside of the computational domain. This can readily occur because the integration scheme applied here does not check for local errors.

In this situation, if Δt is reduced to 0.0015, the non-filtered streamline can also be fully evaluated (i.e. it continues past the leading edge and exits the domain) suggesting that the problem is a numerical integration issue. It should be noted that the calculation time of the filtered solution is roughly four times longer than the simple integration scheme. For this particular streamline applying the simple integrator requires seven times the number of iterations than the SIAC scheme. Therefore the overall performance of the SIAC scheme is better and results in fewer line segments allowing for faster rendering.

4.2 Three-dimensional Transonic Airfoil

All of the test cases and examples have been two-dimensional. The strength of the SIAC filtering scheme described in this paper is that the difference between streamlining in 2D and 3D is minor. This is a major contrast to using filter pre-processing in which the filter must be applied in each coordinate direction. This is because all filtering is done during the integration and along the trace as the new positions are computed.

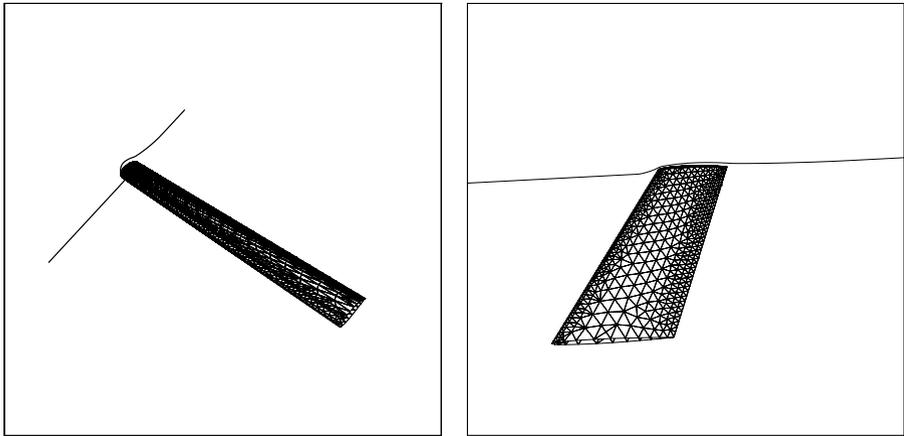
In this example, SIAC streamlining is shown from the results of an inviscid (Euler Equations) three-dimensional simulation. The geometry is that of an isolated Onera M6 wing and the flow conditions were $Ma = 1.5$. The DG solution employs second-order polynomials within each element of a tetrahedral mesh containing 45,937 cells. All tetrahedra have planar faces interior to the domain and curved edges and faces that sit on the wing in an isoparametric (i.e. quadratic geometry) setting.

A streamline was traced close to the wingtip, and the results are shown in Fig. 12. Again, we emphasize that this example demonstrates that we can go from two-dimensional streamline filtering to three-dimensional streamline filtering with little extra effort.

4.3 Viscous Flow

The last example that we present is a two-dimensional viscous flow (using the Reynolds-Averaged Navier-Stokes Equations) calculated around a NACA 0012 airfoil with the following conditions: $\alpha = 2.0$, $Ma = 0.5$ and $Re = 5,000$. The mesh consists of 4068 quadratic adapted elements with a $P = 2$ solution interpolation. Figure 13 shows the type of streamline that we can expect to obtain; separation can be observed toward the trailing edge. It should be noted that the SIAC scheme was used to generate these streamlines.

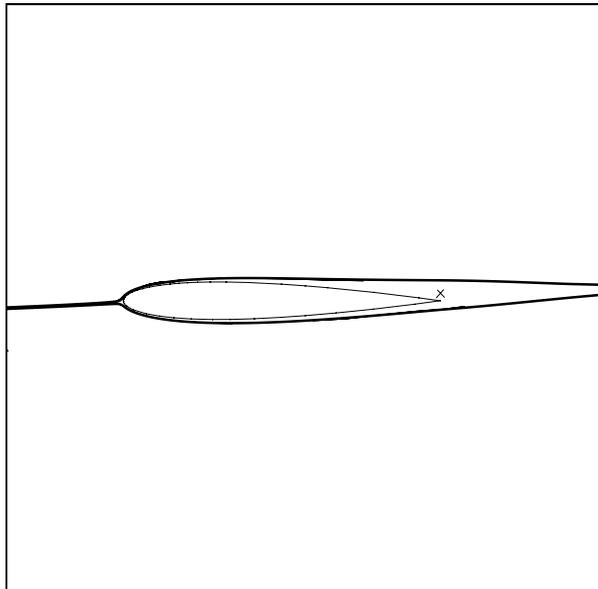
Figure 14 shows a streamline in the separated region at the trailing edge. $\Delta t = 0.1$ is used with a second-order Runge-Kutta scheme in both cases. It can be seen that the non-filtered streamline (top two images) spirals inward toward a critical point. This is a physically incorrect situation. However, for the streamline calculation done with filtering (bottom two images), the streamline spirals out and finally leaves the airfoil. This example demonstrates how, when using ODE integrators, errors can accumulate and produce physically erroneous results (especially in the presence of critical points). Filtering of the solution ameliorates this problem in both an accurate and efficient manner.



(a)Streamline and wing looking from above. (b)Streamline and wing discretization looking from the wing-root.

Fig. 12 Streamline close to the wingtip for the 3D transonic case

Fig. 13 Streamline close to the airfoil. Separation can be seen toward the trailing edge. The x location marks the starting point of the streamlines on Fig. 14



5 Conclusions and Future Work

A smoothness-increasing accuracy-conserving filtering technique was introduced to obtain accurate streamline calculations by introducing continuity at element interfaces. The method uses an on-the-fly accuracy conserving, smoothness-increasing filter with a second-order Runge-Kutta (RK2) method to trace streamlines. By eliminating the element-wise jumps in-

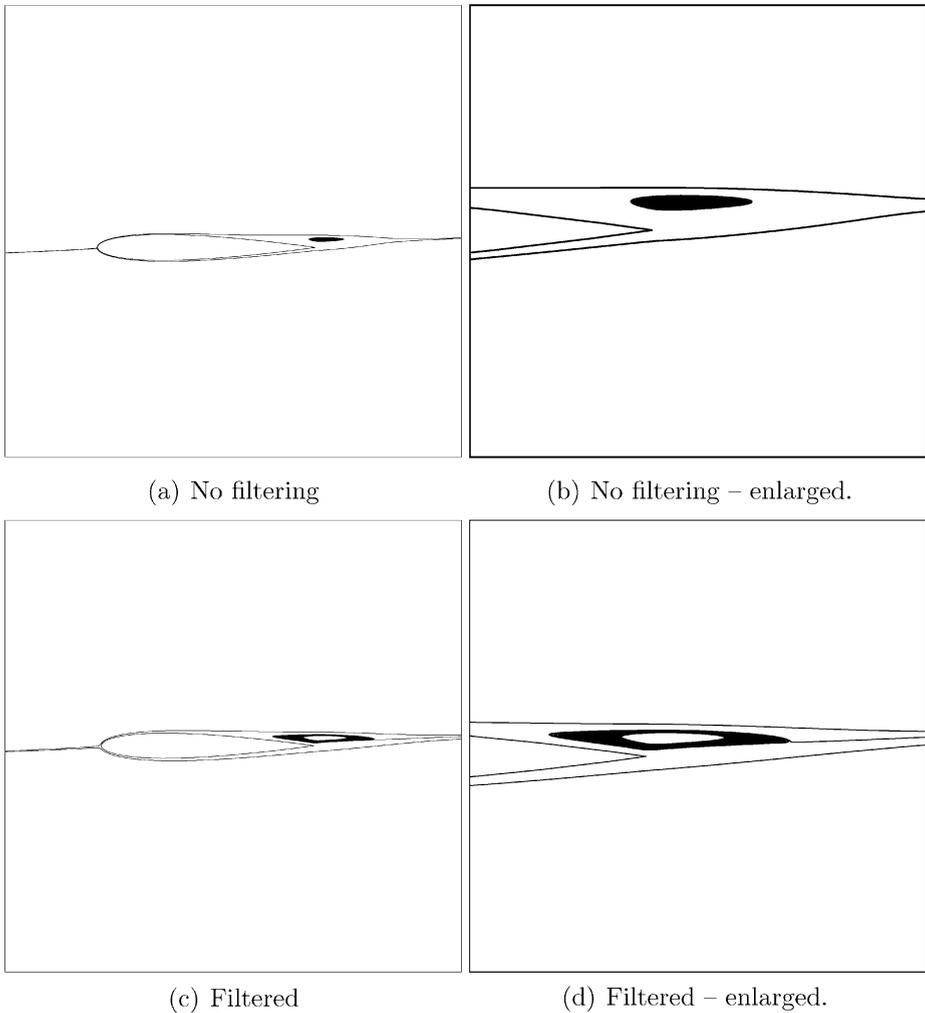


Fig. 14 Streamlines at the separated region. Streamline starting location is the same for both cases

herent in discontinuous Galerkin solutions, the order of convergence of time-integrators can be conserved. With this method, incorrect integration close to critical points can be avoided, and accurate streamlines can be traced with larger time steps. Although the filter is computationally expensive, it is a necessary tool to be employed in the DG post-processing settings, so the time integrator does not *see* the “artificial” discontinuities in the flow solution.

Clearly, a more accurate streamline time integration scheme is desirable; the currently used RK2 solver is explicit and displays only second order convergence. Finding a higher-order multi-stage or multi-step scheme that requires sample points only along the path would lead to a more accurate streamlining method. For streaklining, it has been shown that implicit methods, in particular Backward Differentiation, are most effective [14]. In order to be able to use these algorithms, a continuous solution is not sufficient; the first derivative of the solution also needs to be smooth. As a requirement of the implicit nature of these schemes

and a desire to efficiently converge, a Newton-Raphson iteration is used to find the desired location of the sample points. Without a smooth and accurate velocity gradient tensor, the Newton-Raphson scheme does not display quadratic convergence and may not be stable. By applying a modification to this filtering technique [3], continuity of the first derivative can be achieved and more effective time-integrators can be implemented for both streamlining and streaklining.

Additionally, examining the errors in the interpolation procedure needs to be performed since the convolution procedure requires the solution at quadrature points along the streamline. Currently, the location of these points is calculated using linear interpolation along that segment of the streamline. The solution, however, is a higher-order curve. In order to improve the accuracy of the quadrature point coordinates calculation, a curve fitting of the higher-order curve should be applied to the sample points of the streamline.

However, given the limitations of current visualization techniques for streamline integration of higher-order data, we are able to obtain results that suggest this is a promising filtering technique. We have used existing knowledge of post-processing discontinuous Galerkin methods to increase smoothness of the approximation using a combination of one-sided post-processing and a characteristic length for non-uniform meshes. Since this technique takes into account the information about the approximation, we are able to ensure that it conserves the accuracy while improving smoothness.

Acknowledgements The first, third and fourth authors would like to acknowledge the support of ARO grant number W911NF-05-1-0395. The third author also acknowledges the support of NSF Career Award (Kirby) NSF-CCF0347791. This work was performed while the second author was in Department of Mathematics, Virginia Tech, Blacksburg, VA, USA. The authors would also like to thank Mike Steffen for useful comments and suggests.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Burden, R., Faires, J.: Numerical Analysis. PWS, Boston (1993)
2. Cockburn, B.: Discontinuous Galerkin methods for convection-dominated problems. In: High-Order Methods for Computational Physics. Lecture Notes in Computational Science and Engineering, vol. 9, pp. 69–224. Springer, Berlin (1999)
3. Cockburn, B., Ryan, J.K.: Local derivative post-processing for discontinuous Galerkin methods (2008, in preparation)
4. Cockburn, B., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework. *Math. Comput.* **52**, 411–435 (1989)
5. Cockburn, B., Shu, C.-W.: The Runge-Kutta local projection P^1 -discontinuous-Galerkin finite element method for scalar conservation laws. *Math. Model. Numer. Anal.* **25**, 337–361 (1991)
6. Cockburn, B., Shu, C.-W.: The Runge-Kutta discontinuous Galerkin method for conservation laws. V: Multidimensional systems. *J. Comput. Phys.* **141**, 199–224 (1998)
7. Cockburn, B., Shu, C.-W.: Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comput.* **16**, 173–261 (2001)
8. Cockburn, B., Lin, S.-Y., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems. *J. Comput. Phys.* **84**, 90–113 (1989)
9. Cockburn, B., Hou, S., Shu, C.-W.: The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case. *Math. Comput.* **54**, 545–581 (1990)
10. Cockburn, B., Luskin, M., Shu, C.-W., Süli, E.: Post-processing of Galerkin methods for hyperbolic problems. In: Proceedings of the International Symposium on Discontinuous Galerkin Methods, pp. 291–300. Springer, Berlin (1999)

11. Cockburn, B., Karniadakis, G.E., Shu, C.-W.: *Discontinuous Galerkin Methods: Theory, Computation and Applications*. Springer, Berlin (2000)
12. Cockburn, B., Luskin, M., Shu, C.-W., Suli, E.: Enhanced accuracy by post-processing for finite element methods for hyperbolic equations. *Math. Comput.* **72**, 577–606 (2003)
13. Curtis, S., Kirby, R.M., Ryan, J.K., Shu, C.-W.: Post-processing for the discontinuous Galerkin method over non-uniform meshes. *SIAM J. Sci. Comput.* **30**, 272–289 (2007)
14. Darmofal, D.L., Haines, R.: An analysis of 3-d particle path integration algorithms. *J. Comput. Phys.* **123**, 182–195 (1996)
15. Darmofal, D.L., Haines, R.: *Towards the next generation in computational fluid dynamics* (2005), AIAA, 205-0007
16. Gear, C.W.: Solving ordinary differential equations with discontinuities. *ACM Trans. Math. Softw.* **10**(1), 23–44 (1984)
17. Gottlieb, S., Shu, C.-W.: Total variation diminishing Runge-Kutta schemes. *Math. Comput.* **67**, 73–85 (1998)
18. Gottlieb, S., Shu, C.-W., Tadmor, E.: Strong stability preserving high-order time discretization methods. *SIAM Rev.* **43**, 89–112 (2001)
19. Hou, H.S., Andrews, H.C.: Cubic splines for image interpolation and digital filtering. *IEEE Trans. Acoust. Speech Signal Process.* **26**, 508–517 (1978)
20. Karniadakis, G.E., Sherwin, S.J.: *Spectral/hp Element Methods for CFD*, 2nd edn. Oxford University Press, London (2005)
21. Laidlaw, D.H., Kirby, R.M., Jackson, C.D., Scott Davidson, J., Miller, T.S., da Silva, M., Warren, W.H., Tarr, M.J.: Comparing 2d vector field visualization methods: A user study. *IEEE Trans. Vis. Comput. Graph.* **11**(1), 59–70 (2005)
22. Ma, K.-L., Smith, P.J.: Virtual smoke: An interactive 3d flow visualization technique. In: *Visualization '92*. IEEE Comput. Soc., Los Alamitos (1992)
23. Max, N., Crawfis, R.: Textured splats for 3d scalar and vector field visualization. In: *Visualization '93*, pp. 261–272. IEEE Comput. Soc., Los Alamitos (1993)
24. Mitchell, D.P., Netravali, A.N.: Reconstruction filters in computer-graphics. In: *SIGGRAPH '88: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 221–228. Assoc. Comput. Mech., New York (1988)
25. Moller, T., Machiraju, R., Mueller, K., Yagel, R.: Evaluation and design of filters using a Taylor series expansion. *IEEE Trans. Vis. Comput. Graph.* **3**(2), 184–199 (1997)
26. Ryan, J.K., Shu, C.-W.: On a one-sided post-processing technique for the discontinuous Galerkin methods. *Methods Appl. Anal.* **10**, 295–307 (2003)
27. Ryan, J.K., Shu, C.-W., Atkins, H.L.: Extension of a post-processing technique for the discontinuous Galerkin method for hyperbolic equations with application to an aeroacoustic problem. *SIAM J. Sci. Comput.* **26**, 821–843 (2005)
28. Steffen, M., Curtis, S., Kirby, R.M., Ryan, J.K.: Investigation of smoothness enhancing accuracy-conserving filters for improving streamline integration through discontinuous fields. *IEEE Trans. Vis. Comput. Graph.* **14**, 680–692 (2008)
29. Szabó, B.A., Babuska, I.: *Finite Element Analysis*. Wiley, New York (1991)
30. van Walsum, T., Post, F.J.: Iconic techniques for feature visualization. In: *Visualization '95*, pp. 288–295. IEEE Comput. Soc., Los Alamitos (1995)
31. van Wijk, J.J.: Rendering surface-particles. In: *Visualization '92*, pp. 54–61. IEEE Comput. Soc., Los Alamitos (1992)
32. van Wijk, J.J., de Leeuw, W.C.: Enhanced spot noise for vector field visualization. In: *Visualization '95*, pp. 233–239. IEEE Comput. Soc., Los Alamitos (1995)
33. van Wijk, J.J., de Leeuw, W.C.: A probe for local flow field visualization. In: *Visualization '93*, pp. 39–45. IEEE Comput. Soc., Los Alamitos (1993)
34. Vetterling, W.T., Press, W.H., Teukolsky, S.A., Flannery, B.P.: *Numerical Recipes in C*. Cambridge University Press, Cambridge (1997)
35. Weiskopf, D., Erlebacher, G.: Overview of flow visualization. In: Hansen, C.D., Johnson, C.R. (eds.) *The Visualization Handbook*. Elsevier, Amsterdam (2005)