




Adaptive Characteristic Length for L-SIAC Filtering of FEM Data

Ashok Jallepalli¹  · Robert Haimés² · Robert M. Kirby¹

Received: 22 December 2017 / Revised: 29 September 2018 / Accepted: 31 October 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Treating discontinuities at element boundaries is a significant problem in understanding high-order FEM simulation data since the physics used to model the simulation is often continuous. Recently, the family of SIAC filters, especially the L-SIAC filter, has been gaining popularity for its use in postprocessing. The computational math community, with its focus on improving the theoretical aspects of the SIAC filter, has applied the filter only on simple, fairly uniform unstructured meshes, where the largest element in the mesh is less than or equal to twice the smallest element. In many engineering applications, the unstructured meshes have varying orders of mesh resolution, but there is no literature for adapting the characteristic length of the SIAC filter to address these real-world simulation data. The central contribution of this paper is an algorithm used to calculate the characteristic length dynamically at any point in the mesh. We demonstrate that our approach has a lower error and is computationally faster than using maximum edge length over the mesh.

Keywords cG (continuous Galerkin) · dG (discontinuous Galerkin) · Higher order methods · Higher order data · Smoothness increasing accuracy conserving filter (SIAC filter) · Line SIAC (L-SIAC)

1 Introduction

The high-order FEM techniques used in applications such as Formula-1 race car design [1] or bioengineering [2] are becoming increasingly popular to solve real-world engineering

✉ Ashok Jallepalli
ashokj@sci.utah.edu

Robert Haimés
haimés@mit.edu

Robert M. Kirby
kirby@sci.utah.edu

¹ School of Computing, Scientific Computing and Imaging (SCI) Institute, University of Utah, Salt Lake City, UT, USA

² Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA

problems. The discontinuous Galerkin (dG) data and/or the derivative quantities output by the FEM techniques are discontinuous at the element boundaries, but the underlying physics often imply the data are continuous. These discontinuities are in some cases an artifact of the simulation technique and hence may not hold any significance for anyone other than numerical simulation experts. Moreover, most of the data extraction techniques and visualization tools need the simulation data to be continuous to be effective. Hence, treating these discontinuities at the element boundary while conserving accuracy is important to make decisive conclusions from the simulation. Recently, a family of SIAC filters, especially the Line SIAC filter (L-SIAC [3]), has been introduced as a postprocessor to compute and visualize continuous derived quantities from simulation data [4].

In the computational math community, the main focus has been to improve the theoretical aspects of the SIAC filter, and therefore it has been applied on simple structured and unstructured meshes with small changes in resolution of the mesh, i.e., the ratio of the maximum to minimum element is usually less than three. Almost all the meshes used in real-world engineering simulations are unstructured, with element sizes varying possibly several orders of magnitude. Hence, in this paper, we showcase the results for unstructured meshes due to their interest in real-world applications. The different resolutions in the unstructured mesh help capture essential features of the simulation while keeping the computational cost of the simulation at a minimum. To postprocess these unstructured meshes, the current recommendation is to use maximum edge length in the mesh as the characteristic length of the SIAC filter. This technique is valid for regular unstructured meshes [5], but maximum edge length as the characteristic length is not sufficient on meshes with resolutions varying several orders of magnitude.

The difficulty in using the L-SIAC filter for real-world applications is selecting an adequate characteristic length and filter orientation to reduce error on all parts of the unstructured mesh, with minimum computational overhead. In this paper, we propose an algorithm to compute the characteristic length that adaptively scales the L-SIAC filter across different resolutions of the mesh. The proposed adaptive characteristic length produces a lower error on the nonuniform unstructured mesh and generates smaller if not the same error on uniform meshes. We also show that our algorithm is computationally efficient in high-resolution regions of the mesh.

The paper is organized as follows: In Sect. 1.1, we present previously suggested alternatives for the characteristic length with their drawbacks. In Sect. 2, we define the L-SIAC filter and its implementation. In Sect. 3, we present our proposed algorithm to calculate the characteristic length dynamically. We also show that using the recommended adaptive characteristic length is computationally efficient on high-resolution regions of the mesh. In Sect. 4, we compare our adaptive characteristic length against the maximum edge length on unstructured and structured meshes present in the literature. We also demonstrate the effectiveness of using adaptive characteristic length on simulations data output by Nektar++ [6]. We then conclude in Sect. 5 by discussing the pros and cons of adaptive characteristic length as revealed through this implementation and comparison study.

1.1 Previous Work

In the mathematical literature, Cockburn et al. [7,8] introduced SIAC filters as a postprocessing technique for increasing the accuracy of the discontinuous Galerkin (dG) solution by exploiting superconvergence rates in their negative norm; these filters, as a secondary consequence of the postprocessing, increase the continuity in the solution. In [9–11], variations of the SIAC filter, called one-sided SIAC filters, are introduced to deal with boundaries and

shocks. Mirzargar et al. in [12] introduced hexagonal SIAC filters, which are better suited for hexagonal meshes. Li et al. in [13] discussed effective ways to calculate the derivative quantities using the SIAC and one-sided SIAC filter.

To extend the work done in the mathematical literature to the field of engineering, Mirzaee et al. [5,14] have shown effective ways to intersect the SIAC kernel and mesh geometry for structured and unstructured meshes, but it is still difficult to use the SIAC filter for unstructured meshes beyond 2D due to geometric and computational complexity. In [15], Steffen et al. applied the SIAC filter to postprocess fields prior to streamline integration, and in [16], Walfisch et al. proposed a one-sided SIAC filtering approach for visualizing continuous streamlines both accurately and efficiently. Later in [3], Docampo-Sanchez et al. proved that the L-SIAC filter could be used to achieve higher order convergence and continuity for higher dimensional solutions. Jallepalli et al. [4] exploited the geometric simplicity of the L-SIAC filter to calculate derived quantities for realistically sized 3D simulation data and acknowledged that efficient application of the L-SIAC filter depends on selecting an effective characteristic length and filter orientation.

Previous studies [5,14] have applied the SIAC filter to structured and unstructured meshes, but there is no literature on applying the L-SIAC filter for these meshes. Hence, we will review the literature on choosing the characteristic length for the SIAC filter to gain insight into its application for the L-SIAC filter.

Since it is difficult to prove the correct characteristic length for a nonuniform structured and unstructured mesh, three approaches have been used for applying the SIAC filter:

1. Mirzaee et al. [5] used the maximum edge length of the mesh as the characteristic length. In [17], King et al. showed that using a characteristic length equal to two or three times the edge length produces smoother results but at the cost of increased error. Hence, although the computational mathematics community often uses a global characteristic length for regular and smoothly varying meshes, it is our experience that using a single global characteristic length, however it is chosen, frequently fails for meshes with element ratios of 1:10 or higher.
2. Curtis et al. [18] used a piecewise constant characteristic length, for which the characteristic length is constant inside each element and discontinuous at the element boundaries for a nonuniform mesh. For meshes with a large difference in maximum and minimum element sizes, using a piecewise constant characteristic length for the L-SIAC filter could produce better errors compared to the maximum edge length, but the error appears discontinuous at the element boundaries due to discontinuity in the characteristic length.
3. In [19], Li studied the connection between the error of the filtered solution and the filter length used to postprocess uniform and nonuniform meshes. He sought to extend the existing theoretical SIAC filtering estimates to nonuniform meshes and boundary filtering by studying their theoretical building blocks such as the divided differences and their relation to the approximation space. His experimental results (in 2D and 3D) on general uniform and nonuniform meshes are quite promising and consistent with his theoretical results, but it is left as an open question how to extend his theoretical results beyond 1D.

To address the shortcomings of previous approaches, we propose an algorithm that adaptively scales the characteristic length depending on location in the mesh, while maintaining continuity at element boundaries. Thus, the proposed algorithm has a similar or lower error compared to the L-SIAC filter with maximum edge length, maintains the continuity of error at element boundaries, and has a lower computational cost than the L-SIAC filter with maximum edge length.

2 L-SIAC

Postprocessing using an L-SIAC filter can be defined as the convolution of the given field with a 1D SIAC filter, which is rotated along a direction ($\hat{\mathbf{d}}$) and scaled with a characteristic length (H). Mathematically, the L-SIAC filter can be defined as

$$u^*(\mathbf{x}) = \int_{-\infty}^{\infty} K_H(t) * u_h(\mathbf{x} - \Gamma(t)) dt \quad (1)$$

$$\Gamma(t) = t\hat{\mathbf{d}}, \hat{\mathbf{d}} \text{ is a unit vector} \quad (2)$$

where u^* is the postprocessed solution, u_h is the FEM solution of polynomial degree k , H is the characteristic length, $\hat{\mathbf{d}}$ is the direction of the line along which the L-SIAC filter is applied, and $K_H(t)$ is the SIAC kernel defined as a linear combination of B-splines (Ψ) along the line Γ by the parameter t :

$$K_H^{2k+1,k+1}(t) = \sum_{\gamma=-k}^k c_\gamma \Psi_H^{k+1}(t - \gamma) = \frac{1}{H} \sum_{\gamma=-k}^k c_\gamma \Psi^{k+1}\left(\frac{t}{H} - \gamma\right),$$

$$K_H^{2k+1,k+1}(t) = \frac{1}{H} K^{2k+1,k+1}\left(\frac{t}{H}\right). \quad (3)$$

The B-splines used in the postprocessor are well studied and can be computed using the recurrence relation

$$\Psi^1 = X_{[-1/2, 1/2]},$$

$$\Psi^{k+1}(\eta) = \frac{1}{k} \left(\left(\frac{k+1}{2} + \eta \right) \Psi^k \left(\eta + \frac{1}{2} \right) + \left(\frac{k+1}{2} - \eta \right) \Psi^k \left(\eta - \frac{1}{2} \right) \right), \quad k \geq 1 \quad (4)$$

$$K * x^r = x^r, \quad r \in (0, 1, \dots, 2k). \quad (5)$$

The coefficients of the kernel, c_λ , can be found by using the property shown in Eq. 5, that the kernel must not destroy the accuracy of the approximation (i.e., the polynomial reproduction property). More specifically, they reproduce polynomials of degree $2k$ by convolution. When using a symmetric B-spline kernel, the coefficients can be solved once and stored for reuse. All discussions of errors described in this paper assume that a symmetric kernel fits at the location where the L-SIAC filter needs to be applied to avoid the additional error due to the one-sided L-SIAC filter at the boundaries; all computations are carried out in the interior of the domain.

2.1 Implementation of the L-SIAC Filter

To apply the L-SIAC filter at any point in the field, we carry out the convolution along the parametric line $\Gamma(t)$ centered at the given point. Numerically, the FEM data and the SIAC kernel are piecewise polynomials at element boundaries and B-spline knots, respectively. First, the parametric line needs to be divided into line segments such that each line segment can be represented as a polynomial function. Next, we integrate over each line segment using the appropriate Gauss-quadrature. The Gauss-quadrature required is the average degree of the B-splines (k) and the FEM field (k) plus one. The division of the line segments is determined

by the knot locations of the kernel and the intersection of the element boundaries of the mesh with $\Gamma(t)$. Thus, the total number of line segments depends on the L-SIAC filter direction, characteristic length and location at which the L-SIAC filter is applied on the mesh. Hence, the convolution can be rewritten as the summation

$$u^*(\mathbf{x}) = \sum_s^{Sup(\Gamma(t))} \sum_q^{q \in Q} w_q K_H(t_{q,s}) u_h(\mathbf{x} - \Gamma(t_{q,s})) \tag{6}$$

or

$$u^*(\mathbf{x}) = \sum_s^{Sup(\Gamma(t))} \sum_q^{q \in Q} w_q K_H(t_{q,s}) u(x - t_{q,s}d_x, y - t_{q,s}d_y)$$

$$t_{q,s} = a + (b - a)q, s = [\Gamma(a), \Gamma(b)], q \in Q$$

where s are the line segments created by splitting the line Γ at the SIAC filter knot positions and the intersection of the line (Γ) with the mesh element boundaries; Q denotes the set of quadrature points; w_q are the weights for the quadrature point; and $t_{q,s}$ is a quadrature point on the line segment s .

3 Proposed Adaptive Characteristic Length

The adaptive characteristic length ($\mathcal{H}(p)$) at a point p is evaluated in the following two steps:

Preprocessing Step This step needs to be evaluated before applying the L-SIAC filter. The step evaluates the effective characteristic length ($\mathcal{H}(v_i)$) at all vertices (v_i) in the mesh. It is accomplished by first calculating the longest edge (L_{e_j}) for each element e_j and then calculating the area-weighted longest edge of all the triangles connected at each vertex. The area-weighted largest edge is the characteristic length at the vertex and can be described mathematically as follows:

$$\mathcal{H}(v_i) = \frac{\sum_j L_{e_j} \cdot Area(e_j)}{\sum_j Area(e_j)} \tag{7}$$

where v_i is the i th vertex in the given mesh (M), e_j is j th the element connected to vertex v_i and L_{e_j} is largest edge length of element e_j . Observe that the characteristic length definition above is consistent with the mesh function proposed in [20,21] for facilitating convergence proofs on unstructured meshes, further corroborating our choice of this functional form.

Postprocessing Step This step is evaluated each time the L-SIAC filter is computed at a new point (p) in the mesh. First, we calculate the barycentric coordinates of the point (p) with respect to the element (e_l), which contains the point. Next, we interpolate the characteristic length from the vertices ($\mathcal{H}(v)$) of the element (e_l) using the barycentric coordinates ($\lambda_0, \dots, \lambda_k, \dots, \lambda_N$) to the given point (p). This step can be described mathematically as follows:

$$\mathcal{H}(p) = \sum_{l=0}^N \lambda_l \cdot \mathcal{H}_{v_l} \tag{8}$$

where N is the number of vertices in the element (e_l). From this point on, we use \mathcal{H} for adaptive characteristic length and H for maximum edge length as the characteristic length.

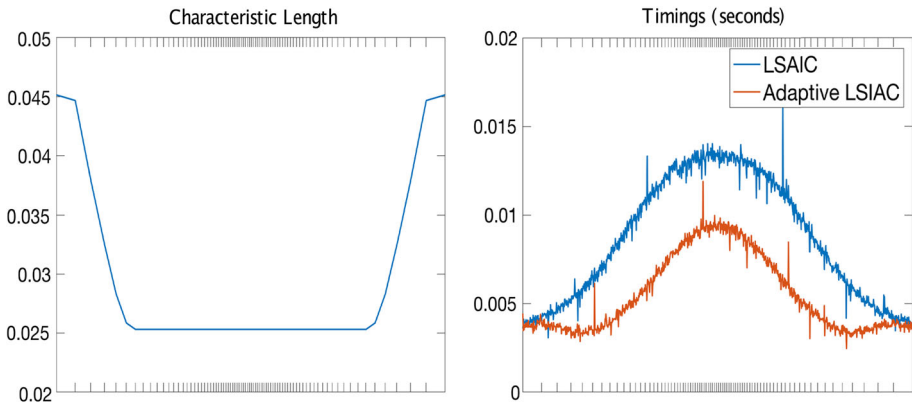


Fig. 1 Left: visualization of adaptive scaling calculated at 1000 points on a 1D mesh. Right: time in seconds to apply the L-SIAC filter using adaptive scaling at 1000 points. The tick marks on the x-axis indicate the element boundaries

3.1 Cost of Applying the Adaptive L-SIAC Filter

As demonstrated in Eq. 6, the cost of applying the L-SIAC filter at a single point depends on the number line segments in the summation, the number of quadrature points in each line segment, and the cost of evaluating the FEM field $u_h(x)$ and kernel $K(t)$ at a single location. Thus, the cost of calculating the L-SIAC at a single point is given by

$$O \left(\underbrace{N}_{\# \text{ line segments}} \cdot \overbrace{(k+1)}^{\# \text{ quadrature points}} \cdot \left(\underbrace{B}_{\text{cost of } u_h(x) \text{ eval}} + \underbrace{D}_{\text{cost of } K(t) \text{ eval}} \right) \right). \quad (9)$$

The number of line segments required to apply the L-SIAC filter at a single location on a mesh depends on the L-SIAC kernel knots and the intersection of the kernel with the mesh. The length of the L-SIAC filter of degree k and characteristic length H is $(3k + 1) \cdot H$. The ratio $\frac{L}{e_l}$ gives the number of intersections of a line segment of length L on a mesh with element lengths e_l . Hence, the total number of line segments required for applying L-SIAC filter at given point is given by

$$\begin{aligned} \# \text{ line segments} &= (3k + 1) + \frac{L}{e_l} \\ &= (3k + 1) \left(1 + \frac{H}{e_l} \right) \end{aligned} \quad (10)$$

where $3k + 1$ is the number of knots in the L-SIAC kernel and $\frac{L}{e_l}$ is the number of line segments due to the intersection of the elements with the L-SIAC kernel.

Therefore, if we apply a constant scaling proportional to the maximum edge length, i.e., $H = r e_l$, where $r \gg 1$, then the computational cost grows as the mesh resolution grows. On the other hand, a local scaling, which is not proportional to the mesh, remains at a lower cost. In Fig. 1, we evaluated the L-SIAC filter on a 1D mesh with a smooth variation in mesh resolution to demonstrate that the local characteristic length has a lower computational cost compared to the constant characteristic length.

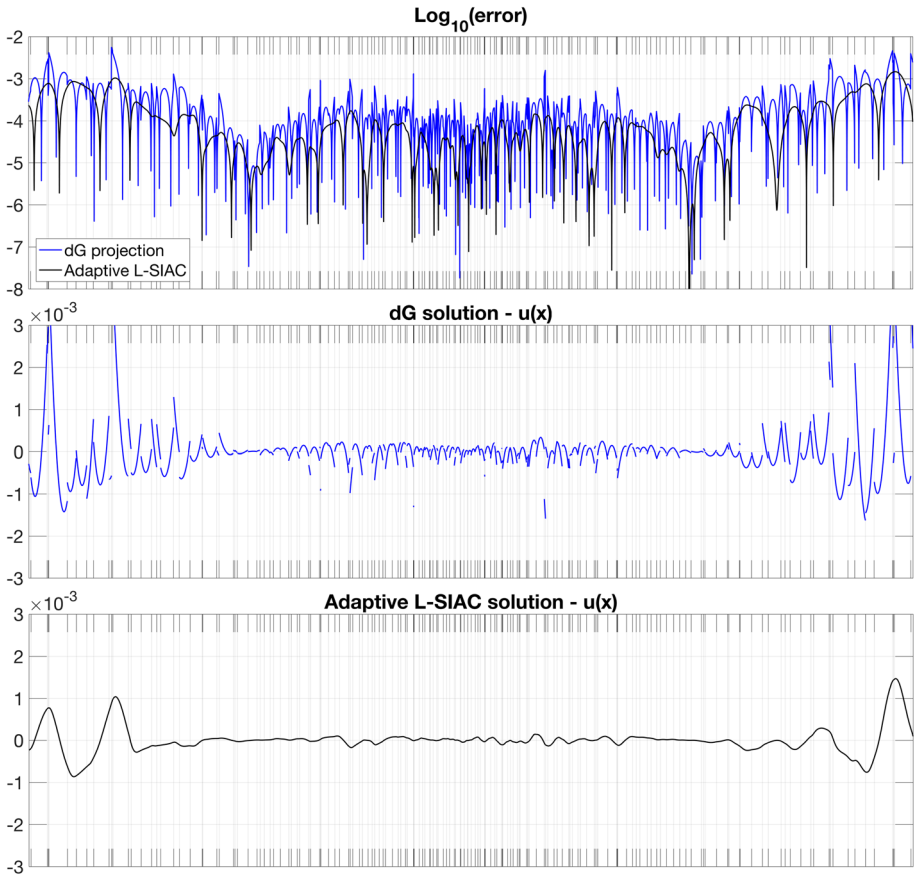


Fig. 2 Pointwise error contours for the dG and the adaptive L-SIAC filter across a horizontal line ($y = 0.5$) at the center of mesh ($SvMeshR10, \mathcal{P}^1$) using the function described in Sect. 4.1. The tick marks along the x-axis represent the element boundaries. Top: pointwise error contours using a logarithmic scale for the dG and the adaptive L-SIAC filter; middle: pointwise contour of the dG solution minus the true solution to highlight the discontinuity at element interfaces; lower: pointwise contour of the adaptive L-SIAC solution minus the true solution to emphasize the continuity at element interfaces

3.2 Continuity of the Adaptive L-SIAC Filter

One of the primary objectives of the L-SIAC filter is to achieve continuity at element interfaces. In the case of the adaptive L-SIAC filter, since the characteristic length maintains C^0 continuity across the element interfaces, the post-processed solution and the resulting error are continuous across element interfaces.

To further illustrate, let us consider the mesh ($SvMeshR10$) and the projected solution described in Sect. 4.1 and apply the adaptive L-SIAC filter along the dotted yellow line as shown in Fig. 4. The error is plotted using the logarithmic scale in Fig. 2. Although the log plot provides insight into the accuracy, the log plots are not helpful to visualize the continuity in the solution profile. To show the continuity in the solution profile at the element interfaces, the difference between the approximate and true solution for each element are

Table 1 Meshes used in the results section along with their number of elements and time to compute the preprocessing step (Sect. 3)

Mesh names	Elements	Time (s)
Unstructured smooth variational mesh, Sect. 4.1		
SvMeshR2	2127	0.02
SvMeshR10	6743	0.07
SvMeshR100	12,542	0.1
Unstructured variable-sized mesh, Sect. 4.2		
VsMeshR005	2321	0.05
VsMeshR025	10,138	0.23
VsMeshR125	34,720	0.84
Unstructured uniform mesh, Sect. 4.3		
UMeshR005	1107	0.03
UMeshR025	4883	0.16
UMeshR125	16,769	0.66
Unstructured diagonal variable-sized mesh, Sect. 4.4		
DvsMeshR5	7378	0.07
DvsMeshR10	26,970	0.27
DvsMeshR20	93,750	1.18
Flow past cylinder example, Sect. 4.5		
2DcylinderMesh	830	0.01
Counter rotating vortices example, Sect. 4.6		
3DMesh	223,837	5.15

plotted individually to generate the middle and lower plots in Fig. 2. We observe the solution is discontinuous for dG and appears continuous in the case of the adaptive L-SIAC filter.

4 Results

In this section, we present the postprocessing results for a comparison of the efficiency of the L-SIAC filter used with adaptive characteristic length versus constant maximum edge characteristic length. From here on, we use “L-SIAC” to refer to applying the L-SIAC filter with constant characteristic length equal to maximum edge length in the mesh, and “adaptive L-SIAC” to applying the L-SIAC filter with proposed adaptive characteristic length.

In Table 1, we list all the meshes used in this section along with the time to compute the preprocessing step described in Sect. 3. The results in this paper are generated on a machine with a 2.4 GHz (Intel CPU E-7-4870) processor. In general, we observed that the preprocessing time is negligible compared to the time to apply the L-SIAC filter; hence, we have not included the preprocessing time while reporting the time by the adaptive L-SIAC filter. For the examples below, unless specified, we have used symmetric L-SIAC filters, by extending the meshes beyond $[0, 1] \times [0, 1]$ whenever possible to avoid error due to one-sided L-SIAC filters at the boundaries.

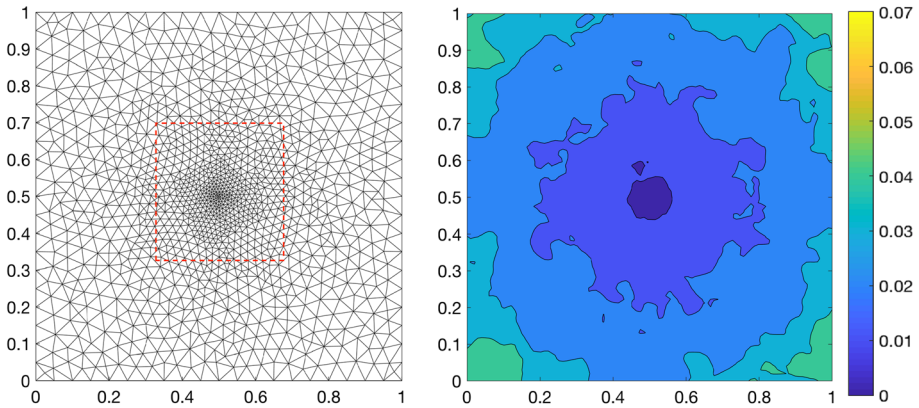


Fig. 3 Left: example of an unstructured triangular mesh with smooth variation in mesh resolution. Right: adaptive characteristic length for the mesh on the left

4.1 Delaunay triangulation with a smooth variation in mesh resolution

In many engineering applications, most meshes have continuous variation in mesh resolution, with high resolution in the area of interest to the simulation. Maintaining the accuracy of the simulation data after postprocessing is crucial so that the interpretation of the data is preserved. Hence, in this example, we consider meshes as shown in Fig. 3, where the mesh has high resolution at the center and gradually coarsens toward its boundaries. We used the Gmsh tool [22] to create three meshes with ratios of mesh resolution at the boundary and the center of the mesh equal to 2, 10, and 100: we refer to them as *SvMeshR2*, *SvMeshR10*, and *SvMeshR100*, respectively.

For the meshes with smooth variation in mesh resolution, we project the function $\cos(2\pi(x + y))$ onto the mesh with domain $[0, 1] \times [0, 1]$ using the dG methodology (i.e., using the L^2 projection), and analyze the postprocessed solution using L-SIAC and adaptive L-SIAC filters, with the direction parameter set along the x-axis for both filters. In Fig. 4, we visualize the postprocessing error due to the filters using log plots for *SvMeshR10* with different polynomial orders. The line plots in this figure are created by sampling the error plots along the dotted line, and ticks at the top and bottom of the plot indicate the intersections of element boundaries with the dotted line. From these error plots, especially the line plots, it is clear that the L-SIAC filter has attained a high and equal error in all parts of the mesh, which means it has lower accuracy at high resolutions of the mesh. In contrast, the adaptive L-SIAC filter has attained a lower error at the center of the mesh, i.e., in the finer resolution region of the mesh.

To further understand the behavior of the postprocessing filters in the finer regions of the mesh, in Table 2 we show L^2 and L^∞ errors for the projection, the L-SIAC filter, and the adaptive L-SIAC filter, a square region indicated by the red box in Fig. 3. The adaptive L-SIAC filter produces better accuracy than the dG projection (except for \mathcal{P}^3 , L^∞ , *SvMeshR100* Mesh). In terms of L^2 error, except in the case of \mathcal{P}^1 *SvMeshR2* mesh, the adaptive L-SIAC filter has better accuracy compared to the L-SIAC filter. Regarding L^∞ error, the accuracy of the adaptive L-SIAC filter is similar accuracy to that of the L-SIAC filter. In Table 3, we show that the adaptive L-SIAC filter is computationally more efficient than the L-SIAC filter.

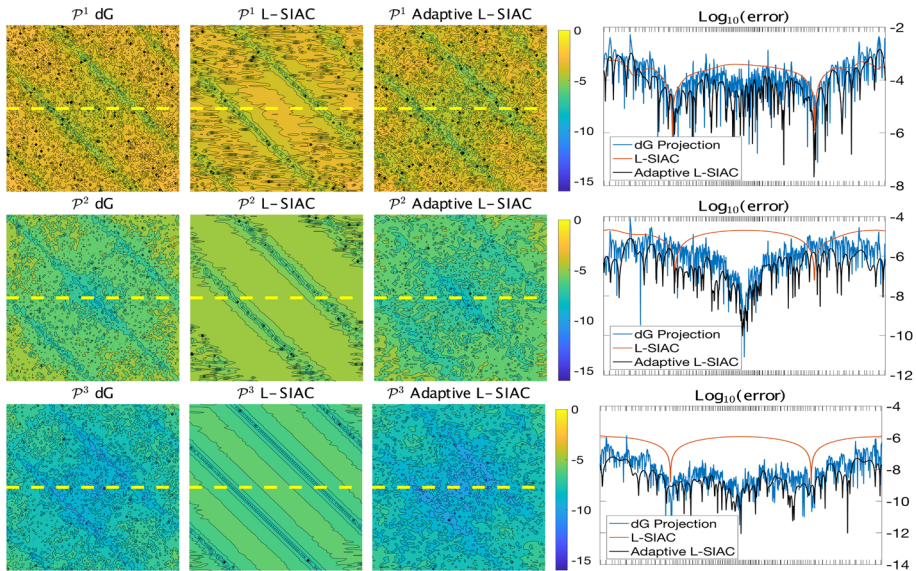


Fig. 4 Pointwise error ($\log_{10}(Error)$) contour plots over a smoothly varying resolution with 6743 elements (SvMeshR10). 1st column: dG error; 2nd column: postprocessed with the L-SIAC filter; 3rd column: postprocessed with the adaptive L-SIAC filter; 4th column: errors along the line $y = 0.5$. Top row: \mathcal{P}^1 polynomials; middle row: \mathcal{P}^2 polynomials; bottom row: \mathcal{P}^3 polynomials

Table 2 L^2 and L^∞ errors over the high-resolution region of meshes described in Sect. 4.1

Mesh	L^2			L^∞		
	Proj	L-SIAC	Adaptive L-SIAC	Proj	L-SIAC	Adaptive L-SIAC
\mathcal{P}^1						
<i>SvMeshR2</i>	6.1e-04	2.1e-04	2.5e-04	1.5e-02	2.6e-03	5.0e-03
<i>SvMeshR10</i>	1.4e-04	1.2e-04	5.6e-05	5.0e-03	7.9e-04	1.7e-03
<i>SvMeshR100</i>	7.8e-05	1.2e-04	3.2e-05	2.6e-03	6.3e-04	1.4e-03
\mathcal{P}^2						
<i>SvMeshR2</i>	1.4e-05	7.2e-06	3.9e-06	6.1e-04	5.7e-05	7.3e-05
<i>SvMeshR10</i>	1.3e-06	6.2e-06	4.0e-07	6.6e-05	2.2e-05	1.2e-05
<i>SvMeshR100</i>	6.3e-07	6.3e-06	1.8e-07	4.9e-05	2.2e-05	7.0e-06
\mathcal{P}^3						
<i>SvMeshR2</i>	2.0e-07	3.9e-07	5.6e-08	1.9e-05	2.2e-06	1.9e-06
<i>SvMeshR10</i>	1.4e-08	3.6e-07	3.5e-09	2.2e-06	1.3e-06	1.6e-07
<i>SvMeshR100</i>	4.9e-09	3.7e-07	1.3e-09	6.7e-07	1.2e-06	1.6e-06

4.2 Delaunay Triangulation with Variable-Sized Mesh

In this example, we choose nonuniform unstructured meshes as shown in Fig. 5 with the element size at the center of the mesh equal to 0.5 times that at the boundaries. Mirzaee et al. [5] used the Gmsh tool [22] to show the performance of the SIAC filter on unstructured meshes. We created three meshes with the largest element sizes equal to 0.05, 0.025, and

Table 3 Cost (seconds) of evaluating L^2 and L^∞ errors over the high-resolution region of meshes described in Sect. 4.1

Mesh	L-SIAC	Adaptive L-SIAC	Speed-up
\mathcal{P}^1			
<i>SvMeshR2</i>	50.25	35.89	1.4001
<i>SvMeshR10</i>	699.74	345.9	2.023
<i>SvMeshR100</i>	3192.27	1594.97	2.0015
\mathcal{P}^2			
<i>SvMeshR2</i>	210.48	134.54	1.5644
<i>SvMeshR10</i>	2743.95	1365.78	2.0091
<i>SvMeshR100</i>	11,868.6	4641.63	2.557
\mathcal{P}^3			
<i>SvMeshR2</i>	511.21	325.9	1.5686
<i>SvMeshR10</i>	6131.68	2706.05	2.2659
<i>SvMeshR100</i>	25,099.8	9074.14	2.7661

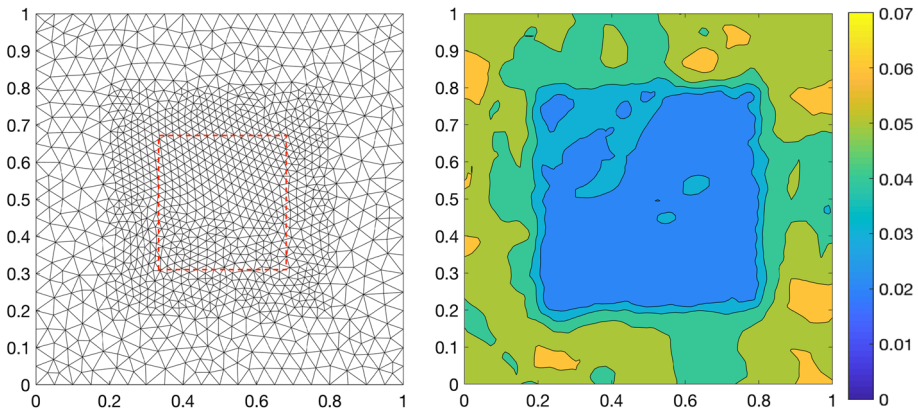


Fig. 5 Left: example of an unstructured, variable-sized, triangular mesh described in Sect. 4.2. Right: adaptive characteristic length for the mesh on the left

0.0125; we refer to them as *VsMeshR005*, *VsMeshR025*, and *VsMeshR125*, respectively. We then projected the function $\cos(2\pi(x + y))$ over the domain $[0, 1] \times [0, 1]$ for each mesh using the dG methodology. We postprocess the projected function using L-SIAC filters with the direction parameter along the x-axis.

In Fig. 6, we show the postprocessing results of L-SIAC and adaptive L-SIAC filters for the mesh with 2321 elements (*VsMeshR005*) and an element size of 0.05 units at the boundaries for different polynomial orders. The line plots in this figure are created by sampling the error plots along the dotted line, and ticks at the top and bottom of the plot indicate the intersections of element boundaries with the dotted line. In the line plots, we can see that L-SIAC and adaptive L-SIAC filters have similar error in coarse regions of the mesh, but the adaptive L-SIAC filter has a lower error in the high-resolution area.

To further compare the efficiency of adaptive L-SIAC and L-SIAC filters, we have selected a rectangular region, shown by the red box in Fig. 5, to calculate the L^2 and L^∞ errors in Table 4. The adaptive L-SIAC filter and the L-SIAC filter have slightly better accuracy than the dG approximation, the accuracy of the adaptive L-SIAC filter and the L-SIAC filter is

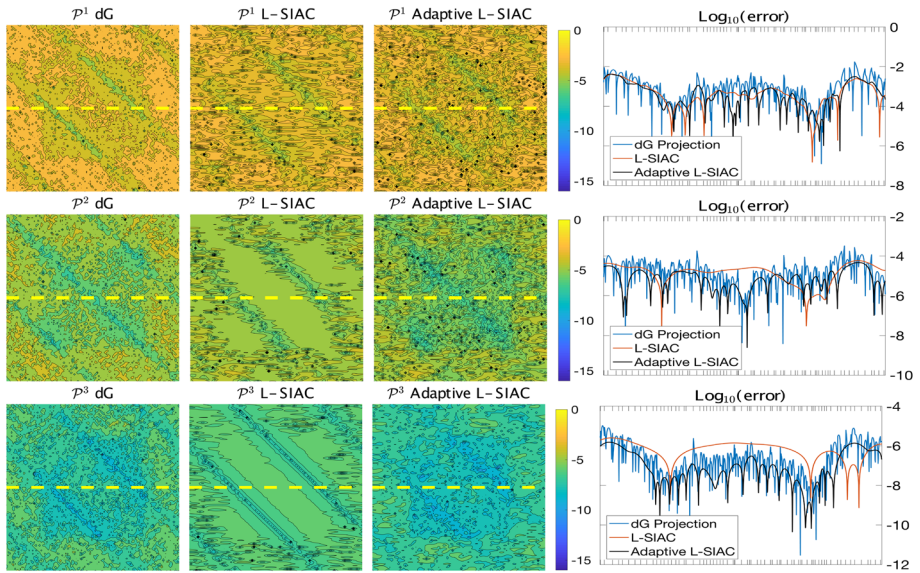


Fig. 6 Pointwise error ($\log_{10}(\text{Error})$) contour plots over a variable-sized mesh shown in Fig. 5 with 2321 elements (VsMeshR005). 1st column: dG error; 2nd column: postprocessed with the L-SIAC filter; 3rd column: postprocessed with the adaptive L-SIAC filter; 4th column: errors along the line $y = 0.5$. Top row: \mathcal{P}^1 polynomials; middle row: \mathcal{P}^2 polynomials; bottom row: \mathcal{P}^3 polynomials

Table 4 L^2 and L^∞ errors over the high-resolution region of meshes described in Sect. 4.2

Mesh	L^2			L^∞		
	Proj	L-SIAC	Adaptive L-SIAC	Proj	L-SIAC	Adaptive L-SIAC
\mathcal{P}^1						
<i>VsMeshR005</i>	4.7e-04	1.8e-04	2.1e-04	8.8e-03	2.5e-03	5.0e-03
<i>VsMeshR025</i>	6.7e-05	1.2e-05	2.1e-05	1.2e-03	2.8e-04	8.0e-04
<i>VsMeshR125</i>	2.7e-05	7.5e-06	1.1e-05	5.3e-04	1.7e-04	2.8e-04
\mathcal{P}^2						
<i>VsMeshR005</i>	7.1e-06	7.2e-06	2.2e-06	2.7e-04	3.3e-05	4.4e-05
<i>VsMeshR025</i>	4.5e-07	1.3e-07	1.3e-07	2.6e-05	2.9e-06	5.7e-06
<i>VsMeshR125</i>	1.2e-07	2.3e-08	3.7e-08	5.1e-06	2.0e-05	3.6e-05
\mathcal{P}^3						
<i>VsMeshR005</i>	1.1e-07	4.2e-07	2.9e-08	7.3e-06	1.9e-06	1.1e-06
<i>VsMeshR025</i>	2.2e-09	1.7e-09	5.4e-10	1.7e-07	1.5e-08	2.8e-08

similar for polynomial degrees 1 and 2, and the adaptive L-SIAC filter has better accuracy for polynomial degree 3. In Table 5, we observe that the adaptive L-SIAC is computationally more efficient than the L-SIAC filter.

Table 5 Cost (seconds) of evaluating L^2 and L^∞ errors over the high-resolution region of meshes described in Sect. 4.2

Mesh	L-SIAC	Adaptive L-SIAC	Speed-up
\mathcal{P}^1			
<i>VsMeshR005</i>	81.28	47.4	1.7148
<i>VsMeshR025</i>	950.92	707.72	1.3436
<i>VsMeshR125</i>	9269.78	8904.8	1.041
\mathcal{P}^2			
<i>VsMeshR005</i>	299.66	179.78	1.6668
<i>VsMeshR025</i>	4252.49	3209.35	1.325
<i>VsMeshR125</i>	41,464.9	36,708.8	1.1296
\mathcal{P}^3			
<i>VsMeshR005</i>	757.57	421.02	1.7994
<i>VsMeshR025</i>	8438.2	5750.8	1.4673

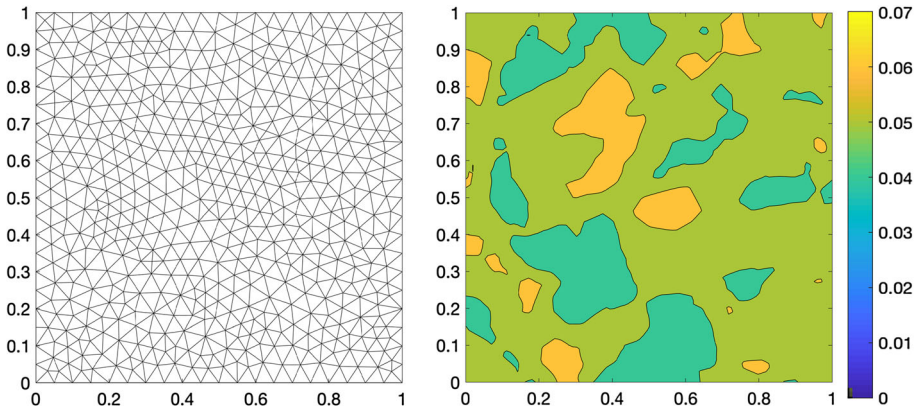


Fig. 7 Left: example of unstructured triangular mesh using a simple Delaunay triangulation described in Sect. 4.3. Right: adaptive characteristic length for the mesh on the left

4.3 Simple Delaunay Triangulation

In this example, we compare the performance of L-SIAC filters on simple unstructured triangular meshes. We show a sample mesh in Fig. 7 with the Gmsh tool [22] using a Delaunay triangulation. We created three meshes with the largest element sizes equal to 0.05, 0.025, and 0.0125; we refer to them as *UMeshR005*, *UMeshR025*, and *UMeshR125*, respectively. We then projected the function $\cos(2\pi(x + y))$ over the domain $[0, 1] \times [0, 1]$ for each mesh using the dG methodology, and analyzed the postprocessing results using L-SIAC filters with the direction parameter set along the x-axis.

In Fig. 8, we show the contour plot for pointwise errors on the mesh with 1107 elements (*UMeshR005*) using L-SIAC filters for different polynomial orders. We observe that L-SIAC and adaptive L-SIAC filters have similar errors, but the L-SIAC filter displays slightly smoother results than the adaptive L-SIAC filter with the same overall accuracy. In general, when there is no significant variation in resolution of the mesh, the L-SIAC filter produces slightly smoother results than the adaptive L-SIAC filter. This variation in smoothness is due to differences in the adaptive characteristic length as a consequence of the element sizes

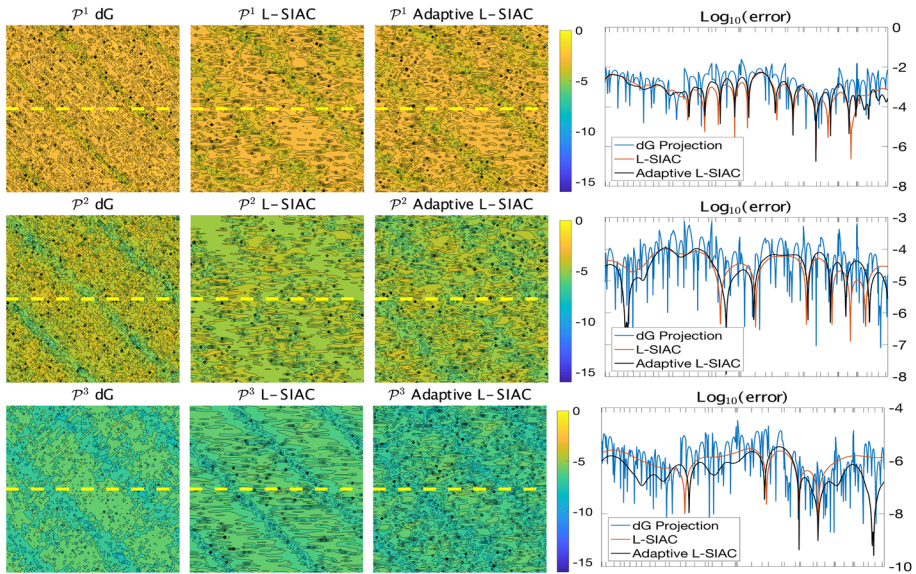


Fig. 8 Pointwise error ($\log_{10}(\text{Error})$) contour plots over the unstructured Delaunay triangular mesh shown in Fig. 7 with 1107 elements (*UMeshR005*). 1st column: dG error; 2nd column: postprocessed with the L-SIAC filter; 3rd column: postprocessed with the adaptive L-SIAC filter; 4th column: errors along the line $y = 0.5$. Top row: \mathcal{P}^1 polynomials; middle row: \mathcal{P}^2 polynomials; bottom row: \mathcal{P}^3 polynomials

Table 6 L^2 and L^∞ errors for the simple Delaunay triangular meshes described in Sect. 4.3

Mesh	L^2			L^∞		
	Proj	L-SIAC	Adaptive L-SIAC	Proj	L-SIAC	Adaptive L-SIAC
\mathcal{P}^1						
<i>UMeshR005</i>	4.3e-03	1.6e-03	1.8e-03	3.2e-02	1.6e-02	1.8e-02
<i>UMeshR025</i>	9.1e-04	3.6e-04	4.2e-04	7.9e-03	4.8e-03	5.5e-03
<i>UMeshR125</i>	2.7e-04	9.7e-05	1.1e-04	2.4e-03	1.3e-03	1.4e-03
\mathcal{P}^2						
<i>UMeshR005</i>	1.4e-04	4.1e-05	4.2e-05	1.9e-03	3.3e-04	3.6e-04
<i>UMeshR025</i>	1.5e-05	3.9e-06	4.4e-06	3.8e-04	5.8e-05	6.3e-05
<i>UMeshR125</i>	2.3e-06	6.1e-07	6.7e-07	4.3e-05	9.2e-06	9.7e-06
\mathcal{P}^3						
<i>UMeshR005</i>	3.9e-06	1.3e-06	9.5e-07	1.0e-04	1.6e-05	1.8e-05
<i>UMeshR025</i>	2.0e-07	5.0e-08	5.5e-08	7.5e-06	1.2e-06	1.3e-06
<i>UMeshR125</i>	1.6e-08	3.8e-09	4.3e-09	5.8e-07	6.8e-08	7.4e-08

in the unstructured mesh. In Tables 6 and 7, the overall accuracy for L-SIAC and adaptive L-SIAC filters is very close, and in most cases, the adaptive L-SIAC takes slightly less time to compute.

Table 7 Cost (seconds) of evaluating L^2 and L^∞ errors for the simple Delaunay triangular meshes described in Sect. 4.3

Mesh	L-SIAC	Adaptive L-SIAC	Speed-up
\mathcal{P}^1			
<i>UMeshR005</i>	46.75	36.09	1.2954
<i>UMeshR005</i>	487.68	481.58	1.0127
<i>UMeshR125</i>	7505.23	6693.08	1.1213
\mathcal{P}^2			
<i>UMeshR005</i>	194.04	194.17	0.99933
<i>UMeshR025</i>	3327.84	3007.23	1.1066
<i>UMeshR125</i>	50,888.5	48,079.7	1.0584
\mathcal{P}^3			
<i>UMeshR005</i>	615.75	548.37	1.1229
<i>UMeshR025</i>	7125.95	6075.18	1.173
<i>UMeshR125</i>	98,615.8	70,847.9	1.3919

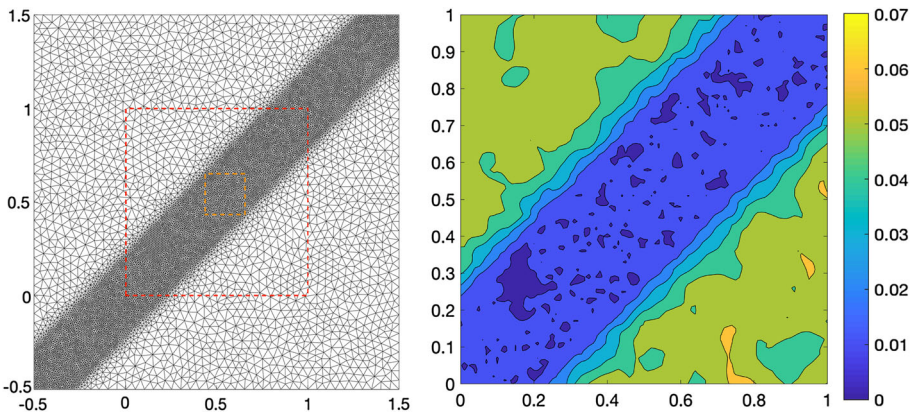


Fig. 9 Left: example of an unstructured, diagonally variable-sized, triangular mesh described in Sect. 4.4. Right: adaptive characteristic length for the mesh on the left within the dotted red lines (Color figure online)

4.4 Delaunay Triangulation for Diagonally Variable-Sized Mesh

In this example, we choose a nonuniform mesh as shown in Fig. 9, such that the mesh has high resolution until a distance 0.2 units from the diagonal and low resolution away from the diagonal. We perform a L^2 projection of $\sin(2\pi(x + y))$ and advect it using the following equation:

$$u_t + u_x + u_y = 0, \quad (x, y) \in (-0.5, 1.5) \times (-0.5, 1.5), \quad T = 1.5$$

We obtain the simulation results over three different meshes, where the ratios of elements are 2:1, 5:1, and 10:1; we refer to them as *DvsMeshR2*, *DvsMeshR5*, and *DvsMeshR10*, respectively. We postprocess the simulation results using the L-SIAC filters, with the direction parameter along the diagonal, i.e., 45° with the x -axis. In Fig. 10, we visualize the postprocessing error caused by the filters using log plots for *DvsMeshR5* for different polynomial orders. The line plots in this figure are created by sampling the error plots along the dotted line. The ticks at the top and bottom of the plot indicate the intersections of element

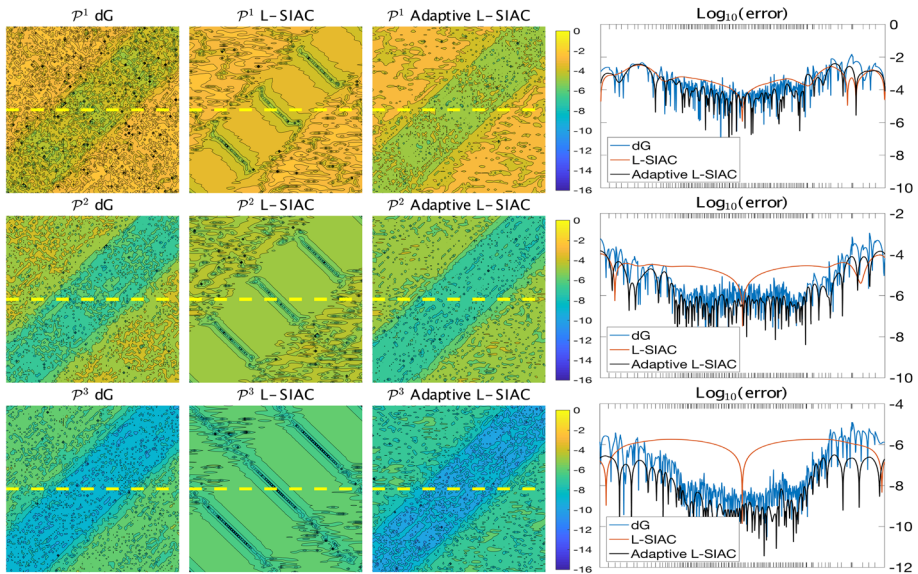


Fig. 10 Pointwise error ($\log_{10}(\text{Error})$) contour plots over a mesh ($DvsMeshR5$) shown in Fig. 9 and described in Sect. 4.4. 1st column: dG error output by the simulation; 2nd column: postprocessed with the L-SIAC filter; 3rd column: postprocessed with the adaptive L-SIAC filter; 4th column: errors along the line $y = 0.5$. Top row: \mathcal{P}^1 polynomials; middle row: \mathcal{P}^2 polynomials; bottom row: \mathcal{P}^3 polynomials

Table 8 L^2 and L^∞ errors for the simple diagonally variable-sized meshes described in Sect. 4.4

Mesh	L^2			L^∞		
	dG	L-SIAC	Adaptive L-SIAC	dG	L-SIAC	Adaptive L-SIAC
\mathcal{P}^1						
<i>DvsMeshR2</i>	3.3e-04	4.5e-04	2.1e-04	8.9e-03	4.4e-03	3.2e-03
<i>DvsMeshR5</i>	7.2e-05	3.2e-04	5.5e-05	3.3e-03	2.0e-03	1.9e-03
<i>DvsMeshR10</i>	5.4e-05	3.0e-04	5.2e-05	2.6e-03	2.2e-03	1.9e-03
\mathcal{P}^2						
<i>DvsMeshR2</i>	8.6e-06	4.5e-05	5.1e-06	4.5e-04	2.6e-04	1.6e-04
<i>DvsMeshR5</i>	2.1e-06	3.7e-05	1.2e-06	3.4e-04	2.2e-04	1.1e-04
<i>DvsMeshR10</i>	1.45e-06	3.84e-05	9.9e-07	2.1e-04	2.0e-04	7.4e-05
\mathcal{P}^3						
<i>DvsMeshR2</i>	1.0e-07	5.7e-06	8.1e-08	7.0e-06	3.09e-05	2.5e-06
<i>DvsMeshR5</i>	1.1e-08	4.7e-06	9.1e-09	1.4e-06	2.7e-05	7.8e-07

boundaries with the dotted line. From these error plots, especially the line plots, it is clear that the adaptive L-SIAC filter has high accuracy in high-resolution areas.

In Table 8, the L^2 and L^∞ errors for the L-SIAC filters are calculated in a rectangular region shown by the orange box in Fig. 9. The dG, L-SIAC filter, and adaptive L-SIAC filter have similar accuracy for \mathcal{P}^1 polynomials, and the adaptive L-SIAC filter has better accuracy compared to the dG and the L-SIAC filter for \mathcal{P}^2 and \mathcal{P}^3 polynomials. Regarding

Table 9 Cost (seconds) of evaluating L^2 and L^∞ errors for the diagonally variable-sized meshes described in Sect. 4.4

Mesh	L-SIAC	Adaptive L-SIAC	Speed-up
\mathcal{P}^1			
<i>DvsMeshR2</i>	48.4	12.8	3.78
<i>DvsMeshR5</i>	1288.4	79.9	16.12
<i>DvsMeshR10</i>	17,821.8	287.9	60.02
\mathcal{P}^2			
<i>DvsMeshR2</i>	245.2	59.3	4.13
<i>DvsMeshR5</i>	6625.75	362.17	18.29
<i>DvsMeshR10</i>	96,539.7	1339.97	72.04
\mathcal{P}^3			
<i>DvsMeshR2</i>	740.29	181.85	4.07
<i>DvsMeshR5</i>	19,646.1	1113.35	17.64

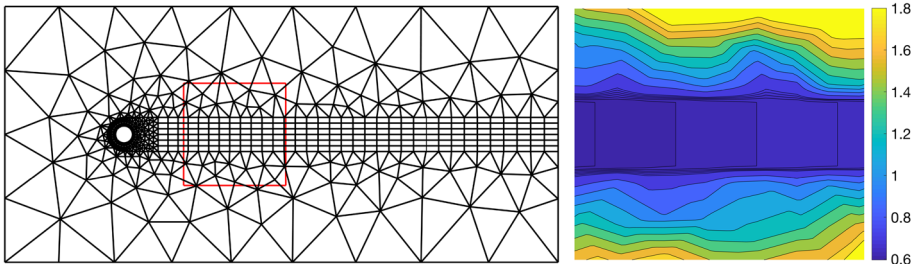


Fig. 11 Left: mesh file used for simulating flow past the cylinder. Right: visualization of adaptive characteristic length in the area shown by the red box on the left (Color figure online)

the computational cost, Table 9 shows the speed-up for the adaptive L-SIAC filter increases with the increase in the resolution of the mesh.

4.5 2D simulation of flow past a cylinder

The incompressible Navier–Stokes solver from the Nektar++ [6] suite is used to create the fluid flow data used in this example. The flow past a circular cylinder at constant viscosity is a transient problem. The simulation is created on the mesh shown in Fig. 11, which has been provided as a part of the Nektar++ suite. The mesh consists of 500 triangles and 330 quadrilaterals of polynomial degree 2 (\mathcal{P}^2) with 81 as the ratio of the largest to the smallest edge. The continuous Galerkin methodology is used, which guarantees C^0 continuity at the element boundaries. The simulation (input) parameters were set to use the Velocity Correction Scheme outlined in [23], the convective form of the nonlinear terms, and IMEX order one. The Reynolds number is set to $Re = 500$, and the shedding behind the cylinder generates consistently shaped vortices. A single snapshot of the simulation data inside the rectangular box in Fig. 11 is used to calculate the vorticity.

The contours of vorticity calculated using the element derivatives (cG derivatives) are shown in the left-hand portion of Fig. 12. The right-hand portion is computed by sampling the data along the dotted line. The line segments are colored differently for each element to highlight the discontinuities. The discontinuities at the element boundaries are due to applying

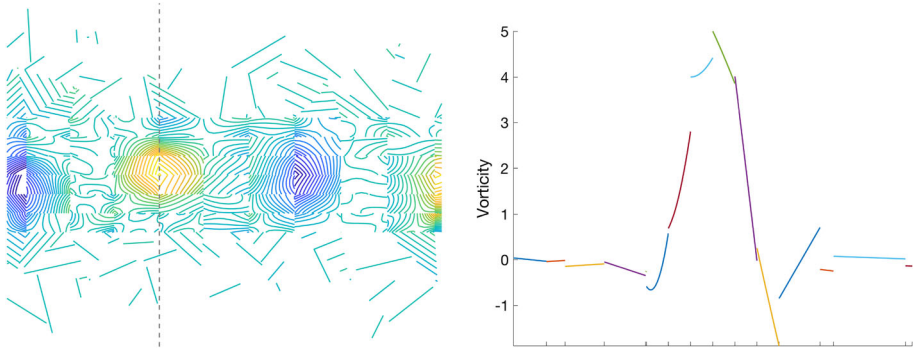


Fig. 12 Left: contours for vorticity using element derivatives in the wake of the 2D cylinder. Right: line plot of the vorticity extracted along the dashed line shown in the contour image

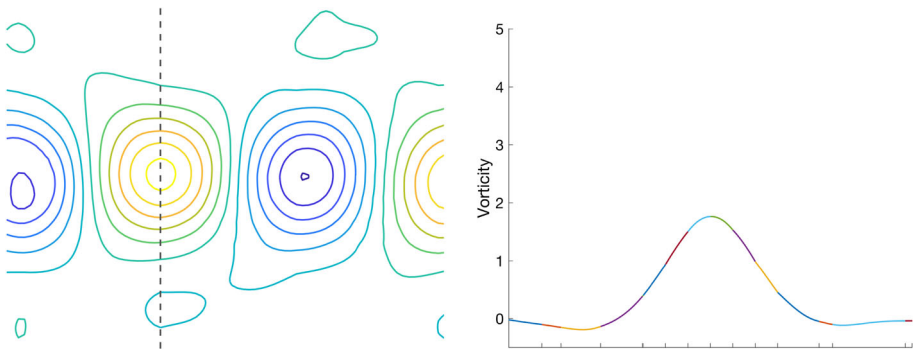


Fig. 13 Left: contours for vorticity using the L-SIAC filter in the wake of the 2D cylinder. Right: line plot of the vorticity extracted along the dashed line shown in the contour image

the derivative on cG simulation data. The ticks along the x-axis show the intersection of the element boundaries with the dotted line.

To calculate the vorticity using the L-SIAC filter, we used the approach of Jallepalli et al. [4], with the characteristic length equal to largest element in the rectangular box in Fig. 11 ($H = 1.45$). The contours of the vorticity thus obtained are shown in Fig. 13; they are continuous, but the magnitude of vorticity along the dotted line is diminished (over smoothed) at the peak.

Again, by applying the technique of Jallepalli et al. [4] to calculate the vorticity, but this time using the adaptive L-SIAC filter, we show the contours for vorticity in Fig. 14. These contours have a pronounced peak and are continuous, in contrast to the contours in Fig. 13, using the L-SIAC filter, which are also continuous but have a diminished peak. This pronounced peak is recovered due to adaptively reducing the characteristic length equal to the element length in the high-resolution region of the mesh. The time to compute the vorticity using the L-SIAC filter on the 2D plane at 10^4 locations is 42 seconds compared to 31.4 seconds for the adaptive L-SIAC filter.

4.6 3D Simulation of Counter-Rotating Vortices

To qualitatively show the use of the adaptive L-SIAC filter on an unstructured nonuniform 3D mesh, we give another Nektar++ simulation example. An incompressible Navier–Stokes

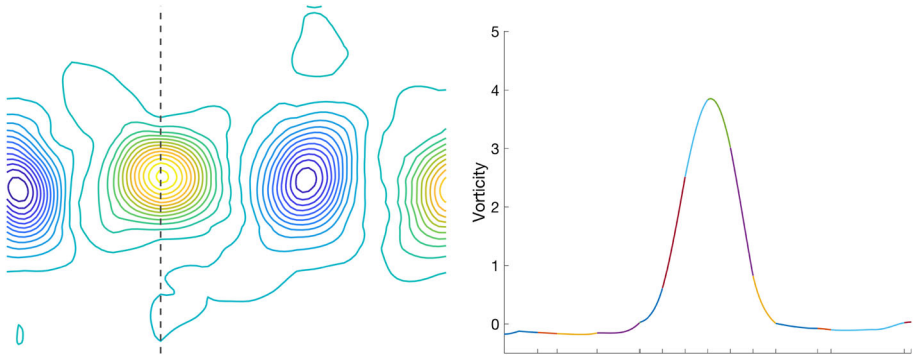


Fig. 14 Left: contours for vorticity using the adaptive L-SIAC filter in the wake of the 2D cylinder. Right: line plot of the vorticity extracted along the dashed line shown in the contour image

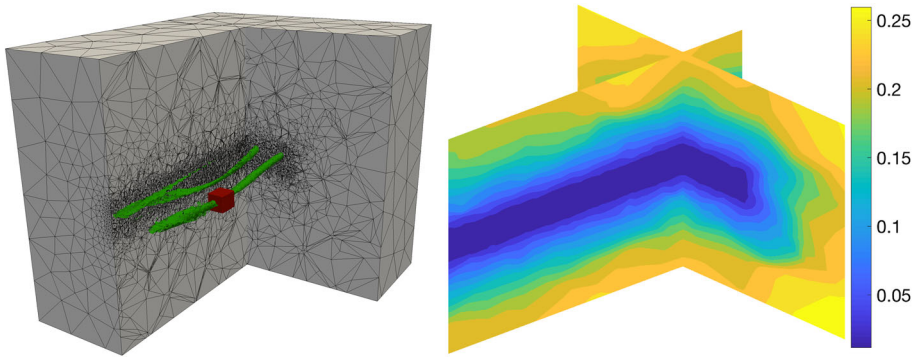


Fig. 15 Left: mesh file used for simulating counter-rotating vortices. Middle: visualization of counter-rotating vortices. Right: adaptive characteristic length across the mesh on the left

solver with cG discretization is used to generate the two primary vortices and their secondary counter-rotating vortices shown in Fig. 15. The simulation (input) parameters were set to use the Velocity Correction Scheme outlined in [23], the convective form of the nonlinear terms with SVV de-aliasing, and IMEX order two. Further details on the simulations are given in [24]. The simulation mesh is adaptively refined for high resolution at the location of the vortices, i.e., the region of interest, and gradually coarsened to increase the efficiency of the simulation. The simulation mesh contains 223,837 polynomial degree five (P^5) tetrahedra, with the ratio of the largest to the smallest edge equal to 22.

The red cube shown on the vortices in the right-hand portion of Fig. 15 indicates the selected region we used to more closely examine the vorticity field. To produce the left image of Fig. 16, the box is sampled, and the vorticity using the element derivatives is computed at a resolution of $100 \times 100 \times 100$. This 3D lattice is then iso-surfaced (value = 12 units) to generate the vortex tubes; the iso-surface suffers from discontinuity at the element boundaries. The right-hand portion of the Fig. 16 shows the data from a 2D slice, which in this case is computed from the data on an $10^2 \times 10^2$ lattice. This image shows the discontinuities at the element boundaries. At the chosen iso-value, we can identify the primary vortex and secondary vortex, but it is difficult to distinguish between them due to the discontinuities in the iso-surface and the contours.

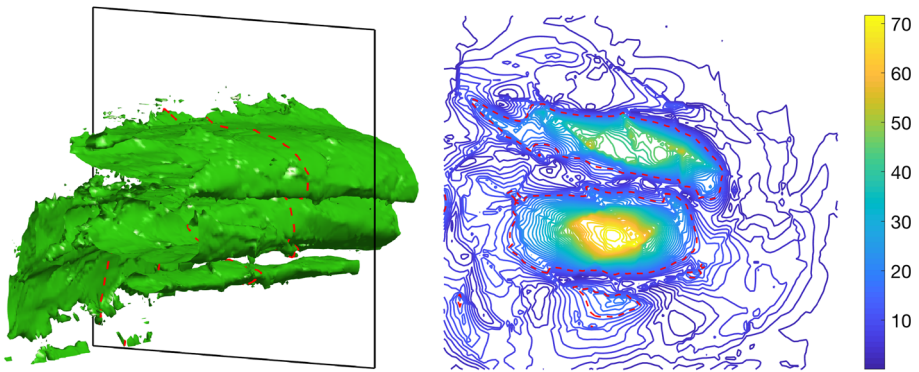


Fig. 16 Left: an iso-contour of vorticity extracted using the element derivatives. Right: contour lines showing the vorticity field extracted over the plane in the left image

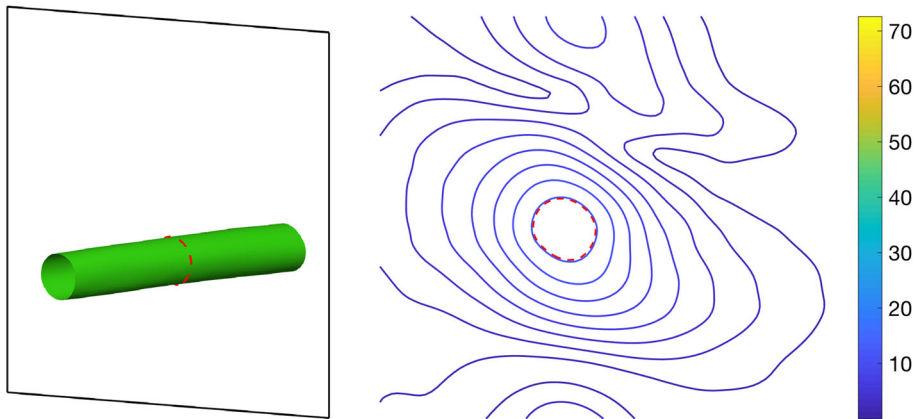


Fig. 17 Left: an iso-contour of vorticity extracted using the L-SIAC filter. Right: contour lines showing the vorticity field extracted over the plane in the left image

We calculate the vorticity of the field in the same subset using the L-SIAC methodology proposed in Jallepalli et al. [4]. All the derivatives required— u_y^* , u_z^* , v_x^* , v_y^* , w_x^* , w_z^* —are computed using the L-SIAC filter, where u , v , w are the components of the velocity in the given vector field. The parameters used for the L-SIAC filter are B-splines of order seven ($D^1K(11, 7)$, where D^1 represents the total derivative), and the characteristic length is $H = 0.034$, which is the maximum edge length in the box. In Fig. 17, we calculate the magnitude of vorticity at all the sampled points, generate the iso-surface of the vorticity at the same value used for the cG derivatives (12 units), and draw the contours at the same plane we selected for the cG derivatives. The iso-surface and the contours using the L-SIAC filter in Fig. 17 are continuous, but the secondary vortex is undetectable due to the error introduced by a characteristic length that is too large.

Following the technique proposed by Jallepalli et al. [4], using the adaptive L-SIAC filter with B-splines of order seven ($D^1K(11, 7)$), and the adaptive characteristic length discussed in this paper we can calculate the magnitude of vorticity. In Fig. 18, we show the iso-surface and contours of vorticity using the adaptive L-SIAC filter. Unlike the L-SIAC filter in Fig. 17, the iso-surface and contours are continuous, and both the primary and secondary vortices are detected.

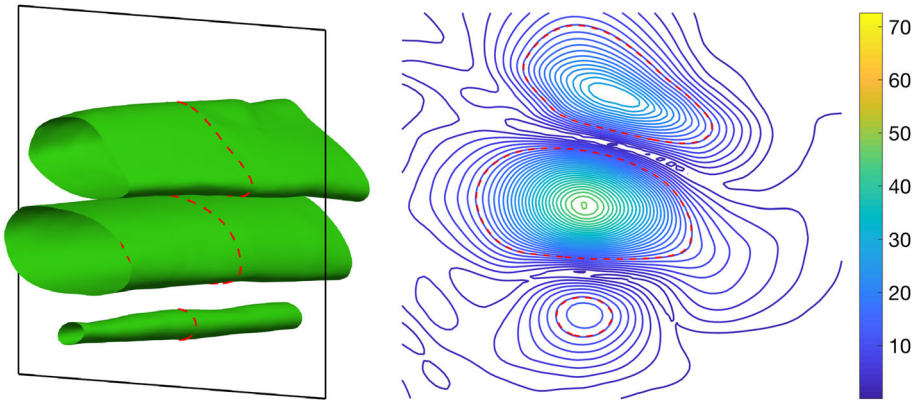


Fig. 18 Left: an iso-contour of vorticity extracted using the adaptive L-SIAC filter. Right: contour lines showing the vorticity field extracted over the plane in the left image

5 Conclusion

The motivation for this work was to produce an algorithm for adaptively scaling the L-SIAC filter to maintain the accuracy of simulations obtained using engineering applications; in particular, to focus on postprocessing simulation data obtained from unstructured meshes with the resolution varying over several orders of magnitude. The paper shows that the algorithm presented herein displays similar or better accuracy and lower computation time on general unstructured meshes, through a quantitative analysis of analytical examples and qualitative results for the simulation data. In both cases, the adaptive L-SIAC filter is similar to or more accurate than the L-SIAC filter with maximum edge length as the characteristic length. We also acknowledge that choosing the direction parameter for the L-SIAC filters in the case of unstructured meshes is still an open issue and needs to be addressed in future work.

One of the drawbacks of this algorithm is that it does not take into account the anisotropy of elements in the mesh; future work needs to include the anisotropy of the mesh as an input to the algorithm to help estimate the best direction(s) and scaling for the L-SIAC filter. An additional area of future research is to further develop the mathematical connections between our work, the mesh function proposed in [20,21] and the characteristic length scaling factor proposed in [13,19].

Acknowledgements The authors would like to thank in particular the reviewers for their valuable comments and insights; their comments very much helped us improve the paper. The authors thank Dr. Jennifer Ryan and Dr. Xiaozhou Li for their insights and recommendations. The authors also wish to thank Professor Spencer Sherwin (Imperial College London, UK), Mr. Alexandre Sidot, and the Nektar++ Group for the counter-rotating vortex data and helpful discussions. The authors acknowledge support from ARO W911NF-15-1-0222 (Program Manager Dr. Mike Coyle).

References

1. Lombard, J.-E.W., Moxey, D., Sherwin, S.J., Hoessler, J.F., Dhandapani, S., Taylor, M.J.: Implicit large-eddy simulation of a wingtip vortex. *AIAA J.* **54**(2), 506–518 (2015)
2. Chooi, K., Comerford, A., Sherwin, S., Weinberg, P.: Intimal and medial contributions to the hydraulic resistance of the arterial wall at different pressures: a combined computational and experimental study. *J. R. Soc. Interface* **13**(119), 20160234 (2016)

3. Docampo-Sánchez, J., Ryan, J.K., Mirzargar, M., Kirby, R.M.: Multi-dimensional filtering: reducing the dimension through rotation. *SIAM J. Sci. Comput.* **39**(5), A2179–A2200 (2017)
4. Jallepalli, A., Docampo-Sánchez, J., Ryan, J.K., Haimes, R., Kirby, R.M.: On the treatment of field quantities and elemental continuity in fem solutions. *IEEE Trans. Vis. Comput. Graph.* **24**(1), 903–912 (2017)
5. Mirzaee, H., King, J., Ryan, J.K., Kirby, R.M.: Smoothness-increasing accuracy-conserving filters for discontinuous Galerkin solutions over unstructured triangular meshes. *SIAM J. Sci. Comput.* **35**(1), A212–A230 (2013)
6. Cantwell, C.D., Moxey, D., Comerford, A., Bolis, A., Rocco, G., Mengaldo, G., De Grazia, D., Yakovlev, S., Lombard, J.-E., Ekelschot, D., Jordi, B., Xu, H., Mohamied, Y., Eskilsson, C., Nelson, B., Vos, P., Biotto, C., Kirby, R.M., Sherwin, S.J.: Nektar++: an open-source spectral/hp element framework. *Comput. Phys. Commun.* **192**, 205–219 (2015)
7. Cockburn, B., Luskin, M., Shu, C.-W., Süli, E.: Post-processing of Galerkin methods for hyperbolic problems. In: Karniadakis, G., Cockburn, B., Shu, C.-W. (eds.) *Proceedings of the International Symposium on Discontinuous Galerkin Methods. Lecture Notes in Computational Science and Engineering*, Newport, RI, 1999, vol. 11, pp. 291–300. Springer-Verlag, Berlin (1999)
8. Cockburn, B., Luskin, M., Shu, C.-W., Süli, E.: Enhanced accuracy by post-processing for finite element methods for hyperbolic equations. *Math. Comput.* **72**(242), 577–606 (2003)
9. Ryan, J.K., Li, X., Kirby, R.M., Vuik, C.: One-sided position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering over uniform and non-uniform meshes. *J. Sci. Comput.* **64**(3), 773–817 (2015)
10. van Slingerland, P., Ryan, J.K., Vuik, C.: Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering for improving discontinuous Galerkin solutions. *SIAM J. Sci. Comput.* **33**(2), 802–825 (2011)
11. Nguyen, D.-M., Peters, J.: Nonuniform discontinuous Galerkin filters via shift and scale. *SIAM J. Numer. Anal.* **54**(3), 1401–1422 (2016)
12. Mirzargar, M., Jallepalli, A., Ryan, J.K., Kirby, R.M.: Hexagonal smoothness-increasing accuracy-conserving filtering. *J. Sci. Comput.* **73**(2–3), 1072–1093 (2017)
13. Li, X., Ryan, J.K., Kirby, R.M., Vuik, C.: Smoothness-increasing accuracy-conserving (SIAC) filters for derivative approximations of discontinuous Galerkin (DG) solutions over nonuniform meshes and near boundaries. *J. Comput. Appl. Math.* **294**, 275–296 (2016)
14. Mirzaee, H., Ji, L., Ryan, J.K., Kirby, R.M.: Smoothness-increasing accuracy-conserving (SIAC) post-processing for discontinuous Galerkin solutions over structured triangular meshes. *SIAM J. Numer. Anal.* **49**(5), 1899–1920 (2011)
15. Steffen, M., Curtis, S., Kirby, R.M., Ryan, J.K.: Investigation of smoothness-increasing accuracy-conserving filters for improving streamline integration through discontinuous fields. *IEEE Trans. Vis. Comput. Graph.* **14**(3), 680–692 (2008)
16. Walfisch, D., Ryan, J.K., Kirby, R.M., Haimes, R.: One-sided smoothness-increasing accuracy-conserving filtering for enhanced streamline integration through discontinuous fields. *J. Sci. Comput.* **38**(2), 164–184 (2009)
17. King, J., Mirzaee, H., Ryan, J.K., Kirby, R.M.: Smoothness-increasing accuracy-conserving (SIAC) filtering for discontinuous Galerkin solutions: improved errors versus higher-order accuracy. *J. Sci. Comput.* **53**(1), 129–149 (2012)
18. Curtis, S., Kirby, R.M., Ryan, J.K., Shu, C.-W.: Postprocessing for the discontinuous Galerkin method over nonuniform meshes. *SIAM J. Sci. Comput.* **30**(1), 272–289 (2007)
19. Li, X.: *Smoothness-Increasing Accuracy-Conserving Filters for Discontinuous Galerkin Methods: Challenging the Assumptions of Symmetry and Uniformity* (2015)
20. Demlow, A., Stevenson, R.: Convergence and quasi-optimality of an adaptive finite element method for controlling L^2 errors. *Numer. Math.* **117**(2), 185–218 (2011)
21. Makridakis, C.G.: On the Babuška–Osborn approach to finite element analysis: L^2 estimates for unstructured meshes. *Numer. Math.* **139**(4), 831–844 (2018)
22. Geuzaine, C., Remacle, J.-F.: *Gmsh: a 3-D finite element mesh generator with built-in pre-and post-processing facilities.* *Int. J. Numer. Methods Eng.* **79**(11), 1309–1331 (2009)
23. Karniadakis, G.E., Sherwin, S.J.: *Spectral/hp Element Methods for Computational Fluid Dynamics*, 2nd edn. Oxford University Press, Oxford (2005)
24. Sidot, A.: *Spectral/hp Element Methods Applied to Vortex Structures in a Low Incidence Delta Wing Wake Compared to Experiments.* M.S. thesis in Aeronautics, Imperial College London (2017)