J Sci Comput (2014) 58:690–704
DOI 10.1007/s10915-013-9748-2

Smoothness-Increasing Accuracy-Conserving (SIAC) Filters for Discontinuous Galerkin Solutions: Application to Structured Tetrahedral Meshes

Hanieh Mirzaee · Jennifer K. Ryan · Robert M. Kirby

Received: 31 January 2013 / Revised: 24 June 2013 / Accepted: 28 June 2013 /
Published online: 25 July 2013

Abstract In this paper, we attempt to address the potential usefulness of smoothness-increasing accuracy-conserving (SIAC) filters when applied to real-world simulations. SIAC filters as a class of post-processors were initially developed in Bramble and Schatz (Math Comput 31:94, 1977) and later applied to discontinuous Galerkin (DG) solutions of linear hyperbolic partial differential equations by Cockburn et al. (Math Comput 72:577, 2003), and are successful in raising the order of accuracy from $k+1$ to $2k+1$ in the L^2—norm when applied to a locally translation-invariant mesh. While there have been several attempts to demonstrate the usefulness of this filtering technique to nontrivial mesh structures (Curtis et al. in SIAM J Sci Comput 30(1):272, 2007; Mirzaee et al. in SIAM J Numer Anal 49:1899, 2011; King et al. in J Sci Comput, 2012), the application of the SIAC filter never exceeded beyond two-space dimensions. As tetrahedral meshes are often the type considered in more realistic simulations, we contribute to the class of SIAC post-processors by demonstrating the effectiveness of SIAC filtering when applied to structured tetrahedral meshes. These types of meshes are generated by tetrahedralizing uniform hexahedra and therefore, while maintaining the structured nature of a hexahedral mesh, they exhibit an unstructured tessellation within each hexahedral element. Moreover, we address the computationally intensive task of performing numerical integrations when one considers tetrahedral elements for SIAC filtering and provide guidelines on how to ameliorate these challenges through the use of more general cubature rules. We consider two examples of a hyperbolic equation and confirm the usefulness of SIAC filters in obtaining the superconvergence accuracy of $2k+1$ when applied to structured tetrahedral meshes. Additionally, the DG methodology merely requires weak constraints on the fluxes between elements. As SIAC filters improve this weak continuity to

H. Mirzaee (✉) · R. M. Kirby
School of Computing, University of Utah, Salt Lake City, UT, USA
e-mail: mirzaee@cs.utah.edu

R. M. Kirby
e-mail: kirby@cs.utah.edu

J. K. Ryan
Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands
e-mail: J.K.Ryan@tudelft.nl

$\mathcal{C}^{k-1}$—continuity at the element interfaces, we provide results that show how post-processing is useful in extracting smooth isosurfaces of DG fields.

Keywords Discontinuous Galerkin · SIAC filtering · Accuracy enhancement · Structured tetrahedral meshes · Post-processing

1 Introduction and Motivation

A class of post-processing techniques was developed in [1] and later continued by Cockburn et al. [2,6] for discontinuous Galerkin (DG) solutions of linear hyperbolic partial differential equations. This post-processor, later known as smoothness-increasing accuracy-conserving filter [7], is successful in raising the order of accuracy from $k + 1$ to $2k + 1$ in the L^2—norm when applied to a locally translation-invariant mesh. While there have been several attempts to demonstrate the usefulness of this filtering technique to nontrivial mesh structures, the application of the SIAC filter never exceeded beyond two-space dimensions. Thereby, we consider the contribution of this paper to be the very first attempt of its kind in demonstrating the potential usefulness of SIAC filtering when applied to real-world simulations. In this work, we examine the effect of filtering over three-dimensional structured tetrahedral meshes. These types of meshes are generated by tetrahedralizing uniform hexahedra and therefore while maintaining the structured nature of a hexahedral mesh, they exhibit an unstructured tessellation within each hexahedral element. We consider two examples of a hyperbolic equation and demonstrate that it is indeed possible to obtain the superconvergence accuracy of $2k+1$ through the application of the SIAC filter. Moreover, we address the computationally intensive task of performing numerical integrations when one considers tetrahedral elements for SIAC filtering and provide guidelines on how to ameliorate these challenges through the use of more general cubature rules. Additionally, the DG methodology merely requires weak constraints on the fluxes between elements. As SIAC filters improve this weak continuity to $\mathcal{C}^{k-1}$—continuity at the element interfaces, we provide results that show how post-processing is useful in extracting smooth isosurfaces of DG fields.

In order to extend the application of the SIAC filter to more general mesh structures, the authors in [3] considered a smoothly-varying mesh as well as a random mesh and demonstrated the $2k + 1$ order of accuracy for the smoothly-varying mesh. However, numerical results were only provided for one-dimensional and two-dimensional tensor-product meshes. The application of the SIAC filter to structured triangulations was extended both theoretically and numerically in [4]. Numerical examples confirming the effectiveness of SIAC filtering over structured triangular meshes were given. In [5], theoretical proof demonstrating the applicability of the SIAC filter to adaptive meshes was provided. Moreover, numerical examples were given confirming the accuracy enhancing capability of SIAC filtering to a broader range of translation-invariant meshes. However, the authors in [5] only considered two-dimensional uniform triangulations. In [8], Mirzaee et al. provided the computational extension of this filtering technique to general unstructured triangulations. Four different mesh types were considered and numerical results were presented in two-dimensions indicating lower errors and increased smoothness through a proper choice of kernel scaling. Consequently, to the best of authors' knowledge there has never been a published work regarding SIAC post-processing over tetrahedral mesh structures.

While the theoretical extension of the SIAC filter to tetrahedral meshes is straightforward, the computational extension is challenging due to the non-tensor product nature of these mesh types. The filtering kernel in higher dimensions is a tensor-product of one-dimensional kernels. This fact can be exploited when filtering over tensor-product translation-invariant meshes (such as quadrilateral or hexahedral meshes) to gain more efficiency by evaluating the higher dimensional integrals as products of 1D integrals. However, in the case of triangular or tetrahedral meshes, the higher dimensional integrals are not separable which in turn lead to a substantial increase in the computational cost. This issue is more pronounced for tetrahedral meshes where the kernel/mesh intersection will result in many integration regions. Consequently, using convenient tensor-product quadrature rules will result in a non-tractable postprocessing for tetrahedral meshes. In this work, we have taken advantage of the more general cubature rules that yield substantially fewer integration points comparing to the tensor-product rules. With these integration rules, we were able to collect results for three different mesh resolutions and polynomial orders. Detail of the implementation is provided in the following sections.

We proceed in this paper by providing the basics of the DG scheme over tetrahedral meshes followed by a brief review of the SIAC filter. In Sect. 2, we briefly discuss the theoretical foundations in SIAC filtering of DG solutions and how it applies to structured tetrahedral meshes. We then continue by presenting the detail of the implementation as well as important practical considerations. Section 3, provides numerical results confirming the usefulness of post-processing over structured tetrahedral meshes. Finally Sect. 4, concludes this paper. We further note that throughout this document, we use the terms *filtering* and *post-processing* interchangeably.

1.1 The Discontinuous Galerkin Formulation for Tetrahedral Mesh Structures

The discontinuous Galerkin method has become increasingly popular in the last decades, while the first report on the DG finite-element method dates back to 1970s [9]. Since its introduction by Reed and Hill in 1973 [10] for solving the linear neutron transport equation on triangular meshes, its flexibility in handling complex geometries and its high-order accuracy have resulted in successive development and application of the DG methodology to the solution problems in fluid and solid mechanics, acoustics, electromagnetics and so on. Although the governing equations of these applications are generally represented by systems of linear hyperbolic equations, the DG has been also extended to systems of nonlinear equations (see [11,12]).

In this paper we consider accuracy enhancement of numerical solutions to three-dimensional linear hyperbolic equations of the form

$$
\begin{aligned}
u_t + \sum_{i=1}^{3} \frac{\partial}{\partial x_i}(A_i(\mathbf{x})u) = 0, \quad \mathbf{x} \in \Omega \times [0, T],\\
u(\mathbf{x}, 0) = u_o(\mathbf{x}), \quad \mathbf{x} \in \Omega
\end{aligned} \tag{1}
$$

where $\Omega \in \mathbb{R}^3$, and $A_i(\mathbf{x}),\ i = 1, 2, 3$ is bounded in the L^∞−norm. We also assume smooth initial conditions are given along with periodic boundary conditions.

To start the discretization of Eq. (1) in space with DG methodology, we develop a suitable weak formulation of (1) on a partition of Ω that we denote by $\bar{\Omega}$ which consists of N non-overlapping tetrahedral elements τ_e and is given by

$$
\bar{\Omega} = \bigcup_{e=1}^{N} \tau_e. \tag{2}
$$

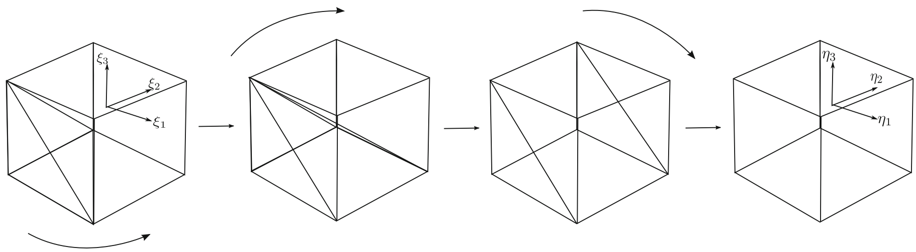


Fig. 1 Tetrahedron to Hexahedron transformation for evaluating the DG approximation using tensorial basis functions

Moreover, in the semi-discrete formulation we obtain an approximate solution by defining the approximation space $V_h \subset V$. In DG methodology the space V_h is continuous in each element but discontinuous across element interfaces and is defined such that

$$V_h = \left\{ \varphi \in L^2(\Omega) : \quad \varphi|_{\tau_e} \in \mathbb{P}^k(\tau_e), \qquad \forall \tau_e \in \bar{\Omega} \right\}, \tag{3}$$

i.e., the space of L^2-integrable functions defined over τ_e with degree at most k. Consequently, the weak form of Eq. (1) will be

$$\int_{\tau_e} \frac{\partial u_h}{\partial t} v_h \, d\mathbf{x} - \sum_{i=1}^{3} \int_{\tau_e} f_i(\mathbf{x}, t) \frac{\partial v_h}{\partial x_i} \, d\mathbf{x} + \sum_{i=1}^{3} \int_{\partial \tau_e} \hat{f}_i \hat{n}_i v_h \, ds = 0, \tag{4}$$

where $\partial \tau_e$ is the boundary of the element τ_e, and $\hat{n_i}$ is the unit outward normal to the element boundary in the ith direction. Furthermore, $f_i(x, t) = A_i(x) u_h(x, t)$ is the flux function. We note that the flux function is multiply defined at element interfaces and, therefore, we impose the definition that $\hat{f}_i \hat{n}_i = h(u_h(\mathbf{x}^{exterior}, t), u_h(\mathbf{x}^{interior}, t))$ be a consistent two-point monotone Lipschitz flux as in [13].

The approximate solution u_h is given by

$$u_h(\mathbf{x}, t) = \sum_{p=0}^{k} \sum_{q=0}^{k-p} \sum_{r=0}^{k-p-q} u_{\tau_e}^{(p,q,r)}(t) \phi^{(p,q,r)}(\mathbf{x}), \quad \mathbf{x} \in \tau_e. \tag{5}$$

Here $u_{\tau_e}^{(p,q,r)}(t)$ is the expansion coefficient at time t corresponding to the expansion polynomial $\phi^{(p,q,r)}$ in the tetrahedral element τ_e.

We note that for $\phi^{(p,q,r)}$, we choose a hierarchical expansion basis using Jacobi polynomials of mixed weight that retains the tensor-product property and is key in obtaining computational efficiency via the *sum-factorisation* technique [14]. In order to implement such expansion basis we perform a mapping to the so-called *collapsed coordinate* system. Fig. 1 depicts how a tetrahedron in the Cartesian coordinate system (ξ_1, ξ_2, ξ_3) is transformed to a hexahedron in the collapsed coordinate system (η_1, η_2, η_3). The advantage of such a system is that we can then define one-dimensional functions upon which we can construct our multidomain tensorial basis. Once in the collapsed coordinate system, we can evaluate the DG approximation u_h of degree k at a single point in $O(k^4)$ floating point operations using the sum-factorisation technique. For more information regarding tensorial basis functions and the collapsed coordinate system consult [14].

By substituting Eq. (5) into Eq. (4), we solve for the expansion coefficients $u_{\tau_e}^{(p,q,r)}(t)$ and we complete our DG scheme. We add that for our DG solver, we used the Nektar++ implementation given in [14] and available at http://www.nektar.info.

1.2 Overview of the Smoothness-Increasing Accuracy-Conserving Filter

Built upon the framework initially established by Bramble and Schatz [1] and Mock and Lax [15], Cockburn et al. [2,6], introduced a class of post-processors, later known as smoothness-increasing accuracy-conserving (SIAC) filter, for hyperbolic PDEs using the discontinuous Galerkin method. The post-processor consists of a convolution kernel applied to the approximation only once, at the final time, and is independent of the partial differential equation under consideration as long as the necessary *negative-order norm* error estimate can be proven. The negative-order norm error estimates give us information on the oscillatory nature of the error and should be of higher order than the L^2-norm error estimates for the post-processor to be applicable. The post-processor extracts this information and works to filter out oscillations in the error and to enhance the accuracy in the usual L^2-norm, up to the order of the error estimates in the negative-order norm. Indeed, the smoothness will increase to $\mathcal{C}^{k-1}$ and the order will improve to $\mathcal{O}(h^{2k+1})$ for a locally translation-invariant mesh. SIAC filtering has been extended to a broader set of applications such as being used for filtering within streamline visualization algorithms in [7,16,17]. In this section we provide a brief introduction of this filtering technique. A more detailed background can be found in [2,18,19].

The post-processor is simply the discontinuous Galerkin solution u_h at the final time T, convolved against a B-spline kernel $K_H^{2k+1,k+1}$. That is, in one-dimension,

$$\begin{aligned} u^\star(x) &= K_H^{2k+1,k+1} \star u_h, \\ &= \frac{1}{H}\int_{-\infty}^{\infty} K^{2k+1,k+1}\left(\frac{y-x}{H}\right) u_h(y)dy, \end{aligned} \tag{6}$$

where $u^\star$ is the post-processed solution and H is the kernel scaling parameter. The superscript $2k+1, k+1$ typically represents the number of B-splines used in the kernel as well as the B-spline order. We shall drop this superscript for simplicity. k is also the degree of the numerical approximation.

The convolution kernel $K(x)$ is a linear combination of B-splines and is given by

$$K(x) = \sum_{\gamma=-k}^{k} c_\gamma \psi^{k+1}(x-\gamma), \tag{7}$$

where c_γ are the kernel coefficients and are assigned such that the kernel reproduces polynomials up to degree $2k$. That is $K \star p = p$, where p is a polynomial of degree $0, \cdots, 2k$. ψ^{k+1} is the B-spline of degree k, and

$$K_H = \frac{1}{H} K\left(\frac{x}{H}\right). \tag{8}$$

The definition in Eq. (7), provides a *symmetric* form of the kernel. This type of the kernel acquires equal amount of information from both sides of an evaluation point x and is therefore not suitable for post-processing near boundaries and shocks. A one-sided form of the kernel was introduced in [20] and further developed in [18] that only collects information from one side of the boundary or shock. In this paper, we only consider periodic boundary conditions and consequently the symmetric form could be applied everywhere in our computational domain, however, all the discussions herein could easily extend to accommodate one-sided filtering. We further add that in higher dimensions the convolution kernel is a tensor-product of one-dimensional kernels.

2 Smoothness-Increasing Accuracy-Conserving Filter for Structured Tetrahedral Meshes

We begin this section by stating the main theorem in SIAC filtering of DG solutions [5]:

Theorem 1 *Let u_h be the DG solution to the linear hyperbolic equation given in Eq.* (1) *at the final time T. The approximation is taken over a mesh whose elements are of size $h = \frac{1}{\ell}H$ in each coordinate direction where ℓ is a multi-index of a dimension equal to the number of elements in one direction. For example, if $\ell = [\ell_1, \cdots, \ell_N]$, where N is the number of elements in one direction, then the size of the element i will be $\frac{1}{\ell_i}H$. Consequently, H represents the macro-element size of which any particular element is generated by hierarchical integer partitioning of the macro-element. Given sufficient smoothness in the initial data we obtain the following estimate for the L_2-error,*

$$\|u - K_H \star u_h\|_\Omega \leq \mathsf{C} H^{2k+1}, \tag{9}$$

where K_H is the post-processing kernel given in Eq. (8) *scaled by H.*

Comparing to the original theorem in [2], Theorem 1 encompasses a broader range of mesh structures. More detail along with several numerical examples can be found in [5].

We note that if within each macro-element of size H we assume equal partitioning, the resulting mesh will have a *translation invariant* structure. A translation invariant mesh is a type of mesh in which we can identify a repeating pattern [21]. Consequently, according to Theorem 1, we are able to obtain higher order accuracy in the L^2-norm when post-processing translation invariant meshes. The structured tetrahedral mesh we consider in this paper will fall into this category.

To generate a translation invariant structured tetrahedral mesh, we first split the domain into uniform hexahedral elements and then subdivide each hexahedral element into tetrahedra. As it is mentioned in [22], there are eight different ways to divide a hexahedron into tetrahedra. These configurations are shown in Fig. 2. As it is understood this figure, the top configuration leads to five tetrahedra per element while the rest lead to six tetrahedra. Another point worth mentioning is that in the first five configurations (first and second row in Fig. 2), we sometimes need to flip the hexahedral element when constructing the entire mesh so that the diagonal edges of adjacent hexahedra align. We are however not required to do that if we choose any of the last three configurations (bottom row in Fig. 2). In either case, we are always able to identify a repeating pattern within these structured meshes. In this paper we have chosen the lower left configuration in Fig. 2 as our structured tetrahedral mesh. We continue by providing the detail of the implementation of the SIAC filter over structured tetrahedral meshes.

The convolution kernel in three dimensions is formed by the tensor-product of one-dimensional kernels. That is

$$\hat{K}(x, y, z) = K(x) \times K(y) \times K(z). \tag{10}$$

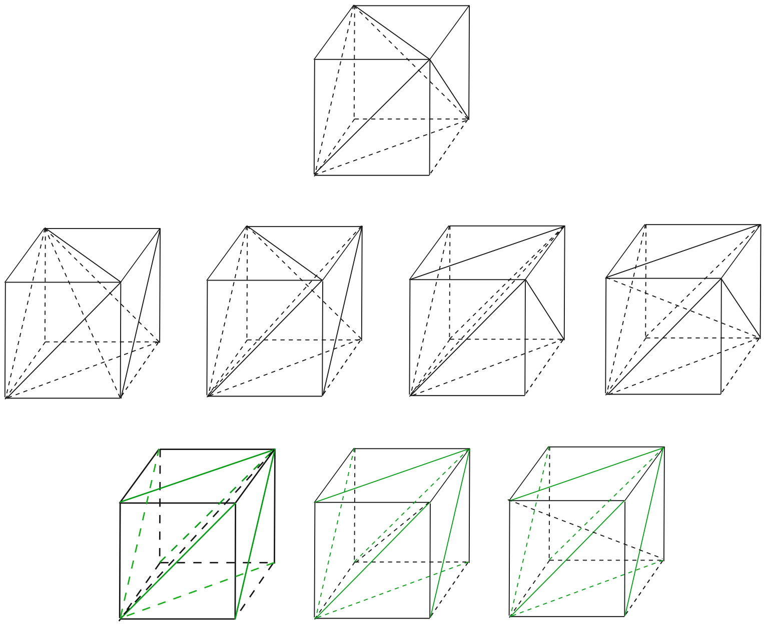


Fig. 2 All possible subdivisions of a hexahedral element into five and six tetrahedra. When juxtaposing the hexahedral elements, it is necessary to flip the hexahedron in x-, y- or z-direction with the first five configurations. No flipping is required with the configurations in the *bottom row*. We consider the *lower left* configuration as our structured tetrahedral mesh

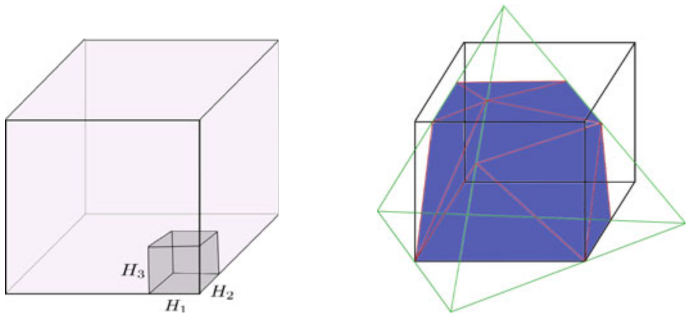
Consequently the post-processor in three-dimensions will have the following form:

$$u^{\star}(x, y, z) = \frac{1}{H_1 H_2 H_3}$$
$$\times \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} K\left(\frac{x_1 - x}{H_1}\right) K\left(\frac{x_2 - y}{H_2}\right) K\left(\frac{x_3 - z}{H_1}\right) u_h(x_1, x_2, x_3) dx_1 dx_2 dx_3, \qquad (11)$$

where u_h is the approximate DG solution of the numerical simulation and H_i, $i = 1, 2, 3$ are the kernel scaling parameters in each direction.

We note that in [4] and [8], we thoroughly discussed the extension of the SIAC filter to structured and unstructured triangular meshes respectively. Similarly, here we have taken the existing implementation of this filtering technique and applied it to structured tetrahedral meshes.

The convolution kernel as presented in Eq. (7) along with the DG approximation u_h are piecewise polynomials. Therefore to numerically evaluate the integral in Eq. (11) exactly to machine precision, we need to subdivide the integration domain into regions of sufficient continuity, where the integrand does not have any break in regularity. Previously in [8], we demonstrated that these integration regions can be found by solving a geometric intersection problem between a square and a triangle for triangular meshes. In three dimensions, the footprint of the kernel is contained in a cube that is further subdivided by the kernel knots into smaller cubes of $H_1 \times H_2 \times H_3$ dimensions. As a result, to find the regions of continuity as shown in Fig. 3, we find the intersection region between a tetrahedral element and a cube. For this we apply the Sutherland-Hodgman clipping algorithm from computer graphics [23].



(a) Kernel footprint **(b)** Kernel-mesh intersection

Fig. 3 Footprint of a three-dimensional kernel (**a**). Demonstration of an intersection between a tetrahedral element and a cube of the kernel footprint (**b**)

Following Theorem 1, the scaling parameters H_i, $i = 1, 2, 3$, which determine the extent of the kernel on the DG mesh, will be equal to the translation invariance of the mesh. This is necessary in order to observe the proper order of convergence in the L^2—norm after the application of the SIAC filter. It is therefore clear that H_i is equal to the uniform mesh spacing for the configurations in the bottom row of Fig. 2 as the entire mesh could be constructed by exactly repeating the hexahedral element. However, for the other configurations, whenever we perform a flip in a direction i, the scaling H_i will be twice as large as the uniform mesh spacing.

To evaluate the post-processed solution at a point denoted by (x, y, z), we center the kernel at that point. We then find the intersection regions and evaluate the resulted integrals. Therefore the integral in Eq. (11) now becomes

$$\begin{aligned} u^\star(x, y, z) &= \frac{1}{H^3} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \bar{K}(x_1)\bar{K}(x_2)\bar{K}(x_3) u_h(x_1, x_2, x_3) dx_1 dx_2 dx_3 \\ &= \frac{1}{H^3} \sum_{T_j \in Supp\{\hat{K}\}} \int_{T_j} \bar{K}(x_1)\bar{K}(x_2)\bar{K}(x_3) u_h(x_1, x_2, x_3) dx_1 dx_2 dx_3 \end{aligned} \tag{12}$$

where we have denoted $\bar{K}(x_i) = K(\frac{x_i - x}{H})$ for simplicity, and $Supp\{\hat{K}\}$ contains all the tetrahedral elements T_j that intersect with the kernel footprint.

We note that the final integration region resulted as the kernel-mesh intersection is itself a polyhedron. For ease of implementation, we further tetrahedralize this polyhedron by triangulating its faces (as shown in Fig. 3b) and connecting the resulting triangles to the centroid of the polyhedron. Consequently, the integral in Eq. (12) becomes

$$\begin{aligned} &\int_{T_j} \bar{K}(x_1)\bar{K}(x_2)\bar{K}(x_3) u_h(x_1, x_2, x_3) dx_1 dx_2 dx_3 \\ &\quad = \sum_{n=0}^{N} \int_{\tau_n} \bar{K}(x_1)\bar{K}(x_2)\bar{K}(x_3) u_h(x_1, x_2, x_3) dx_1 dx_2 dx_3 \end{aligned} \tag{13}$$

where N is the total number of tetrahedral subregions formed in the tetrahedral element T_j as the result of kernel-mesh intersection.

By numerically computing the integral in Eq. (13) using a quadrature technique, we can now evaluate the post-processed solution $u^*(x, y, z)$.

2.1 Practical Considerations

From the computational perspective, post-processing over tetrahedral meshes is a very challenging task. Let us consider again the post-processing formula given in Eq. (11). Following our discussion in the previous section, in computing the post-processed value at a single point (x, y, z), there exist three distinct steps:

1. Centering the kernel at (x, y, z) and identifying the support of the kernel over the DG mesh.
2. Solving a geometric intersection problem to obtain the integration regions.
3. Numerically evaluating the integrals by a means of quadrature rule.

In the case of structured tetrahedral meshes, it suffices to find the extent of the kernel on the basic hexahedral mesh which has a uniform structure. It is trivial that the footprint of the kernel over such a uniform mesh can be found in constant computational time. In the case of unstructured meshes, as it was performed in [8], we can always group the unstructured mesh elements within a regular grid. Consequently, we conclude that Step 1 has always a constant computational complexity.

To find the integration regions in Step 2, as mentioned in the previous section, we perform the Sutherland-Hodgman clipping algorithm. In this algorithm we loop through the faces of one polyhedron and clip it against the second polyhedron. The computational complexity of this algorithm is $\mathcal{O}(n)$, where $n = f_1 \times f_2$ and is equal to 24 when finding the intersection between a cube and a tetrahedron. $f_i, i = 1, 2$ indicate the number of faces in each polyhedron. For structured tetrahedral meshes, the support of the kernel spans $3k + 1$ hexahedral elements in each direction $(2 \times \left(\lfloor \frac{2k+1}{2} \rfloor + \frac{k+1}{2}\right))$, k being the degree of the approximation. That is, for each evaluation point we need to process $6 \times (3k + 1)^3$ tetrahedra for the configuration we chose in this paper. As each tetrahedral element intersects with at most 8 cubes of the kernel, the cost of finding all the intersection regions will therefore be $8 \times 6 \times 24(3k + 1)^3$ or $\mathcal{O}(k^3)$.

Given the amount of processing we need to perform in Step 2, we should maintain the computational cost of Step 3 as low as possible in order to have a tractable post-processing algorithm. It appears that the main computational bottleneck in post-processing over tetrahedral meshes lies in evaluation of the integrals. The key point to consider here is that the three-dimensional integral over a tetrahedral region τ_n as given in Eq. (13) is in fact an expensive operator to evaluate due to several function evaluations. To understand why this is the case, consider quadrilateral and hexahedral meshes. For these mesh structures, the tensor product nature of the convolution kernel in higher dimensions would result in separation of the integrals and ultimately the multidimensional integral could essentially be evaluated by computing one-dimensional integrals (see [19]). This evaluation could even further be simplified if one also considers tensorial basis functions to represent the DG approximation in multi-dimensions [14]. Consequently, using tensor-product Gaussian quadrature rules provides a convenient way to numerically evaluate the integrals involved in the convolution when dealing with tensor-product mesh structures such as quadrilateral and hexahedral meshes. However, in the case of triangular or tetrahedral meshes, due to the dependency of the coordinate directions, the convolution integral is not separable (see [19] for triangles) and therefore, we can not reduce the cost of computing the 3D integral through evaluating 1D integrals. As a result, using the conventional tensor-product quadrature rules will not be optimal in the

Table 1 Number of quadrature points required in each integration technique for triangular elements

Triangles		
k	Tensor-product	Cubature
P^2	16	12
P^3	25	19
P^4	49	33

k indicates the degree of the numerical approximation

Table 2 Number of quadrature points required in each integration technique for tetrahedral elements

Tetrahedra		
k	Tensor-product	Cubature
P^2	125	46
P^3	343	140
P^4	729	–

k indicates the degree of the numerical approximation. Cubature points for the P^4 approximation were not available

sense of using the fewest function evaluations for a given approximation degree. A suitable alternative here is to use non-tensor product formulas -generally known as *cubature* rules. The cubature rules are complicated to derive and are not known to very high-orders. For our post-processing experiments we used the pregenerated points and weights by Zhang et al. in [24] and available for polynomials up to degree 14. We note that from Eq. (13), it is clear that the cubature rule we apply should be exact to integrate a polynomial of degree $4k$. Table 1 and Table 2, provide the number of points (see [24]) required to evaluate the convolution integral over each region of continuity using the aforementioned quadrature strategies. While there is no substantial difference in the number of quadrature points for the case of triangular elements, there is a noticeable difference in terms of computational efficiency when using cubature rules for tetrahedral meshes over tensor-product quadrature. Note that we could not find the cubature points for a $k = 4$ DG approximation (polynomial integrand of degree 16). However, in practice we were able to use even fewer cubature points, that are required to integrate a lower degree polynomial, to evaluate the convolution operator. In fact, using only 24 points for a P^2 approximation, 36 points for a P^3, and 46 points for a P^4 approximation seemed to be enough to provide the accuracy predicted by theory.

We further emphasize the use of the sum-factorization technique introduced in Sect. 1.1 for evaluating our DG approximation at the cubature points. The application of this technique along with the cubature rules, led to a substantial decrease in the computational intensity of Step 3 of the post-processing algorithm.

3 Numerical Results

In this section we provide numerical results that demonstrate the effectiveness of SIAC filtering when applied to structured tetrahedral meshes. We consider a constant coefficient and a variable coefficient advection equation and show that it is indeed possible to gain the optimal convergence rate of $2k+1$ in the L^2 and L_∞ norms after the application of the SIAC

Table 3 Errors before and after post-processing the solutions of the constant coefficient advection equation over a structured tetrahedral mesh

Mesh	L_2-error	Order	L_∞-error	Order	L_2-error	Order	L_∞-error	Order
–	Before post-processing				After post-processing			
$\mathbb{P}^2$								
6,000	1.06E−03	–	6.78E−03	–	2.50E−04	–	7.50E−04	–
48,000	1.28E−04	3.04	8.96E−04	2.91	5.52E−06	5.50	1.54E−05	5.60
384,000	1.60E−05	3.00	1.13E−04	2.98	1.40E−07	5.30	3.82E−07	5.33
$\mathbb{P}^3$								
6,000	1.21E−04	–	1.30E−03	–	3.72E−05	–	8.60E−05	–
48,000	7.51E−06	4.01	8.41E−05	3.95	2.15E−07	7.43	4.43E−07	7.60
384,000	4.76E−07	3.98	5.43E−06	3.99	1.07E−09	7.65	3.01E−09	7.20
$\mathbb{P}^4$								
6,000	2.02E−05	–	1.70E−04	–	not valid	–	not valid	–
48,000	6.53E−07	4.95	5.65E−06	4.91	2.02E−09	–	6.50E−09	–
384,000	2.21E−08	4.88	1.89E−07	4.90	5.43E−12	8.56	1.79E−11	8.50

filter. Moreover, to demonstrate the effectiveness of SIAC filtering in introducing smoothness back to our numerical approximation, we provide an example of isosurfaces of a DG field before and after the application of the post-processor.

3.1 Constant Coefficient Advection Equation

For this example we consider the following advection equation

$$u_t + u_x + u_y + u_z = 0, \qquad (x, y, z) \in (0, 1) \times (0, 1) \times (0, 1), \qquad T = 6.28, \qquad (14)$$

with initial condition $u(0, x, y, z) = \sin(2\pi(x+y+z))$. Table 3 provides the error results for three different mesh resolutions and polynomial degrees. From these results it is completely clear that SIAC filtering has been effective in raising the order of accuracy to $2k + 1$ both in the L_2 and L_∞ norms.

3.2 Variable Coefficient Advection Equation

For this example we consider solutions of the equation

$$u_t + (au)_x + (au)_y + (au)_z = f, \quad (x, y, z) \in (0, 1) \times (0, 1), \times (0, 1) \quad T = 6.28. (15)$$

We implement a smooth coefficient $a(x, y, z) = 2 + \sin(2\pi(x + y + z))$, with an initial condition of $u(x, y, z, 0) = \sin(2\pi(x + y + z))$. Periodic boundary conditions are implemented in both directions and the forcing function, $f(x, y, z, t)$, is chosen so that the exact solution is $u(x, y, z, t) = \sin(2\pi(x + y + z - 2t))$. Table 4 demonstrates the error results before and after post-processing. Similarly to the previous example, we see a clear improvement in the order of accuracy. Moreover, the magnitudes of the errors are lower after the application of the post-processor.

Table 4 Errors before and after post-processing the solutions of the variable coefficient advection equation over a structured tetrahedral mesh

Mesh	L_2-error	Order	L_∞-error	Order	L_2-error	Order	L_∞-error	Order
–	Before post-processing				After post-processing			
$\mathbb{P}^2$								
6,000	1.78E−03	–	6.50E−03	–	3.02E−04	–	6.90E−04	–
48,000	2.24E−04	2.99	8.83E−04	2.88	7.61E−06	5.31	1.58E−05	5.45
384,000	2.82E−05	2.98	1.14E−04	2.95	1.93E−07	5.30	3.71E−07	5.41
$\mathbb{P}^3$								
6,000	2.00E−04	–	1.10E−03	–	4.50E−05	–	9.10E−05	–
48,000	1.32E−05	3.92	6.97E−05	3.98	3.06E−07	7.20	7.36E−07	6.95
384,000	8.60E−07	3.94	4.51E−06	3.95	2.10E−09	7.18	5.79E−09	6.99
$\mathbb{P}^4$								
6,000	2.91E−05	–	2.00E−04	–	not valid	–	not valid	–
48,000	9.74E−07	4.90	6.79E−06	4.88	5.50E−09	–	8.43E−09	–
384,000	3.15E−08	4.95	2.32E−07	4.87	1.68E−11	8.50	2.67E−11	8.30

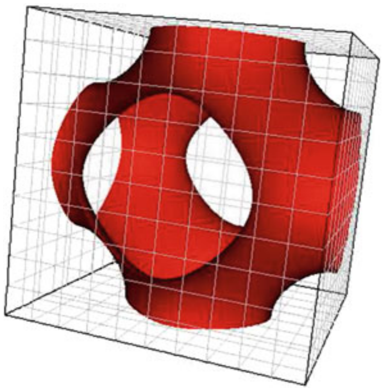


Fig. 4 Isosurface constructed based on the analytical solution $u(x, y, z) = \cos(2\pi x) + \cos(2\pi y) + \cos(2\pi z)$ for $isovalue = 0.2$

3.3 Isosurfaces of a DG Field

Here we again consider the advection equation given in Eq. (14) but this time with $u(x, y, z) = \cos(2\pi x) + \cos(2\pi y) + \cos(2\pi z)$ as the initial condition. We consider an isosurface of the numerical approximation of this equation before and after the application of the SIAC filter. Figure 4 demonstrates an isosurface extracted from the analytical field for $isovalue = 0.2$.

To extract an isosurface we follow the approach of the traditional Marching Cubes (MC) algorithm [25] with some modifications. For a given MC mesh (which is a hexahedral mesh), we loop through individual cubes and identify the cube that contains part of the isosurface for a given isovalue. In traditional MC, linear interpolation is used to find the surface/edge intersection along an edge of the cube. However, in our modified algorithm we perform a higher-order root-finding scheme. That is, we find the intersection of the higher-order DG approximation with the edge

(a) Isosurface based on the DG solution.

(b) Isosurface based on the post-processed solution.

Fig. 5 Comparison of isosurfaces before and after the application of the SIAC filter. **a** Isosurface based on the DG solution, **b** Isosurface based on the post-processed solution

of the cube by a means of root-finding. Therefore, along an edge of the cube that contains the isosurface, we find the intersection point by finding the roots of the following equation:

$$u_h(\mathbf{x}) - isovalue = 0, \tag{16}$$

where u_h is our numerical approximation as given in Eq. (5). When generating isosurfaces using the post-processed data, u_h is replaced by $u^\star$, the post-processed value given in Eq. (11). We add that by applying a root-finding mechanism, we are able to observe the discontinuities that exist in the solution data as long as the MC grid overlaps with the hexahedral mesh that was used to construct our structured tetrahedral mesh.

Figure 5 depicts a zoomed-in portion of the isosurface in Fig. 4, but this time using the approximate DG solution u_h (Fig. 5a) and the post-processed solution $u^\star$ (Fig. 5b) to find the point of intersection in Eq. (16). As you notice there are visible discontinuities in the isosurface constructed on the DG approximation whereas in the isosurface extracted using the post-processed value, there is no discontinuity. In other words, through the application of the SIAC filter we are indeed able to introduce smoothness back to our numerical solution.

4 Conclusion

From its early introduction by Bramble and Schatz in [1] to its later development for linear hyperbolic equations by Cockburn et al. in [2,6], there has never been a demonstration of the effectiveness of the Smoothness-Increasing Accuracy-Conserving filter over three-dimensional mesh structures. In fact, the very first attempt of applying this filtering technique to meshes of nontrivial structures, mainly in one-dimension, was in [3] . Later in a series of papers [4,5,8], the extension to structured triangular meshes, general translation-invariant meshes as well as adaptive meshes and unstructured triangulations were provided; all in two-space dimensions. As our ultimate goal is the application of this filter to real-world simulations, we provided in this paper for the first time, computational results confirming the accuracy-conserving and smoothness-increasing capabilities of the SIAC filter over structured tetrahedral meshes. We considered two variants of a hyperbolic PDE and presented error results which indicates that it is indeed possible to obtain the optimal $2k + 1$ order of accuracy through post-processing. We further demonstrated how post-processing is useful in extracting smooth isosurfaces of DG fields. We believe this is a significant contribution and a major step in extending the application of the SIAC filter beyond conventional 2D mesh structures.

Acknowledgments The first, second and third authors are sponsored in part by the Air Force Office of Scientific Research (AFOSR), Computational Mathematics Program (Program Manager: Dr. Fariba Fahroo), under grant number FA9550-08-1-0156, and by the Department of Energy (DOE NET DE-EE0004449). The second author is sponsored by the Air Force Office of Scientific Research (AFOSR), Air Force Material Command, USAF, under Grant Number FA8655-09-1-3055. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

1. Bramble, J.H., Schatz, A.H.: Higher order local accuracy by averaging in the finite element method. Math. Comput. **31**, 94–111 (1977)
2. 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)
3. 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**(1), 272–289 (2007)
4. 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**, 1899–1920 (2011)
5. 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. Available online (2012)
6. 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, New York (1999)
7. 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**(3), 680–692 (2008)
8. Mirzaee, H., King, J., Ryan, J.K., Kirby, R.M.: Smoothness-increasing accuracy-conserving (SIAC) filters for discontinuous Galerkin solutions over unstructured triangular Meshes. SIAM J. Sci. Comput. (2012). Accepted under revision
9. Cockburn, B., Karniadakis, G., Shu, C.W.: The development of the discontinuous Galerkin methods. In: Discontinuous Galerkin Methods: Theory, Computation and Applications. Lecture Notes on Computational Science and Engineering, vol. 11, 3–50 Springer, New York (2000)
10. Reed, W., Hill, T.: Triangular Mesh Methods for the Neutron Transport Equation. Tech. rep, Los Alamos Scientific Laboratory Report, Los Alamos, NM (1973)
11. Cockburn, B.: Devising discontinuous Galerkin methods for non-linear hyperbolic conservation laws. J. Comput. Appl. Math. (128), 187–204 (2001)
12. Cockburn, B., Shu, C.W.: Runge–Kutta discontinous Galerkin methods for convection-dominated problems. J. Sci. Comput. **16**, 173–261 (2001)
13. Cockburn, B.: Discontinuous Galerkin methods for convection-dominated problems. In: Barth, T.J., Deconinck, H. (eds). High-Order Methods for Computational Physics, Lect. Notes Comput. Sci. Eng.vol. 9, pp. 69–224. Springer, Berlin (1999)
14. Karniadakis, G.E., Sherwin, S.J.: Spectral/hp Element Methods for CFD, 2nd edn. Oxford University Press, UK (2005)
15. Mock, M.S., Lax, P.D.: The computation of discontinuous solutions of linear hyperbolic equations. Commun. Pure Appl. Math. **18**, 423–430 (1978)
16. 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)
17. 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)

18. van Slingerland, P., Ryan, J.K., Vuik, C.: Position-depnedent smoothness-increasing accuracy-conserving (SIAC) filtering for improving discontinuous Galerkin solutions. SIAM J. Sci. Comput. **33**, 802–825 (2011)
19. Mirzaee, H., Ryan, J.K., Kirby, R.M.: Efficient implementation of smoothness-increasing accuracy-conserving (SIAC) filters for discontinuous Galerkin solutions. J. Sci. Comput. **52**(1), 85–112 (2011)
20. 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)
21. Babuška, I., Rodriguez, R.: The problem of the selection of an a-posteriori error indicator based on smoothing techniques. Int. J. Numer. Methods Eng. **36**(4), 539–567 (1993)
22. Dompierre, R., Labbe, P., Vallet, M.G., Camarero, R.: How to subdivide pyramids, prisms and hexahedra into tetrahedra. Tech. rep, Centre de Recherche en Calcul Appliqué, Montréal, Québec (1999)
23. Sutherland, I.E., Hodgman, G.W.: Reentrant polygon clipping. Commun. ACM **17**(1), 32–42 (1974)
24. Zhang, L., Cui, T., Liu, H.: A set of symmetric quadrature rules on triangles and tetrahedra. J. Comput. Math **27**(1), 89–96 (2009)
25. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resulotion 3D surface construction algorithm. In: SIGGRAPH '87 Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, vol. 21, pp. 163–169 ACM, New York, (1987)