SIAM J. SCI. COMPUT.
Vol. 33, No. 2, pp. 802–825


# POSITION-DEPENDENT SMOOTHNESS-INCREASING ACCURACY-CONSERVING (SIAC) FILTERING FOR IMPROVING DISCONTINUOUS GALERKIN SOLUTIONS*

PAULIEN VAN SLINGERLAND†, JENNIFER K. RYAN†, AND C. VUIK†

**Abstract.** Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering is a promising technique not only in improving the order of the numerical solution obtained by a discontinuous Galerkin (DG) method but also in increasing the smoothness of the field and improving the magnitude of the errors. This was initially established as an accuracy enhancement technique by Cockburn et al. for linear hyperbolic equations to handle smooth solutions [*Math. Comp.*, 72 (2003), pp. 577–606]. By implementing this technique, the quality of the solution can be improved from order $k+1$ to order $2k+1$ in the $L^2$-norm. Ryan and Shu used these ideas to extend this technique to be able to handle postprocessing near boundaries as well as discontinuities [*Methods Appl. Anal.*, 10 (2003), pp. 295–307]. However, this presented difficulties as the resulting error had a stair-stepping effect and the errors themselves were not improved over those of the DG solution unless the mesh was suitably refined. In this paper, we discuss an improved filter for enhancing DG solutions that easily switches between one-sided postprocessing to handle boundaries or discontinuities and symmetric postprocessing for smooth regions. We numerically demonstrate that the magnitude of the errors using the modified postprocessor is roughly the same as that of the errors for the symmetric postprocessor itself, regardless of the boundary conditions.

**Key words.** high-order methods, discontinuous Galerkin, filtering, accuracy enhancement

**AMS subject classification.** 65M60

**DOI.** 10.1137/100782188

## 1. Introduction.

Smoothness-increasing accuracy-conserving (SIAC) filtering is a technique that has had initial successful applications in the areas of visualization and aeroacoustics [16, 20]. However, a limitation of this technique becomes evident when it is applied near a discontinuity or domain boundary, due to the symmetric nature of the filter, which is discussed in more detail below. This paper addresses this issue and redefines the previous postprocessor as a position-dependent SIAC filter which consists of a convex combination of different kernel types.

The typical application of SIAC filtering is to improve the order of the numerical solution obtained by a discontinuous Galerkin (DG) method. This is accomplished by using information that is already contained in the numerical solution to increase the smoothness of the DG field and improve the magnitude of the errors.

The foundations for this postprocessor were established by Bramble and Schatz [2]. They showed that the accuracy of Ritz–Galerkin discretizations can be doubled by convolving the solution against a certain symmetric kernel function, only once, at the final time. Thomée [21] provided alternative proofs for the results in [2] by using Fourier transforms. Furthermore, he constructed modified versions of the symmetric

*Submitted to the journal's Computational Methods in Science and Engineering section January 11, 2010; accepted for publication (in revised form) January 19, 2011; published electronically April 7, 2011. 

http://www.siam.org/journals/sisc/33-2/78218.html

†Delft Institute of Applied Mathematics, Delft University of Technology, 2628 CD Delft, The Netherlands (P.vanSlingerland@tudelft.nl, J.K.Ryan@tudelft.nl, C.Vuik@tudelft.nl). The work of the first and second authors was supported by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant FA8655-09-1-3055.

kernel to extract derivative approximations.

Cockburn et al. [6] used the ideas of Bramble and Schatz and those of Mock and Lax [14] to demonstrate that the symmetric postprocessor is also suitable for DG schemes. They proved that, for a certain class of linear hyperbolic equations with sufficiently smooth solutions, the postprocessor enhances the accuracy from order $k+1$ to order $2k+1$ in the $\mathcal{L}^2$-norm, where $k$ is the polynomial degree of the original DG approximation. This postprocessor relies on a symmetric convolution kernel consisting of $2k+1$ B-splines of order $k+1$.

A disadvantage of this symmetric kernel is that it cannot be applied near boundaries and shocks as it requires an equal amount of information from both sides of the point that is being postprocessed. To address this issue, Ryan and Shu [15] extended the ideas in [6] to obtain a one-sided postprocessor that can be applied near boundaries as well as discontinuities in the exact solution. This was achieved by modifying the symmetric kernel such that the support is located on one side of the origin. Although this made it possible to apply the postprocessor near boundaries and shocks, the results obtained were not satisfactory: the errors had a stair-stepping–type structure, and the errors themselves were not improved over those of the unfiltered solution unless the mesh was sufficiently fine (cf. Figure 10).

In this paper, we discuss an improved filter for enhancing DG solutions that easily switches between one-sided postprocessing to handle boundaries or discontinuities and symmetric postprocessing for smooth regions. This is the position-dependent nature of our filtering kernel that relies on different kernels for different domain regions. The improvements to the one-sided kernel are accomplished by combining previous concepts used in one-sided postprocessing for DG solutions with those from spectral methods and finite difference methods [3, 11, 18] to improve the one-sided filter. We obtain the improved one-sided kernel by redefining the basis of our kernel nodes so that it depends upon a smooth shift function, $\lambda(\bar{x})$, as well as using more kernel nodes (B-splines). We can then recast the definition of the postprocessor as a position-dependent SIAC filter using a convex combination of filter types. We numerically demonstrate that this modified position-dependent SIAC filter has boundary errors that are roughly the same as the errors for the symmetric postprocessor itself, regardless of boundary conditions.

We present this paper as follows: In section 2, we briefly review the basic tools used to generate a DG solution as well as the uniform mesh symmetric SIAC filters. In section 3, we introduce improvements to the one-sided SIAC filter and define how to combine this new filter with the symmetric filter. We present our numerical results in section 4, which verifies the improved nature of our position-dependent SIAC filter. We conclude with a discussion of our results in section 5.

## 2. Background.

**2.1. A summary of discontinuous Galerkin methods.** The discontinuous Galerkin method (DG) is numerically a well-established tool that is obtaining greater prominence in computational fluid dynamics applications. It can be thought of as a combination of a finite volume and finite element method. We present a basic outline below. The reader is asked to consult [4, 5, 7, 8, 9, 10] for further details about the DG method.

For the purposes of this paper, consider a simple linear hyperbolic equation,

$$\begin{aligned} u_t + f(u)_x &= 0, \quad x \in \Omega, \quad t \leq T, \\ u(x,0) &= u_0(x), \quad x \in \Omega, \end{aligned} \tag{1}$$

where $f(u) = a\,u$. It is well known that if the discretization of the mesh is defined as $I_j = (x_j - \frac{\triangle x_j}{2}, x_j + \frac{\triangle x_j}{2})$, $j = 1, \ldots, N$, and the basis functions, $\phi_j^\ell(x)$, $\ell = 0, \ldots, k$, are piecewise polynomials of degree less than or equal to $k$, the approximation space is then given by

$$V_h = \{\phi_j^{(\ell)}(x) \in \mathbb{P}^k|_{I_j},\ j = 1, \ldots, N\}. \tag{2}$$

Using this, the DG formulation is then

$$\int_{I_j} (u_h)_t v dx = \int_{I_j} f(u_h) v_x dx - \hat{f}_{j+\frac{1}{2}} v^-_{j+\frac{1}{2}} + \hat{f}_{j-\frac{1}{2}} v^+_{j-\frac{1}{2}} \tag{3}$$

for all $v \in V_h$, with the DG approximation on element $I_j$ being given by

$$u_h(x,t) = \sum_{\ell=0}^{k} u_j^{(\ell)}(t)\, \phi_j^{(\ell)}(x), \qquad j = 1, \ldots, N. \tag{4}$$

Choosing an upwind monotone flux for $\hat{f}$, a system of differential equations is then obtained. This system is then integrated in time using a third-order strong-stability-preserving (SSP) Runge–Kutta scheme such as those in [12, 13, 19].

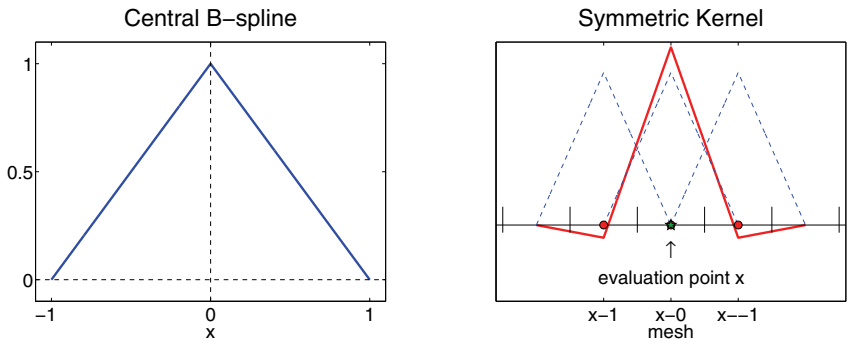
After the DG solution is obtained at the desired final time, we can then apply a smoothness-increasing accuracy-conserving (SIAC) filter. By postprocessing the solution we can obtain higher-order accuracy, provided that the initial condition $u_h(x, 0)$ is the $\mathcal{L}^2$-projection of $u_0(x)$ onto the test space $V_h$. The latter is required by the theoretical error estimates in [6]. However, it should be noted that this preprocessing is required computationally only for the convergence of the filtered solution and not for the convergence of the original DG approximation.

In addition to enhancing the accuracy, the postprocessor also introduces smoothness into the DG field. Note that continuity is only weakly enforced through the numerical fluxes. By convolving the numerical approximation against our kernel consisting of B-splines of order $\ell$ that have $\mathcal{C}^{\ell-2}$-continuity, we are introducing levels of smoothness into the field. As a consequence, it allows for better visualization of streamlines or isosurfaces from a DG field. In light of this we seek to improve upon the DG solution by implementing a position-dependent SIAC filter.

**2.2. SIAC filters.** The symmetric postprocessor for the DG method was introduced by Cockburn et al. [6] to handle periodic linear hyperbolic equations. The general theoretical foundations of this technique were established by Bramble and Schatz [2] and Mock and Lax [14]. The main idea is to convolve the piecewise polynomial DG approximation with a kernel function that is a linear combination of B-splines. It has been shown both theoretically and numerically that this strategy can improve the convergence rate from $\mathcal{O}(h^{k+1})$ to $\mathcal{O}(h^{2k+1})$. Furthermore, the postprocessed approximation is rendered $k-1$ times continuously differentiable. Here, we provide a brief summary of the symmetric postprocessor that provides such an improvement. Further details on the postprocessor and the implementation can be found in [6, 16, 20, 22].

A DG approximation, $u_h$, of polynomial degree $k$ can be postprocessed at the evaluation point $\bar{x}$ by convolving it against a kernel function,

$$u_h^*(\bar{x}) = \frac{1}{H} \int K\left(\frac{\bar{x} - x}{H}\right) u_h(x)\, \mathrm{d}x, \tag{5}$$




(a) Building blocks: $\psi^{(2)}(x)$ (b) Application: $K\left(\frac{\bar{x}-x}{H}\right)$

FIG. 1. *A symmetric kernel that is uniquely determined by the kernel nodes* $\{-1, 0, 1\}$ *(indicated by the circles), scaled by the mesh size* $H = h$*, and the central B-spline of order* $\ell = 2$.

where the scaling $H > 0$ is chosen dependent upon the maximum diameter $h$ of the mesh elements. For a uniform mesh implementation, this scaling is given by $H = h$. The symmetric kernel $K$ is a linear combination of $r + 1$ central B-splines, $\psi^{(\ell)}$, of order $\ell$ of the form

$$K(x) = \sum_{\gamma=0}^{r} c_\gamma \psi^{(\ell)}(x - x_\gamma) \quad \forall x \in \mathbb{R}. \tag{6}$$

In practical applications, $r$ is typically chosen to be $r = 2k$ with $\ell = k+1$. In (6), the kernel nodes $x_0, \ldots, x_r$ are defined to be evenly distributed about the origin, that is,

$$x_\gamma = -\frac{r}{2} + \gamma \quad \forall \gamma = 0, \ldots, r. \tag{7}$$

The kernel nodes $x_0, \ldots, x_r$ will be modified later (cf. (10), (11), and (12)) to obtain other types of kernel functions. The kernel coefficients $c_0, \ldots, c_r$ are defined as the unique solution to the linear system

$$\sum_{\gamma=0}^{r} c_\gamma \int_{\mathbb{R}} \psi^{(\ell)}(x)(x + x_\gamma)^q \, \mathrm{d}x = \begin{cases} 1 & \text{for } q = 0, \\ 0 & \forall\ q = 1, \ldots, r. \end{cases} \tag{8}$$

Finally, the central B-splines, $\psi^{(\ell)}$, can be constructed by convolving the characteristic function on the interval $[-1/2, 1/2]$ with itself $\ell - 1$ times, by a recursion relation, or by using divided differences [17, 22]. Figure 1 displays an illustration of the symmetric form of the SIAC filter based on $r + 1 = 3$ kernel nodes and B-splines of order $\ell = 2$. This is the filter that is usually applied to piecewise linear DG solutions ($k = 1$).

For general polynomial degrees $k$, this SIAC filter which uses $r = 2k$ and $\ell = k+1$ has two main advantages. First, it improves the convergence rate from $\mathcal{O}(h^{k+1})$ to $\mathcal{O}(h^{2k+1})$. Second, it carries the smoothness of the B-spline $\psi^{(k+1)} \in \mathcal{C}^{k-1}$ over to the approximation. In other words, after postprocessing, the discontinuous approximation has become $k - 1$ times continuously differentiable.

A drawback of the symmetric postprocessor is that it cannot be applied near a boundary or a shock. This is because it requires the evaluation of the DG approxi-

mation in the support of $x \mapsto K\left(\frac{\bar{x}-x}{H}\right)$ to be

$$\left[\bar{x} - H\frac{r+\ell}{2}, \bar{x} + H\frac{r+\ell}{2}\right], \tag{9}$$

where $\bar{x}$ is the evaluation point under consideration (cf. (5) and Figure 1(b)). Notice that the kernel takes a symmetric amount of information around the point that is being postprocessed, as noted in [15]. As a consequence, the symmetric postprocessor cannot always be applied, depending on the location of the evaluation point $\bar{x}$ in the domain. For instance, near the boundary the postprocessor should not require unavailable information outside the spatial domain. Similarly, the postprocessor should not take information that requires crossing a shock, since the theoretical error estimates rely on the smoothness of the exact solution.

For this reason, a different type of kernel with a more suitable support is required near boundaries and shocks. In the next section we discuss just such a position-dependent SIAC filter. It takes the basic ideas from those of [15] and improves upon them to create a filter that obtains the same error magnitude at the boundary of the domain as it does in the interior.

**3. Position-dependent SIAC filtering.** To filter near boundaries and shocks, a one-sided postprocessor was proposed by Ryan and Shu in [15] which satisfies (5), (6), and (8). The difference between the one-sided and the symmetric postprocessors is the choice of the kernel nodes, which then leads to different kernel coefficients. While the symmetric kernel uses kernel nodes that are distributed evenly around the origin as in (7), the one-sided kernel is based on kernel nodes such that the support of the kernel is located entirely on one side of the origin. For example, choosing the kernel nodes

$$x_\gamma = \left\lceil \frac{\ell}{2} \right\rceil + \gamma \quad \forall \gamma = 0, \ldots, r = 2k \tag{10}$$

yields the so-called right-sided kernel, whose kernel support is located on the right side of the origin. The resulting postprocessor can be applied to the left of a boundary or a shock (cf. Figure 2, noting the difference between the support of $K(x)$ and the support of $K(\frac{\bar{x}-x}{H})$). Similarly, choosing the kernel nodes to be

$$x_\gamma = -r - \left\lceil \frac{\ell}{2} \right\rceil + \gamma \quad \forall \gamma = 0, \ldots, r = 2k \tag{11}$$

results in the left-sided kernel, which is suitable for application on the right side of a boundary or a shock.

Ryan and Shu observed that the one-sided postprocessor renders the discontinuous approximation $k-1$ times differentiable and that the convergence rate can be improved from $\mathcal{O}(h^{k+1})$ to $\mathcal{O}(h^{2k+1})$, similarly to the symmetric case. Unfortunately, an improvement of the convergence rate does not necessarily imply an improvement of the error. Indeed, it was also observed that the one-sided postprocessor tends to worsen the errors for coarse meshes (cf. Figures 7(c) and 10(b)). The symmetric postprocessor does not suffer from this drawback and is much more accurate. For this reason, the symmetric kernel was applied in the domain interior whenever possible. Only near boundaries and shocks, where the symmetric kernel cannot be applied, was the one-sided kernel used. This type of switching between different kernels made

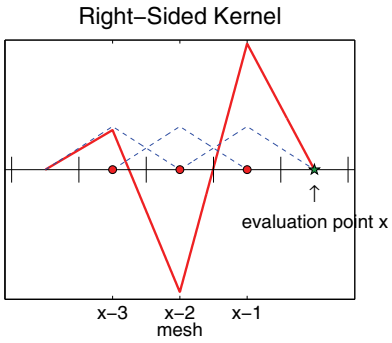



FIG. 2. *Application of the right-sided kernel,* $K\left(\frac{\bar{x}-x}{H}\right)$*, that is uniquely determined by the kernel nodes* $\{1,2,3\}$ *(indicated by the circles) and the central B-spline of order* $\ell = 2$.

it possible to postprocess the entire domain while preserving the accurate nature of the symmetric postprocessor whenever feasible. Details of this switching will be explained in the next section. However, near the boundaries and shocks, the errors were generally worse than the DG solution. Additionally, the switching of the kernels introduced a second problem: the errors showed a stair-stepping nature, indicating that unwanted discontinuities were reintroduced in the postprocessed solution (cf. Figures 7(c) and 10(b)). We seek to improve these two drawbacks in the existing one-sided postprocessor and to cast the kernel in the context of a *position-dependent SIAC filter*. This will be done in two steps. First, the *location* of the kernel nodes $x_0, \ldots, x_r$ is redefined to be position-dependent (section 3.1). By doing this in a smooth manner we remove the stair-stepping errors. Next, the *number* of kernel nodes (or B-splines), $r+1$, are also chosen to be position-dependent (section 3.2). This creates the possibility of using extra kernel nodes in a local neighborhood of boundaries and shocks, which benefits the accuracy, as noted in [3, 11]. These ideas are then combined in a convex combination of filter types to create our position-dependent SIAC filter.

**3.1. Location and scaling of the kernel nodes.** To define the new position-dependent SIAC filter, we begin by redefining the kernel nodes and discussing the scaling of the kernel support. We maintain the postprocessor such that it continues to satisfy (5), (6), and (8). However, the kernel nodes are now given by the generalized form,

$$x_\gamma = -\frac{r}{2} + \gamma + \lambda(\bar{x}) \quad \forall \gamma = 0, \ldots, r. \tag{12}$$

Note that the kernel nodes depend on the evaluation point $\bar{x}$ through a shift function $\lambda$. By redefining the kernel nodes, this leads to a modified kernel support (cf. (9)), which means that the values of the kernel coefficients in (6) using (8) also change. The main subject of this section is the choice of shift function for the kernel nodes as well as control of the kernel support by the choice of $\lambda$ and $H$.

We begin by noting that choosing $\lambda(\bar{x}) = 0$ leads to the symmetric kernel (cf. (7)). Similarly, for $\lambda(\bar{x}) = -\left\lceil \frac{r+\ell}{2} \right\rceil$ and $\lambda(\bar{x}) = \left\lceil \frac{r+\ell}{2} \right\rceil$, the left-sided (cf. (11)) and right-sided (cf. (10)) kernels are obtained. By choosing $\lambda(\bar{x})$ between zero and $\left\lceil \frac{r+\ell}{2} \right\rceil$, the partly right-sided kernels can be obtained. This is illustrated in Figure 3 for $\lambda(\bar{x}) = 1$ and $r = 2$. Similarly, partly left-sided kernels are also defined. Using this information, we want to control the location of the kernel nodes depending on the

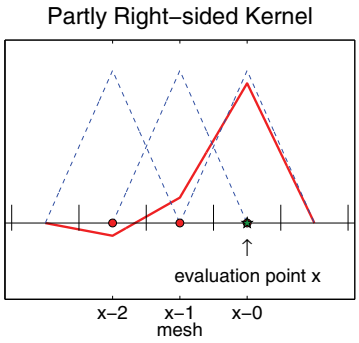



FIG. 3. *Application of a partly right-sided kernel,* $K\left(\frac{\bar{x}-x}{H}\right)$*, that is uniquely determined by the kernel nodes* $\{0,1,2\}$ *(indicated by the circles) and the central B-spline of order* $\ell = 2$*.*

.

evaluation point $\bar{x}$. This also controls the accuracy and applicability of the kernel. For example, the symmetric kernel is known to be more accurate than the one-sided kernel, and this is the kernel that we want to apply in the largest part of the domain interior as possible. This is equivalent to $\lambda(\bar{x})$ being zero. Therefore, it makes sense to choose $\lambda(\bar{x})$ as close to zero as possible when not implementing the symmetric kernel. We thus say that the kernel is applicable if the support of $x \mapsto K\left(\frac{\bar{x}-x}{H}\right)$, which is given by

$$\left[\bar{x} - H\frac{r+\ell}{2} + \lambda(\bar{x}), \bar{x} + H\frac{r+\ell}{2} - \lambda(\bar{x})\right], \tag{13}$$

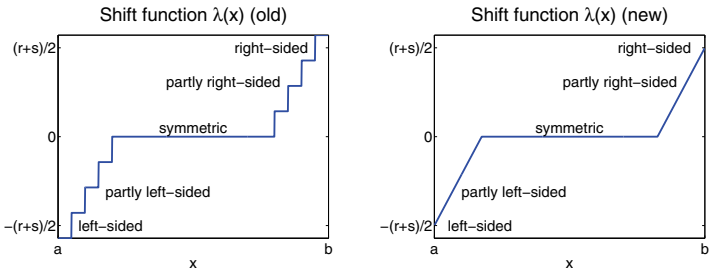
does not contain a shock or a boundary. Note that the kernel support is the result of *shifting the support* of the symmetric kernel in (9) by precisely a distance $\lambda(\bar{x})$. This means that $\lambda$ affects only the location of the kernel support and not its diameter. For now, we delay the discussion of the support diameter and discuss a strategy to obtain a suitable explicit expression for the shift function $\lambda$.

To clarify the idea, consider a subinterval $[a,b]$ of our spatial domain that lies precisely between two subsequent boundaries or shocks. The shift function is obtained by using the following strategy: for every evaluation point $\bar{x} \in [a,b]$, the value $\lambda(\bar{x})$ is chosen as close to zero as possible, to make the kernel "as symmetric as possible" and to maximize the accuracy, yet such that the support of $x \mapsto K\left(\frac{\bar{x}-x}{H}\right)$ lies within $[a,b]$ to ensure the applicability of the kernel.

We begin the investigation of $\lambda$ by noting that in the original one-sided postprocessor, the shift function was designed such that the kernel nodes are integers. Under this assumption, the strategy above leads to a *piecewise constant* shift function (cf. Figure 4(a)),

$$\lambda(\bar{x}) = \begin{cases} \min\{0, -\lceil\frac{r+\ell}{2}\rceil + \lfloor\frac{\bar{x}-a}{H}\rfloor\} & \text{for } \bar{x} \in \left[a, \frac{a+b}{2}\right), \\ \max\{0, \lceil\frac{r+\ell}{2}\rceil + \lceil\frac{\bar{x}-b}{H}\rceil\} & \text{for } \bar{x} \in \left[\frac{a+b}{2}, b\right]. \end{cases}$$

Note that this shift function uses the symmetric kernel in the interior of the domain as often as possible. Near the boundary, where the symmetric kernel cannot be applied, partly one-sided kernels are used. Furthermore, observe that this shift function is discontinuous. As a result, the postprocessed solution is generally also discontinuous precisely at the locations where $\lambda$ is discontinuous. This reveals the cause of the




(a) (b)

FIG. 4. *Choosing the shift function between two subsequent boundaries or shocks* ($r = 2k$, $\ell = k+1$, $k = 2$).

aforementioned stair-stepping character of the errors, which is precisely what we seek to overcome in this section. The discontinuous nature of the shift function is an immediate consequence of the assumption that the kernel nodes are chosen as integers.

By dropping the assumption that $\lambda \in \mathbb{Z}$, we can now improve upon the shift function. We use the strategy above, which sets $\lambda$ as close to zero as possible, but we also consider noninteger kernel nodes. This leads to the following *continuous* shift function (cf. Figure 4):

$$\lambda(\bar{x}) = \begin{cases} \min\{0, -\frac{r+\ell}{2} + \frac{\bar{x}-a}{H}\} & \text{for } \bar{x} \in \left[a, \frac{a+b}{2}\right), \\ \max\{0, \frac{r+\ell}{2} + \frac{\bar{x}-b}{H}\} & \text{for } \bar{x} \in \left[\frac{a+b}{2}, b\right]. \end{cases} \tag{14}$$

Observe that the new shift function takes values closer to zero than the original one. This results in partly one-sided kernels that are "more symmetric" and more accurate than before and kernel coefficients that depend continuously on the evaluation point. Furthermore, observe that the function is infinitely smooth everywhere, except for two locations where it is continuous but not differentiable. As a consequence, the postprocessed solution maintains the same smoothness as the B-splines except in those two locations where it is only continuous. We note that it is possible to design a shift function that has the same smoothness as the B-splines throughout the entire domain, but then $\lambda$ is no longer chosen as close to zero as possible, which results in partly one-sided kernels that are "less symmetric" and less accurate.

We now address the second issue with respect to applicability of the kernel, that is, the diameter of the kernel support, which is controlled by $H$. Note that, in order for $\lambda(\bar{x})$ defined above to exist, it is necessary to choose the scale $H$ sufficiently small so that the diameter of the kernel support, which does not depend on $\lambda$, is not larger than the diameter of the subinterval. That is,

$$H(r+\ell) \leq b - a. \tag{15}$$

It has already been noted that for uniform meshes, $H = h$. However, depending on the locations of domain boundaries or discontinuities in the solution, this may not always be feasible. Testing of the kernel scaling suggests that increasing the kernel size, $H > h$, increases the smoothness of the postprocessed solution but the $\mathcal{L}^2$-errors are larger than for that of the DG solution, unless the mesh is sufficiently refined. This type of scaling is not implemented for our purposes. However, in section 4.5, a scaling less than the uniform element size is used for the coarser meshes. In such a

scaling, oscillations in the error of the DG solution reappear for the filtered solution and the same order of convergence is obtained for both errors. Additionally, the errors are closer to that of the DG solution, although they are slightly worse. The scaling of the convolution kernel for filtering between a domain boundary and discontinuity requires further investigation.

**3.2. Number of kernel nodes.** The previous section introduced a new SIAC filter using a position-dependent choice of the *location* of the kernel nodes. This eliminated the stair-stepping nature of the errors of the original filter. However, the (partly) one-sided kernels tend to worsen the errors for coarse meshes. In this section we further improve the SIAC filter by addressing this issue. This improvement consists of using a position-dependent choice of the *number* of kernel nodes (or B-splines).

The current version of the SIAC filter, defined by (5), (6), (8), (12), and (14), can be applied for any number of kernel nodes, $r+1$. Previously, it was standard to use $r+1 = 2k+1$ kernel nodes. We now drop this convention and consider implementing general integer values of $r$. The motivation for this is that increasing the number of kernel nodes can lead to higher accuracy. This was noted in [3, 11] and is illustrated in sections 4.1 and 4.2. For purposes of clarification, we emphasize that we refer to using a greater number of B-splines as using a greater number of kernel nodes. Moreover, extra kernel nodes increase the kernel support (cf. (13)). This inevitably increases the computational costs. For example, consider using small matrix-vector multiplications for the filtered solution where the postprocessing matrix is precomputed. The postprocessing matrix size is $(r+1)\times(k+1)$ and contains inner products of B-splines with polynomial basis functions. This is multiplied by a vector of length $(k+1)$ that contains the DG modes for a mesh element in the kernel support. Therefore, to postprocess one evaluation point in a one-dimensional uniform mesh, $(r+\ell+1)$ small matrix vector multiplications are required, with a summation of all elements in the resulting vectors. We stress that the postprocessor is applied only once, at the final time of the simulation.

To keep the extra computational costs at a minimum, extra kernel nodes should be used only where necessary, i.e., where the (partly) one-sided kernels are applied. The symmetric kernel, which is applied in the largest part of the domain interior, is sufficiently accurate for the standard number of kernel nodes, i.e., $r+1 = 2k+1$, where $k$ is the polynomial degree of the DG approximation, $u_h$. We emphasise that we use extra kernel nodes only near a boundary or a shock.

Because the number of kernel nodes is an integer, simply letting $r$ depend on the evaluation point $\bar{x}$ would introduce discontinuities. For this reason, we propose considering two postprocessed solutions: $(u_h^*)_{r_1}$, which is based on a relatively small constant number of kernel nodes $r_1+1$, and $(u_h^*)_{r_2}$, which is based on a relatively large constant number of kernel nodes $r_2+1$. These two solutions are obtained from (5), (6), (8), (12), and (14) using $r=r_1$ and $r=r_2$, respectively. Note that both solutions are based on a combination of symmetric and (partly) one-sided kernels. The only difference is the number of kernel nodes. We then combine these two kernels to define the new position-dependent SIAC filter to be a smooth convex combination of these two solutions:

$$u_h^*(\bar{x}) = \theta(\bar{x})(u_h^*)_{r_1}(\bar{x}) + (1-\theta(\bar{x}))(u_h^*)_{r_2}(\bar{x}).$$

In this equation, the coefficient function $\theta$ depends on the position of the evaluation point $\bar{x}$. It takes values in $[0,1]$ in the following manner (see Figure 5 for an illustration). First, choose $\theta$ equal to one in the largest part of the interior of the domain,

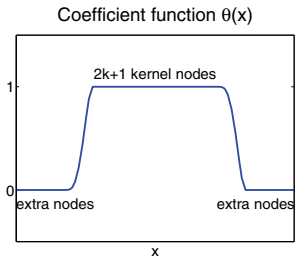



FIG. 5. *Choosing the coefficient function between two subsequent boundaries or shocks.*

where the symmetric kernel already provides enough accuracy for the relatively small number of kernel nodes $r_1 + 1$. In this paper, we use $r_1 + 1 = 2k + 1$, where $k$ is the polynomial degree of the DG approximation $u_h$. Additionally, choose $\theta$ equal to zero near a boundary or a shock, where the (partly) one-sided kernels need a relatively large number of kernel nodes, $r_2 + 1$, to obtain sufficient accuracy. In this paper, we use $r_2 + 1 = 4k + 1$.

It should be noted that other values of $r_2 > r_1$ could be used as well. For example, we also ran tests for $r_2 = 3k+1$ and $r_2 = 5k+1$. As expected, we found that a larger number of kernel nodes leads to more accuracy. However, the difference between using $5k+1$ nodes and using $4k+1$ nodes was relatively small. For $3k+1$ nodes, the errors were not improved over the unfiltered errors for the coarser meshes. Based on these experiments, using $r_2 + 1 = 4k + 1$ is a natural choice for enhancing the errors where necessary, without increasing the kernel support and the computational costs too much.

Moreover, to preserve the smoothness of the kernel, the coefficient function should be at least as differentiable as the applied B-splines, i.e., $\ell - 2$ times (in this paper, we apply the usual B-spline order $\ell = k + 1$). This requires two smooth transition regions, where $\theta(\bar{x}) \in (0, 1)$. In this paper, these regions each consist of two mesh elements in the "symmetric part" of the domain, i.e., where $\lambda(\bar{x}) = 0$, adjacent to the "one-sided part" of the domain, i.e., where $\lambda(\bar{x}) \neq 0$. Furthermore, $\theta$ is chosen to be a polynomial of degree $2\ell + 1$, which is uniquely defined under the given smoothness conditions. For example, one choice of the coefficient function $\theta$ is

$$\theta(\bar{x}) = \begin{cases} 0 & \text{for } \bar{x} \in [a, a_1), \\ p(\bar{x}) & \text{for } \bar{x} \in [a_1, a_2], \\ 1 & \text{for } \bar{x} \in (a_2, b_2), \\ q(\bar{x}) & \text{for } \bar{x} \in [b_2, b_1], \\ 0 & \text{for } \bar{x} \in (b_1, b], \end{cases} \tag{16}$$

where

$$a_1 = a + \tfrac{3k+1}{2}h, \qquad a_2 = a + \left(\tfrac{3k+1}{2} + 2\right)h,$$
$$b_1 = b - \tfrac{3k+1}{2}h, \qquad b_2 = b - \left(\tfrac{3k+1}{2} + 2\right)h,$$

and where $p$ is a polynomial of degree $2\ell+1 = 2k+3$. We note that we further require that $p(a_1) = 0$, $p(a_2) = 1$, and $\frac{\mathrm{d}^n p}{\mathrm{d}x^n}(a_1) = \frac{\mathrm{d}^n p}{\mathrm{d}x^n}(a_2) = 0$ for all $n = 1, \ldots, \ell = k + 1$. A similar definition holds for $q$. We remark that other choices for $\theta$ may also work in practice.

We emphasize that it is not necessary to compute both intermediate postprocessed solutions in the entire domain. This is required only in the four mesh elements that make up the two transition regions where $\theta \in (0, 1)$.

**4. Numerical validation.** This section illustrates the performance of our new position-dependent SIAC filter compared to the original filter for six different test cases. We consider the $\mathcal{L}^2$-projection of a sine function in section 4.1 and the DG solution at the final time $t = 12.5$ of the following four one-dimensional hyperbolic PDEs: a constant coefficient equation with periodic boundary conditions, a constant coefficient equation with Dirichlet boundary conditions, a variable coefficient equation, and a discontinuous coefficient equation in sections 4.2–4.5, and a two-dimensional system in section 4.6. The examples demonstrate that the convergence rate can be improved from $\mathcal{O}(h^{k+1})$ to $\mathcal{O}(h^{2k+1})$ throughout the entire domain and that the errors for the filtered solution can be better than those for the DG solution.

We implement both the old and new postprocessors that are found in section 3 and compare the results. For all test cases, the DG solution was based on first order upwind fluxes, monomial basis functions, and a uniform mesh. Furthermore, for the time-discretization, a third order SSP–RK scheme was applied [12, 13], using a sufficiently small time step to ensure that the time-stepping errors were not dominating the spatial errors. The computation of the convolution in (5) was performed exactly using Gaussian quadrature, as described in [15]. Finally, we note that we made use of the ARPREC multiprecision package in order reduce round-off errors appropriately [1].

**4.1. $\mathcal{L}^2$-projection of a sine function.** The first test case is the $\mathcal{L}^2$-projection of $u(x) = \sin(x)$ onto the space of piecewise polynomials of degree $k = 1, 2, 3$. This test case can also be interpreted as a DG approximation at the initial time. It is the most elementary case that we can test in order to ensure the reliability of our filter.

Table 1 illustrates that the new postprocessor improves the convergence rate from $\mathcal{O}(h^{k+1})$ to at least $\mathcal{O}(h^{2k+1})$. This is also illustrated in Figure 6 for piecewise polynomials of degree $k = 2$. In this figure, we can see that the old and new versions of the postprocessor have the same convergence rate, which is better than that of the DG solution. Additionally, we do see that the old implementation of the one-sided postprocessor is sufficient to improve upon the errors of the DG solution when the mesh is suitably refined. However, the magnitude of the errors obtained from the new implementation of the SIAC filter is considerably better for all meshes, both for the $\mathcal{L}^2$- and $\mathcal{L}^\infty$-norms.

Figure 7 shows the absolute error per evaluation point, which reveals the differences in the local accuracy of the two postprocessors. In the interior of the domain, there are no differences, which stems naturally from the fact that both postprocessors apply a symmetric kernel with $2k + 1$ nodes in that region. As a consequence, the errors are sufficiently smooth and accurate, as expected. The differences between the two postprocessors occur at the boundary of the domain.

In Figures 7(c) and 7(b) we examine the boundary regions. We see that the old postprocessor (cf. Figure 7(c)) shows the two main problems that were discussed at the beginning of section 3. The first is that not all discontinuities have been removed, which can be seen by observing the stair-stepping nature of the errors. The cause of these discontinuities is the choice of the kernel nodes, which has been resolved by introducing a continuous shift function in the definition of the kernel nodes. Indeed, the discontinuities are no longer introduced by the new postprocessor (cf. Figure 7(b)). Second, the old postprocessor has worse errors near the boundary for coarse meshes. This was resolved by changing the number of kernel nodes. By increasing the number

TABLE 1

*Comparison of the global accuracy of the old and new postprocessors for the $\mathcal{L}^2$-projection of the sine function. The new postprocessor improves the convergence rate from $\mathcal{O}(h^{k+1})$ to $\mathcal{O}(h^{2k+1})$.*

| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k = 1$ | | | | | | | | | | | | |
| 20 | 6.510e-03 | - | 5.953e-03 | - | 1.601e-02 | - | 2.213e-02 | - | 4.876e-04 | - | 1.258e-03 | - |
| 40 | 1.629e-03 | 2.00 | 1.500e-03 | 1.99 | 1.714e-03 | 3.22 | 3.111e-03 | 2.83 | 1.899e-05 | 4.68 | 5.352e-05 | 4.56 |
| 80 | 4.074e-04 | 2.00 | 3.759e-04 | 2.00 | 1.584e-04 | 3.43 | 4.000e-04 | 2.96 | 9.024e-07 | 4.40 | 1.792e-06 | 4.90 |
| 160 | 1.019e-04 | 2.00 | 9.402e-05 | 2.00 | 1.416e-05 | 3.48 | 5.035e-05 | 2.99 | 5.330e-08 | 4.08 | 5.694e-08 | 4.98 |
| polynomial degree $k = 2$ | | | | | | | | | | | | |
| 20 | 1.729e-04 | - | 1.279e-04 | - | 3.949e-03 | - | 6.681e-03 | - | 4.186e-06 | - | 3.144e-06 | - |
| 40 | 2.163e-05 | 3.00 | 1.613e-05 | 2.99 | 2.111e-04 | 4.23 | 3.921e-04 | 4.09 | 8.687e-08 | 5.59 | 6.710e-08 | 5.55 |
| 80 | 2.704e-06 | 3.00 | 2.021e-06 | 3.00 | 5.474e-06 | 5.27 | 1.387e-05 | 4.82 | 1.384e-09 | 5.97 | 7.872e-10 | 6.41 |
| 160 | 3.381e-07 | 3.00 | 2.528e-07 | 3.00 | 1.256e-07 | 5.45 | 4.464e-07 | 4.96 | 2.173e-11 | 5.99 | 1.231e-11 | 6.00 |
| polynomial degree $k = 3$ | | | | | | | | | | | | |
| 20 | 3.423e-06 | - | 2.146e-06 | - | 1.059e-04 | - | 2.263e-04 | - | 3.747e-07 | - | 9.841e-07 | - |
| 40 | 2.141e-07 | 4.00 | 1.354e-07 | 3.99 | 4.712e-06 | 4.49 | 8.963e-06 | 4.66 | 6.304e-10 | 9.22 | 3.885e-10 | 11.31 |
| 80 | 1.338e-08 | 4.00 | 8.486e-09 | 4.00 | 3.412e-08 | 7.11 | 8.718e-08 | 6.68 | 2.673e-12 | 7.88 | 1.526e-12 | 7.99 |
| 160 | 8.363e-10 | 4.00 | 5.307e-10 | 4.00 | 2.004e-10 | 7.41 | 7.163e-10 | 6.93 | 1.056e-14 | 7.98 | 5.970e-15 | 8.00 |

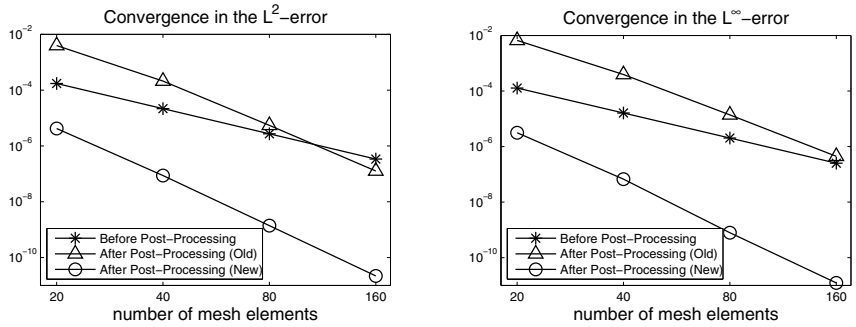
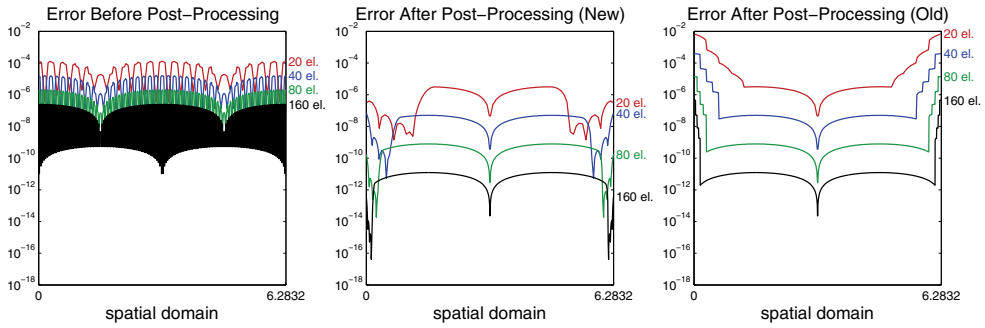



FIG. 6. *Comparison of the global accuracy of the old and new postprocessors for the $\mathcal{L}^2$-projection of the sine function for $k = 2$. An illustration of Table 1 for showing the improved convergence rate and error magnitude for the new one-sided postprocessor.*

of kernel nodes, we are able to improve on the errors over that of the DG solution (cf. section 3.2). Indeed, the new postprocessor obtains higher accuracy near the boundary. This is due to the fact that the latter uses $4k + 1$ kernel nodes instead of $2k + 1$ in that region.

The effect of using extra kernel nodes is isolated in Table 2 and Figures 8 and 9. These results were obtained by applying the fully one-sided kernel in the entire domain using $2k + 1$ kernel nodes for the old postprocessor and $4k + 1$ kernel nodes for the new version of the SIAC filter. In other words, the only difference between the two postprocessors is the number of kernel nodes. For this specific illustration, a periodic extension of $u_h$ was used. It is clear from the figures that the extra kernel nodes lead to a better convergence rate and better accuracy of $\mathcal{O}(h^{4k+1})$. The same phenomenon occurs at the boundary using the new postprocessor (cf. Figure 7(b)), which explains the improvement of the errors in that region.

It should be noted that the accuracy of $\mathcal{O}(h^{4k+1})$ cannot be expected in general. Instead, the theory in [6] predicts an error that is $\mathcal{O}(h^{4k+1}) + \mathcal{O}(h^{2k+2})$. We speculate that the meshes considered in this test case are sufficiently coarse so that the first




(a) (b) (c)

FIG. 7. *Comparison of the local accuracy of the old and new postprocessors for the $\mathcal{L}^2$-projection of the sine function for $k = 2$. The new filter improves both the smoothness and the accuracy in the entire domain, including the boundary.*

TABLE 2

*Comparison of the global accuracy of the fully one-sided kernel using $2k + 1$ kernel nodes for the old one-sided postprocessor and $4k + 1$ kernel nodes for the new one-sided postprocessor. The results are for the $\mathcal{L}^2$-projection of the sine function and clearly demonstrate that the use of extra kernel nodes for the new one-sided postprocessor leads to higher accuracy.*

| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k = 1$ | | | | | | | | | | | | |
| 20 | 6.510e-03 | - | 5.953e-03 | - | 4.500e-02 | - | 2.538e-02 | - | 3.717e-03 | - | 2.105e-03 | - |
| 40 | 1.629e-03 | 2.00 | 1.500e-03 | 1.99 | 5.700e-03 | 2.98 | 3.215e-03 | 2.98 | 1.185e-04 | 4.97 | 6.711e-05 | 4.97 |
| 80 | 4.074e-04 | 2.00 | 3.759e-04 | 2.00 | 7.148e-04 | 3.00 | 4.033e-04 | 3.00 | 3.722e-06 | 4.99 | 2.110e-06 | 4.99 |
| 160 | 1.019e-04 | 2.00 | 9.402e-05 | 2.00 | 8.942e-05 | 3.00 | 5.045e-05 | 3.00 | 1.166e-07 | 5.00 | 6.625e-08 | 4.99 |
| polynomial degree $k = 2$ | | | | | | | | | | | | |
| 20 | 1.729e-04 | - | 1.279e-04 | - | 1.026e-02 | - | 5.790e-03 | - | 9.516e-05 | - | 5.368e-05 | - |
| 40 | 2.163e-05 | 3.00 | 1.613e-05 | 2.99 | 3.286e-04 | 4.97 | 1.854e-04 | 4.96 | 1.929e-07 | 8.95 | 1.088e-07 | 8.95 |
| 80 | 2.704e-06 | 3.00 | 2.021e-06 | 3.00 | 1.033e-05 | 4.99 | 5.828e-06 | 4.99 | 3.815e-10 | 8.98 | 2.163e-10 | 8.97 |
| 160 | 3.381e-07 | 3.00 | 2.528e-07 | 3.00 | 3.233e-07 | 5.00 | 1.824e-07 | 5.00 | 7.607e-13 | 8.97 | 4.421e-13 | 8.93 |
| polynomial degree $k = 3$ | | | | | | | | | | | | |
| 20 | 3.423e-06 | - | 2.146e-06 | - | 2.584e-03 | - | 1.458e-03 | - | 2.818e-06 | - | 1.590e-06 | - |
| 40 | 2.141e-07 | 4.00 | 1.354e-07 | 3.99 | 2.090e-05 | 6.95 | 1.179e-05 | 6.95 | 3.622e-10 | 12.93 | 2.044e-10 | 12.93 |
| 80 | 1.338e-08 | 4.00 | 8.486e-09 | 4.00 | 1.647e-07 | 6.99 | 9.294e-08 | 6.99 | 4.468e-14 | 12.98 | 2.526e-14 | 12.98 |
| 160 | 8.363e-10 | 4.00 | 5.307e-10 | 4.00 | 1.290e-09 | 7.00 | 7.277e-10 | 7.00 | 5.507e-18 | 12.99 | 3.209e-18 | 12.94 |

error dominates the second. For finer meshes, it is likely that the second error will start to dominate, so that the error then becomes $\mathcal{O}(h^{2k+2})$. A similar effect will be encountered in the next section.

**4.2. Constant coefficients and periodic boundary conditions.** We now consider a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions,

$$\begin{aligned} u_t + u_x &= 0, \\ u(x,0) &= \sin(x) \end{aligned}$$

for all $x \in [0, 2\pi]$ and $t \geq 0$. As a consequence, the exact solution is the periodic translation of the sine function,

$$u(x,t) = \sin(x - t).$$

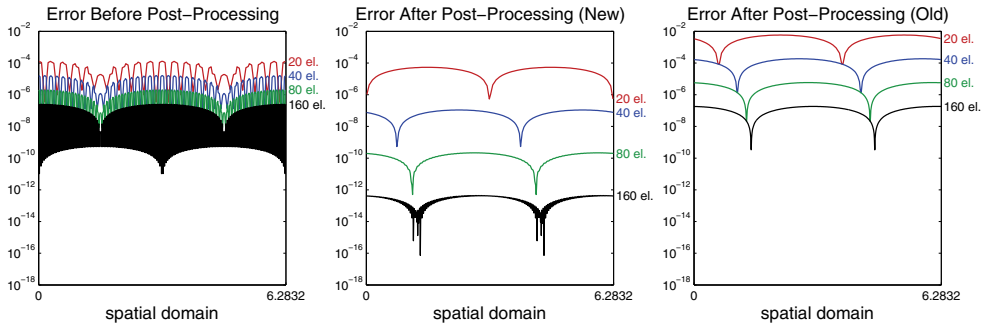



FIG. 8. *Comparison of the local accuracy of the fully one-sided kernel using* $2k+1$ *kernel nodes for the old one-sided postprocessor and* $4k+1$ *kernel nodes for the new one-sided postprocessor. The results are for the* $\mathcal{L}^2$*-projection of the sine function and clearly demonstrate that the use of extra kernel nodes for the new one-sided postprocessor leads to increased smoothness and higher accuracy.*

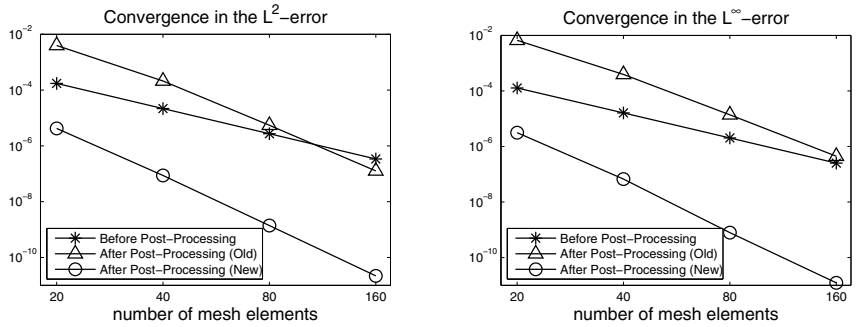
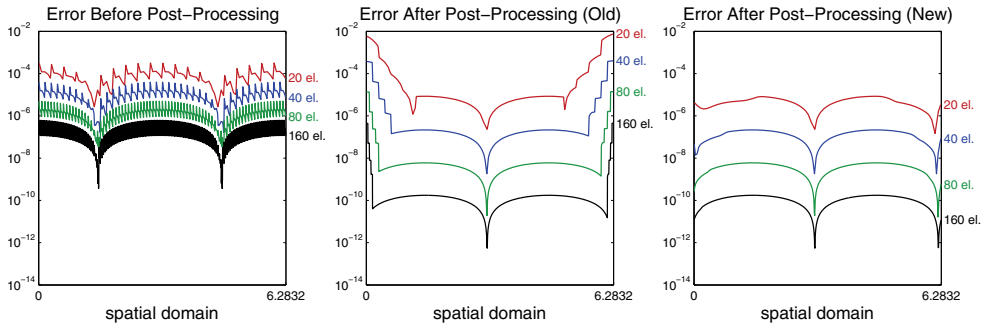



FIG. 9. *Comparison of the global accuracy of the fully one-sided kernel using both the old and new postprocessors for the* $\mathcal{L}^2$*-projection of the sine function for* $k = 2$*. An illustration of Table* 2 *for showing the improved convergence rate and error magnitude for the new one-sided postprocessor.*

For $t = 0$, this test case is equivalent to the one discussed in the previous section. Here, we consider the final time $t = 12.5$. This test case is more challenging than the previous one because $u_h$ now contains information of the physics of the PDE and the numerics of the DG method.

Comparing the plots in Figure 10, we can see that the behavior of the two postprocessors is similar to what was observed for the $\mathcal{L}^2$-projection in section 4.1. That is, we are able to obtain better errors than both the DG solution and the old one-sided postprocessor. In fact, the magnitude of the errors is improved throughout the *entire* domain when the new position-dependent postprocessor is applied to the DG approximation. Furthermore, the convergence rate is improved from $\mathcal{O}(h^{k+1})$ to at least $\mathcal{O}(h^{2k+1})$ (cf. Table 3 and Figure 11).

We again isolate the effect of using extra kernel nodes for this example in Table 4 and Figures 12 and 13 by applying the fully one-sided kernel throughout the entire domain. Here, a difference from the previous case is observed. The lines corresponding to the new position-dependent SIAC filter in the convergence rate plot (cf. Figure 13) are no longer straight, demonstrating that the convergence rate is no longer constant. In other words, as before, the new postprocessor shows a higher convergence rate




(a) (b) (c)

FIG. 10. *Comparison of the local accuracy of the old and new postprocessors for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions for $k = 2$. The new filter improves both the smoothness and the accuracy in the entire domain, including the boundary.*

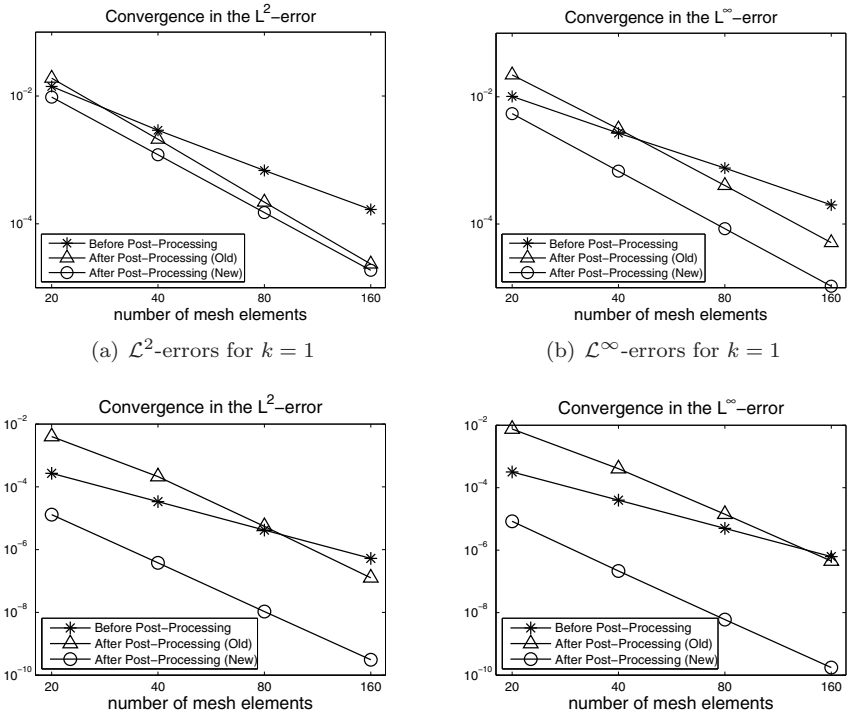
TABLE 3

*Comparison of the global accuracy of the old and new postprocessors for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions. The new postprocessor improves the convergence rate from $\mathcal{O}(h^{k+1})$ to $\mathcal{O}(h^{2k+1})$.*

| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k = 1$ | | | | | | | | | | | | |
| 20 | 1.410e-02 | - | 1.015e-02 | - | 1.894e-02 | - | 2.207e-02 | - | 9.596e-03 | - | 5.439e-03 | - |
| 40 | 2.907e-03 | 2.28 | 2.687e-03 | 1.92 | 2.105e-03 | 3.17 | 3.125e-03 | 2.82 | 1.201e-03 | 3.00 | 6.780e-04 | 3.00 |
| 80 | 6.814e-04 | 2.09 | 7.570e-04 | 1.83 | 2.184e-04 | 3.27 | 4.033e-04 | 2.95 | 1.497e-04 | 3.00 | 8.450e-05 | 3.00 |
| 160 | 1.674e-04 | 2.03 | 1.999e-04 | 1.92 | 2.344e-05 | 3.22 | 5.096e-05 | 2.98 | 1.868e-05 | 3.00 | 1.054e-05 | 3.00 |
| polynomial degree $k = 2$ | | | | | | | | | | | | |
| 20 | 2.683e-04 | - | 3.176e-04 | - | 4.003e-03 | - | 7.501e-03 | - | 1.301e-05 | - | 8.408e-06 | - |
| 40 | 3.352e-05 | 3.00 | 3.981e-05 | 3.00 | 2.108e-04 | 4.25 | 4.068e-04 | 4.20 | 3.767e-07 | 5.11 | 2.158e-07 | 5.28 |
| 80 | 4.190e-06 | 3.00 | 4.973e-06 | 3.00 | 5.464e-06 | 5.27 | 1.409e-05 | 4.85 | 1.056e-08 | 5.16 | 5.972e-09 | 5.18 |
| 160 | 5.238e-07 | 3.00 | 6.221e-07 | 3.00 | 1.254e-07 | 5.45 | 4.495e-07 | 4.97 | 3.090e-10 | 5.10 | 1.744e-10 | 5.10 |
| polynomial degree $k = 3$ | | | | | | | | | | | | |
| 20 | 5.176e-06 | - | 4.402e-06 | - | 1.304e-04 | - | 3.213e-04 | - | 3.757e-07 | - | 1.048e-06 | - |
| 40 | 3.236e-07 | 4.00 | 2.760e-07 | 4.00 | 4.712e-06 | 4.79 | 9.451e-06 | 5.09 | 6.634e-10 | 9.15 | 4.094e-10 | 11.32 |
| 80 | 2.023e-08 | 4.00 | 1.725e-08 | 4.00 | 3.406e-08 | 7.11 | 8.913e-08 | 6.73 | 2.957e-12 | 7.81 | 1.689e-12 | 7.92 |
| 160 | 1.264e-09 | 4.00 | 1.078e-09 | 4.00 | 1.999e-10 | 7.41 | 7.232e-10 | 6.95 | 1.287e-14 | 7.84 | 7.277e-15 | 7.86 |

than the old postprocessor, but only for the coarser meshes. This change in the convergence rate can also be observed for the new position-dependent postprocessor in Table 3 for $k = 3$. We speculate that this change occurs because the errors in the negative order norm of the DG solution begin to dominate, and these errors are of $\mathcal{O}(h^{2k+1})$.

**4.3. Constant coefficients and Dirichlet boundary conditions.** The previous section discussed a test case with periodic boundary conditions. Due to the periodicity, it is not necessary to use a one-sided approach near the boundary. The more favorable symmetric postprocessor could be applied by using a periodic extension of the DG solution. However, in most real-life applications, the boundary conditions are not periodic. For this reason, we revisit the test case of the previous section but




(a) $\mathcal{L}^2$-errors for $k = 1$ (b) $\mathcal{L}^\infty$-errors for $k = 1$

(c) $\mathcal{L}^2$-errors for $k = 2$ (d) $\mathcal{L}^\infty$-errors for $k = 2$

FIG. 11. *Comparison of the global accuracy of the old and new postprocessors for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions. An illustration of Table* 3 *for* $k = 1, 2$ *shows the improved convergence rate over the DG solution and the improved error magnitude compared to that of the old one-sided postprocessor.*

now use Dirichlet boundary conditions. That is,

$$\begin{aligned} u_t + u_x &= 0, \\ u(x,0) &= \sin(x), \\ u(0,t) &= \sin(-t) \end{aligned}$$

for all $x \in [0, 2\pi]$ and $t \geq 0$. As a consequence, the exact solution is still a periodic translation of the sine function,

$$u(x,t) = \sin(x - t).$$

Similar to the periodic case, we observe that the convergence rate is improved from $\mathcal{O}(h^{k+1})$ to better than $\mathcal{O}(h^{2k+1})$ (cf. Table 5). Furthermore, the smoothness and the accuracy are improved in the entire domain, including the boundary (cf. Figure 14).

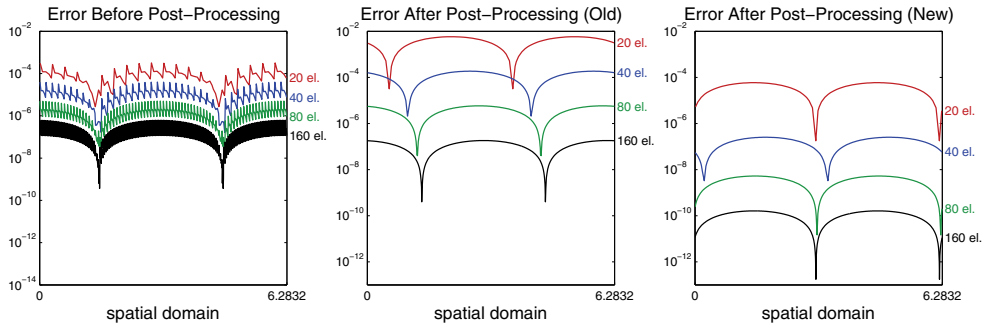
**4.4. Smoothly varying coefficients and periodic boundary conditions.** We now consider the DG solution to a one-dimensional linear hyperbolic equation with smoothly varying coefficients and periodic boundary conditions,

$$\begin{aligned} u_t + (a\,u)_x &= f, \\ a(x,t) &= 2 + \sin(x + t), \\ u(x,0) &= \sin(x) \end{aligned}$$

TABLE 4

*Comparison of the global accuracy of the fully one-sided kernel using both $2k+1$ kernel nodes for the old postprocessor and $4k+1$ kernel nodes for the new position-dependent SIAC filter for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions. As expected, the use of extra kernel nodes for the new filter leads to higher accuracy.*

| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k=1$ | | | | | | | | | | | | |
| 20 | 1.410e-02 | - | 1.015e-02 | - | 4.186e-02 | - | 2.361e-02 | - | 1.219e-02 | - | 6.868e-03 | - |
| 40 | 2.907e-03 | 2.28 | 2.687e-03 | 1.92 | 5.574e-03 | 2.91 | 3.144e-03 | 2.91 | 1.236e-03 | 3.30 | 6.979e-04 | 3.30 |
| 80 | 6.814e-04 | 2.09 | 7.570e-04 | 1.83 | 7.149e-04 | 2.96 | 4.033e-04 | 2.96 | 1.497e-04 | 3.05 | 8.446e-05 | 3.05 |
| 160 | 1.674e-04 | 2.03 | 1.999e-04 | 1.92 | 9.039e-05 | 2.98 | 5.100e-05 | 2.98 | 1.864e-05 | 3.01 | 1.052e-05 | 3.01 |
| polynomial degree $k=2$ | | | | | | | | | | | | |
| 20 | 2.683e-04 | - | 3.176e-04 | - | 1.027e-02 | - | 5.794e-03 | - | 1.045e-04 | - | 5.898e-05 | - |
| 40 | 3.352e-05 | 3.00 | 3.981e-05 | 3.00 | 3.287e-04 | 4.97 | 1.854e-04 | 4.97 | 4.490e-07 | 7.86 | 2.535e-07 | 7.86 |
| 80 | 4.190e-06 | 3.00 | 4.973e-06 | 3.00 | 1.033e-05 | 4.99 | 5.829e-06 | 4.99 | 9.340e-09 | 5.59 | 5.272e-09 | 5.59 |
| 160 | 5.238e-07 | 3.00 | 6.221e-07 | 3.00 | 3.233e-07 | 5.00 | 1.824e-07 | 5.00 | 2.875e-10 | 5.02 | 1.623e-10 | 5.02 |
| polynomial degree $k=3$ | | | | | | | | | | | | |
| 20 | 5.176e-06 | - | 4.402e-06 | - | 2.584e-03 | - | 1.457e-03 | - | 2.821e-06 | - | 1.591e-06 | - |
| 40 | 3.236e-07 | 4.00 | 2.760e-07 | 4.00 | 2.090e-05 | 6.95 | 1.179e-05 | 6.95 | 3.964e-10 | 12.80 | 2.236e-10 | 12.80 |
| 80 | 2.023e-08 | 4.00 | 1.725e-08 | 4.00 | 1.647e-07 | 6.99 | 9.294e-08 | 6.99 | 3.161e-13 | 10.29 | 1.785e-13 | 10.29 |
| 160 | 1.264e-09 | 4.00 | 1.078e-09 | 4.00 | 1.290e-09 | 7.00 | 7.277e-10 | 7.00 | 2.318e-15 | 7.09 | 1.308e-15 | 7.09 |




(a) (b) (c)

FIG. 12. *Comparison of the local accuracy of the fully one-sided kernel using $2k+1$ kernel nodes (old) and $4k+1$ kernel nodes (new) for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions for $k=2$. The pointwise errors show that the use of extra kernel nodes for the new filter leads to higher accuracy and smoothness than the previous one-sided filter.*

for all $x \in [0, 2\pi]$ and $t \geq 0$. The forcing term $f(x,t)$ is chosen such that the exact solution is the periodic translation of the sine function,

$$u(x,t) = \sin(x-t).$$

Because the coefficients are no longer constant, this linear test case forms a first step toward examining the effects of position-dependent SIAC filtering for nonlinear problems.

Similar to all of the previous test cases, we observe that the convergence rate is improved from from $\mathcal{O}(h^{k+1})$ to at least $\mathcal{O}(h^{2k+1})$ (cf. Table 6 and Figure 15). In Figure 15 we can see that the smoothness and the accuracy are improved in the entire domain, including the boundary.

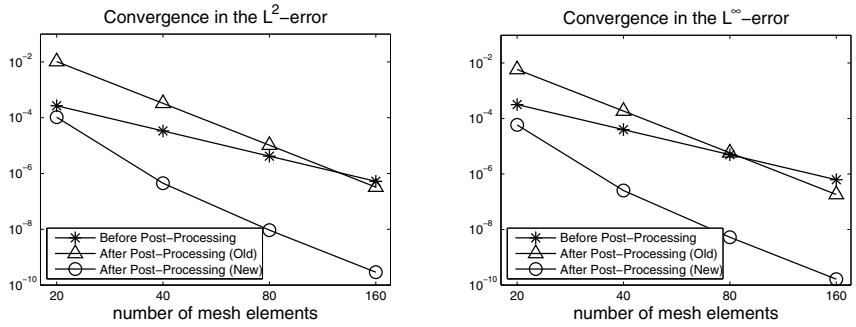



FIG. 13. *Comparison of the global accuracy of the fully one-sided kernel using* $2k+1$ *kernel nodes (old) and* $4k+1$ *kernel nodes (new) for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and periodic boundary conditions. An illustration of Table* 4 *for* $k=2$ *shows the effect of using extra kernel nodes.*

TABLE 5

*Comparison of the global accuracy of the old and new postprocessors for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and Dirichlet boundary conditions. The new postprocessor improves the convergence rate from* $\mathcal{O}(h^{k+1})$ *to* $\mathcal{O}(h^{2k+1})$.

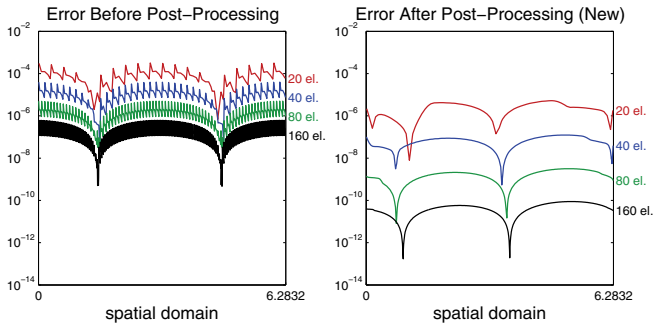
| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k=1$ | | | | | | | | | | | | |
| 20 | 1.100e-02 | - | 1.286e-02 | - | 1.633e-02 | - | 2.335e-02 | - | 3.365e-03 | - | 2.406e-03 | - |
| 40 | 2.685e-03 | 2.03 | 3.286e-03 | 1.97 | 1.757e-03 | 3.22 | 3.230e-03 | 2.85 | 4.138e-04 | 3.02 | 2.995e-04 | 3.01 |
| 80 | 6.669e-04 | 2.01 | 8.317e-04 | 1.98 | 1.659e-04 | 3.40 | 4.128e-04 | 2.97 | 5.133e-05 | 3.01 | 3.717e-05 | 3.01 |
| 160 | 1.664e-04 | 2.00 | 2.092e-04 | 1.99 | 1.547e-05 | 3.42 | 5.190e-05 | 2.99 | 6.391e-06 | 3.01 | 4.627e-06 | 3.01 |
| polynomial degree $k=2$ | | | | | | | | | | | | |
| 20 | 2.681e-04 | - | 3.171e-04 | - | 4.003e-03 | - | 7.500e-03 | - | 6.984e-06 | - | 5.229e-06 | - |
| 40 | 3.352e-05 | 3.00 | 3.978e-05 | 2.99 | 2.108e-04 | 4.25 | 4.068e-04 | 4.20 | 1.837e-07 | 5.25 | 1.238e-07 | 5.40 |
| 80 | 4.190e-06 | 3.00 | 4.972e-06 | 3.00 | 5.464e-06 | 5.27 | 1.409e-05 | 4.85 | 4.627e-09 | 5.31 | 3.163e-09 | 5.29 |
| 160 | 5.238e-07 | 3.00 | 6.20e-07 | 3.00 | 1.254e-07 | 5.45 | 4.494e-07 | 4.97 | 1.279e-10 | 5.18 | 8.832e-11 | 5.16 |
| polynomial degree $k=3$ | | | | | | | | | | | | |
| 20 | 5.176e-06 | - | 4.405e-06 | - | 1.304e-04 | - | 3.213e-04 | - | 3.751e-07 | - | 1.047e-06 | - |
| 40 | 3.236e-07 | 4.00 | 2.761e-07 | 4.00 | 4.712e-06 | 4.79 | 9.451e-06 | 5.09 | 6.387e-10 | 9.20 | 3.965e-10 | 11.37 |
| 80 | 2.023e-08 | 4.00 | 1.725e-08 | 4.00 | 3.406e-08 | 7.11 | 8.913e-08 | 6.73 | 2.747e-12 | 7.86 | 1.589e-12 | 7.96 |
| 160 | 1.264e-09 | 4.00 | 1.078e-09 | 4.00 | 1.999e-10 | 7.41 | 7.232e-10 | 6.95 | 1.118e-14 | 7.94 | 6.479e-15 | 7.94 |

**4.5. Discontinuous variable coefficients and periodic boundary conditions.** For all of the previous test cases, the exact solution was infinitely smooth. In section 3, we emphasized that our position-dependent SIAC filter can be applied near a shock, similar to the application near a boundary. To test this numerically, we consider a one-dimensional linear hyperbolic equation with discontinuous coefficients and periodic boundary conditions,

$$u_t + (a\,u)_x = 0,$$

$$a(x) = \begin{cases} \frac{1}{2}, & x \in [-\frac{1}{2}, \frac{1}{2}], \\ 1 & \text{else}, \end{cases}$$

$$u(0,t) = \begin{cases} -2\cos(4\pi x), & x \in [-\frac{1}{2}, \frac{1}{2}], \\ \cos(2\pi x) & \text{else} \end{cases}$$




(a) (b)

FIG. 14. *Comparison of the local accuracy of the new position-dependent postprocessor for the DG solution to a one-dimensional linear hyperbolic equation with constant coefficients and Dirichlet boundary conditions for $k = 2$. The plots demonstrate that new SIAC filter improves both the smoothness and the accuracy in the entire domain, including the boundary.*

TABLE 6

*Comparison of the global accuracy of the old and new postprocessors for the DG solution to a one-dimensional linear hyperbolic equation with smooth variable coefficients and periodic boundary conditions.*

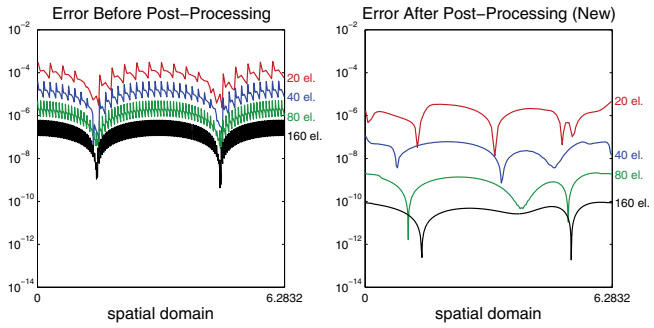
| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k = 1$ | | | | | | | | | | | | |
| 20 | 1.091e-02 | - | 1.462e-02 | - | 1.634e-02 | - | 2.466e-02 | - | 2.745e-03 | - | 2.947e-03 | - |
| 40 | 2.678e-03 | 2.03 | 3.526e-03 | 2.05 | 1.749e-03 | 3.22 | 3.380e-03 | 2.87 | 3.483e-04 | 2.98 | 2.766e-04 | 3.41 |
| 80 | 6.663e-04 | 2.01 | 8.616e-04 | 2.03 | 1.642e-04 | 3.41 | 4.306e-04 | 2.97 | 4.384e-05 | 2.99 | 2.985e-05 | 3.21 |
| 160 | 1.664e-04 | 2.00 | 2.130e-04 | 2.02 | 1.517e-05 | 3.44 | 5.398e-05 | 3.00 | 5.497e-06 | 3.00 | 3.626e-06 | 3.04 |
| polynomial degree $k = 2$ | | | | | | | | | | | | |
| 20 | 2.684e-04 | - | 3.312e-04 | - | 4.004e-03 | - | 7.504e-03 | - | 4.576e-06 | - | 4.668e-06 | - |
| 40 | 3.354e-05 | 3.00 | 4.066e-05 | 3.03 | 2.108e-04 | 4.25 | 4.068e-04 | 4.21 | 1.011e-07 | 5.50 | 1.176e-07 | 5.31 |
| 80 | 4.191e-06 | 3.00 | 5.027e-06 | 3.02 | 5.464e-06 | 5.27 | 1.409e-05 | 4.85 | 2.767e-09 | 5.19 | 2.147e-09 | 5.78 |
| 160 | 5.238e-07 | 3.00 | 6.255e-07 | 3.01 | 1.254e-07 | 5.45 | 4.494e-07 | 4.97 | 1.237e-10 | 4.48 | 9.124e-11 | 4.56 |
| polynomial degree $k = 3$ | | | | | | | | | | | | |
| 20 | 5.171e-06 | - | 4.411e-06 | - | 1.305e-04 | - | 3.213e-04 | - | 1.106e-05 | - | 3.912e-05 | - |
| 40 | 3.235e-07 | 4.00 | 2.763e-07 | 4.00 | 4.712e-06 | 4.79 | 9.451e-06 | 5.09 | 6.627e-10 | 14.03 | 1.119e-09 | 15.09 |
| 80 | 2.023e-08 | 4.00 | 1.725e-08 | 4.00 | 3.406e-08 | 7.11 | 8.913e-08 | 6.73 | 2.650e-12 | 7.97 | 1.532e-12 | 9.51 |
| 160 | 1.264e-09 | 4.00 | 1.079e-09 | 4.00 | 1.999e-10 | 7.41 | 7.232e-10 | 6.95 | 1.060e-14 | 7.97 | 7.131e-15 | 7.75 |

for all $x \in [-1, 1]$ and $t \geq 0$. The exact solution is given by

$$u(x,t) = \begin{cases} -2\cos\left(4\pi(x - \frac{1}{2}t)\right), & x \in [-\frac{1}{2}, \frac{1}{2}], \\ \cos\left(2\pi(x - t)\right) & \text{else}, \end{cases}$$

which has two stationary shocks [15].

The results for this test case are displayed in Table 7 and Figures 16 and 17. As is evident from Table 7 and Figures 16 and 17, the accuracy of the new postprocessor is better than that of the DG solution as long as the mesh is sufficiently fine. For the coarser meshes, some errors have a rather uncommon value. This is because, for those meshes, the kernel scale $H$ is chosen smaller than the mesh diameter $h$, as a consequence of the requirement in (15). This is one of the drawbacks of the new postprocessor: the kernel support is twice as large due to the extra kernel nodes (cf. (13)). This causes difficulties for the new postprocessor when the distance between




(a) (b)

FIG. 15. *Comparison of the local accuracy of the new postprocessor for the DG solution to a one-dimensional linear hyperbolic equation with smooth variable coefficients and periodic boundary conditions for $k = 2$. The new filter improves both the smoothness and the accuracy in the entire domain, including the boundary.*

TABLE 7

*Comparison of the global accuracy of the old and new postprocessors for the DG solution to a one-dimensional linear hyperbolic equation with discontinuous coefficients and periodic boundary conditions. There is a clear improvement in both the order of accuracy and the magnitude of the errors.*

| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k = 1$ | | | | | | | | | | | | |
| 20 | 1.207e+00 | - | 1.558e+00 | - | 1.153e+00 | - | 1.544e+00 | - | 1.204e+00 | - | 1.624e+00 | - |
| 40 | 2.716e-01 | 2.15 | 3.771e-01 | 2.05 | 2.539e-01 | 2.18 | 3.491e-01 | 2.14 | 2.744e-01 | 2.13 | 4.330e-01 | 1.91 |
| 80 | 3.827e-02 | 2.83 | 5.744e-02 | 2.71 | 3.635e-02 | 2.80 | 4.879e-02 | 2.84 | 3.750e-02 | 2.87 | 5.024e-02 | 3.11 |
| 160 | 5.201e-03 | 2.88 | 8.619e-03 | 2.74 | 4.705e-03 | 2.95 | 6.187e-03 | 2.98 | 4.753e-03 | 2.98 | 6.170e-03 | 3.03 |
| polynomial degree $k = 2$ | | | | | | | | | | | | |
| 20 | 3.645e-02 | - | 5.143e-02 | - | 6.808e+00 | - | 1.623e+01 | - | 5.709e-01 | - | 2.944e+00 | - |
| 40 | 2.052e-03 | 4.15 | 4.841e-03 | 3.41 | 1.672e-01 | 5.35 | 6.778e-01 | 4.58 | 1.249e-03 | 8.84 | 1.825e-03 | 10.66 |
| 80 | 2.173e-04 | 3.24 | 6.272e-04 | 2.95 | 6.027e-03 | 4.79 | 2.795e-02 | 4.60 | 4.164e-05 | 4.91 | 1.398e-04 | 3.71 |
| 160 | 2.682e-05 | 3.02 | 7.936e-05 | 2.98 | 8.414e-05 | 6.16 | 5.793e-04 | 5.59 | 1.178e-06 | 5.14 | 1.693e-06 | 6.37 |
| polynomial degree $k = 3$ | | | | | | | | | | | | |
| 20 | 1.085e-03 | - | 2.451e-03 | - | 3.579e+00 | - | 1.247e+01 | - | 2.270e-01 | - | 6.612e-01 | - |
| 40 | 6.602e-05 | 4.04 | 1.369e-04 | 4.16 | 1.865e-02 | 7.58 | 9.948e-02 | 6.97 | 2.640e-03 | 6.43 | 1.847e-02 | 5.16 |
| 80 | 4.132e-06 | 4.00 | 8.741e-06 | 3.97 | 6.502e-04 | 4.84 | 2.915e-03 | 5.09 | 5.205e-06 | 8.99 | 6.981e-05 | 8.05 |
| 160 | 2.584e-07 | 4.00 | 5.510e-07 | 3.99 | 2.623e-06 | 7.95 | 1.772e-05 | 7.36 | 4.669e-09 | 10.12 | 8.703e-08 | 9.65 |

two subsequent boundaries/shocks is smaller than the support of the kernel, which is scaled by $H$. This can lead to an inconvenient restriction on the kernel scale, which becomes more restrictive as the number of elements between the two subsequent boundaries/shocks becomes smaller.

Nevertheless, for $k = 2$ and $k = 3$, the magnitude of the errors is much smaller for the new postprocessor than for the old one. For $k = 1$, this is not the case: the errors are slightly worse. To understand this, we compare Figure 11 and Figure 17. For the linear smooth periodic test case, the improvement over the old postprocessor is stronger for $k = 2$ (cf., e.g., Figure 11(c)) than for $k = 1$ (cf., e.g., Figure 11(a)). We speculate that this is due to the fact that *more* extra kernel nodes are used for $k = 2$ than for $k = 1$. A similar, yet stronger, effect is observed for the discontinuous case in Figure 17: the differences between the old and new postprocessors are quite small for $k = 1$ (cf., e.g., Figure 17(a)). Improvement of this issue is left for future research.

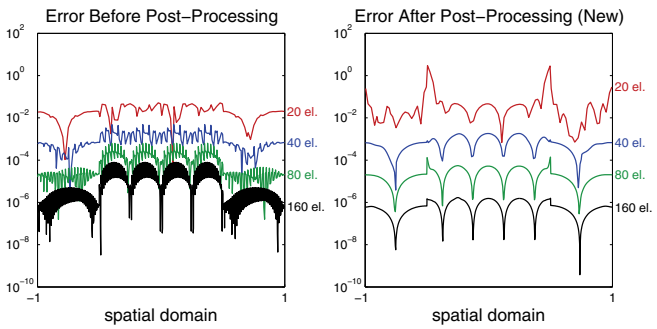
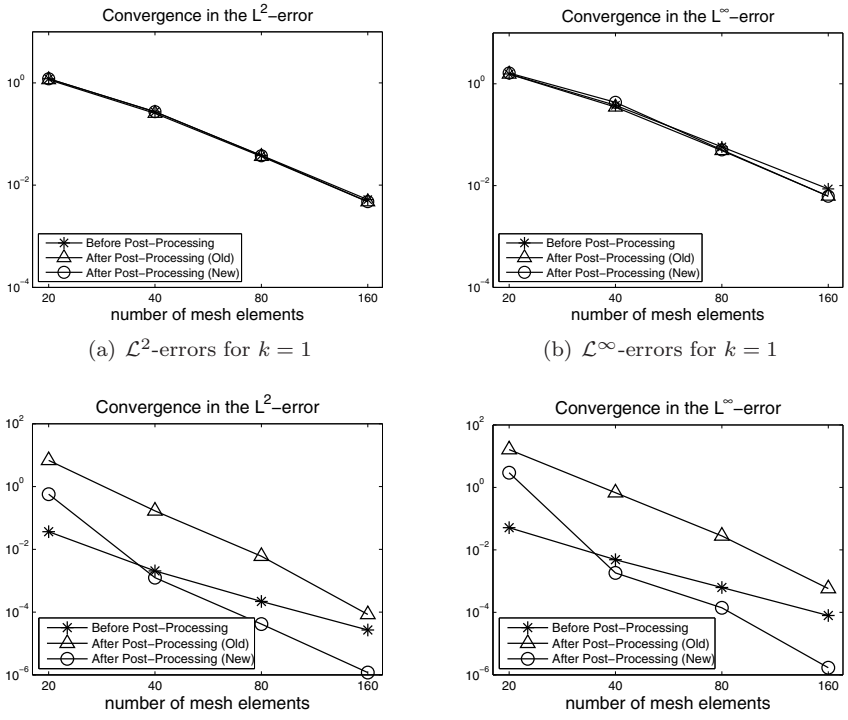



FIG. 16. *Comparison of the local accuracy of the new postprocessor and the DG solution for a one-dimensional linear hyperbolic equation with discontinuous coefficients and periodic boundary conditions for $k = 2$. The improved smoothness throughout the domain is evident for a sufficiently fine mesh.*




(a) $\mathcal{L}^2$-errors for $k = 1$ (b) $\mathcal{L}^\infty$-errors for $k = 1$

(c) $\mathcal{L}^2$-errors for $k = 2$ (d) $\mathcal{L}^\infty$-errors for $k = 2$

FIG. 17. *Comparison of the global accuracy of the old and new postprocessors for the DG solution to a one-dimensional linear hyperbolic equation with discontinuous coefficients and periodic boundary conditions. In this plot, we illustrate the convergence rate of the errors given in Table 7 for $k = 1, 2$. We can see that a sufficiently fine mesh is required for improved errors and an improved convergence rate.*

**4.6. Two-dimensional system.** In the previous sections, we considered only one-dimensional problems. However, higher-dimensional fields can also be filtered by applying the one-dimensional kernel in a tensor-product manner (cf., e.g., [16, p. 827]). In this section, we apply such a strategy to a two-dimensional system with periodic boundary conditions:

$$\begin{bmatrix} u \\ v \end{bmatrix}_t + \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_x + \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_y = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

for all $x \in [-1, 1]$ and $t \geq 0$. The initial condition is given by

$$u(x, y, 0) = \frac{1}{2\sqrt{2}}(\sin(x + y) - \cos(x + y)),$$
$$v(x, y, 0) = \frac{1}{2\sqrt{2}}\big((\sqrt{2} - 1)\sin(x + y) + (1 + \sqrt{2})\cos(x + y)\big).$$

The results for this test case are displayed in Table 8. Similar to the linear one-dimensional problems, we observe that the convergence rate is improved from $\mathcal{O}(h^{k+1})$ to at least $\mathcal{O}(h^{2k+1})$. For $k = 1$, the magnitude of the errors in the $\mathcal{L}^\infty$-norm is slightly worse for the new postprocessor than for the unfiltered case but

TABLE 8

*Comparison of the global accuracy of the old and new postprocessors for the DG solution to a two-dimensional system. There is a clear improvement in both the order of accuracy and the magnitude of the errors.*

$u$-component

| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k = 1$ | | | | | | | | | | | | |
| 20 | 1.218e-01 | - | 3.939e-02 | - | 1.155e-01 | - | 2.770e-02 | - | 2.726e-02 | - | 2.726e-02 | - |
| 40 | 1.772e-02 | 2.78 | 7.119e-03 | 2.47 | 1.549e-02 | 2.90 | 4.578e-03 | 2.60 | 3.482e-03 | 2.97 | 3.482e-03 | 2.97 |
| 80 | 2.945e-03 | 2.59 | 1.382e-03 | 2.36 | 1.960e-03 | 2.98 | 6.331e-04 | 2.85 | 4.389e-04 | 2.99 | 4.389e-04 | 2.99 |
| polynomial degree $k = 2$ | | | | | | | | | | | | |
| 20 | 1.579e-03 | - | 1.244e-03 | - | 1.959e-02 | - | 1.768e-02 | - | 1.035e-04 | - | 1.035e-04 | - |
| 40 | 1.946e-04 | 3.02 | 1.618e-04 | 2.94 | 4.288e-04 | 5.51 | 6.003e-04 | 4.88 | 2.293e-06 | 5.50 | 2.293e-06 | 5.50 |
| 80 | 2.436e-05 | 3.00 | 2.045e-05 | 2.98 | 9.463e-06 | 5.50 | 1.763e-05 | 5.09 | 7.031e-08 | 5.03 | 7.031e-08 | 5.03 |
| polynomial degree $k = 3$ | | | | | | | | | | | | |
| 20 | 7.868e-05 | - | 5.576e-05 | - | 1.997e-03 | - | 1.790e-03 | - | 1.927e-06 | - | 1.927e-06 | - |
| 40 | 4.982e-06 | 3.98 | 3.297e-06 | 4.08 | 1.099e-05 | 7.51 | 1.545e-05 | 6.86 | 1.688e-09 | 10.16 | 1.688e-09 | 10.16 |
| 80 | 3.109e-07 | 4.00 | 1.971e-07 | 4.06 | 6.066e-08 | 7.50 | 1.168e-07 | 7.05 | 1.158e-11 | 7.19 | 1.158e-11 | 7.19 |

$v$-component

| | Before postprocessing | | | | After postprocessing (old) | | | | After postprocessing (new) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mesh | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order | $L^2$-error | order | $L^\infty$-error | order |
| polynomial degree $k = 1$ | | | | | | | | | | | | |
| 20 | 1.427e-01 | - | 3.893e-02 | - | 1.358e-01 | - | 3.645e-02 | - | 3.211e-02 | - | 3.211e-02 | - |
| 40 | 2.036e-02 | 2.81 | 6.132e-03 | 2.67 | 1.807e-02 | 2.91 | 5.421e-03 | 2.75 | 4.065e-03 | 2.98 | 4.065e-03 | 2.98 |
| 80 | 3.312e-03 | 2.62 | 1.631e-03 | 1.91 | 2.275e-03 | 2.99 | 7.226e-04 | 2.91 | 5.093e-04 | 3.00 | 5.093e-04 | 3.00 |
| polynomial degree $k = 2$ | | | | | | | | | | | | |
| 20 | 2.218e-03 | - | 2.453e-03 | - | 2.161e-02 | - | 1.434e-02 | - | 1.184e-04 | - | 1.184e-04 | - |
| 40 | 2.719e-04 | 3.03 | 3.075e-04 | 3.00 | 4.895e-04 | 5.46 | 6.639e-04 | 4.43 | 2.629e-06 | 5.49 | 2.629e-06 | 5.49 |
| 80 | 3.388e-05 | 3.00 | 3.839e-05 | 3.00 | 1.090e-05 | 5.49 | 2.177e-05 | 4.93 | 8.086e-08 | 5.02 | 8.086e-08 | 5.02 |
| polynomial degree $k = 3$ | | | | | | | | | | | | |
| 20 | 1.144e-04 | - | 1.259e-04 | - | 2.210e-03 | - | 1.405e-03 | - | 1.860e-06 | - | 1.860e-06 | - |
| 40 | 7.494e-06 | 3.93 | 8.209e-06 | 3.94 | 1.251e-05 | 7.46 | 1.604e-05 | 6.45 | 2.036e-09 | 9.84 | 2.036e-09 | 9.84 |
| 80 | 4.754e-07 | 3.98 | 5.226e-07 | 3.97 | 6.984e-08 | 7.48 | 1.398e-07 | 6.84 | 1.371e-11 | 7.21 | 1.371e-11 | 7.21 |

still improved over the results obtained for the old postprocessor. In all other cases, the position-dependent postprocessor improves the DG errors. In other words, the results for this two-dimensional problem are similar to those for the previous (linear) one-dimensional cases.

**5. Summary discussion.** Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering is a promising tool that has been recast to allow for post-processing near a boundary or solution discontinuity. This is done through a combination of a shift function, $\lambda$, using more kernel nodes (or B-splines), and writing this in the context of a convex combination of filter types. This convex combination of filters allows for easy transitioning from one-sided postprocessing that uses an extra $2k$ kernel nodes to handle domain boundaries and discontinuities in the solution to symmetric postprocessing for handling smooth interior solutions. The use of extra kernel nodes at the boundary increases the accuracy of the postprocessor to that of the symmetric postprocessor. This was verified by numerical results that demonstrate that the magnitude of the errors using the modified postprocessor is roughly the same as the errors for the symmetric postprocessor itself, regardless of boundary conditions.

REFERENCES

[1] D. H. Bailey, Y. Hida, X. S. Li, and B. Thompson, *ARPREC: An Arbitrary Precision Computation Package*, Tech. report 53651, Lawrence Berkeley National Laboratory, Berkeley, CA, 2002.

[2] J. H. Bramble and A. H. Schatz, *Higher order local accuracy by averaging in the finite element method*, Math. Comp., 31 (1977), pp. 94–111.

[3] W. Cai, D. Gottlieb, and C.-W. Shu, *On one-sided filters for spectral Fourier approximations of discontinuous functions*, SIAM J. Numer. Anal., 29 (1992), pp. 905–916.

[4] B. Cockburn, S. Hou, and C.-W. Shu, *The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws* IV: *The multidimensional case*, Math. Comp., 54 (1990), pp. 545–581.

[5] B. Cockburn, S.-Y. Lin, and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws* III: *One dimensional systems*, J. Comput. Phys., 84 (1989), pp. 90–113.

[6] B. Cockburn, M. Luskin, C.-W. Shu, and E. Suli, *Enhanced accuracy by post-processing for finite element methods for hyperbolic equations*, Math. Comp., 72 (2003), pp. 577–606.

[7] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws* II: *General framework*, Math. Comp., 52 (1989), pp. 411–435.

[8] B. Cockburn and C.-W. Shu, *The Runge-Kutta local projection $P^1$-discontinuous-Galerkin finite element method for scalar conservation laws*, RAIRO Modél. Math. Anal. Numér., 25 (1991), pp. 337–361.

[9] B. Cockburn and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws* V: *Multidimensional systems*, J. Comput. Phys., 141 (1998), pp. 199–224.

[10] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, J. Sci. Comput., 16 (2001), pp. 173–261.

[11] D. Gottlieb and C.-W. Shu, *On the Gibbs phenomenon* I: *Recovering exponential accuracy from the Fourier partial sum of a nonperiodic analytical function*, J. Comput. Appl. Math., 43 (1992), pp. 81–98.

[12] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Math. Comp., 67 (1998), pp. 73–85.

[13] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.

[14] M. S. Mock and P. D. Lax, *The computation of discontinuous solutions of linear hyperbolic equations*, Comm. Pure Appl. Math., 18 (1978), pp. 423–430.

[15] J. K. Ryan and C.-W. Shu, *On a one-sided post-processing technique for the discontinuous Galerkin methods*, Methods Appl. Anal., 10 (2003), pp. 295–307.

[16] J. Ryan, C.-W. Shu, and H. Atkins, *Extension of a postprocessing technique for the discontinuous Galerkin method for hyperbolic equations with application to an aeroacoustic problem*, SIAM J. Sci. Comput., 26 (2005), pp. 821–843.
[17] L. L. Schumaker, *Spline Functions: Basic Theory*, John Wiley & Sons, New York, 1981.
[18] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.
[19] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.
[20] M. Steffen, S. Curtis, R. M. Kirby, and J. K. Ryan, *Investigation of smoothness enhancing accuracy-conserving filters for improving streamline integration through discontinuous fields*, IEEE Trans. Visualization and Computer Graphics, (2007).
[21] V. Thomée, *High order local approximations to derivatives in the finite element method*, Math. Comp., 31 (1977), pp. 652–660.
[22] P. van Slingerland, J. K. Ryan, and C. Vuik, *Smoothness-Increasing Accuracy-Conserving Spline Filters Applied to Streamline Visualisation of DG Approximations*, Tech. report 09-06, Delft University of Technology, Delft, The Netherlands, 2009.