One-Sided Position-Dependent Smoothness-Increasing Accuracy-Conserving (SIAC) Filtering Over Uniform and Non-Uniform Meshes

Jennifer K. Ryan[§][¶][**] Xiaozhou Li[†][¶] Robert M. Kirby[‡][||]
Kees Vuik[†]

October 31, 2014

Abstract

In this paper, we introduce a new position-dependent Smoothness-Increasing Accuracy-Conserving (SIAC) filter that retains the benefits of position dependence as proposed in [20] while ameliorating some of its shortcomings. As in the previous position-dependent filter, our new filter can be applied near domain boundaries, near a discontinuity in the solution, or at the interface of different mesh sizes; and as before, in general, it numerically enhances the accuracy and increases the smoothness of approximations obtained using the discontinuous Galerkin (dG) method. However, the previously proposed position-dependent one-sided filter had two significant disadvantages: (1) increased computational cost (in terms of function evaluations), brought about by the use of $4k+1$ central B-splines near a boundary (leading to increased kernel support) and (2) increased numerical conditioning issues that necessitated the use of quadruple precision for polynomial degrees of $k \geq 3$ for the reported accuracy benefits to be realizable numerically. Our new filter addresses both of these issues — maintaining the same support size and with similar function evaluation characteristics as the symmetric filter in a way that has better numerical conditioning — making it, unlike its predecessor, amenable for GPU computing. Our new filter was conceived by revisiting the original error analysis for superconvergence of SIAC filters and by examining the role of the B-splines and their weights in the SIAC filtering kernel. We demonstrate, in the uniform mesh case, that our new filter is globally superconvergent for

[†]Delft Institute of Applied Mathematics, Delft University of Technology, 2628 CD Delft, The Netherlands.({X.Li-2,C.Vuik}@tudelft.nl)

[‡]School of Computing, University of Utah, Salt Lake City, Utah, USA. (kirby@cs.utah.edu)

[§]School of Mathematics, University of East Anglia, Norwich NR4 7TJ, United Kingdom. (Jennifer.Ryan@uea.ac.uk)

[¶]Supported by the Air Force Office of Scientific Research (AFOSR), Air Force Material Command, USAF, under grant number FA8655-13-1-3017.

[||]Supported by the Air Force Office of Scientific Research (AFOSR), Computational Mathematics Program (Program Manager: Dr. Fariba Fahroo), under grant number FA9550-08-1-0156.

[**]Corresponding author.

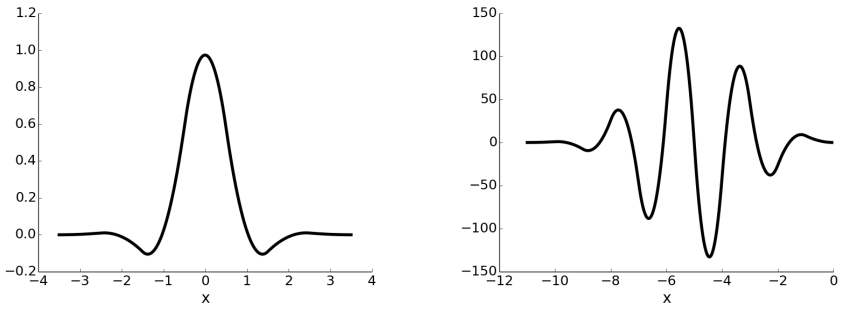
$k = 1$ and superconvergent in the interior (e.g., region excluding the boundary) for $k \geq 2$. Furthermore, we present the first theoretical proof of superconvergence for postprocessing over smoothly varying meshes, and explain the accuracy-order conserving nature of this new filter when applied to certain non-uniform meshes cases. We provide numerical examples supporting our theoretical results and demonstrating that our new filter, in general, enhances the smoothness and accuracy of the solution. Numerical results are presented for solutions of both linear and nonlinear equation solved on both uniform and non-uniform one- and two-dimensional meshes.

Index terms— discontinuous Galerkin method, post-processing, SIAC filtering, superconvergence, uniform meshes, smoothly-varying meshes, non-uniform meshes

1 Introduction

Computational considerations are always a concern when dealing with the implementation of numerical methods that claim to have practical (engineering) value. The focus of this paper is the formerly introduced Smoothness-Increasing Accuracy-Conserving (SIAC) class of filters, a class of filters that exhibit superconvergence behavior when applied to discontinuous Galerkin (dG) solutions. Although the previously proposed position-dependent filter (which we will, henceforth, call the SRV filter) introduced in [20] met its stated goals of demonstrating superconvergence, it contained two deficiencies which often made it impractical for implementation and usage within engineering scenarios. The first deficiency of the SRV filter was its reliance on $4k + 1$ central B-splines, which increased both the width of the stencil generated and increased the computational cost (in terms of functions evaluations) a disproportionate amount compared to the symmetric SIAC filter. The second deficiency is one of numerical conditioning: the SRV filter requires the use of quadruple precision to obtain consistent and meaningful results, which makes it unsuitable for practical CPU-based computations and for GPU computing. In this paper, we introduce a position-dependent SIAC filter that, like the SRV filter, allows for one-sided post-processing to be used near boundaries and solution discontinuities and which exhibits superconvergent behavior; however, our new filter addresses the two stated deficiencies: it has a smaller spatial support with a reduced number of function evaluations, and it does not require extended precision for error reduction to be realized.

To give context to what we will propose, let us review how we arrived at the currently available and used one-sided filter given in [20]. The SIAC filter has its roots in the finite element superconvergence extraction technique for elliptic equations proposed by Bramble and Schatz [2], Mock and Lax [18], and Thomée [19]. The linear hyperbolic system counterpart for discontinuous Galerkin (dG) methods was introduced by Cockburn, Luskin, Shu, and Süli [6]. The post-processing technique can enhance the accuracy order of dG approximations from $k+1$ to $2k+1$ in the L^2-norm. This symmetric post-processor uses $2k + 1$ central B-splines of order $k + 1$. However, a limitation of this symmetric post-processor was that it required a symmetric amount of information around the location being post-processed. To overcome this problem, Ryan and Shu [14] used the same ideas in [6] to develop a one-sided post-processor that could be applied near boundaries and discontinuities in the exact solution. However, their results were not very satisfactory as the errors had a stair-stepping-type structure, and the errors themselves were not reduced when the post-processor was applied to some dG solutions over coarse meshes. Later, van Slingerland, Ryan and Vuik [20] recast this formulation as a position-dependent SIAC filter by introducing a smooth shift function $\lambda(\bar{x})$ that aided in redefining the filter nodes and helped to ease the errors from the stair-stepping-type structure. In an attempt to reduce the errors, the authors doubled to $4k+1$ the number of central B-splines used in the filter when near a boundary. Further, they introduced a convex function that allowed for a smooth transition between boundary and symmetric regions.



(a) Symmetric filter (b) Boundary filter

Figure 1.1: Comparison of (a) the symmetric filter centered around $x = 0$ when the kernel is applied in the domain interior and (b) the position-dependent filter at the boundary (represented by $x = 0$) before convolution with a quadratic approximation. Notice that the boundary filter requires a larger spatial support, the amplitude is significantly larger in magnitude and the filter does not emphasize the point $x = 0$, which is the point being post-processed.

The results obtained with this strategy were good for linear hyperbolic equations over uniform meshes, but new challenges arose. Issues were manifest when the position-dependent filter was applied to equations whose solution lacked the (high) degree of regularity required for valid post-processing. In some cases, this filter applied to certain dG solutions gave worse results than when the original one-sided filter which used only $2k + 1$ central B-splines [14] was used. Furthermore, it was observed that in order for the superconvergence properties expressed in [20] to be fully realized, extended precision (beyond traditional double precision) had to be used. Lastly, the addition of more B-splines did not come without cost. Figure 1.1 shows the difference between the symmetric filter, which was introduced in [2, 6] and is applied in the domain interior, and the SRV filter when applied to the left boundary. The solution being filtered is at $x = 0$, and the filter extends into the domain. Upon examination, one sees that the position-dependent filter has a larger filter support and, by necessity, is not symmetric near the boundary. The vast discrepancy in spatial extent is due to the number of B-splines used: $2k + 1$ in the symmetric case versus $4k + 1$ in the one-sided case. The practical implication of this discrepancy is two-fold: (1) filtering at the boundary with the $4k + 1$ filter is noticeably more costly (in terms of function evaluations) than filtering in the interior with the symmetric filter; and (2) the spatial extent of the one-sided filter forces one to use the one-sided filter over a larger area/volume before being able to transition over to the symmetric filter.

Recall that the SRV filter added *two* new features when attempting to improve the original Ryan and Shu one-sided filter: they added position-dependence to the filter *and* increased the number of B-splines used. In attempting to overcome the deficiencies of the filter in [20], we reverted to the $2k+1$ B-spline approach as in [14] but added position-dependence. Although going back to $2k+1$ B-splines does make the filter less costly in the function evaluation sense, unfortunately, this approach did not lead to a filter that reduced the errors in the way we had hoped (i.e. at least error-conserving, if not also superconvergent). We were forced to reconsider altogether how superconvergence is obtained in the SIAC filter context; we outline a sketch of our thinking chronologically in what follows.

To conceive of a new one-sided filter containing all the benefits mentioned above with none of the deficiencies, we had to harken back to the fundamentals of SIAC filtering. We not only examined the filter itself (e.g., its construction, etc.), but also the error analysis in [6, 8]. From the analysis in [6, 8] we can conclude that the main source of the aforementioned conditioning issue (expressed in terms of the necessity for increased precision) is the constant term found in the expression for the error, which relies on the B-spline coefficients $c_\gamma^{(2k+1,\ell)}$ in the SIAC filtering kernel

$$\sum_\gamma |c_\gamma^{(2k+1,\ell)}|.$$

This quantity increases when the number of B-splines is increased. Further, the condition number of the system used to calculate $c_\gamma^{(2k+1,\ell)}$ becomes quite large, on the order of 10^{24} for $\mathbb{P}^4$, increasing the possibility for round-off errors and thus requiring higher levels of precision and increasing inefficiency. As mentioned before, we attempted to still use $2k+1$ position-dependent central B-splines, but this approach did not lead to error reduction. Indeed, the constant in the error term remains quite large at the boundaries. To reduce the error term, we concluded that one needed to add to the set of central B-splines the following: one *non-central* B-spline near the boundary. This general B-spline allows the filter to maintain the same spatial support throughout the domain, including boundary regions; it provides only a slight increase in computational cost as there are now $2k+2$ B-splines to evaluate as part of the filter; and possibly most importantly, it allows for error reduction. We note that our modifications to the previous filter (e.g., going beyond central B-splines) do come with a compromise: we must relax the assumption of obtaining superconvergence. Instead, we merely require a reduction in error and a smoother solution. This new filter remains globally superconvergent for $k=1$.

The new contributions of this paper are:

- A new one-sided position-dependent SIAC filter that allows filtering up to boundaries and that ameliorates the two principle deficiencies identified in the previous $4k+1$ one-sided position-dependent filter, which we refer to as the SRV filter throughout this manuscript;

- Examination and documentation of the reasoning concerning the constant term in the error analysis that led to the proposed work;

- Demonstration that for the linear polynomial case the filtered approximation is always superconvergent for a uniform mesh; and

- Application of the scaled new filter to both smoothly-varying and non-uniform (random) meshes. In the smoothly-varying case, we prove and demonstrate that we obtain superconvergence. For the general non-unform case, we still observe significant improvement in the smoothness and an error reduction over the original dG solution, although full superconvergence is not always achieved. We show however that we remain accuracy-order conserving.

These important results are presented as follows: first, as we present the SIAC filter in the context of discontinuous Galerkin approximations, we review the important properties of the dG method and the position-dependent SIAC filter in Section 2. In Section 3, we introduce the newly proposed filter and further establish some theoretical error estimates for the uniform and non-uniform (smoothly-varying) cases. We present the numerical results over uniform and non-uniform one-dimensional mesh structures in Section 4 and two-dimensional quadrilateral mesh structures in Section 5. Finally, conclusions are given in Section 6.

2 Background

In this section, we present the relevant background for understanding how to improve the Smoothness-Increasing Accuracy-Conserving filter, which includes the important properties of the discontinuous Galerkin (dG) method that make the application of SIAC filtering attractive as well as the building blocks of SIAC filtering – B-splines and the symmetric filter.

2.1 Important Properties of Discontinuous Galerkin Methods

We frame the discussion of the properties of dG methods in the context of a one-dimensional problem as the ideas easily extend to multiple dimensions. Further details about the discontinuous Galerkin method can be found in [3, 4].

Consider a one-dimensional hyperbolic equation such as

$$u_t + a_1 u_x + a_0 u = 0, \qquad x \in \Omega = [x_L, x_R] \tag{2.1}$$

$$u(x,0) = u_0(x). \tag{2.2}$$

To obtain a dG approximation, we first decompose Ω as $\Omega = \bigcup_{j=1}^{N} I_j$ where $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] = [x_j - \frac{1}{2}\Delta x_j, x_j + \frac{1}{2}\Delta x_j]$. Then Equation (2.1) is multiplied by a test function and integrated by parts. The test function is chosen from the same function space as the trial functions, a piecewise polynomial basis. The approximation can then be written as

$$u_h(x,t) = \sum_{\ell=0}^{k} u_j^{(\ell)}(t)\varphi_j^{(\ell)}(x), \quad \text{for } x \in I_j.$$

Herein, we choose the basis functions $\varphi_j^{(\ell)}(x) = P^{(\ell)}(2(x-x_j)/\Delta x_j)$, where $P^{(\ell)}$ is the Legendre polynomial of degree ℓ over $[-1,1]$. For simplicity, throughout this paper we represent polynomials of degree less than or equal to ℓ by $\mathbb{P}^\ell$.

In order to investigate the superconvergence property of the dG solution, it is important to look at the usual convergence rate of the dG method. By estimating the error of the dG solution, we obtain $u - u_h \sim \mathcal{O}(h^{k+1})$ in the L^2-norm for sufficiently smooth initial data u_0 [3]:

$$\|u-u_h\|_0 \le C\, h^{k+1}\|u_0\|_{H^{k+2}},$$

where h is the measure of the elements, $h = \Delta x$ for a uniform mesh and $h = \max_j \Delta x_j$ for non-uniform meshes. Another useful property is the superconvergence of the dG solution in the negative-order norm [6], where we have

$$\|\partial_h^\alpha(u-u_h)\|_{-\ell,\Omega} \le C\, h^{2k+1}\, \|\partial_h^\alpha u_0\|_{k+1,\Omega}$$

for linear hyperbolic equations. This expression represents why accuracy enhancement through post-processing is possible. Unfortunately, this superconvergence property does not hold for non-uniform meshes when $\alpha \ge 1$, which makes extracting the superconvergence over non-uniform meshes challenging. However, we prove that, for certain non-uniform meshes containing smoothness in their construction (i.e. smoothly-varying meshes), the accuracy enhancement through the SIAC filtering is still possible.

2.2 A Review of B-splines

As the SIAC filter relies heavily on B-splines, here we review the definition of B-splines given by de Boor in [1] as well as central B-splines.

Definition 2.1 (B-spline).

Let $\mathbf{t} := (t_j)$ *be a nondecreasing sequence of real numbers that create a so-called knot sequence. The* j*th* **B-spline** *of order* ℓ *for the knot sequence* $\mathbf{t}$ *is denoted by* $B_{j,\ell,\mathbf{t}}$ *and is defined, for* $\ell = 1$*, by the rule*

$$B_{j,1,\mathbf{t}}(x) = \begin{cases} 1, & t_j \le x < t_{j+1}; \\ 0, & \textit{otherwise.} \end{cases}$$

In particular, $t_j = t_{j+1}$ *leads to* $B_{j,1,\mathbf{t}} = 0$*. For* $\ell > 1$*,*

$$B_{j,\ell,\mathbf{t}}(x) = \omega_{j,l,\mathbf{t}} B_{j,\ell-1,\mathbf{t}} + (1-\omega_{j+1,\ell,\mathbf{t}}) B_{j+1,\ell-1,\mathbf{t}}, \tag{2.3}$$

with

$$\omega_{j,\ell,\mathbf{t}}(x) = \frac{x-t_j}{t_{j+\ell-1}-t_j}.$$

This notation will be used to create a new kernel near the boundaries.

The original symmetric filter [6, 15] relied on central B-splines of order ℓ whose knot sequence was uniformly spaced and symmetrically distributed $\mathbf{t} = -\frac{\ell}{2}, -\frac{\ell-2}{2}, \cdots, \frac{\ell-2}{2}, \frac{\ell}{2}.$, yielding the following recurrence relation for central B-splines:

$$\begin{aligned} \psi^{(1)}(x) &= \chi_{[-1/2,1/2]}(x), \\ \psi^{(\ell+1)}(x) &= (\psi^{(1)} \star \psi^{(\ell)})(x) = \frac{(\frac{\ell+1}{2}+x)\psi^{(\ell)}(x+\frac{1}{2}) + (\frac{\ell+1}{2}-x)\psi^{(\ell)}(x-\frac{1}{2})}{\ell}, \quad \ell \ge 1. \end{aligned} \tag{2.4}$$

For the purposes of this paper, it is convenient to relate the recurrence relation for central B-splines to the definition of general B-splines given in Definition 2.1. Relating the recurrence relation to the definition can be done by defining $\mathbf{t} = t_0, \ldots, t_\ell$ to be a knot sequence, and denoting $\psi_{\mathbf{t}}^{(\ell)}(x)$ to be the 0^{th} B-spline of order ℓ for the knot sequence $\mathbf{t}$,

$$\psi_{\mathbf{t}}^{(\ell)}(x) = B_{0,\ell,\mathbf{t}}(x).$$

Note that the knot sequence $\mathbf{t}$ also represents the so-called breaks of the B-spline. The B-spline in the region $[t_i, t_{i+1})$, $i = 0, \ldots, \ell - 1$ is a polynomial of degree $\ell - 1$, but in the entire support $[t_0, t_\ell]$, the B-spline is a piecewise polynomial. When the knots (t_j) are sampled in a symmetric and equidistant fashion, the B-spline is called a central B-spline. Notice that Definition (2.4) for a central B-spline is a subset of the general B-spline definition where the knots are equally-spaced. This new notation provides more flexibility than the previous central B-spline notation.

2.3 Position-Dependent SIAC Filtering

The original position-dependent Smoothness-Increasing Accuracy-Conserving filter is a convolution of the dG approximation with a central B-spline kernel

$$u^\star(\bar{x}) = (K_h^{(2k+1,\ell)} \star u_h)(\bar{x}). \tag{2.5}$$

The convolution kernel is given by

$$K^{(2k+1,\ell)}(x) = \sum_{\gamma=0}^{2k} c_\gamma^{(2k+1,\ell)} \psi^{(\ell)}(x - x_\gamma), \tag{2.6}$$

where $2k+1$ represents the number of *central* B-splines, ℓ the order of the B-splines and $K_h = \frac{1}{h} K\left(\frac{x}{h}\right)$. The coefficients $c_\gamma^{(2k+1,\ell)}$ are obtained from the property that the kernel reproduces polynomials of degree $\leq 2k$. For the symmetric central B-spline filter [6, 15], $\ell = k+1$ and $x_\gamma = -k + \gamma$, where k is the highest degree of the polynomial used in the dG approximation. More explicitly, the symmetric kernel is given by

$$K^{(2k+1,\ell)}(x) = \sum_{\gamma=0}^{2k} c_\gamma^{(2k+1,\ell)} \psi^{(\ell)}(x - (-k + \gamma)). \tag{2.7}$$

Note that this kernel is by construction symmetric and uses an equal amount of information from the neighborhood around the point being post-processed. While being symmetric is suitable in the interior domain when the function is smooth, it is not suitable for application near a boundary, or when the solution contains a discontinuity.

The one-sided position-dependent SRV filter defined in [20] is named after its change of support according to the location of the point being post-processed. For example, near a boundary or discontinuity, a translation of the filter is done so that the support of the kernel remains inside the domain. Furthermore, in these regions, a greater number of central B-splines is required. Using more B-splines aids in improving the magnitude of the errors near the boundary, while allowing superconvergence. In addition, the authors in [20] increased the number of B-splines used in the construction of the kernel to be

$4k+1$. The position-dependent (SRV) filter for elements near the boundaries can then be written as[*]

$$K^{(4k+1,\ell)}(x) = \sum_{\gamma=0}^{4k} c_\gamma^{(4k+1,\ell)} \psi^{(\ell)}(x - x_\gamma), \tag{2.8}$$

where x_γ depends on the location of the evaluation point $\bar{x}$ used in Equation (2.5) and at the boundaries is given by

$$x_\gamma = -4k + \gamma + \lambda(\bar{x}),$$

with

$$\lambda(\bar{x}) = \begin{cases} \min\{0, -\frac{4k+\ell}{2} + \frac{\bar{x}-x_L}{h}\}, & \bar{x} \in [x_L, \frac{x_L+x_R}{2}), \\ \max\{0, \frac{4k+\ell}{2} + \frac{\bar{x}-x_R}{h}\}, & \bar{x} \in [\frac{x_L+x_R}{2}, x_R]. \end{cases} \tag{2.9}$$

Here x_L and x_R are the left and right boundaries, respectively.

The authors chose $4k+1$ central B-splines because, in their experience, using fewer (central) B-splines was insufficient for enhancing the error. Furthermore, in order to provide a smooth transition from the boundary kernel to the interior kernel, a convex combination of the two kernels was used:

$$u_h^\star(x) = \theta(x)\left(K_h^{(2k+1,l)} \star u_h\right)(x) + (1-\theta(x))\left(K_h^{(4k+1,l)} \star u_h\right)(x), \tag{2.10}$$

where $\theta(x) \in \mathcal{C}^{k-1}$ such that $\theta = 1$ in the interior and $\theta = 0$ in the boundary regions. This position-dependent filter demonstrated better behavior in terms of error than the original one-sided filter given by Ryan and Shu in [14]. Throughout the article, we will refer to the position-dependent filter using $4k+1$ central B-splines as the SRV filter.

3 Proposed One-Sided Position-Dependent SIAC Filter

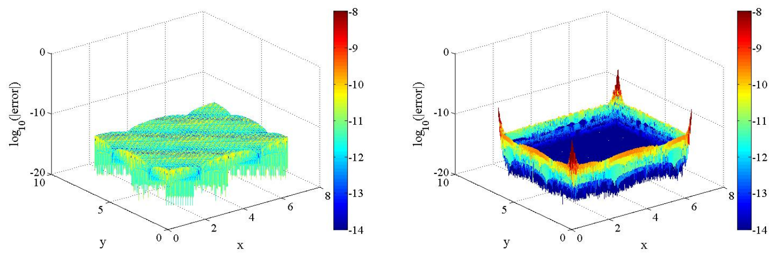
In this section, we propose a new one-sided position-depenent filter for application near boundaries. We first discuss the deficiencies in the current position-dependent SIAC filter. We then propose a new position dependent filter that ameliorates the deficiencies of the SRV filter; however, our new filter must make some compromises with regards to superconvergence (which will be discussed). Lastly, we prove that our new filter is globally superconvergent for $k=1$ and superconvergent in the interior of the domain for $k \geq 2$.

3.1 Deficiencies of the Previous Position-Dependent SIAC Filter

The SRV filter was reported to reduce the errors when filtering near a boundary. However, applying this filter to higher-order dG solutions (e.g., $\mathbb{P}^4-$ or even $\mathbb{P}^3-$polynomials

[*]Note that the notation used in the current manuscript is slightly different from the notation used in [20]. Instead of using $r_2 = 4k$ to denote the SRV filter we chose to use the number of B-splines directly for the clarity of the discussion.

in some cases) required using a multi-precision package (or at least quadruple precision) to reduce round-off error, leading to significantly increased computational time. Figure 3.1 shows the significant round-off error near the boundaries when using double precision for post-processing the initial condition. The multi-precision requirement also makes the position-depdendent kernel [20], near the boundaries, unsuitable for GPU computing.



(a) dG Error (b) Post-processed (SRV)

Figure 3.1: Comparison of the pointwise errors in log scale of the (a) original L^2 projection solution, (b) the SRV filter in [20] for the 2D L^2 projection using basis polynomials of degree $k = 4$, mesh 80×80. Double precision was used in these computations.

To discover why this challenge arises requires revisiting the foundations of the filter – in particular, the existing error estimates. The L^2-error estimate given in [6],

$$\|u - u_h^\star\|_{0,\Omega} \le Ch^{2k+1}, \tag{3.1}$$

provides us insight into the cause of the issue by examining the constant C in more detail. The constant C depends on:

$$\kappa^{(r+1,\ell)} = \sum_{\gamma=0}^{r} |c_\gamma^{(r+1,\ell)}|, \tag{3.2}$$

where c_γ denotes the kernel coefficients and the value of r depends on the number of B-splines used to construct the kernel. The kernel coefficients are obtained by ensuring that the kernel reproduces polynomials of degree r by the convolution:

$$K^{(r+1,\ell)}(x) \star (x)^p = x^p, \quad p = 0, 1, \ldots, r. \tag{3.3}$$

We note that for the error estimates to hold, it is enough to ensure that the kernel reproduces polynomials up to degree $2k$ (for $r \ge 2k$), although near the boundaries it was required that the kernel reproduces polynomials of degree $4k$ ($r = 4k$) in [8, 20]. For filtering near the boundary, this value κ defined in Equation (3.2) is on the order of 10^5 for $k = 3$ and 10^7 for $k = 4$, as can be seen in Figure 3.3. This indicates one possible avenue (i.e., lowering κ) by which we might generate an improved filter. A second avenue is by investigating the round-off error stemming from the large condition number of the linear system generated to satisfy Equation (3.3) and solved to find the kernel coefficients. The condition number of the generated matrix is on the order of 10^{24} for $k = 4$. This leads to significant round-off error (e.g., the rule of thumb in this particular case being that 24 digits of accuracy are lost due to the conditioning of this system), hence requiring the use of high-precision / extended precision libraries for SIAC filtering to remain accurate (in both its construction and usage).

The requirement of using extended precision in our computations increases the computational cost. In addition, the aforementioned discrepancy in the spatial extent of the filters due to the number of B-splines used – $2k+1$ in the symmetric case versus $4k+1$ in the one-sided case – leads the boundary filter costing even more due to extra function evaluations. The extra computational cost has led us to reconsider the position-dependent filter and propose a better conditioned and less computationally intensive alternative.

In order to apply SIAC filters near boundaries, we first no longer restrict ourselves to using only central B-splines. Secondly, we seek to maintain a constant support size for both the interior of the domain and the boundaries. The idea we propose is to add one general B-spline for boundary regions, which is located within the already defined support size. Using general B-splines provides greater flexibility and improves the numerical conditions (eliminating the explicit need for precision beyond double precision). To introduce our new position-dependent one-sided SIAC filter, we discuss the one-dimensional case and how to modify the current definitions of the SIAC filter. Multi-dimensional SIAC filters are a tensor product of the one-dimensional case.

3.2 The New Position-Dependent One-Sided Kernel

Before we provide the definition of the new position-dependent one-sided kernel, we first introduce a new concept, that of a *knot matrix*. The definition of a knot matrix helps us to introduce the new position-dependent one-sided kernel in a concise and compact form. It also aids in demonstrating the differences between the new position-dependent kernel, the symmetric kernel and the SRV filter. Informally, the idea behind introducing a knot matrix is to exploit the definition of B-splines in terms of their corresponding knot sequence $\mathbf{t} := (t_j)$, in the definition of the SIAC filter. In order to introduce a knot matrix, we will use the following notation: $\psi^{(\ell)}_{\mathbf{t}}(x) = B_{0,\ell,\mathbf{t}}(x)$.

Definition 3.1 (Knot matrix).

A **knot matrix, T**, *is an $n \times m$ matrix such that the $\gamma-$th row, $\mathbf{T}(\gamma)$, of the matrix* $\mathbf{T}$ *is a knot sequence with $\ell+1$ elements (i.e., $m = \ell+1$) that are used to create the B-spline $\psi^{(\ell)}_{\mathbf{T}(\gamma)}(x)$. The number of rows n is specified based on the number of B-splines used to construct the kernel.*

To provide some context for needing the definition of a knot matrix, we first redefine some of the previous SIAC kernels discussed in terms of their knot matrices. Recall that the general definition of the SIAC kernel relies on $r+1$ *central* B-splines of order ℓ. Therefore, we can use Definition 3.1 to rewrite the symmetric kernel given in Equation (2.7) in terms of a knot matrix as follows

$$K^{(2k+1,\ell)}_{\mathbf{T}_{sym}}(x) = \sum_{\gamma=0}^{2k} c^{(2k+1,\ell)}_{\gamma} \psi^{(\ell)}_{\mathbf{T}_{sym}(\gamma)}(x), \tag{3.4}$$

where $\mathbf{T}_{sym}$ in this relation is a $(2k+1)\times(\ell+1)$ matrix. Each row in $\mathbf{T}_{sym}$ corresponds to the knot sequence of one of the constituent B-splines in the symmetric kernel. More specifically, the element of $\mathbf{T}_{sym}$ are defined as

$$\mathbf{T}_{sym}(i,j) = -\frac{\ell}{2} + j + i - k, \qquad i = 0,\dots,2k; \text{ and } j = 0,\dots,\ell.$$

For instance, for the first order symmetric SIAC kernel we have: $\ell = 2$ and $k = 1$. Therefore, the corresponding knot matrix can be defined as

$$\mathbf{T}_{sym} = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}. \tag{3.5}$$

The definition of a SIAC kernel in terms of a knot matrix can also be used to rewrite the original boundary filter [14], which uses only $2k+1$ central B-splines at the left boundary. The knot matrix $\mathbf{T}_{one}$ for this case is given by

$$\mathbf{T}_{one} = \begin{pmatrix} -4 & -3 & -2 \\ -3 & -2 & -1 \\ -2 & -1 & 0 \end{pmatrix}. \tag{3.6}$$

Now we can define our new position-dependent one-sided kernel by generating a knot matrix. The new position-dependent one-sided kernel consists of $r+1 = 2k+1$ central B-splines and one general B-spline, and hence the knot matrix is of size $(2k+2)\times(\ell+1)$. At a high-level, using the scaling of the kernel, the new position-dependent one-sided kernel can be written as

$$K_{h\mathbf{T}}^{(r+1,\ell)}(x) = \sum_{\gamma=0}^{r+1} c_\gamma^{(r+1,\ell)} \psi_{h\mathbf{T}(\gamma)}^{(\ell)}(x), \tag{3.7}$$

where $\mathbf{T}(\gamma)$ represents the γ-th row of the knot matrix $\mathbf{T}$, which is the knot sequence $T(\gamma,0),\dots,T(\gamma,\ell)$. For the central B-spline, $\gamma = 0,\dots,2k$ and

$$\psi_{h\mathbf{T}(\gamma)}^{(\ell)}(x) = \frac{1}{h}\psi_{\mathbf{T}(\gamma)}^{(\ell)}\left(\frac{x}{h}\right).$$

The added B-spline is a monomial defined as

$$\psi_{h\mathbf{T}(r+1)}^{(\ell)}(x) = \frac{1}{h} x_{\mathbf{T}(r+1)}^{\ell-1}\left(\frac{x}{h}\right),$$

where

$$x_{\mathbf{T}(r+1)}^{\ell-1} = \begin{cases} (x - T(r+1,0))^{\ell-1}, & T(r+1,0) \le x \le T(r+1,\ell); \\ 0, & \text{otherwise.} \end{cases}$$

Therefore near the left boundary, the kernel in Equation (3.7) can be rewritten as

$$K_{h\mathbf{T}}^{(r+1,\ell)}(x) = \underbrace{\sum_{\gamma=0}^{r} c_\gamma^{(r+1,\ell)} \psi_{\mathbf{hT}(\gamma)}^{(\ell)}(x)}_{\text{Position-dependent kernel with } r = 2k+1 \text{ central B-splines}} + \underbrace{c_{r+1}^{(r+1,\ell)} \psi_{h\mathbf{T(r+1)}}^{(\ell)}(x)}_{\text{General B-spline}}. \tag{3.8}$$

The kernel coefficients, $c_\gamma^{(r+1,\ell)}$, $\gamma = 0,\dots,r+1$ are obtained through reproducing polynomials of degree up to $r+1$. We have imposed further restrictions on the knot matrix for the definition of the new position-dependent one-sided kernel. First, for convenience we require

$$T(\gamma,0) \le T(\gamma,1) \le \cdots \le T(\gamma,\ell), \quad \text{for} \quad \gamma = 0,\dots,r,+1$$

and

$$T(\gamma+1,0) \le T(\gamma,\ell), \quad \text{for} \quad \gamma = 0,\dots,r.$$

Second, the knot matrix, $\mathbf{T}$, should satisfy

$$T(0,0) \ge \frac{\bar{x}-x_R}{h} \quad \text{and} \quad T(r,\ell) \le \frac{\bar{x}-x_L}{h},$$

where h is the element size in a uniform mesh. This requirement is derived from the support of the B-spline as well as the support needing to remain inside the domain. Recall that the support of the B-spline $\psi^{(\ell)}_{\mathbf{T}(\gamma)}$ is $[T(\gamma,0),T(\gamma,\ell)]$, and the support of the kernel is $[T(0,0),T(r,\ell)]$. For any $\bar{x} \in [x_L,x_R]$, the post-processed solution at point $\bar{x}$ can then be written as

$$u^\star(\bar{x}) = \left(K_{h\mathbf{T}}^{(r+1,\ell)} \star u_h\right)(\bar{x}) = \int_{-\infty}^{\infty} K_{h\mathbf{T}}^{(r+1,\ell)}(\bar{x}-\xi)u_h(\xi)d\xi = \int_{\bar{x}-hT(r,\ell)}^{\bar{x}-hT(0,0)} K_{h\mathbf{T}}^{(r+1,\ell)}(\bar{x}-\xi)u_h(\xi)d\xi, \tag{3.9}$$

where $h\mathbf{T}$ represents the scaled knot matrix. For the boundary regions, we force the interval $[\bar{x}-hT(r,\ell),\bar{x}-hT(0,0)]$ to be inside the domain $\Omega = [x_L,x_R]$. This implies that

$$x_L \le \bar{x}-hT(r,\ell), \qquad \bar{x}-hT(0,0) \le x_R,$$

and hence the requirement of $T(0,0) \ge \frac{\bar{x}-x_R}{h}$ and $T(r,\ell) \le \frac{\bar{x}-x_L}{h}$. Finally, we require that the kernel remain as symmetric as possible. This means the knots should be chosen as

$$\text{Left }: T \leftarrow T - \left(T(r,\ell) - \frac{\bar{x}-x_L}{h}\right), \qquad \text{for } \frac{\bar{x}-x_L}{h} < \frac{3k+1}{2}, \tag{3.10}$$

$$\text{Right}: T \leftarrow T - \left(T(0,0) - \frac{\bar{x}-x_R}{h}\right), \qquad \text{for } \frac{x_R-\bar{x}}{h} < \frac{3k+1}{2}, \tag{3.11}$$

This shifting will increase the error and it is therefore still necessary to increase the number of B-splines used in the kernel.

Because the symmetric filter yields superconvergence results, we wish to retain the original form of the kernel as much as possible. Near the boundary, where the symmetric filter cannot be applied, we keep the $2k+1$ shifted central B-splines and add only one general B-spline. To avoid increasing the spatial support of the filter, we will choose the knots of this general B-spline dependent upon the knots of the $2k+1$ central B-splines in the following way: near the left boundary, we let the first $2k+1$ B-splines be central B-splines whereas the last B-spline will be a general spline. The elements of the knot matrix are then given by

$$T(i,j) = \begin{cases} -\ell - r + j + i + \frac{\bar{x}-x_L}{h}, & 0 \le i \le r, 0 \le j \le \ell; \\ \frac{\bar{x}-x_L}{h} - 1, & i = r+1, j = 0; \\ \frac{\bar{x}-x_L}{h}, & i = r+1, j = 1,\dots,\ell. \end{cases} \tag{3.12}$$

Similarly, we can design the new kernel near the right boundary, where the general B-spline is given by

$$\psi^{(\ell)}_{\mathbf{T}(0)}(x) = x^{\ell-1}_{\mathbf{T}(0)} = \begin{cases} (T(0,\ell) - x)^{\ell-1}, & T(0,0) \le x \le T(r+1,\ell); \\ 0, & \text{otherwise.} \end{cases}$$

The elements of the knot matrix for the right boundary kernel are defined as

$$T(i,j) = \begin{cases} \frac{\bar{x}-x_R}{h}, & i = 0, j = 0, \dots, \ell-1; \\ \frac{\bar{x}-x_R}{h} + 1, & i = 0, j = \ell; \\ j + i - 1 + \frac{\bar{x}-x_R}{h}, & 1 \le i \le r+1, 0 \le j \le \ell, \end{cases} \tag{3.13}$$

and the form of the kernel is then

$$K^{(r+1,\ell)}_{h\mathbf{T}}(x) = c^{(r+1,\ell)}_0 \psi^{(\ell)}_{h\mathbf{T}(0)}(x) + \sum_{\gamma=1}^{r+1} c^{(r+1,\ell)}_\gamma \psi^{(\ell)}_{h\mathbf{T}(\gamma)}(x). \tag{3.14}$$

We note that this "extra" B-spline is only used when $\frac{\bar{x}-x_L}{h} < \frac{3k+1}{2}$ or $\frac{x_R-\bar{x}}{h} < \frac{3k+1}{2}$, otherwise the symmetric central B-spline kernel is used.

We present a concrete example for the $\mathbb{P}^1$ case with $\ell = 2$. In this case, the knot matrices for our newly proposed filter at the left and right boundaries are

$$\mathbf{T}_{Left} = \begin{pmatrix} -4 & -3 & -2 \\ -3 & -2 & -1 \\ -2 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \qquad \mathbf{T}_{Right} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}. \tag{3.15}$$

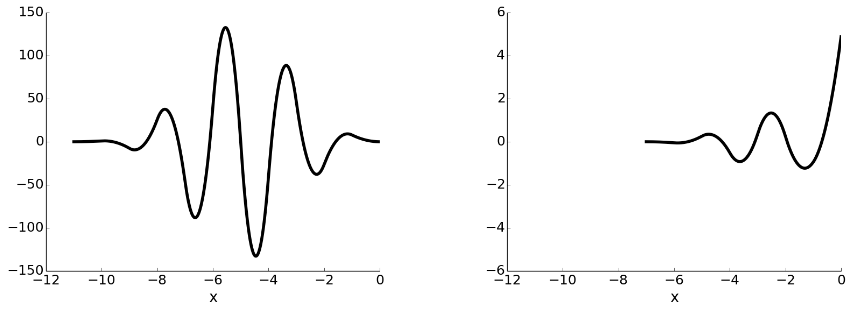
These new knot matrices are 4×3 matrices where, in the case of the filter for the left boundary, the first three rows express the knots of the three central B-splines and the last row expresses the knots of the general B-spline. For the filter applied to the right boundary, the first row expresses the knots of the general B-spline and the last three rows express the knots of the central B-splines.

If we use the same form of the knot matrix to express the SRV kernel introduced in [20] at the left boundary for $k = 1$, we would have

$$\mathbf{T}_{SRV} = \begin{pmatrix} -6 & -5 & -4 \\ -5 & -4 & -3 \\ -4 & -3 & -2 \\ -3 & -2 & -1 \\ -2 & -1 & 0 \end{pmatrix}. \tag{3.16}$$

Comparing the new knot matrix with the one used to obtain the SRV filter, we can see that they have the same number of columns, which indicates that they use the same order of B-splines. There are fewer rows in the new matrix $(2k+2)$ than the number of rows from the original position-dependent filter $(4k+1)$. This indicates that the new filter uses fewer B-splines than the SRV filter.

To compare the new filter and the SRV filter, we plot the kernels used at the left boundary for $k = 2$. Figure 3.2 illustrates that the new position-dependent SIAC kernel places more weight on the evaluation point than the SRV kernel, and the SRV kernel has a significantly larger magnitude and support which we observed to cause problems, especially for higher-order polynomials (such as $\mathbb{P}^3$ or $\mathbb{P}^4$). For this example, using the filter for quadratic approximations, the scaling of the original position-dependent SIAC filter has a range from -150 to 150 versus -7.5 to 7.5 for the newly proposed filter.



(a) Boundary Kernel (SRV) (b) Boundary Kernel (New)

Figure 3.2: Comparison of the two boundary filters before convolution. (a) The SRV kernel and (b) the new kernel at the boundary with $k = 2$. The boundary is represented by $x = 0$. The new filter has reduced support and magnitude.

3.3 Theoretical Results in the Uniform Case

The previous section introduced a new filter to reduce the errors of dG approximations while attempting to ameliorate the issues concerning the old filter. In this section, we discuss the theoretical results for the newly defined boundary kernel. Specifically, for $k = 1$ it is globally superconvergent of order three. For higher degree polynomials, it is possible to obtain superconvergence only in the interior of the domain.

Recall from Equation (3.7) that the new one-sided scaled kernel has the form

$$K_{h\mathbf{T}}^{(r+1,\ell)}(x) = \sum_{\gamma=0}^{r+1} c_\gamma^{(r+1,\ell)} \psi_{h\mathbf{T}(\gamma)}^{(\ell)}(x). \tag{3.17}$$

In the interior of the domain the symmetric SIAC kernel is used which consists of $2k+1$ central B-splines

$$K_{h\mathbf{T}}^{(2k+1,\ell)}(x) = \sum_{\gamma=0}^{2k} c_\gamma^{(2k+1,\ell)} \psi_{h\mathbf{T}(\gamma)}^{(\ell)}(x), \tag{3.18}$$

and, near the left boundary the new one-sided kernel can be written as

$$K_{h\mathbf{T}}^{(r+1,\ell)}(x) = \left(\sum_{\gamma=0}^{r} c_\gamma^{(r+1,\ell)} \psi_{h\mathbf{T}(\gamma)}^{(\ell)}(x) \right) + c_{r+1}^{(r+1,\ell)} \psi_{h\mathbf{T}(r+1)}^{(\ell)}(x), \qquad r = 2k$$

where $2k+1$ central B-splines are used together with one general B-spline. The scaled kernel $K_{h\mathbf{T}}^{(r+1,\ell)}$ has the property that the convolution $K_{h\mathbf{T}}^{(r+1,\ell)} \star u_h$ only uses information inside the domain Ω.

Theorem 3.1. *Let u be the exact solution to the linear hyperbolic equation*

$$u_t + \sum_{i=1}^{d} A_i u_{x_i} + A_0 u = 0, \quad \mathbf{x} \in \Omega \times (0,T], \tag{3.19}$$
$$u(\mathbf{x},0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega,$$

where the initial condition $u_0(\mathbf{x})$ is a sufficiently smooth function. Here, $\Omega \subset \mathbb{R}^d$. Let u_h be the numerical solution to Equation (3.19), obtained using a discontinuous Galerkin scheme with an upwind flux over a uniform mesh with mesh spacing h. Let $u_h^\star(\bar{\mathbf{x}}) = (K_{h\mathbf{T}}^{(r+1,\ell)} \star u_h)(\bar{\mathbf{x}})$ be the solution obtained by applying our newly proposed filter which uses $r+1 = 2k+1$ central B-splines of order $\ell = k+1$ and one general B-spline in boundary regions. Then the SIAC-filtered dG solution has the following properties:

(i) $\|(u - u_h^\star)(\bar{\mathbf{x}})\|_{0,\Omega} \le C\ h^3$ for $k = 1$. That is, $u_h^\star(\bar{\mathbf{x}})$ is globally superconvergent of order three for linear approximations.

(ii) $\|(u - u_h^\star)(\bar{\mathbf{x}})\|_{0,\Omega\setminus \mathrm{supp}\{K_s\}} \le C\ h^{r+1}$ when $r+1 \le 2k+1$ central B-splines are used in the kernel. Here $\mathrm{supp}\{K_s\}$ represents the support of the symmetric kernel. Thus, $u_h^\star(\bar{\mathbf{x}})$ is superconvergent in the interior of the domain.

(iii) $\|(u - u_h^\star)(\bar{\mathbf{x}})\|_{0,\Omega} \le C\ h^{k+1}$ globally.

Proof. We neglect the proof of properties (i) and (ii) as they are similar to the proofs in [6] and [8]. Instead we concentrate on $\|(u - u_h^\star)(\bar{\mathbf{x}})\| \le Ch^{k+1}$, which is rather straightforward.

Consider the one-dimensional case ($d = 1$). Then the error can be written as

$$\|u - K_{h\mathbf{T}}^{(r+1,\ell)} \star u_h\|_{0,\Omega} \le \Theta_{h,1} + \Theta_{h,2},$$

where

$$\Theta_{h,1} = \|u - K_{h\mathbf{T}}^{(r+1,\ell)} \star u\|_{0,\Omega} \quad \text{and} \quad \Theta_{h,2} = \|K_{h\mathbf{T}}^{(r+1,\ell)} \star (u - u_h)\|_{0,\Omega}.$$

The proof of higher order convergence for the first term, $\Theta_{H,1}$, is the same as in [6] as the requirement on $K_{h\mathbf{T}}$ does not change (reproduction polynomials up to degree $r+1$). This means that

$$\Theta_{h,1} \le \frac{h^{r+1}}{(r+1)!} C_1 |u|_{r+1,\Omega}.$$

Now consider the second term, $\Theta_{h,2}$. Without loss of generality, we consider the filter for the left boundary in order to estimate $\Theta_{h,2}$. The proofs for the filter in the interior and right boundary are similar. We use the form of the kernel given in Equation (3.8), which decomposes the new filter into two parts: $2k+1$ central B-splines and one general B-spline. That is, we write

$$K_{h\mathbf{T}}^{(r+1,\ell)}(x) = \underbrace{\left(\sum_{\gamma=0}^{r} c_\gamma^{(r+1,\ell)} \psi_{h\mathbf{T}(\gamma)}^{(\ell)}(x)\right)}_{\text{central B-splines}} + \underbrace{c_{r+1}^{(r+1,\ell)} \psi_{h\mathbf{T}(r+1)}^{(\ell)}(x)}_{\text{general B-spline}}.$$

Setting $e(x) = u(x) - u_h(x)$, then

$$\Theta_{h,2} = \|K_{h\mathbf{T}}^{(r+1,\ell)} \star e\|_{0,\Omega_1} \leq \|K_{h\mathbf{T}}^{(r+1,\ell)}\|_{L_1}\|e\|_0 \leq \sup_{x\in\Omega}\left(\sum_{\gamma=0}^{r}|c_\gamma^{(r+1,\ell)}| + \frac{|c_{2k+1}^{(r+1,\ell)}|}{\ell}\right)\|e\|_0.$$

Hence

$$\Theta_{h,2} \leq \mathsf{C}\sup_{x\in\Omega}\left(\sum_{\gamma=0}^{r}|c_\gamma^{(r+1,\ell)}| + \frac{|c_{2k+1}^{(r+1,\ell)}|}{\ell}\right)h^{k+1}.$$

Remark 3.1. *Note that in this analysis we steered away from the negative-order norm argument. Technically, the terms involving the central B-splines have a convergence rate of $r+1 \leq 2k+1$ as given in [6, 8]. It is the new addition, the term involving the general B-spline that presents the limitation and reduces the convergence rate to that of the dG approximation itself (i.e. is accuracy-order conserving).*

To extend this to the multidimensional case ($d > 1$), given an arbitrary $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$, we set

$$\psi_{\mathbf{T}(\gamma)}^{(\ell)}(\mathbf{x}) = \prod_{i=1}^{d}\psi_{\mathbf{T}(\gamma)}^{(\ell)}(x_i).$$

The filter for the multidimensional space considered is of the form

$$K_{h\mathbf{T}}^{(r+1,\ell)}(\mathbf{x}) = \sum_{\gamma=0}^{r+1}\mathbf{c}_\gamma^{(r+1,\ell)}\psi_{h\mathbf{T}(\gamma)}^{(\ell)}(\mathbf{x}),$$

where the coefficients $\mathbf{c}_\gamma^{(\ell)}$ are tensor products of the one-dimensional coefficients. To emphasize the general B-spline used near the boundary, we assume, without loss of generality, that in the $x_{k_1}, \dots, x_{k_{d_0}}$ directions we need the general B-spline, where $0 \leq d_0 \leq d$. Then

$$\psi_{h\mathbf{T}(2k+1)}^{(\ell)} = \prod_{i=1}^{d_0}\psi_{h\mathbf{T}(2k+1)}^{(\ell)}(x_{k_i}).$$

By applying the same idea we used for the one-dimensional case, the theorem is also true for multi-dimensional case. □

We note that the constant in the final estimation is a product of two other constants, one of them is determined by the filter ($\sum_{\gamma=0}^{r}|c_\gamma^{(r+1,\ell)}| + \frac{|c_{r+1}^{(r+1,\ell)}|}{\ell}$) and the other one is determined by the dG approximation. To further illustrate the necessity of examining the constant in the error term which contributed by the filter, we provide Figure 3.3. This figure demonstrates the difference between $\sum_{\gamma=0}^{r}|c_\gamma^{(r+1,\ell)}|$ for the previously introduced filters and our new filter in which the constant gets modified to $\sum_{\gamma=0}^{r}|c_\gamma^{(r+1,\ell)}| + \frac{|c_{r+1}^{(r+1,\ell)}|}{\ell}$. In Figure 3.3, one can clearly see that by adding a general spline to the $r+1$ central B-splines, we are able, in the boundary regions, to reduce the constant in the error term significantly.

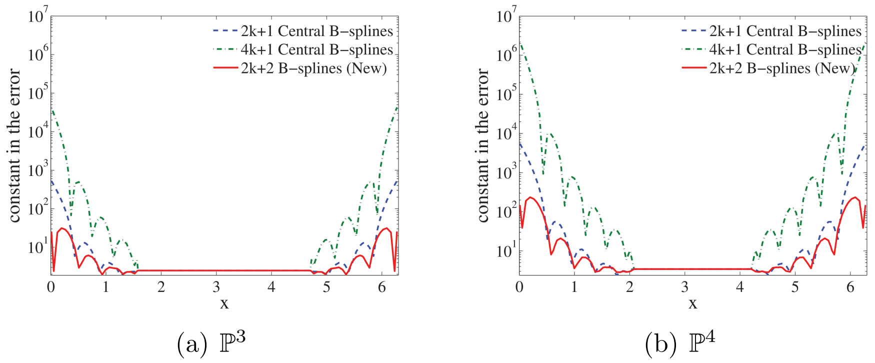


Figure 3.3: Plots demonstrating the effect of the coefficients on the error estimate for (a) $\mathbb{P}^3-$ and (b) $\mathbb{P}^4-$polynomials. Shown is $\sum_{\gamma=0}^{r}|c_\gamma^{(r+1,\ell)}|$ using $2k+1$ central B-splines (blue dashed), using $4k+1$ central B-splines (green dash dot-dot) and $\sum_{\gamma=0}^{r}|c_\gamma^{(r+1,\ell)}|+\frac{|c_{r+1}^{(r+1,\ell)}|}{\ell}$ for the new filter (red line).

3.4 Theoretical Results in the Non-Uniform (Smoothly-Varying) Case

In this section, we give a theoretical interpretation to the computational results presented in [7]. This is done by using the newly proposed filter for non-uniform meshes and showing that the new position-dependent filter maintains the superconvergence property in the interior of the domain for smoothly-varying meshes and is accuracy order conserving near the boundaries for non-uniform meshes. We begin by defining what we mean by a smoothly-varying mesh.

Definition 3.2 (Smoothly-Varying Mesh)**.**
Let ξ be a variable defined over a uniform mesh on domain $\Omega \subset \mathbb{R}$, then a smoothly-varying mesh defined over Ω is a non-uniform mesh whose variable x satisfies

$$x = \xi + f(\xi), \tag{3.20}$$

where f is a sufficiently smooth function and satisfies

$$f'(\xi) > -1, \qquad \xi \in \partial\Omega \Longleftrightarrow \xi + f(\xi) \in \partial\Omega.$$

For example, we can choose $f(\xi) = 0.5\sin(\xi)$ over $[0, 2\pi]$. The multi-dimensional definition can be defined in the same way.

Lemma 3.2. *Let u be the exact solution of a linear hyperbolic equation*

$$u_t + \sum_{n=1}^{d} A_n u_{x_n} + A_0 u = 0, \quad x \in \Omega \times (0, T], \tag{3.21}$$

with a smooth enough initial function and $\Omega \subset \mathbb{R}^d$. Let ξ be the variable for the uniform mesh defined on Ω with size h, and x be the variable of a smoothly-varying mesh defined *in* (3.20). *Let $u_h(\xi)$ be the numerical solution to Equation* (3.21) *over uniform mesh ξ, and $u_h(x)$ be the approximation over smoothly-varying mesh x, both of them obtained by using the discontinuous Galerkin scheme. Then the post-processed solution obtained by applying SIAC filter $K_h(\xi)$ for $u_h(\xi)$ and $K_H(x)$ for $u_h(x)$ with a proper scaling H, are related by*

$$\|u(x) - K_H \star u_h(x)\|_{0,\Omega} \le C\|u(\xi) - K_h \star u_h(\xi)\|_{0,\Omega}.$$

Here, the filter K can be any filter we mentioned in the previous section (symmetric filter, the original position-dependent filter, and newly proposed position-dependent filter). Note that this means that we obtain the full $2k+1$ superconvergence rate behavior for both the SRV and symmetric filters.

Proof. The proof is straightforward. If the scaling H is properly chosen, a simple mapping can be done from the smoothly-varying mesh to the corresponding uniform mesh. The result holds if the Jacobian is bounded (from the definition of smoothly-varying mesh).

$$\|u(x) - K_H \star u_h(x)\|_{0,\Omega}^2 = \int_\Omega (u(x) - K_H \star u_h(x))^2 \, dx$$
$$\overset{x\to\xi}{=} \int_{\tilde{\Omega}} (u(\xi) - u_h^\star(\xi))^2 (1 + f'(\xi)) d\xi \le \|u(\xi) - K_h \star u_h(\xi)\|_{0,\tilde{\Omega}}^2 \cdot \max |1 + f'(\xi)|.$$

According to the definition of smoothly-varying mesh, $\Omega = \tilde{\Omega}$, we have

$$\|u(x) - K_H \star u_h(x)\|_{0,\Omega} \le C\|u(\xi) - K_h \star u_h(\xi)\|_{0,\Omega},$$

where $C = \left(\max_\Omega |1 + f'|\right)^{\frac{1}{2}}$. □

Remark 3.2. *The proof seems obvious, but it is important to choose a proper scaling for H in the computations. Due to the smoothness and computational cost requirements, we need to keep H constant when treating points within the same element. Under this condition, the natural choice is $H = \Delta x_j$ when post-processing the element I_j. It is now easy to see that there exists a c in the element I_j, such that*

$$H = \Delta x_j = h(1 + f'(c)).$$

Remark 3.3. *Note that Theorem 3.1 (iii) still holds for generalized non-uniform meshes. This is due to the proof not relying on the properties (i.e. structure) of the mesh.*

We have now shown that superconvergence can be achieved for interior solutions over smoothly-varying meshes. In the subsequent sections, we present numerical results that confirm our results on uniform and non-uniform (smoothly-varying) meshes.

4 Numerical Results for One Dimension

The previous section introduced a new SIAC kernel by adding a general B-spline to a modified central B-spline kernel. The addition of a general B-spline helps to maintain a consistent support size for the kernel throughout the domain and eliminates the need for a multi-precision package. This section illustrates the performance of the new position-dependent SIAC filter on one-dimensional uniform and non-uniform (smoothly-varying and random) meshes. We compare our results to the SRV filter [20]. In order to provide a fair comparison between the SRV and new filters, we mainly show the results using quadruple precision for a few one-dimensional cases. Furthermore, in order to reduce the computational cost of the filter that uses $4k + 1$ central B-splines, we neglect to implement the convex combination described in Equation (2.10). This is not necessary for the new filter, and was implemented in the SRV filter to ensure the transition from the one-sided filter to the symmetric filter was smooth.

This is the first time that the position-dependent filters have been tested on non-uniform meshes. Although tests were performed using scalings of $H = \Delta x_j$ and $H = \max \Delta x_j$, we only present the results using a scaling of $H = \Delta x_j$. This scaling provides better results in boundary regions, which is one of the motivations of this paper. We note that the errors produced using a scaling of $H = \max \Delta x_j$ are quite similar and often produce smoother errors in the interior of the domain for smoothly-varying meshes.

Remark 4.1. *The SRV filter requires using quadruple precision in the computations to eliminate round-off error, which typically involves more computational effort than natively-supported double precision. The new filter only requires double precision. In order to give a fair comparison between the SRV filter and the new filter, for the one-dimensional examples we have used quadruple precision to maintain a consistent computational environment.*

4.1 Linear Transport Equation Over a Uniform Mesh

The first equation that we consider is a linear hyperbolic equation with periodic boundary conditions,

$$u_t + u_x = 0, \qquad (x,t) \in [0,1] \times (0,T] \tag{4.1}$$

$$u(x,0) = \sin(2\pi x), \qquad x \in [0,1]. \tag{4.2}$$

The exact solution is a periodic translation of the sine function,

$$u(x,t) = \sin(2\pi(x-t)).$$

For $T = 0$, this is simply the L^2-projection of the initial condition. Here, we consider a final time of $T = 1$ and note that we expect similar results at later times.

The discontinuous Galerkin approximation error and the position-dependent SIAC filtered error results are shown in Table 4.1 for both quadruple precision and double precision. Using quadruple precision, both filters reduce the errors in the post-processed solution, although the new filter has only a minor reduction in the quality of the error. However, using double precision only the new filter can maintain this error reduction for $\mathbb{P}^3-$ and $\mathbb{P}^4-$polynomials. We note that we concentrate on the results for $\mathbb{P}^3-$ and $\mathbb{P}^4-$polynomials as there is no noticeable difference between double and quadruple precision for $\mathbb{P}^1-$ and $\mathbb{P}^2-$polynomials.

The pointwise error plots are given in Figures 4.1 and 4.2. When using quadruple precision as in Figure 4.1, the SRV filter can reduce the error of the dG solution better than the new filter for fine meshes. However, it uses $2k-1$ more B-splines than the newly generated filter. This difference is noticeable when using double precision, which is almost ten times faster than using quadruple precision for $\mathbb{P}^3$ and $\mathbb{P}^4$. For such examples the new filter performs better both computationally and numerically (in terms of error). Table 4.1 shows that the SRV filter can only reduce the error for fine meshes when using $\mathbb{P}^4$ piecewise polynomials. The new filter performs as good as when using quadruple precision and reduces the error magnitude at a reduced computational cost.

Additionally, we point out that the accuracy of the SRV filter depends on (1) having higher regularity of $\mathcal{C}^{4k+1}$, (2) a well-resolved dG solution, and (3) a wide enough support (at least $5k+1$ elements). The same phenomenon will also be observed in the following tests such as for a nonlinear equation. For the new filter, the support size remains the same throughout the domain – $3k+1$ elements – and a higher degree of regularity is not necessary.

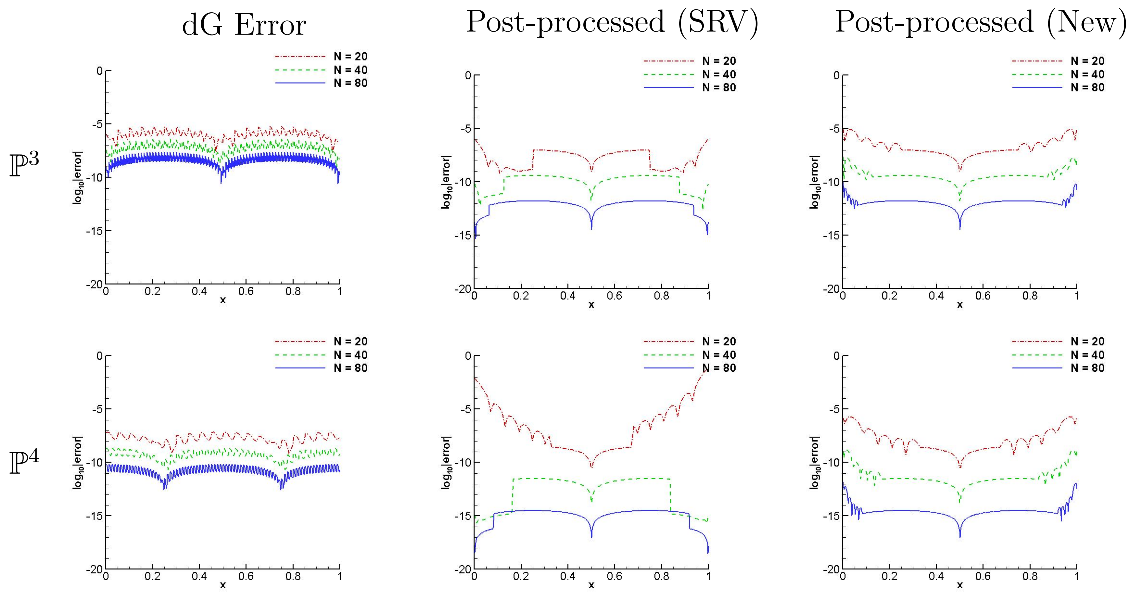


Figure 4.1: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the linear transport equation over uniform meshes using polynomials of degree $k=3,4$ (top and bottom rows, respectively). Quadruple precision was used in the computations.

4.2 Non-Uniform Meshes

We begin by defining three non-uniform meshes that are used in the numerical examples. The meshes tested are:

Mesh 4.2. *Smoothly-Varying Mesh with Periodicity. The first mesh is a simple smoothly-varying mesh. It is defined by* $x=\xi+b\sin(\xi)$, *where* ξ *is a uniform mesh variable and* $b=0.5$ *as in [7]. We note that the tests were also performed for different values of* b*;*

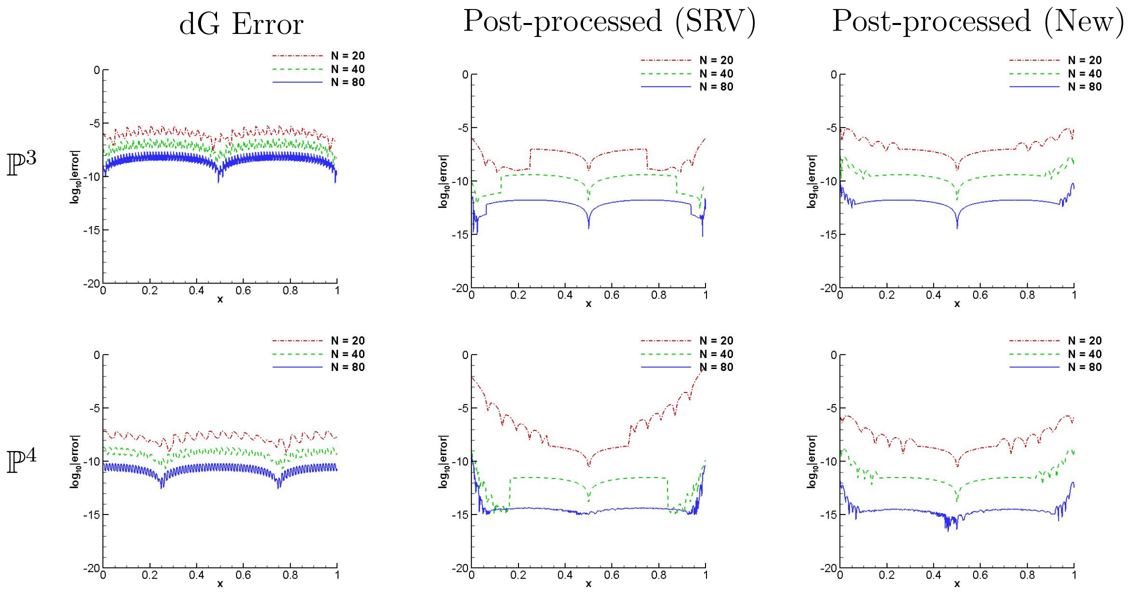


Figure 4.2: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the linear transport equation over uniform meshes using polynomials of degree $k = 3, 4$ (top and bottom rows, respectively). Double precision was used in the computations.

similar results were attained in all cases. This mesh has the nice feature that it is a periodic mesh and that the elements near the boundaries have a larger element size.

Mesh 4.3. *Smooth Polynomial Mesh. The second mesh is also a smoothly-varying mesh but does not have a periodic structure. It is defined by $x = \xi - 0.05(\xi - 2\pi)\xi$. For this mesh, the size of elements gradually decrease from left to right.*

Mesh 4.4. *Randomly-Varying Mesh. The third mesh is a mesh with randomly distributed elements. The element size varies between $[0.8h, 1.2h]$, where h is the uniform mesh size.*

We will now present numerical results demonstrating the usefulness of the position-dependent SIAC filter in [20] and our new one-sided SIAC filter for the aforementioned meshes.

4.3 Linear Transport Equation

The first example that we consider is a linear transport equation,

$$\begin{aligned} u_t + u_x &= 0, \quad (x,t) \in [0, 2\pi] \times (0, T) \\ u(x,0) &= \sin(x), \end{aligned} \tag{4.3}$$

with periodic boundary conditions and the errors calculated at $T = 2\pi$. We calculate the discontinuous Galerkin approximations for this equation over the three different non-uniform meshes (Mesh 4.2, Mesh 4.3 and Mesh 4.4). The approximation is then

Table 4.1: L^2- and $L^\infty-$errors for the dG approximation together with the SRV and new filters for the linear transport equation using polynomials of degree $k = 1, \ldots, 4$ (quadruple precision) and $k = 3, 4$ (double precision) over uniform meshes.

	dG				SRV Filter				New Filter			
Mesh	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order
Quadruple precision												
$\mathbb{P}^1$												
20	4.02E-03	–	1.45E-02	–	1.98E-03	–	2.80E-03	–	1.98E-03	–	2.80E-03	–
40	1.02E-03	1.97	3.82E-03	1.92	2.44E-04	3.02	3.46E-04	3.02	2.44E-04	3.02	3.46E-04	3.02
80	2.58E-04	1.99	9.79E-04	1.96	3.02E-05	3.01	4.28E-05	3.01	3.03E-05	3.01	4.28E-05	3.01
$\mathbb{P}^2$												
20	1.07E-04	–	3.67E-04	–	3.73E-06	–	5.82E-06	–	1.21E-05	–	8.27E-05	–
40	1.34E-05	3.00	4.62E-05	2.99	9.42E-08	5.31	1.34E-07	5.44	5.52E-07	4.45	5.31E-06	3.96
80	1.67E-06	3.00	5.78E-06	3.00	2.48E-09	5.24	3.52E-09	5.26	4.79E-08	3.53	6.19E-07	3.10
$\mathbb{P}^3$												
20	2.06E-06	–	6.04E-06	–	1.53E-07	–	1.02E-06	–	2.30E-06	–	8.71E-06	–
40	1.29E-07	4.00	3.80E-07	3.99	2.70E-10	9.15	4.00E-10	11.32	4.14E-09	9.12	2.27E-08	8.58
80	8.07E-09	4.00	2.38E-08	4.00	1.22E-12	7.79	1.73E-12	7.85	8.18E-12	8.98	1.20E-10	7.56
$\mathbb{P}^4$												
20	3.19E-08	–	7.02E-08	–	7.53E-03	–	7.33E-02	–	5.31E-07	–	1.99E-06	–
40	1.00E-09	4.99	2.25E-09	4.97	1.99E-12	31.82	3.12E-12	34.45	2.97E-10	10.80	1.58E-09	10.30
80	3.14E-11	5.00	7.14E-11	4.98	2.23E-15	9.80	3.19E-15	9.93	1.37E-13	11.08	1.55E-12	9.99
Double precision												
$\mathbb{P}^3$												
20	2.06E-06	–	6.04E-06	–	1.53E-07	–	1.02E-06	–	2.30E-06	–	8.71E-06	–
40	1.29E-07	4.00	3.80E-07	3.99	2.70E-10	9.15	4.00E-10	11.32	4.14E-09	9.12	2.27E-08	8.58
80	8.07E-09	4.00	2.38E-08	4.00	1.25E-12	7.75	3.85E-12	6.70	8.18E-12	8.98	1.20E-10	7.56
$\mathbb{P}^4$												
20	3.19E-08	–	7.02E-08	–	7.53E-03	–	7.33E-02	–	5.31E-07	–	1.99E-06	–
40	1.00E-09	4.99	2.25E-09	4.97	3.97E-11	27.50	6.14E-10	26.83	2.97E-10	10.80	1.58E-09	10.30
80	3.14E-11	5.00	7.14E-11	4.98	1.48E-11	1.42	3.28E-10	0.90	1.37E-13	11.08	1.55E-12	9.99

post-processed at the final time in order to analyze the numerical errors. We note that although the boundary conditions for the equation are periodic, in the boundary regions we implement the one-sided filter in [20] as the SRV filter and compare them with the new filter presented above.

The pointwise error plots for the periodically smoothly varying mesh are given in Figure 4.3 with the corresponding errors presented in Table 4.2. In the boundary regions, the SRV filter behaves slightly better for coarse meshes than the new filter. However, we recall that this filter essentially doubles the support in the boundary regions. Additionally, we see that the new filter has a higher convergence rate than $k + 1$ which is better than the theoretically predicted convergence rate.

For the smooth polynomial mesh 4.3 (without a periodic property), the results of using the scaling of $H = \Delta x_j$ are presented in Figure 4.4 and Table 4.2. Unlike the previous example, without the periodic property, the SRV filter leads to significantly worse performance. The SRV filter no longer enhances the accuracy order and has larger errors near the boundaries. On the other hand, the new filter still improves accuracy when the mesh is sufficiently refined ($N = 40$). Numerically the new filter obtains higher accuracy order than $k + 1$. For higher order polynomials, $\mathbb{P}^3$ and $\mathbb{P}^4$, we see that it achieves accuracy order of $2k + 1$, but this is not theoretically guaranteed.

Lastly, the filters were applied to dG solutions over a randomly distributed mesh. For this randomly-varying mesh, the new filter again reduces the errors except for a very

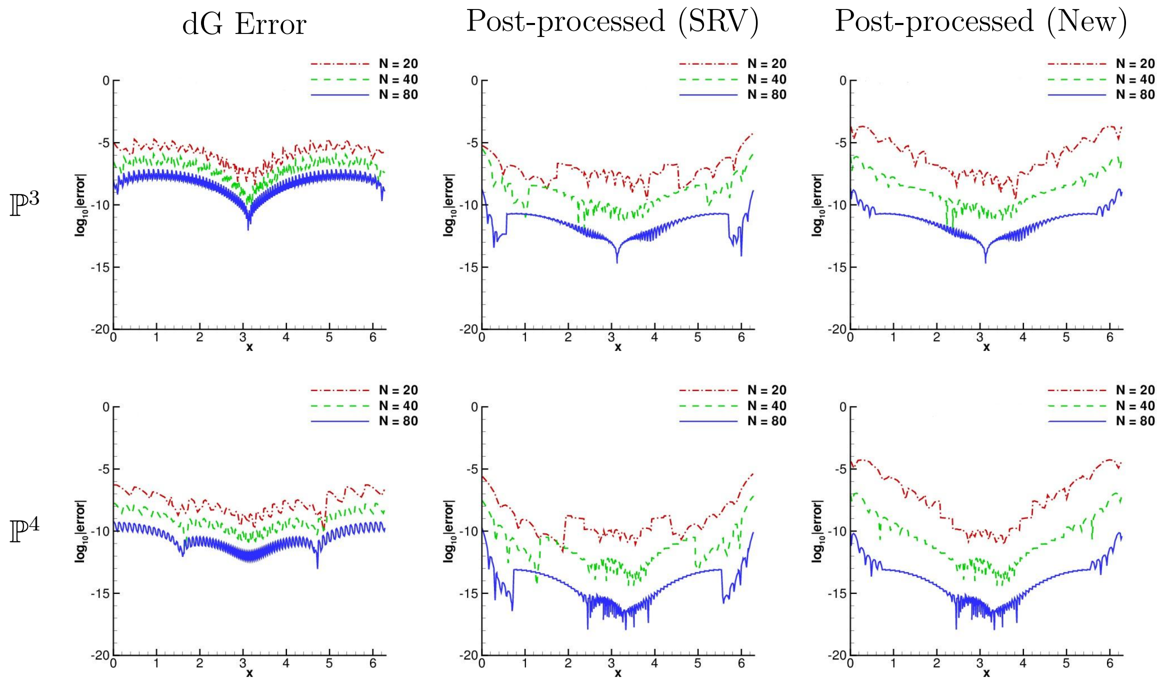


Figure 4.3: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the linear transport equation (4.3) over smoothly-varying mesh (Mesh 4.2). The kernel scaling $H = \Delta x_j$ and quadruple precision was used in the computations.

coarse mesh (see Table 4.2). The accuracy order is decreasing compared to smoothly-varying mesh examples, but it is still higher than $k + 1$. Unlike the smoothly-varying mesh, there are more oscillations in the errors (Figure 4.5). However, the oscillations are still reduced compared to the dG solutions. Note that the SRV filter does not improve the errors from the dG solution at all, and are even worse. This suggests that the SRV filter may be only suitable for uniform meshes.

4.4 Variable Coefficient Equation

In this example, we consider the variable coefficient equation:

$$\begin{aligned} &u_t + (au)_x = f, \quad x \in [0, 2\pi] \times (0, T) \\ &a(x,t) = 2 + \sin(x + t), \\ &u(x, 0) = \sin(x), \end{aligned} \tag{4.4}$$

at $T = 2\pi$. Similar to the previous constant coefficient equation (4.3), we also test this variable coefficient equation (4.4) over three different non-uniform meshes (Mesh 4.2, Mesh 4.3 and Mesh 4.4). Since the results are similar to the previous linear transport equation (4.3), here we do not re-describe the detail of the results. We only note that the results of variable coefficient equation have more wiggles than the constant coefficient equation. This may be an important issue in extending these ideas to nonlinear equations. To save space, we only show the $\mathbb{P}^3$ and $\mathbb{P}^4$ results, $\mathbb{P}^1$ and $\mathbb{P}^2$ are similar to the previous examples.

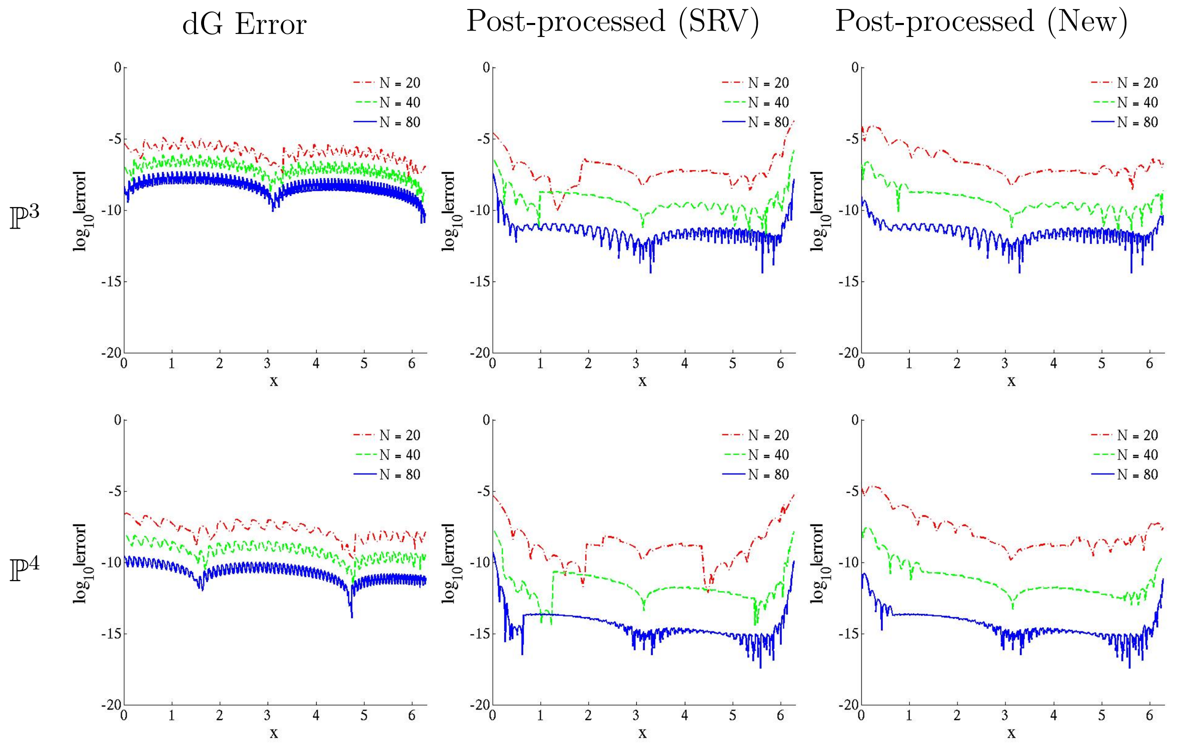


Figure 4.4: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the linear transport equation (4.3) over smooth polynomial mesh (Mesh 4.3). The kernel scaling $H = \Delta x_j$ and quadruple precision was used in the computations.

Figure 4.6 shows the pointwise error plots for the dG and post-processed approximations over a smoothly-varying mesh. The corresponding errors are given in Table 4.3. The results are similar to the linear transport equation. The two filters perform similarly, with the new filter being more computationally efficient.

For the smooth polynomial mesh 4.3, we show the pointwise error plots in Figure 4.7. The corresponding errors are given in Table 4.3. In this example we see that the new filter behaves better at the boundaries than the SRV filter. This may be due to the more compact kernel support size.

Finally, we test the variable coefficient equation (4.4) over randomly-varying mesh 4.4. Similar to the linear transport example, the pointwise errors plots (Figure 4.8) show more oscillations than the smoothly-varying mesh examples. We again see the new filter has better errors at the boundaries than the SRV filter.

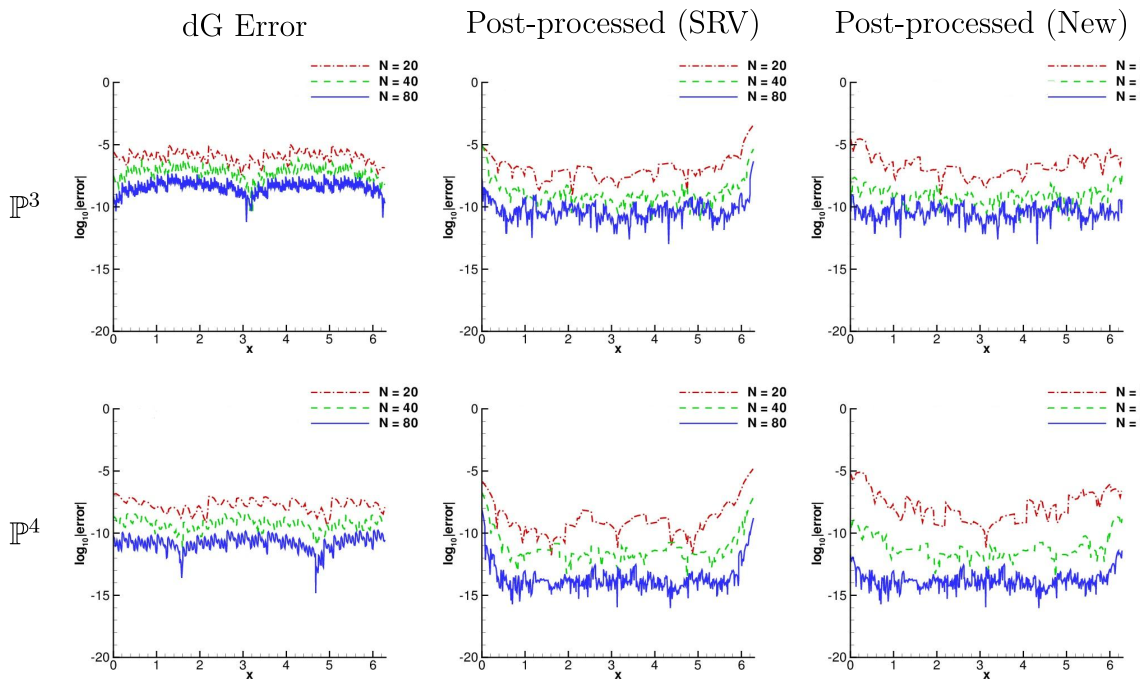


Figure 4.5: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the linear transport equation (4.3) over randomly-varying Mesh 4.4. The kernel scaling $H = \Delta x_j$ and quadruple precision was used in the computations.

5 Numerical Results for Two Dimensions

5.1 Linear Transport Equation Over a Uniform Mesh

To demonstrate the performance of the new filter in two-dimensions, we consider the solution to a linear transport equation,

$$u_t + u_x + u_y = 0, \quad (x, y) \in [0, 2\pi] \times [0, 2\pi], \tag{5.1}$$

$$u(x, y, 0) = \sin(x + y) \tag{5.2}$$

at $T = 2\pi$. Due to the computational cost to obtain the post-processed solution, we only present the 2D results using double precision. Table 5.1 shows that the accuracy is effected by the round-off error, especially for the previous one-sided position-dependent filter. Such significant round-off error appears to destroy the accuracy. Although the error magnitude near the boundaries is larger than the regions where a symmetric filter is used, the new filter reduces the error and improves smoothness of the dG solution.

5.2 Linear Transport Equation Over a Non-Uniform Mesh

For the 2D example, we consider the same linear transport equation as above, now over non-uniform meshes. The non-uniform meshes we consider are rectangular grids, which the tessellations on $x-$ and $y-$ directions are generated by the same way as the Mesh 4.2, Mesh 4.3 and Mesh 4.4. Because of computational cost, we only use double precision

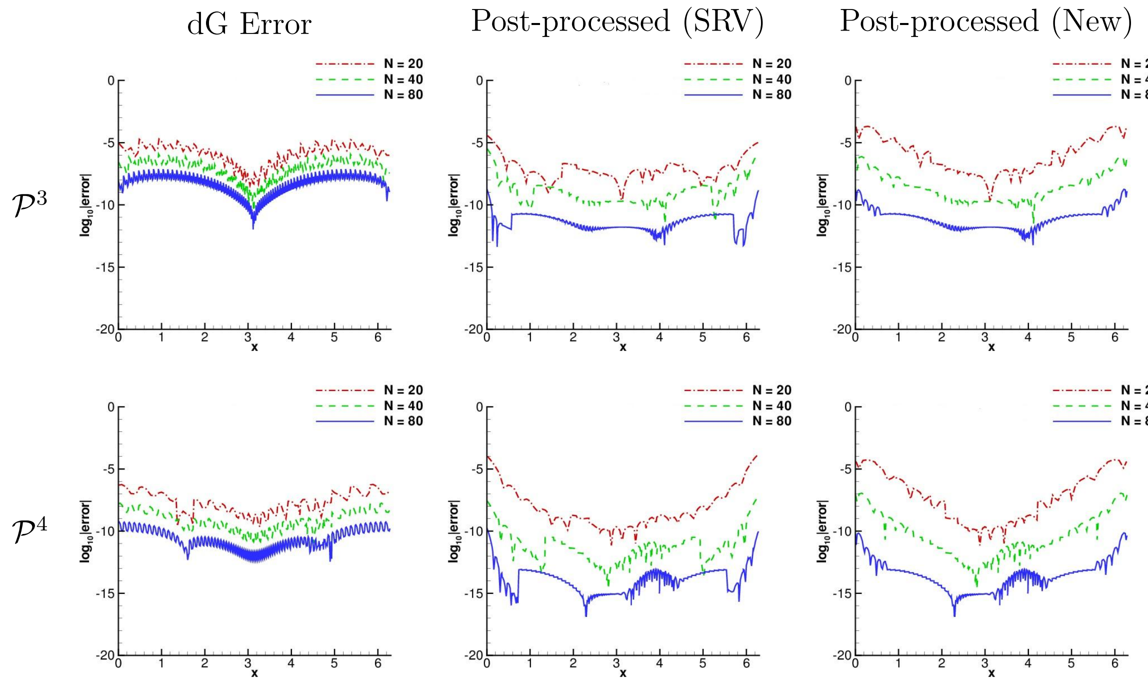


Figure 4.6: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the variable coefficient equation (4.4) over smoothly-varying mesh (Mesh 4.2). The kernel scaling $H = \Delta x_j$ and quadruple precision was used in the computations.

in the two-dimensional computations. Unlike the one-dimensional example, the results of the SRV filter are significantly affected by the round-off error, especially near the four corners of the grids. This round-off error completely destroys the accuracy and smoothness near the boundaries. Compared to the SRV filter, the new filter performs much better. In the following examples, we can clearly see the improvement of the accuracy and smoothness compared to the original dG approximations. From all the tests we performed, it is easy to see that the new filter is more suitable than the SRV filter over non-uniform meshes, and the practical performance of the new filter is better than the theoretical prediction.

For the $\mathbb{P}^3$ case, because of the periodicity, the SRV filter seems slightly better in L^2 norm than the new filter. However, if we look at the L^∞ norm, we can see the new filter still behaves better than the SRV filter (see Table 5.2). For the $\mathbb{P}^4$ case, we can see that even the ideal periodic property can not hide the fact that the SRV filter is not suitable for non-uniform meshes – the SRV filter is worse than the new filter and even the original dG solution. In Figure 5.2, the round-off error of the SRV filter is noticeably demonstrated. The new filter has better errors in L^2 and L^∞ norm when the mesh is sufficiently refined.

Unlike the smoothly-varying mesh we used in the previous example, the smooth-polynomial mesh and the randomly-varying mesh do not have the nice periodic property which is exactly where a one-sided filter is needed. The deficiencies of the SRV filter become significant. The results near the boundaries are worse than the original dG solution (Figures 5.3 and 5.4). The previous filters work well when all of their preconditions

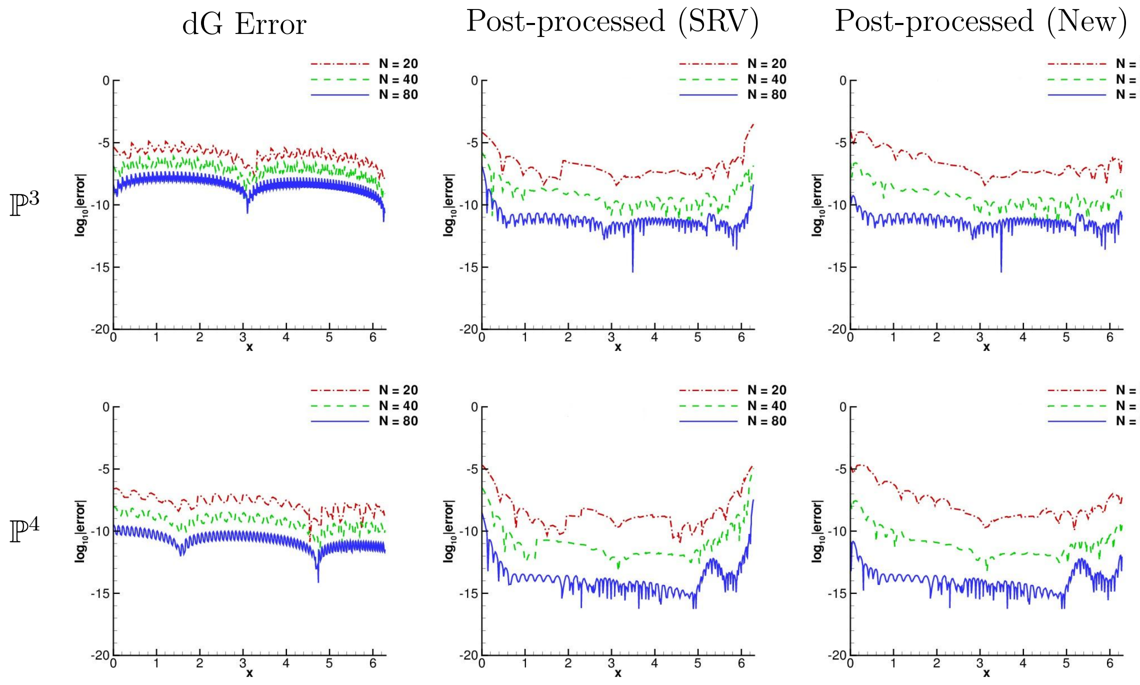


Figure 4.7: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the variable coefficient equation (4.4) over smooth polynomial mesh (Mesh 4.3). The kernel scaling $H = \Delta x_j$ and quadruple precision was used in the computations.

are met; however, if any one of its assumptions are violated, we may not obtain the full benefit of the filter. The new filter appears to still perform well in such circumstances.

6 Conclusion

In this paper, we have proposed a new position-dependent Smoothness-Increasing Accuracy-Conserving (SIAC) filter applied to discontinuous Galerkin approximations over uniform and non-uniform meshes. The new filter was devised as a consequence of analyzing the constant in the previous error estimates. This filter was created by introducing an extra general B-spline to a filter consisting of $2k+1$ central B-splines. This strategy allows us to overcome two shortcomings of the SRV filter: we can now reliably use double-precision to both produce and use our filter, and our new filter has a smaller geometric footprint and hence costs less (in terms of operations) to evaluate. We have, for the first time, proved the accuracy-order conserving nature of the SIAC filter globally and shown that this boundary filter does not affect the interior superconvergence properties. Additionally, we are able, for the first time, to extend our proofs of superconvergence for our symmetric and SVR SIAC filters used over smoothly-varying meshes. We demonstrated the applicability of the position-dependent filter for non-uniform meshes by choosing a proper scaling, H, which is obtained by analyzing smoothly-varying meshes. Numerical results indicate that this scaling idea works, even in the random mesh case (although no proof exists to assert this). Future work will concentrate on extending these concepts to derivative filtering.

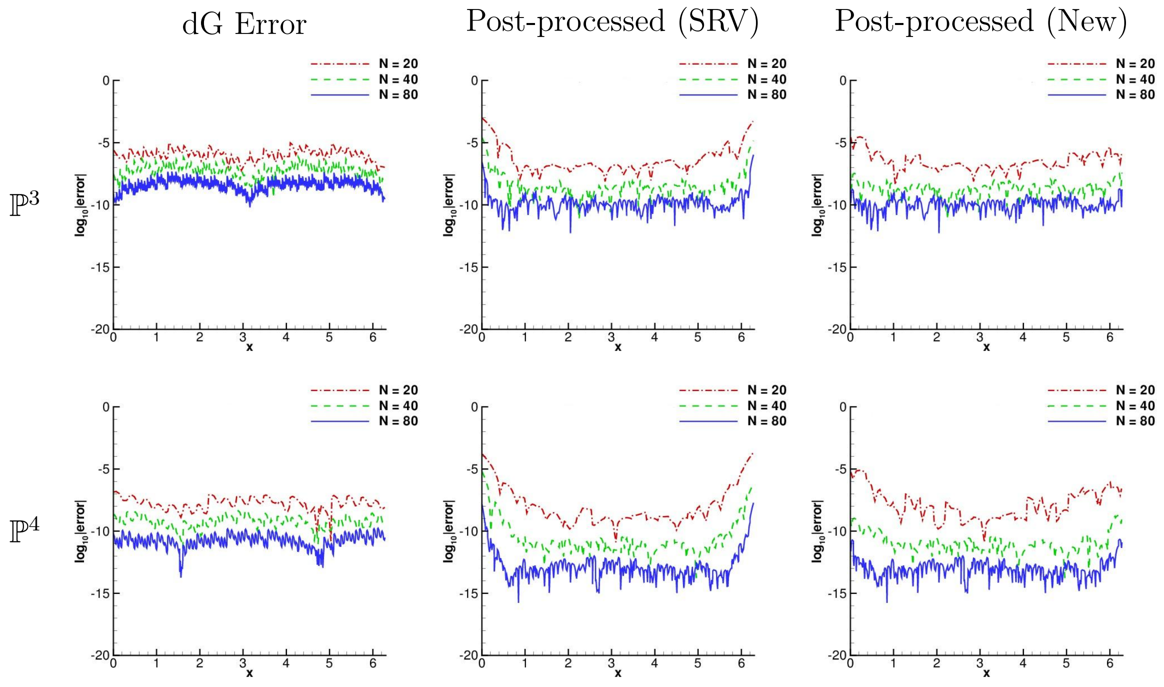
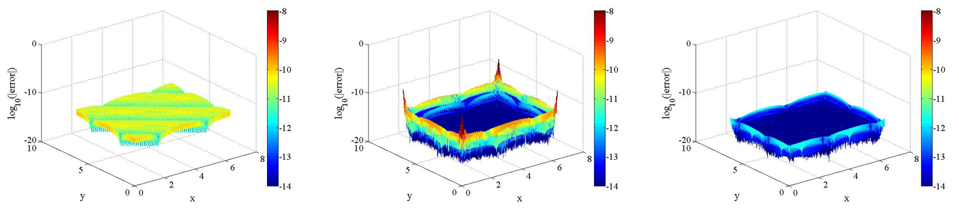


Figure 4.8: Comparison of the pointwise errors in log scale of the original dG solution (left column), the SRV filter (middle column) and the new filter (right column) for the variable coefficient equation (4.4) over randomly-varying mesh (Mesh 4.4). The kernel scaling $H = \Delta x_j$ and quadruple precision was used in the computations.



(a) dG Error (b) Post-processed (SRV) (c) Post-processed (New)

Figure 5.1: Comparison of the pointwise errors in log scale of the original dG solution (left), the SRV filter (middle) and the new filter (right) for the 2D linear transport equation using polynomials of degree $k = 4$ and a uniform 80×80 mesh. Double precision was used in the computations.

Acknowledgments

We would like to first thank Dr. Mahsa Mirzargar for helping us drive this draft to completion. Without her help, we would have possibly never finished this manuscript. We would like to thank Dr. Hanieh Mirzaee for her discovery and discussion concerning the conditioning issues of the SRV filter. The first and second authors are sponsored by the Air Force Office of Scientific Research (AFOSR), Air Force Material Command, USAF, under grant number and FA8655-13-1-3017. The third author is sponsored in part by the Air Force Office of Scientific Research (AFOSR), Computational Mathematics

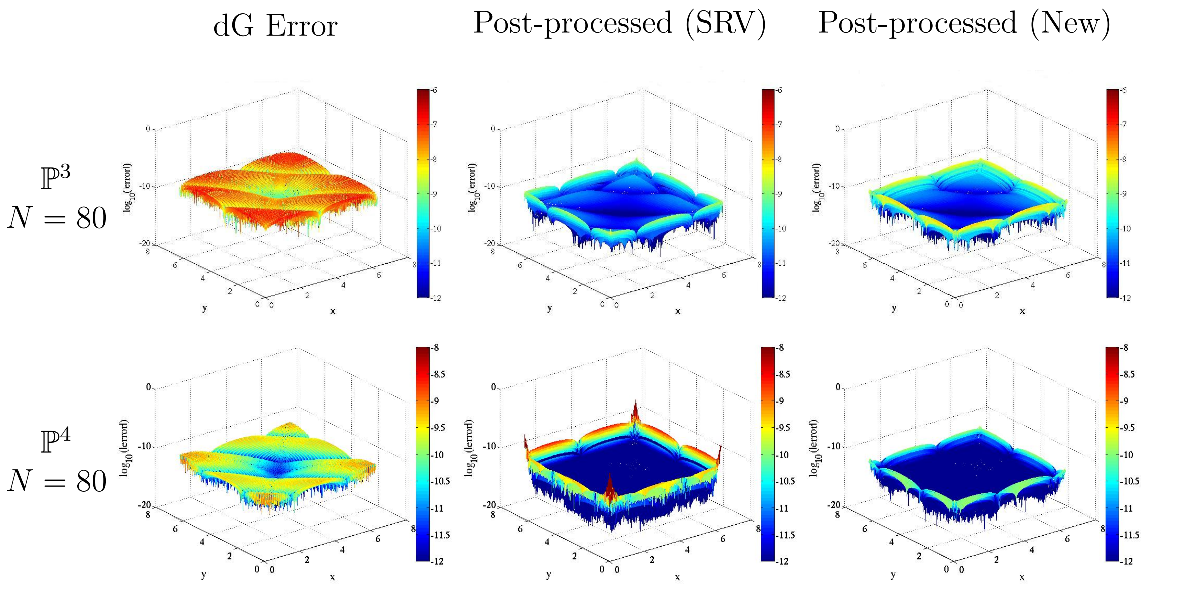


Figure 5.2: Comparison of the pointwise errors in log scale of the original dG solution (left), the SRV filter (middle) and the new filter (right) for the 2D linear transport equation (5.1) using polynomials of degree $k = 3, 4$ over a smoothly-varying 80×80 mesh (Mesh 4.3). The filters use the scaling of $H_x = \Delta x_j$ in x-direction and $H_y = \Delta y_j$ in y-direction. Double precision was used in the computations.

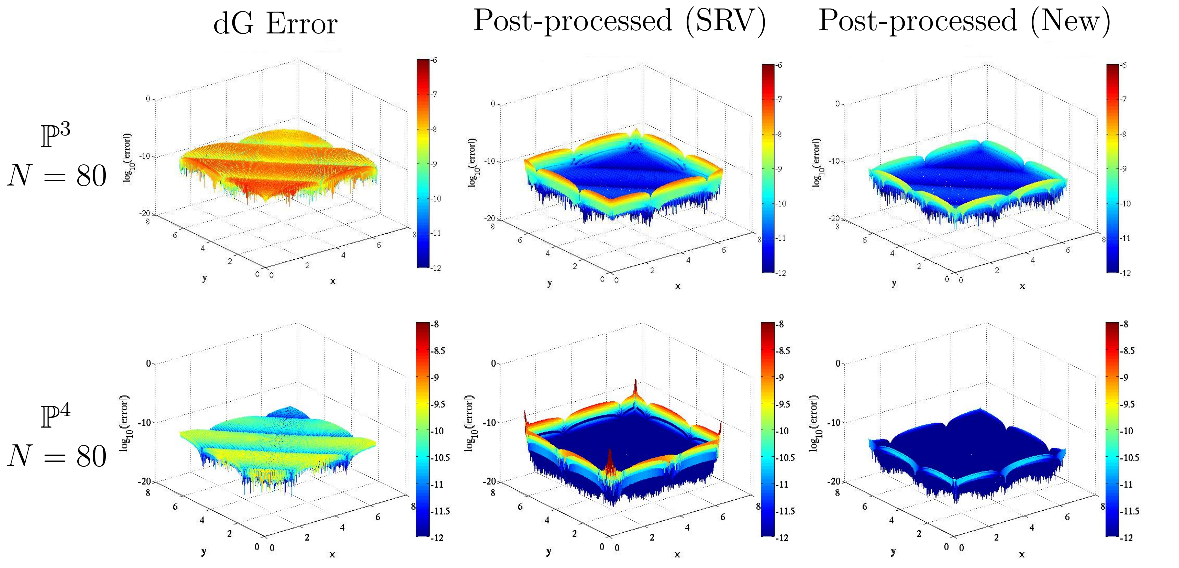


Figure 5.3: Comparison of the pointwise errors on a log scale between the SRV and new filters for the 2D linear transport equation using polynomials of degree $k = 3, 4$. A smooth-varying mesh defined by $x = \xi - b(\xi - 2\pi)\xi$, $y = \xi - b(\xi - 2\pi)\xi$ with $b = 0.05$ was used. Filter scaling was based upon the local element size. Double precision was used in the computations.

Program (Program Manager: Dr. Fariba Fahroo), under grant number FA9550-12-1-0428. The U.S Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

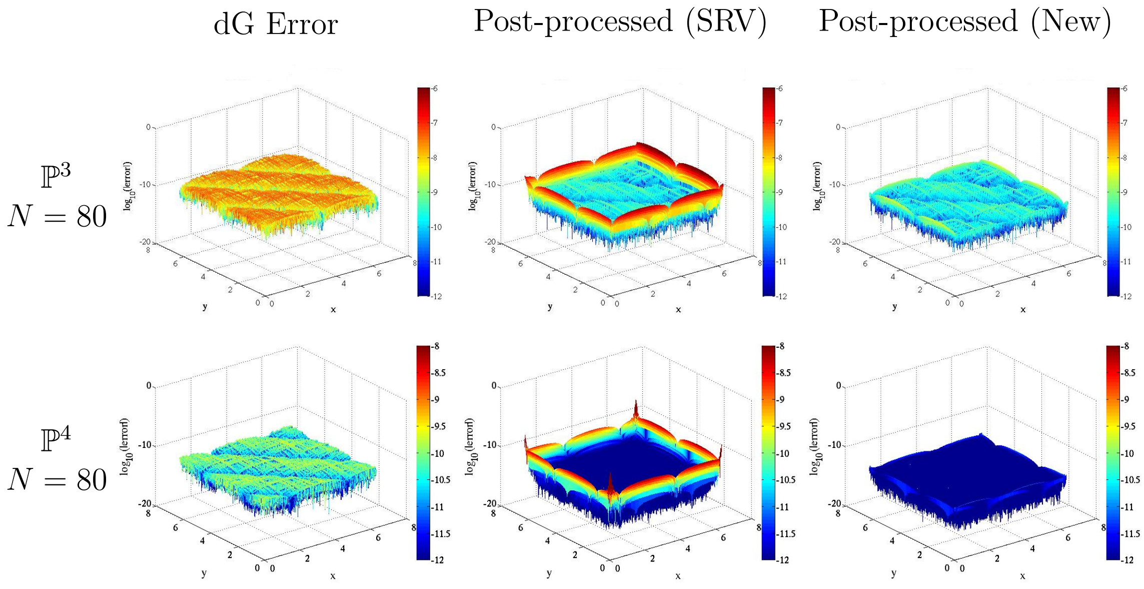


Figure 5.4: Comparison of the pointwise errors in log scale of the original dG solution (left), the SRV filter (middle) and the new filter (right) for the 2D linear transport equation (5.1) using polynomials of degree $k = 3, 4$ over a randomly-varying 80×80 mesh. The filters use the scaling of $H_x = \Delta x_j$ in x-direction and $H_y = \Delta y_j$ in y-direction. Double precision was used in the computations.

References

[1] C. de Boor, "A Practical Guide to Splines". Revised Edition. Springer-Verlag, New York, 2001, pp. 87-108.

[2] J.H. Bramble and A.H. Schatz, "Higher Order Local Accuracy by Averaging in the Finite Element Method", Mathematics of Computation, **31** (1977), pp. 94-111.

[3] B. Cockburn, "Discontinuous Galerkin Methods for Convection-Dominated Problems", High-Order Methods for Computational Physics, vol. 9 of Lecture Notes in Computational Science and Engineering, Springer, 1999.

[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", Mathematics of Computation, **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 II: One Dimensional Systems", Journal of Computational Physics, **84** (1989), pp. 90-113.

[6] B. Cockburn, M. Luskin, C.-W. Shu, and E. Süli, "Enhanced Accuracy by Post-Processing for Finite Element Methods for Hyperbolic Equations", Mathematics of Computation, **72** (2003), pp. 577-606.

[7] S. Curtis, R.M. Kirby, J.K. Ryan, and C.-W. Shu, "Post-Processing for the Discontinuous Galerkin Method over Nonuniform Meshes", SIAM Journal on Scientific Computing, **30** (2007), pp. 272-289.

[8] L. Ji, P. van Slingerland, J.K. Ryan, and C. Vuik, "Superconvergent Error Estimates for a Position-Dependent Smoothness-Increasing Accuracy-Conserving Filter for dG Solutions", Mathematics of Computation, **84** (2014), pp. 2239-2262

[9] James King, Hanieh Mirzaee, J.K. Ryan, and R.M. Kirby, "Smoothness-Increasing Accuracy-Conserving (SIAC) Filtering for Discontinuous Galerkin Solutions: Improved Errors Versus Higher-Order Accuracy", Journal of Scientific Computing, **53** (2012), pp. 129-149.

[10] H. Mirzaee, J. K. Ryan, and R. M. Kirby, "Quantification of Errors Introduced in the Numerical Approximation and Implementation of Smoothness-Increasing Accuracy-Conserving (SIAC) Filtering of Discontinuous Galerkin (dG) Fields", Journal of Scientific Computing, **45** (2010), pp. 447-470.

[11] H. Mirzaee, J. K. Ryan, and R.M. Kirby, "Efficient Implementation of Smoothness-Increasing Accuracy-Conserving Filters for Discontinuous Galerkin Solutions", Journal of Scientific Computing, **52** (2012), pp. 85-112.

[12] J. K. Ryan, "Local Derivative Post-Processing: Challenges for A Non-uniform mesh", Report 10-18, Delft University of Technology, Delft, The Netherlands, 2010.

[13] J.K. Ryan and B. Cockburn, "Local Derivative Post-processing for the discontinuous Galerkin method", Journal of Computational Physics, **228** (2009), pp. 8642 - 8664.

[14] J.K. Ryan and C.-W. Shu, "One-sided Post-processing for the Discontinuous Galerkin Methods", Methods and Applications of Analysis, **10** (2003), pp. 295-307.

[15] J. K. Ryan, C.-W. Shu and H. L. Atkins, "Extension of a Post-processing Technique for the Discontinuous Galerkin method for Hyperbolic Equations with Application to an Aeroacoustic Problem", SIAM Journal on Scientific Computing, **26**(2005), pp. 821-843.

[16] L. Schumaker, "Spline Functions: Basic Theory". Third Edition, Cambridge University Press, 2007.

[17] M. Steffan, S. Curtis, R.M. Kirby, and J.K. Ryan, "Investigation of Smoothness Enhancing Accuracy-Conserving Filters for Improving Streamline Integration through Discontinuous Fields", IEEE Transactions on Visualization and Computer Graphics, **14** (2008), pp. 680-692.

[18] M.S. Mock and P.D. Lax, "The Computation of Discontinuous Solutions of Linear Hyperbolic Equations", Comm. Pure Appl. Math., **31** (1978), pp. 423-430.

[19] V. Thomée, "High Order Local Approximations to Derivatives in the Finite Element Method", Mathematics of Computation, **31** (1977), pp. 652-660.

[20] P. van Slingerland, J.K. Ryan, and C. Vuik, "Position-Dependent Smoothness-Increasing Accuracy-Conserving (SIAC) Filtering for Accuracy for Improving discontinuous Galerkin solutions", SIAM Journal on Scientific Computing **33** (2011) pp. 802-825.

Table 4.2: L^2- and $L^\infty-$errors for the dG approximation together with the SRV and the new filters for the linear transport equation (4.3) over the three meshes 4.2,4.3,4.4. A scaling of $H = \Delta x_j$ along with quadruple precision was used in the computations.

	dG				SRV Filter				New Filter			
Mesh	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order
Mesh 1: Smoothly-Varying Mesh												
$\mathbb{P}^1$												
20	7.13E-03	–	1.97E-02	–	3.57E-03	–	5.83E-03	–	3.54E-03	–	5.29E-03	–
40	1.65E-03	2.11	5.43E-03	1.86	4.35E-04	3.04	6.42E-04	3.18	4.34E-04	3.03	6.42E-04	3.04
80	4.04E-04	2.03	1.43E-03	1.93	5.37E-05	3.02	7.94E-05	3.02	5.37E-05	3.02	7.94E-05	3.02
$\mathbb{P}^2$												
20	2.43E-04	–	1.22E-03	–	2.41E-04	–	1.51E-03	–	1.56E-04	–	7.32E-04	–
40	3.02E-05	3.01	1.55E-04	2.98	9.97E-07	7.92	9.73E-06	7.28	2.55E-06	5.94	2.29E-05	5.00
80	3.77E-06	3.00	1.95E-05	3.00	8.30E-09	6.91	1.34E-08	9.50	1.98E-07	3.68	2.13E-06	3.42
$\mathbb{P}^3$												
20	5.45E-06	–	1.87E-05	–	6.43E-06	–	5.51E-05	–	6.36E-05	–	2.02E-04	–
40	3.39E-07	4.01	1.20E-06	3.96	3.11E-07	4.37	3.25E-06	4.09	1.72E-07	8.53	7.62E-07	8.05
80	2.12E-08	4.00	7.48E-08	4.01	1.45E-10	11.06	1.81E-09	10.81	2.81E-10	9.26	2.16E-09	8.46
$\mathbb{P}^4$												
20	1.56E-07	–	5.20E-07	–	5.15E-07	–	4.11E-06	–	1.72E-05	–	5.41E-05	–
40	4.83E-09	5.01	1.66E-08	4.97	6.43E-09	6.32	6.56E-08	5.97	2.56E-08	9.39	1.12E-07	8.91
80	1.51E-10	5.00	5.22E-10	4.99	1.25E-11	9.01	1.80E-10	8.51	1.15E-11	11.13	7.77E-11	10.50
Mesh 2: Smooth Polynomial Mesh												
$\mathbb{P}^1$												
20	5.36E-03	–	1.68E-02	–	2.38E-03	–	3.48E-03	–	2.39E-03	–	3.48E-03	–
40	1.26E-03	2.10	4.69E-03	1.84	2.93E-04	3.02	4.37E-04	2.99	2.94E-04	3.03	4.37E-04	2.99
80	3.08E-04	2.03	1.23E-03	1.93	3.63E-05	3.01	5.43E-05	3.01	3.64E-05	3.01	5.43E-05	3.01
$\mathbb{P}^2$												
20	1.43E-04	–	8.00E-04	–	2.88E-05	–	2.45E-04	–	4.62E-05	–	2.18E-04	–
40	1.80E-05	2.99	1.03E-04	2.95	2.24E-06	3.68	2.43E-05	3.33	1.22E-06	5.24	9.29E-06	4.55
80	2.25E-06	3.00	1.31E-05	2.99	3.91E-07	2.52	6.37E-06	1.93	9.79E-08	3.64	1.37E-06	2.76
$\mathbb{P}^3$												
20	3.15E-06	–	1.25E-05	–	1.52E-05	–	1.75E-04	–	1.53E-05	–	7.35E-05	–
40	1.96E-07	4.01	8.05E-07	3.96	9.80E-08	7.27	1.50E-06	6.86	3.50E-08	8.77	2.34E-07	8.29
80	1.22E-08	4.00	4.97E-08	4.02	2.31E-09	5.40	3.77E-08	5.32	5.63E-11	9.28	8.26E-10	8.15
$\mathbb{P}^4$												
20	6.25E-08	–	2.67E-07	–	6.79E-07	–	5.38E-06	–	4.45E-06	–	2.13E-05	–
40	1.96E-09	5.00	8.77E-09	4.93	2.13E-09	8.32	2.34E-08	7.85	4.12E-09	10.08	2.76E-08	9.59
80	6.14E-11	5.00	2.79E-10	4.97	3.03E-11	6.13	5.01E-10	5.54	1.74E-12	11.21	1.59E-11	10.76
Mesh 3: Randomly-Varying Mesh												
$\mathbb{P}^1$												
20	4.88E-03	–	1.49E-02	–	2.31E-03	–	6.73E-03	–	2.18E-03	–	3.83E-03	–
40	1.15E-03	2.09	4.45E-03	1.74	2.84E-04	3.02	8.11E-04	3.05	2.76E-04	2.98	6.41E-04	2.58
80	2.87E-04	2.00	1.20E-03	1.89	4.85E-05	2.55	1.60E-04	2.35	4.71E-05	2.55	1.37E-04	2.23
$\mathbb{P}^2$												
20	1.17E-04	–	5.63E-04	–	3.78E-04	–	3.57E-03	–	2.23E-05	–	1.06E-04	–
40	1.52E-05	2.95	7.90E-05	2.83	3.35E-05	3.50	3.45E-04	3.37	9.58E-07	4.54	7.44E-06	3.83
80	1.93E-06	2.98	9.85E-06	3.00	8.04E-06	2.06	1.17E-04	1.56	1.27E-07	2.92	8.51E-07	3.13
$\mathbb{P}^3$												
20	2.49E-06	–	1.06E-05	–	3.35E-05	–	3.61E-04	–	5.64E-06	–	2.90E-05	–
40	1.55E-07	4.00	7.46E-07	3.83	7.42E-07	5.50	9.11E-06	5.31	6.23E-09	9.82	3.80E-08	9.58
80	1.02E-08	3.93	4.67E-08	4.00	2.46E-08	4.91	4.57E-07	4.32	1.54E-10	5.34	8.71E-10	5.45
$\mathbb{P}^4$												
20	4.03E-08	–	1.49E-07	–	1.40E-06	–	1.47E-05	–	1.52E-06	–	7.85E-06	–
40	1.37E-09	4.88	5.25E-09	4.83	1.42E-08	6.63	1.78E-07	6.36	3.20E-10	12.21	1.98E-09	11.95
80	4.40E-11	4.96	1.70E-10	4.95	4.03E-10	5.13	7.93E-09	4.49	3.68E-13	9.77	3.95E-12	8.97

Table 4.3: L^2- and $L^\infty-$errors for the dG approximation together with the SRV and the new filters for the variable coefficient equation (4.4) using a dG approximation of polynomial degree $k = 3, 4$ over the three meshes 4.2,4.3,4.4. The filters are using scaling $H = \Delta x_j$. Quadruple precision was used in the computations.

	dG				SRV Filter				New Filter			
Mesh	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order
Mesh 1: Smoothly-Varying Mesh												
$\mathbb{P}^3$												
20	5.54E-06	–	1.93E-05	–	4.40E-06	–	3.66E-05	–	6.36E-05	–	2.02E-04	–
40	3.41E-07	4.02	1.21E-06	4.00	3.14E-07	3.81	3.25E-06	3.49	1.72E-07	8.53	7.61E-07	8.05
80	2.12E-08	4.01	7.50E-08	4.01	1.45E-10	11.08	1.81E-09	10.81	2.78E-10	9.27	2.05E-09	8.53
$\mathbb{P}^4$												
20	1.62E-07	–	5.69E-07	–	1.89E-05	–	1.44E-04	–	1.72E-05	–	5.41E-05	–
40	4.95E-09	5.03	1.77E-08	5.00	5.74E-09	11.68	5.82E-08	11.28	2.56E-08	9.39	1.12E-07	8.91
80	1.53E-10	5.01	5.48E-10	5.02	1.26E-11	8.83	1.76E-10	8.37	1.16E-11	11.11	7.26E-11	10.59
Mesh 2: Smooth Polynomial Mesh												
$\mathbb{P}^3$												
20	3.15E-06	–	1.27E-05	–	2.70E-05	–	3.05E-04	–	1.53E-05	–	7.36E-05	–
40	1.96E-07	4.01	8.06E-07	3.98	1.31E-07	7.69	1.54E-06	7.62	3.55E-08	8.75	2.38E-07	8.28
80	1.22E-08	4.00	4.98E-08	4.02	7.51E-09	4.13	1.27E-07	3.60	6.25E-11	9.15	7.84E-10	8.24
$\mathbb{P}^4$												
20	6.40E-08	–	2.82E-07	–	2.95E-06	–	2.42E-05	–	4.45E-06	–	2.13E-05	–
40	1.98E-09	5.01	8.94E-09	4.98	6.84E-07	2.11	1.12E-05	1.10	4.12E-09	10.08	2.76E-08	9.60
80	6.18E-11	5.00	2.80E-10	5.00	1.51E-09	8.83	3.50E-08	8.33	1.59E-12	11.34	1.55E-11	10.80
Mesh 3: Randomly-Varying Mesh												
$\mathbb{P}^3$												
20	2.49E-06	–	9.61E-06	–	1.11E-04	–	8.98E-04	–	5.63E-06	–	2.90E-05	–
40	1.56E-07	4.00	7.18E-07	3.74	2.12E-06	5.71	2.55E-05	5.14	7.96E-09	9.47	4.31E-08	9.39
80	1.02E-08	3.93	4.72E-08	3.93	5.91E-08	5.17	1.06E-06	4.59	3.15E-10	4.66	1.91E-09	4.50
$\mathbb{P}^4$												
20	4.07E-08	–	1.56E-07	–	2.45E-05	–	1.96E-04	–	1.52E-06	–	7.85E-06	–
40	1.37E-09	4.89	5.31E-09	4.87	4.48E-07	5.77	6.18E-06	4.99	2.98E-10	12.31	1.79E-09	12.09
80	4.41E-11	4.96	1.73E-10	4.94	1.26E-09	8.48	1.91E-08	8.33	2.64E-12	6.82	2.19E-11	6.35

Table 5.1: L^2- and $L^\infty-$errors for the dG approximation together with the SRV and new filters for a 2D linear transport equation using polynomials of degree $k = 3, 4$. Double precision was used in the computations.

	dG				SRV Filter				New Filter			
Mesh	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order
$\mathbb{P}^3$												
20×20	3.30E-06	–	1.21E-05	–	2.60E-07	–	1.12E-04	–	2.39E-06	–	1.80E-05	–
40×40	2.06E-07	4.00	7.60E-06	3.99	4.69E-10	9.11	3.11E-09	5.17	7.01E-09	8.41	5.11E-08	8.46
80×80	1.29E-08	4.00	4.76E-08	4.00	1.74E-11	4.75	5.50E-09	-0.82	7.97E-11	6.46	1.02E-09	5.65
$\mathbb{P}^4$												
20×20	4.71E-08	–	1.41E-07	–	2.77E-08	–	1.35E-06	–	5.25E-07	–	3.77E-06	–
40×40	1.46E-09	5.01	4.50E-09	4.97	2.55E-08	0.12	2.84E-06	-1.07	3.83E-10	10.42	3.40E-09	10.11
80×80	4.44E-11	5.04	1.43E-10	4.98	2.73E-08	-0.10	7.86E-06	-1.47	3.00E-13	10.31	3.12E-12	10.09

Table 5.2: L^2- and $L^\infty-$errors for the dG approximation together with the SRV and new filters for the 2D linear transport equation (5.1) using polynomials of degree $k = 3, 4$ over the three meshes: Mesh 4.2, Mesh 4.3 and Mesh 4.4. The filters use the scaling of $H_x = \Delta x_j$ in x-direction and $H_y = \Delta y_j$ in y-direction. Double precision was used in the computations.

	dG				SRV Filter				New Filter			
Mesh	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order	L^2 error	order	L^∞ error	order
Mesh 1: Smoothly-Varying Mesh												
$\mathbb{P}^3$												
20×20	8.74E-06	–	5.39E-05	–	6.94E-06	–	1.10E-04	–	6.71E-05	–	4.04E-04	–
40×40	5.45E-07	4.00	3.39E-06	4.00	3.68E-07	4.24	6.49E-06	4.08	2.09E-07	8.33	1.66E-06	7.93
80×80	3.40E-08	4.00	2.06E-07	4.03	1.50E-10	11.76	9.01E-09	9.49	7.33E-10	8.16	7.76E-09	8.92
$\mathbb{P}^4$												
20×20	1.93E-07	–	1.05E-06	–	9.25E-07	–	8.56E-06	–	3.26E-05	–	1.12E-04	–
40×40	6.00E-09	5.01	3.32E-08	4.98	3.38E-08	4.77	4.17E-06	1.04	2.67E-08	10.25	2.31E-07	8.92
80×80	1.88E-10	5.00	1.04E-09	5.00	2.07E-08	0.71	9.13E-06	-1.13	1.90E-11	10.46	1.61E-10	10.49
Mesh 2: Smooth Polynomial Mesh												
$\mathbb{P}^3$												
20×20	4.56E-06	–	3.03E-05	–	1.55E-05	–	3.49E-04	–	1.59E-05	–	1.38E-04	–
40×40	2.85E-07	4.00	1.92E-06	3.98	1.23E-07	6.98	3.04E-06	6.84	4.67E-08	8.41	5.14E-07	8.67
80×80	1.78E-08	4.00	1.20E-07	4.00	3.35E-09	5.20	8.01E-08	5.25	2.43E-10	7.59	4.61E-09	6.80
$\mathbb{P}^4$												
20×20	8.48E-08	–	3.27E-07	–	1.38E-06	–	1.48E-05	–	5.92E-06	–	3.27E-05	–
40×40	2.65E-09	5.00	1.74E-08	4.92	3.21E-08	5.43	4.99E-06	1.57	4.65E-09	10.30	5.79E-08	9.14
80×80	8.31E-11	5.00	5.58E-10	4.96	2.51E-08	0.35	6.88E-06	-0.46	3.29E-12	10.46	3.49E-11	10.27
Mesh 3: Randomly-Varying Mesh												
$\mathbb{P}^3$												
20×20	3.47E-06	–	2.16E-05	–	3.46E-05	–	6.43E-04	–	3.90E-06	–	3.83E-05	–
40×40	2.23E-07	3.96	1.52E-06	3.83	1.90E-06	4.19	3.59E-05	4.16	1.28E-08	8.25	1.25E-07	8.26
80×80	1.41E-08	3.98	9.65E-08	3.98	9.94E-08	4.26	2.71E-06	3.73	2.97E-10	5.43	3.88E-09	5.01
$\mathbb{P}^4$												
20×20	5.83E-08	–	2.84E-07	–	3.06E-06	–	5.19E-05	–	1.06E-06	–	8.87E-05	–
40×40	1.90E-09	4.94	1.04E-08	4.77	2.64E-08	6.86	2.64E-06	4.30	7.80E-10	10.41	1.01E-08	9.78
80×80	6.06E-11	4.97	3.48E-10	4.90	1.46E-08	0.85	7.09E-06	-1.43	6.60E-13	10.20	7.85E-12	10.33