Numerical Testing of a New Positivity-Preserving Interpolation Algorithm

T. A. J. Ouermi
University of Utah School of Computing
U of U Scientific Computing and Imaging Institute
touermi@cs.utah.edu

Robert M. Kirby
University of Utah School of Computing
U of U Scientific Computing and Imaging Institute
kiby@cs.utah.edu

Martin Berzins
University of Utah School of Computing
U of U Scientific Computing and Imaging Institute
mb@sci.utah.edu

September 17, 2020

Abstract

An important component of a number of computational modeling algorithms is an interpolation method that preserves the positivity of the function being interpolated. This report describes the numerical testing of a new positivity-preserving algorithm that is designed to be used when interpolating from a solution defined on one grid to different spatial grid. The motivating application is a numerical weather prediction (NWP) code that uses spectral elements as the discretization choice for its dynamics core and Cartesian product meshes for the evaluation of its physics routines. This combination of spectral elements, which use nonuniformly spaced quadrature/collocation points, and uniformly-spaced Cartesian meshes combined with the desire to maintain positivity when moving between these necessitates our work. This new approach is evaluated against several typical algorithms in use on a range of test problems in one or more space dimensions. The results obtained show that the new method is competitive in terms of observed accuracy while at the same time preserving the underlying positivity of the functions being interpolated.

Introduction

Interpolating from one grid to another is a fundamental part of a number of computational problems. Furthermore, when interpolating solution values, it is important in some applications to conserve properties such as non-negativity of the solution. For example in weather forecasting, when mapping between the different grids used to calculate the dynamics and those used to calculate the physics, the polynomial approximation of positive quantities such as mass, density, or cloud mixing ratio may introduce negative values that are nonphysical. These negative values may cause incorrect representations of other calculations that are dependent on these approximations. In other applications such as parallel resilience [1], combustion simulations and the solution of hyperbolic equations with ENO and WENO schemes, it is important to have polynomial approximations that preserve positivity on general meshes [2, 3]. A somewhat stronger condition is to use methods that preserve the local boundedness of the solution [4] when interpolating.

The particular example motivating this experimental study is the physics-dynamics coupling (PDC) module in the Navy Environmental Prediction System using the NUMA Core (NEPTUNE) [5, 6]. NEPTUNE is a next-generation global NWP system being developed at the Naval Research Laboratory (NRL) and the Naval Postgraduate School (NPS) [6]. In NEPTUNE [5, 6], the dynamics calculations are done on a spectral element mesh, whereas the physics routines require values on a uniform mesh. In the context of NEPTUNE and similar codes, interpolating the solution values produced by the dynamics routines to the spatial points used by the physics routines and vice versa may lead to negative values unless care is taken. This result was shown by Skamarock et al. [7] who demonstrated that not preserving positivity may lead to unphysical results in a predicted physical quantity of interest, such as moisture. Moreover, the nonphysical values introduced through interpolation may lead to spurious values in the results, compared to those produced with positivity-preserving interpolation [8].

This report is concerned with the numerical testing of a new interpolation algorithm that has been proposed for positivity preservation inside the NEPTUNE code. The theoretical basis for the method extends the data-bounded work of Berzins [4] to arbitrary cases and meshes and is further extended to give a new positivity-preserving approach in [9]. The application of this new method to numerical weather prediction is described in [8]. In this report, a number of possible alternative interpolation schemes are introduced. A representative sample of such methods is compared against the new approach on a number of different test functions, including smooth, C^0, discontinuous, and steep-gradient functions. The comparison undertaken focuses on how accurately the different methods are able to represent this underlying set of test functions. In addition, a representative weather model problem is considered in the context of NEPTUNE in [8]. Overall, it will be shown that the new method both preforms well and is suited to the C^0 continuity of the spectral element methods in NEPTUNE.

1 Examples of Existing Interpolation Methods

This section highlights several approaches that have been developed to address the need for data-bounded, positivity-preserving and shape-preserving interpolation. While this selection of methods is not all-inclusive, it is intended to illustrate the main types of polynomial-based approaches.

1.1 Cubic Splines

In Computer-Aided Design (CAD), graphics, and visualization, significant contributions have been made to develop and advance shape-preserving methods. Many of the approaches for shape-preservation are based on cubic splines. In [10] and [11], Schmidt and Heß introduced positive interpolation methods using rational quadratic and cubic splines respectively. Necessary and sufficient conditions for positivity are provided for both the rational quadratic and cubic interpolants. These conditions impose some restrictions on the values of the first derivatives at each node. As in [12], both approaches lead to multiple solutions, and the one with the minimal curvature is selected. The work in [13], [14], and [15] presented positivity-preserving interpolation methods that rely on rational cubic splines. The C^2 continuity in [13] is obtained by solving a tridiagonal system of linear equations. All three methods introduce free parameters that are used to derive and enforce conditions for positivity. Butt and Brodlie [16] provide a method for constructing C^1 cubic Hermite splines. This method is dependent on the availability of values of first derivatives at the nodes, which may not be available in practice. Positivity is enforced by imposing a bound on the values of the derivatives. In the case where bounds on the derivatives are not met, one or two knots are inserted to ensure that the constructed spline is positive. Perhaps the most widely used approach for preserving monotonicity in many applications is PCHIP by Fritch and Carlson [17] who derived necessary and sufficient conditions for monotone cubic interpolation, and provided an algorithm for building a piecewise cubic approximation from data. This algorithm calculates the values of the first derivatives at the nodes based on the necessary and sufficient conditions.

1.2 Quartic and Quintic Splines

Although many shape-preserving interpolation methods are cubic or lower order, a number of approaches target higher-order interpolants, with an emphasis on quartic or quintic polynomial approximations. The work in [18] and [19] presents geometric or visual continuity G^1 and G^2 continuous shape-preserving interpolation using Pythagorean-Hodograph quintic splines curves. This approach uses Bernstein basis functions and a parametric representation of the interpolant in each interval. A sufficient condition for shape preservation is constructed based on free angular parameters that influence the shape of the curve in each interval. The appropriate angular parameters are selected based on the cubic-cubic (CC) criterion introduced in [20]. The G^2 case requires a tridiagonal solve and use of a Newton-Raphson iteration, which potentially affects the computational performance.

Hussain et al. [21] and Hussain et al. [22] introduced C^2 rational quintic interpolation interpolation approaches that preserve positivity. These rational quintic functions are constructed with free parameters that are used to enforce positivity. These methods also both require the approximation of values of first and second derivatives at the nodes if these derivatives are not available. In addition, the rational quintic interpolation methods in [21] and [22] have a $O(h^3)$ order of accuracy.

Heß and Schmidt [12] developed interpolation schemes that preserve positivity and monotonicity using C^2 quartic and quintic splines. Positivity and monotonicity are achieved by imposing some restrictions on the values of the first and second derivatives at each node. This approach leads to a potentially infinite number of solutions that meet the required conditions. Of these solutions, the solution with minimal curvature is selected using global minimization. The global nature of the minimization makes the algorithm challenging to parallelize and may have an impact on computational performance. MQS [23] is an example of a monotonic quintic spline method that was developed by Lux et al. [23] who built on the work of Heß and Schmidt [24], and Ulrich and Watson [25]. This algorithm uses the sufficient conditions from [24] to check for monotonicity and the work in [25] to adjust values of the first and second derivatives to ensure monotonicity. This method requires the values of the first and second derivatives at the nodes, which may not be available in practice. In this report, the first and second derivatives are approximated using a fourth-order finite difference stencil based on [26].

1.3 SPS and B-spline Higher Order Splines

Costantini [27, 28] developed a C^1 and C^2 Shape-Preserving Spline (SPS) interpolation method using Berstein-Bezier polynomials of an arbitrary degree. The desired shape property is obtained by imposing restrictions on the value of the first derivatives at the nodes. The Bezier coefficients for each spline are derived from a linear function. For a given interval, the coefficients of the Berstein-Bezier polynomial interpolant are selected from a linear function. The first derivatives at the nodes are calculated such that the sufficient conditions for shape preservation given in [27] are met. The approximation of the first derivatives at the node is third-order accurate. In addition, Theorem 9 of [29] shows that the spline method presented in [27, 28] has an error of $O(h^4)$. More details on the construction of the splines, an algorithm and a software package for the SPS method can be found in [27, 28]. In addition to the positivity-preserving approaches, conventional B-splines [30] are also used here. Although the B-spline approach does not preserve-positivity, many of the approaches mentioned in this work are based on B-splines and so the use of unmodified B-splines provides an accuracy check on the other spline methods.

1.4 DBI and PPI Methods

The numerical solution of partial differential equations (PDEs), particularly hyperbolic equations, is an another area in which various methods have been developed to enable data-bounded and positivity-preserving approximations. In order to preserve positivity in discontinuous Galerkin (dG) schemes, Zhang et al. [31, 32, 33] and Light et al. [34] introduced a linear rescaling of polynomials

that ensures that the evaluation of the polynomial at the quadrature points is positive. In addition, this linear rescaling of the polynomial conserves mass. The polynomial rescaling, however, does not address the case of interpolating between different meshes, which is the primary focus of this work. Harten et al. [35] developed an essentially nonoscillatory (ENO) piece-wise polynomial reconstruction that is suitable for interpolating between different meshes. ENO methods adaptively build an interpolant based on Newton divided differences and can help remove Gibbs-like effects but do not guarantee positivity. A weighted combination of ENO schemes, (WENO) has been used by Zhang et al. [36] and many others. The Data-Bounded Interpolation (DBI) method was developed by Berzins using evenly spaced meshes from ENO methods [4]. This method was extended by the authors to work for both unevenly spaced meshes and, more importantly, to the Positivity-Preserving Interpolation (PPI) methods used in this report. Sufficient conditions that rely on bounding the ratio of successive divided differences in the polynomial approximation as used by Berzins [4] for uniform meshes are extended in [9] to prove data-boundedness in general. The PPI method further extends the DBI method by relaxing the bounds on the ratio of divided differences and so allows the interpolant to grow beyond the data, while still remaining positive. For a given interval, the DBI and PPI methods successively select stencil points based on the ENO method until the required bounds are violated or $d+1$ points are selected, with d being the target degree of the interpolant. Both the DBI and PPI algorithms are described in [9] and numerical examples pertaining to NEPTUNE [5, 6] and NWP are studied in [8].

2 Comparison Methodology

2.1 Compared Methods

The numerical experiments in this report with cubic splines use PCHIP [17], MSQ [23], Shape-Preserving Splines (SPS) [27, 28], B-splines [30], DBI [4], and the new PPI methods, developed by the authors [9]. These methods are available as follows: **PCHIP:** The version of the PCHIP algorithm used in this report is implemented in Fortran 90 and can be found at `https://people.sc.fsu.edu/~jburkardt/f_src/pchip/pchip.html`.

MQS: The method of Lux et al. [23] is an example of a method for monotonic quintic splines. The algorithm is implemented in Python3 and can be found `https://github.com/tchlux/papers/tree/master/%5B2019-11%5D_HPC_(quintic_spline)`.

SPS: Costantini [27, 28] introduced a high-order Shape-Preserving (monotonicity-, and convexity-preserving) Spline (SPS) method using Berstein-Bezier polynomials of arbitrary degree. The SPS method is implemented in the BVSPIS software package in Fortran 77 and is available from ACM as Algorithm 770 [28] `https://dl.acm.org/action/downloadSupplement?doi=10.1145%2F264029.264059&file=770.gz&download=true`.

B-splines PPPACK, a Fortran 90 library that evaluates piecewise polynomial functions, including cubic splines. The original FORTRAN77 library is by Carl de Boor [30]. The package is available from `https://people.sc.fsu.edu/~jburkardt/f_src/pppack/pppack.html`.

DBI PPI: The DBI and PPI methods have been both developed by the authors and exist only in prototype software form at present.

2.2 Comparison Criteria

The three steps outlined below are used to compare the different methods when used to approximate smooth and nonsmooth functions. The errors are measured in a discrete approximation to the L^2 error norm.

- The first step consists of demonstrating that the various schemes preserve positivity for each of the test functions used. In addition, this step is used to show that a standard polynomial interpolation method does not guarantee positivity.
- The second step experimentally investigates the convergence of the various schemes when using smooth functions. This step tests the ability of the different methods to accurately represent smooth functions as the resolution increases. For the Shape-Preserving Spline (SPS) [27, 28], DBI and PPI methods, we also investigate the approximation accuracy obtained with varying interpolant polynomial degrees.
- The third step focuses on the ability of the different methods to represent a set of challenging test functions with large gradients and/or discontinuities. This step represents situations often encountered in computational science problems, such as mapping between physics and dynamics meshes in NEPTUNE.

3 Positivity-Preserving Interpolants

Preserving positivity while maintaining accuracy is perhaps the key property needed when mapping from one mesh to another in NEPTUNE and similar applications. This section compares the PCHIP, MQS, data-bounded, and positivity-preserving methods against a standard interpolation method using three examples. The standard polynomial interpolation approach (STD) uses the points in each element to build a standard Lagrange interpolant for that element. In each of the examples, the different interpolants are constructed using both a uniform and an LGL mesh. The LGL mesh consists of uniform elements with Legendre Gauss-Lobatto (LGL) quadrature nodes [37] inside each element. In the figures presented in this section, the black and red plots represent the underlying function and its approximation using the different interpolation methods. The results in Figures 1 to 6 below demonstrate that the PCHIP, MQS, DBI, and PPI methods preserve positivity, whereas the standard interpolation methods lead to oscillations and fail to preserve positivity. Using an LGL mesh reduces oscillations compared to the uniform mesh, but does not guarantee that the interpolating polynomials will be positive.

3.1 Example I $f_1(x)$

This example uses the famous Runge function [38] defined as follows:

$$f_1(x) = \frac{1}{1+25x^2}, \quad x \in [-1, 1]. \tag{1}$$

Figures 1 and 2 show the different polynomial approximations for this function using 17 uniformly spaced and LGL points, respectively. The target polynomial degree for the standard interpolation, DBI, and PPI is set to $d = 16$. The standard polynomial interpolation approach, STD, does not preserve positivity with the uniform mesh and generates oscillations in both meshes. The PCHIP, MQS, DBI, SPS and PPI methods preserve positivity for both the uniform and LGL meshes.

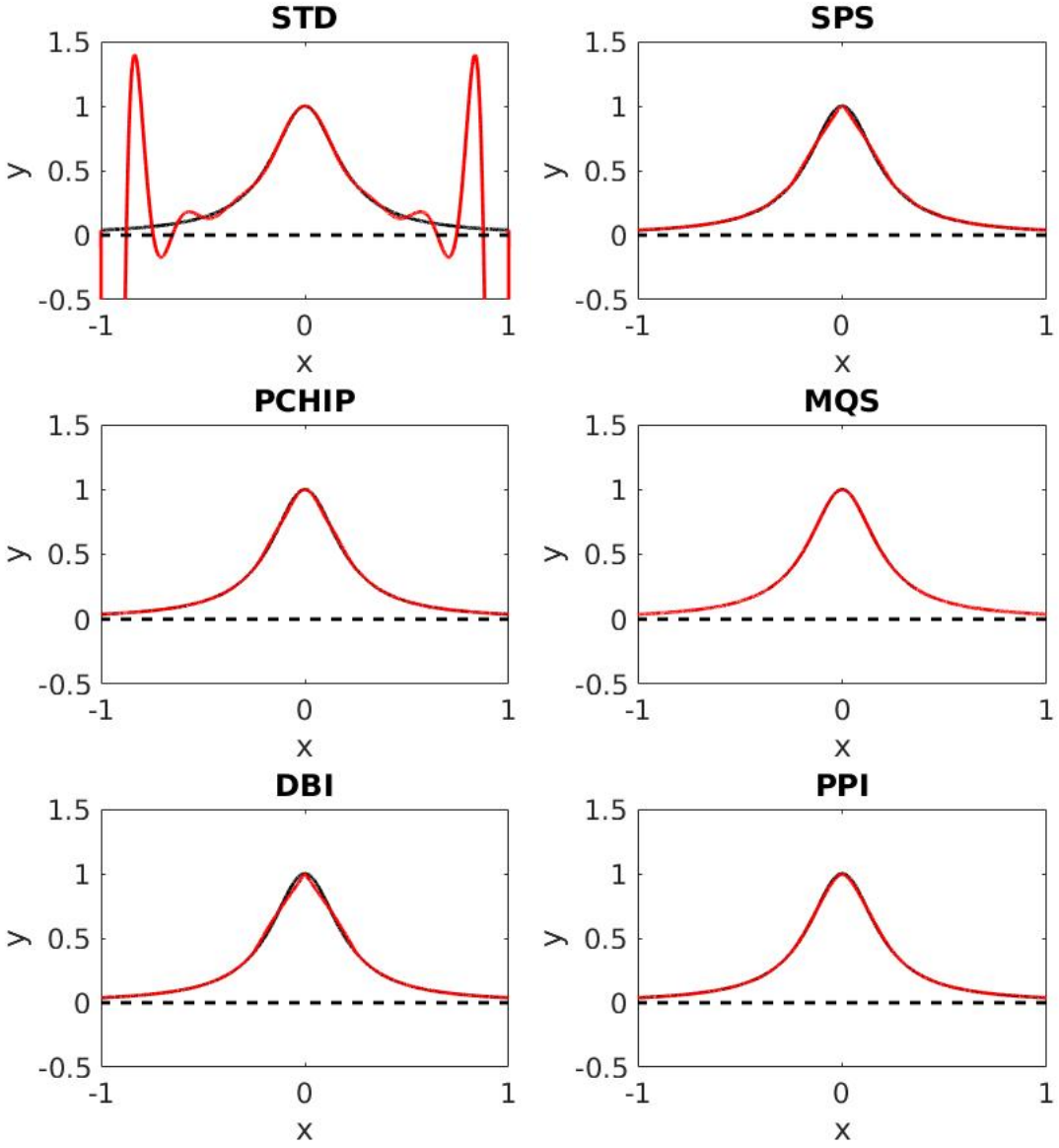

Figure 1: Approximation of the Runge function with the $N = 17$ points that are uniformly distributed on the interval $[-1, 1]$.

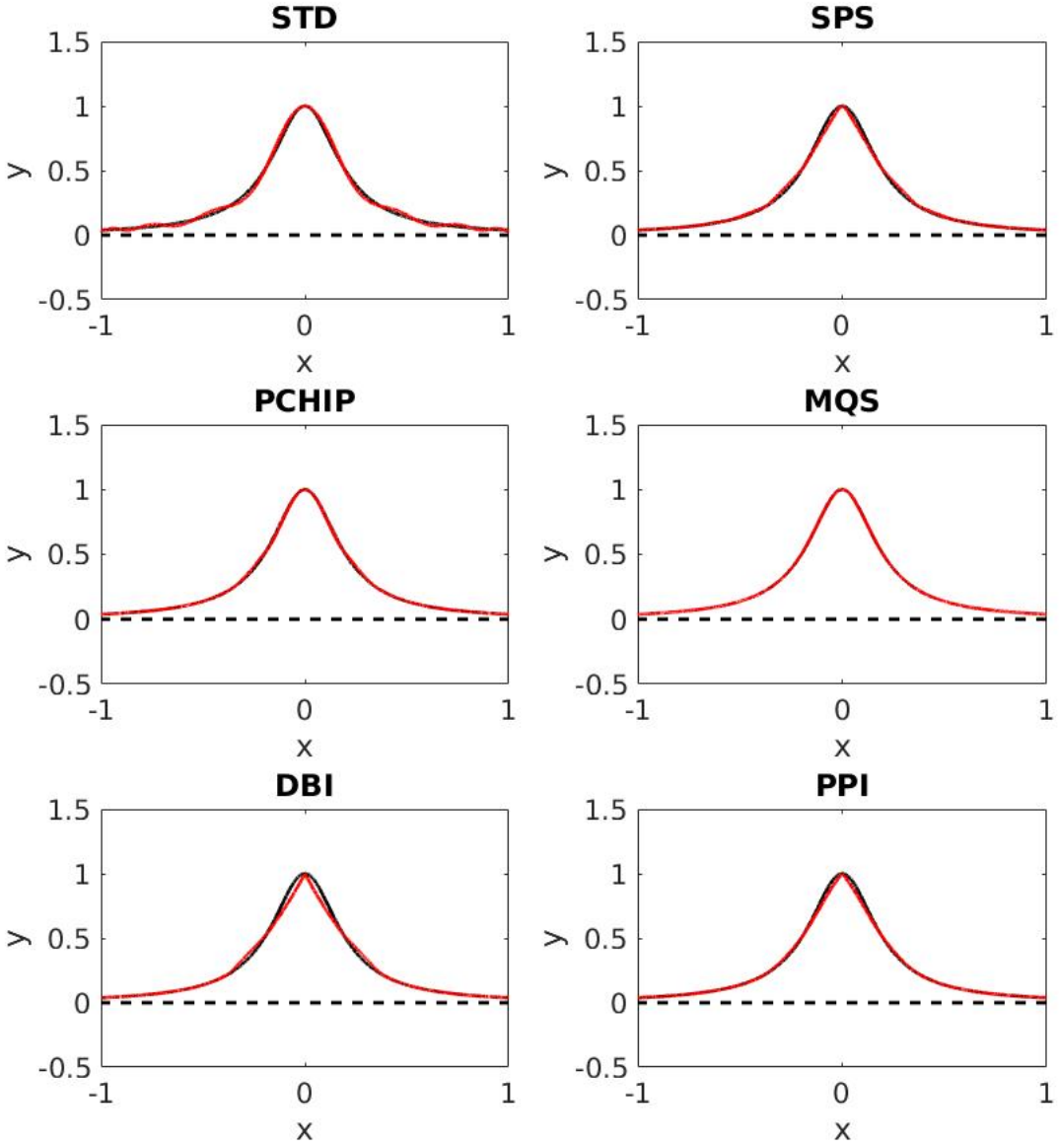

Figure 2: Approximation of the Runge function with the $N = 17$ LGL quadrature points distributed on the interval $[-1, 1]$.

3.2 Example II $f_2(x)$

The second example uses an analytic approximation of the Heaviside function defined as follows:

$$f_2(x) = \frac{1}{1 + e^{-2kx}}, \quad k = 100, \text{ and } x \in [-0.2, 0.2]. \tag{2}$$

A polynomial approximation of $f_2(x)$ is challenging because of the large gradient at about $x = 0$. Attempts to use a global polynomial approximation for this function result in unacceptable oscillations and negative values as also observed in the Runge example above. Figures 3 and 4 show interpolations of $f_2(x)$ using a uniform mesh of 17 points and an LGL mesh with two elements each with nine LGL quadrature points in each element. Standard polynomial interpolation, DBI, and PPI are used with an interpolant of degree $d = 8$ for each interval. Standard polynomial interpolation, STD, fails to preserve positivity in both uniform and LGL meshes. The results demonstrate that the PCHIP, MQS, DBI, SPS and PPI methods preserve positivity with both the uniform and LGL meshes.

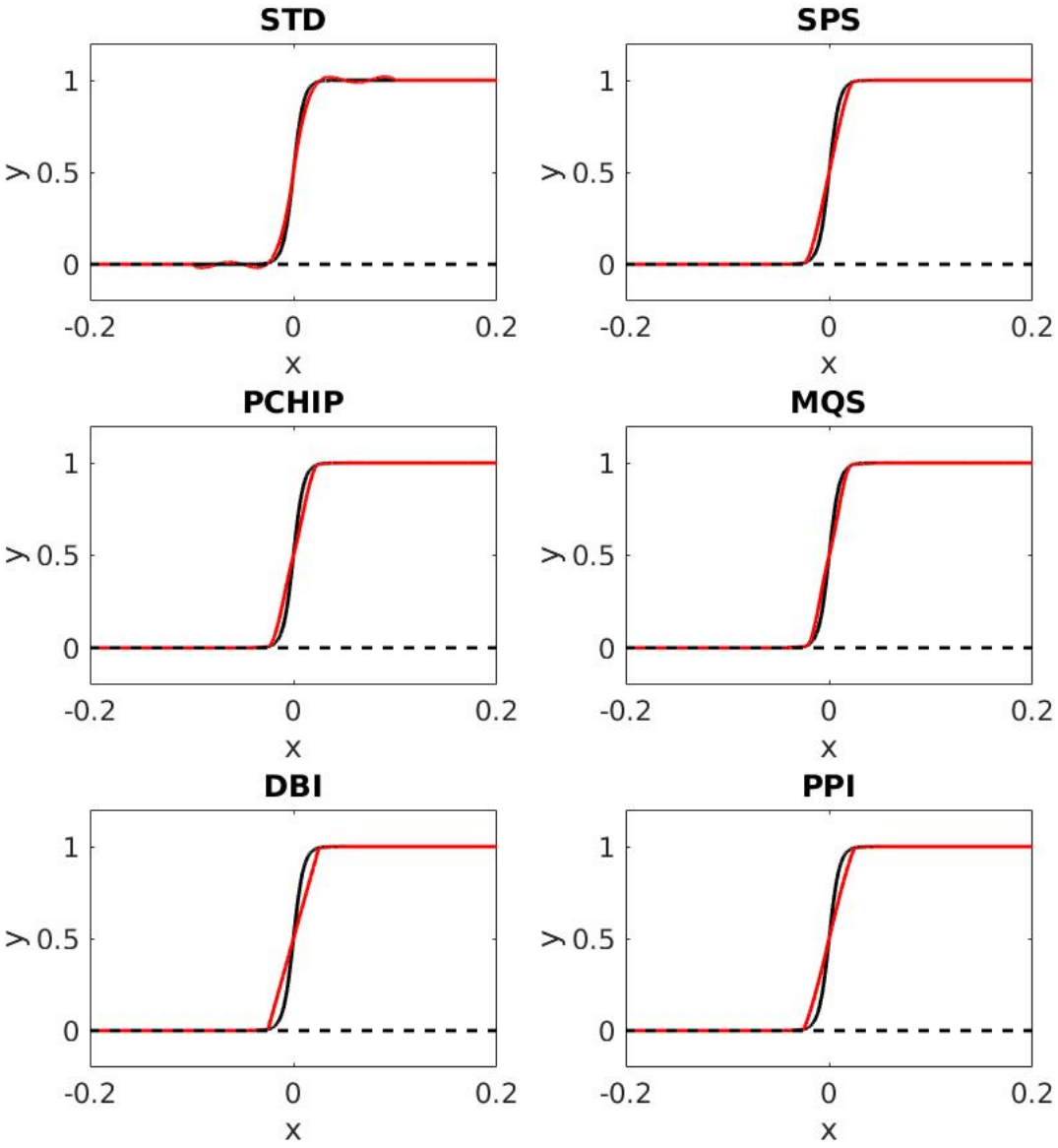

Figure 3: Approximation of $f_2(x) = \frac{1}{1+e^{-2kx}}$, $k = 100$, and $x \in [-0.2, 0.2]$, with $N = 17$ points. The points are uniformly distributed and the target polynomial degree for the DBI and PPI is $d = 8$.

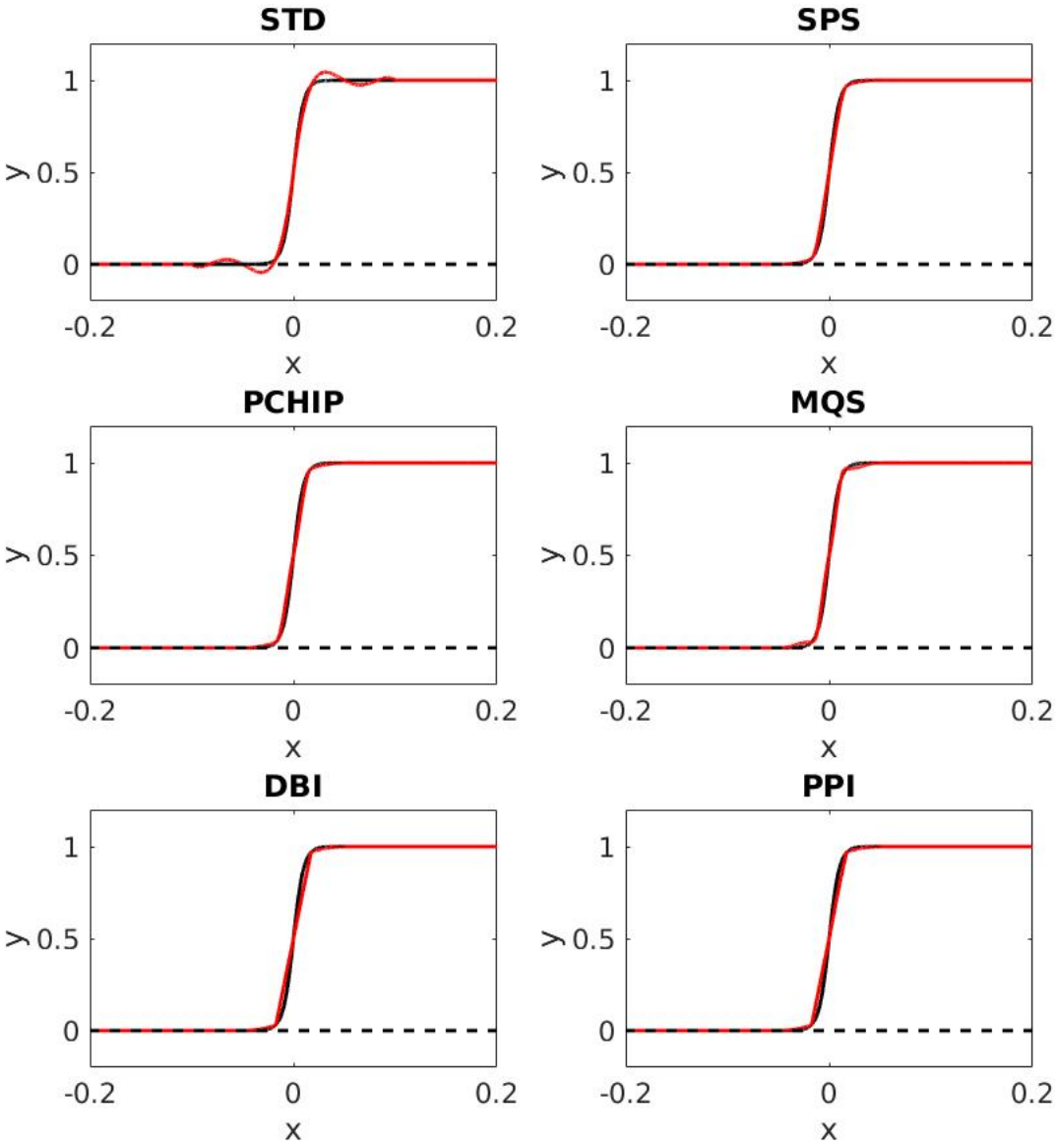

Figure 4: Approximation of $f_2(x) = \frac{1}{1+e^{-2kx}}, \quad k = 100$, and $x \in [-0.2, 0.2]$, with $N = 17$ points. The interval $[-0.2, 0.2]$ is divided in two elements and 9 LGL quadrature points are used in each interval.

3.3 Example III $f_3(x)$

The third example uses a modified version of a function introduced by Tadmor and Tanner [39] and used by Berzins [4] in the context of DBI based upon uniform mesh points. The original function was modified by adding the value one to ensure that the function is positive over the interval $[-1, 1]$.

The modified function is defined as

$$f_3(x) = \begin{cases} 1 + \frac{2e^{2\pi x} - 1 - e^{\pi}}{e^{\pi} - 1}, & x \in [-1, -0.5) \\ \\ 1 - sin\big(\frac{2\pi x}{3} + \frac{\pi}{3}\big), & x \in [-0.5, 1]. \end{cases} \tag{3}$$

This function is particularly challenging because of the discontinuity at $x = -0.5$. This example uses uniform and LGL meshes of 17 points. The LGL mesh consists of four elements and LGL quadrature points are used as the mesh points within each element. The target interpolant degree for the standard interpolation, DBI, and PPI method is $d = 4$. Figures 5 and 6 demonstrate that the interpolants built using the PCHIP, MQS, DBI,SPS and PPI methods remain positive whereas the standard polynomial interpolation approach fails to preserve positivity. In addition, the oscillations observed with the standard polynomial interpolation method are more pronounced with the uniform mesh compared to the LGL mesh.

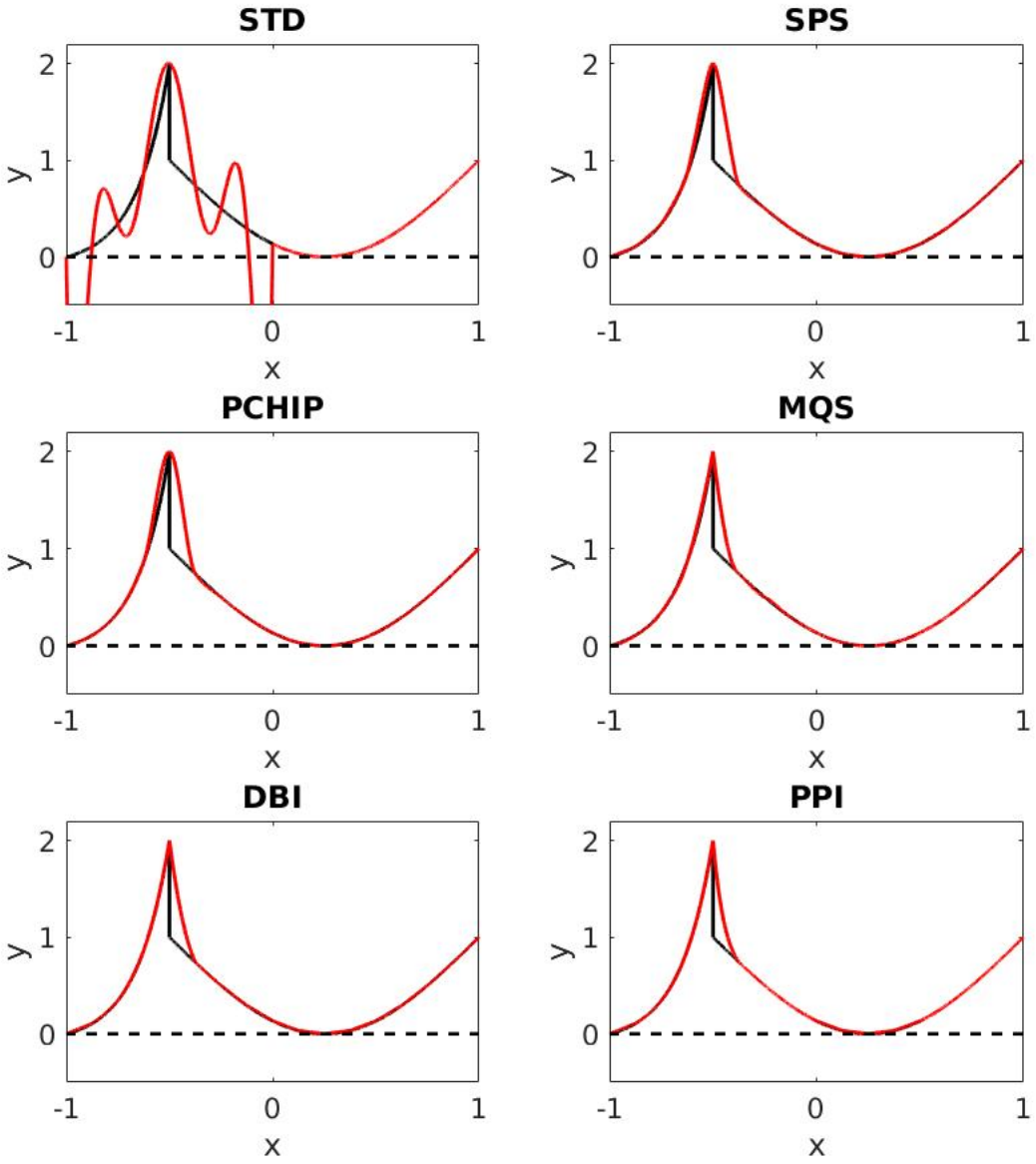

Figure 5: Approximation of $f_3(x)$ with $N = 17$ points. The points are distributed uniformly over the interval $[-1, 1]$.

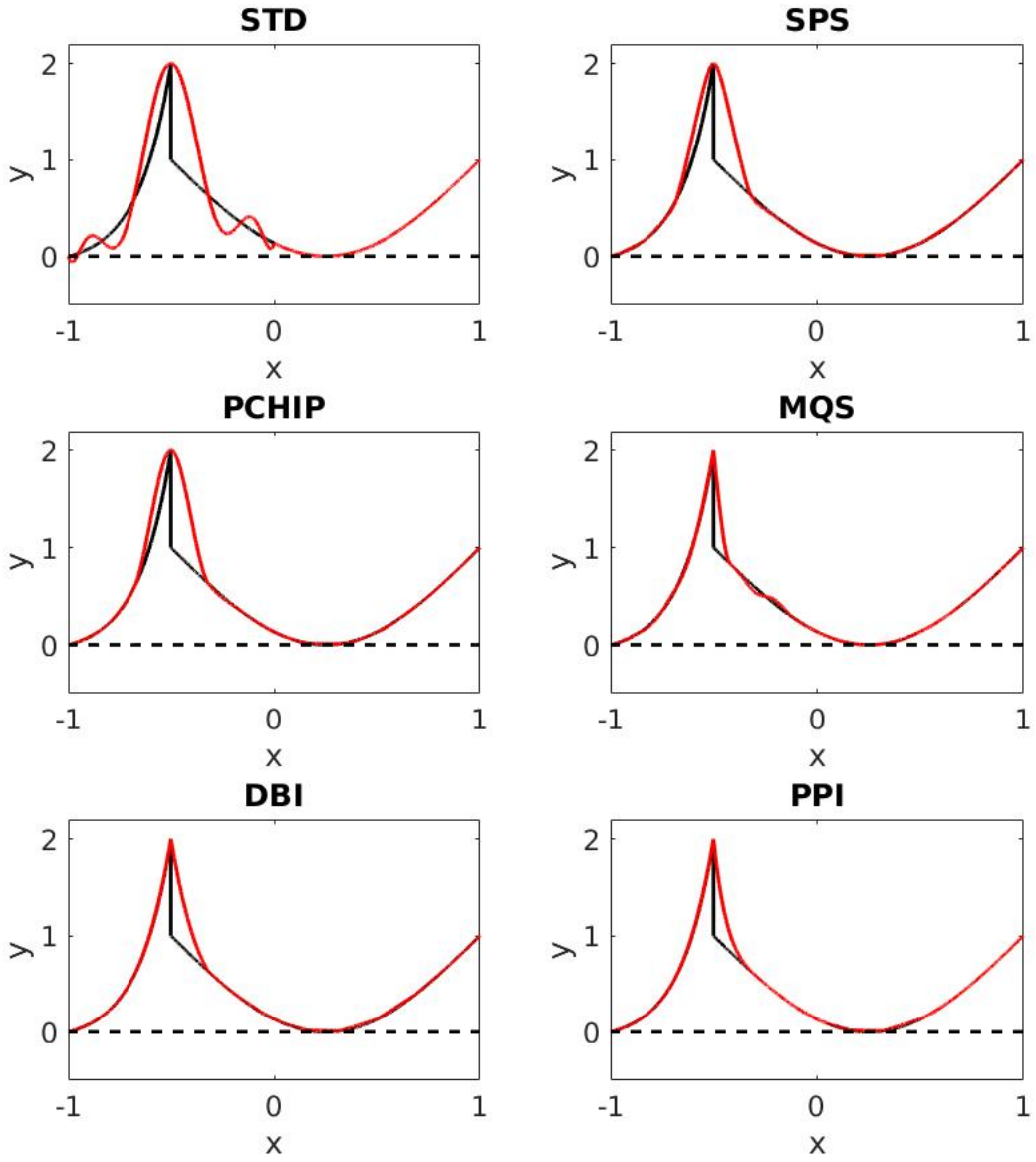

Figure 6: Approximation of $f_3(x)$ with $N = 17$ points. The interval $[-1, 1]$ is divided into four elements, and $d + 1$ LGL quadrature points are used in each element.

3.4 Example IV $f_4(x)$

This example consists of a function with multiple spikes defined as follows:

$$f_4(x) = 1.0 - \left| \frac{2}{\pi} arctan\left(\frac{sin\left(\pi \frac{x}{h}\right)}{\delta} \right) \right|, \quad x \in [0, 1] \tag{4}$$

where h represent the element size, and $\delta = 0.01$. $f_4(x)$ depends on the element size h and therefore, on the number of element in a given interval. At the element boundaries, $f_4(x)$ is C^0-continuous with large gradients of opposite signs. This example uses 33 points, four elements, and nine points in each element. The approximations in Figures 7 and 8 use uniform and LGL quadrature points, respectively. The plots in Figures 7 and 8 show the standard polynomial interpolation approach lead to oscillation and negative values, whereas the PCHIP, MQS, DBI,SPS and PPI methods preserve positivity and remove the oscillations.

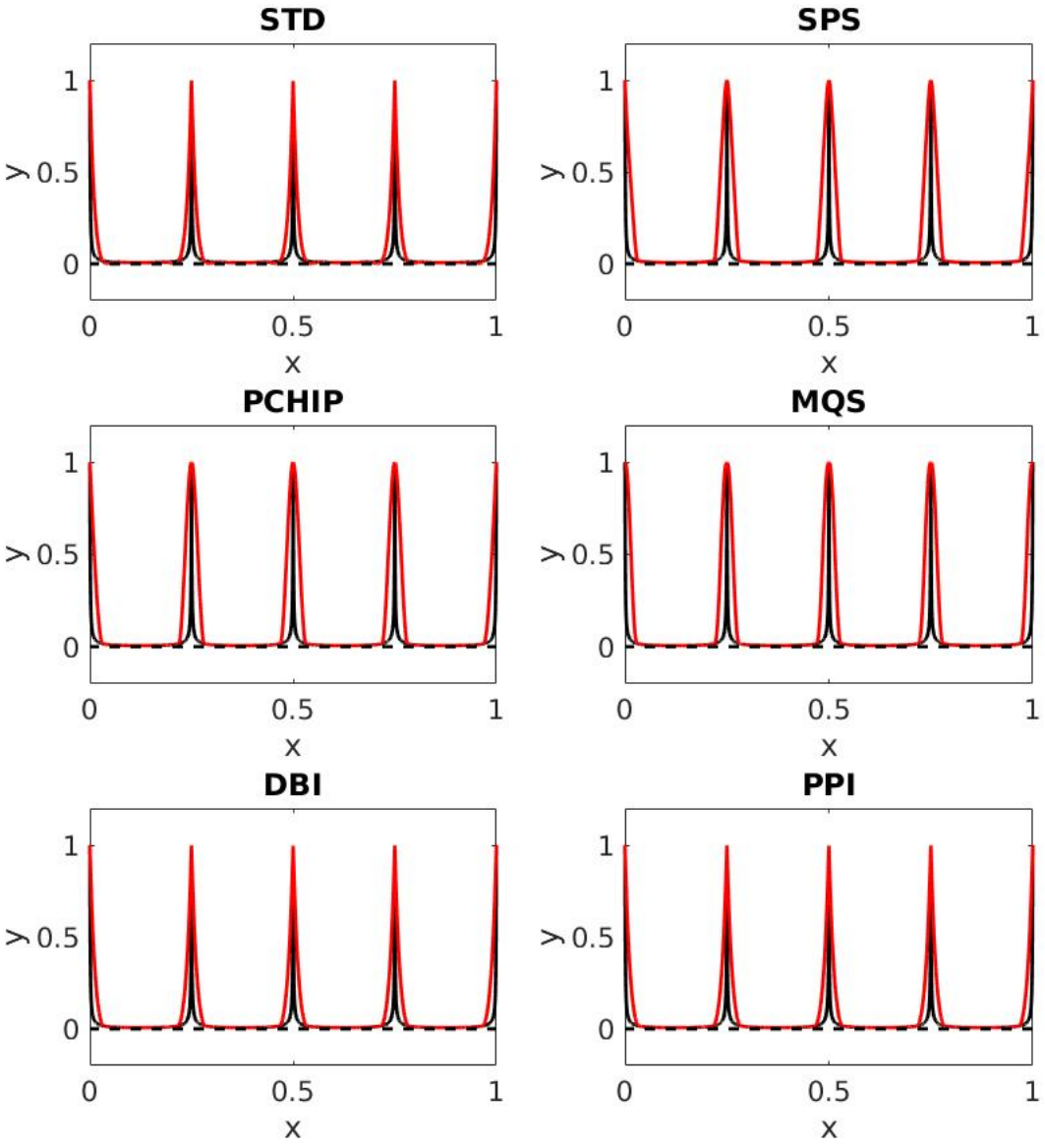

Figure 7: Approximation of $f_4(x)$, with $N = 33$ points. The points are uniformly distributed, and the target polynomial degree for the DBI and PPI is $d = 8$.

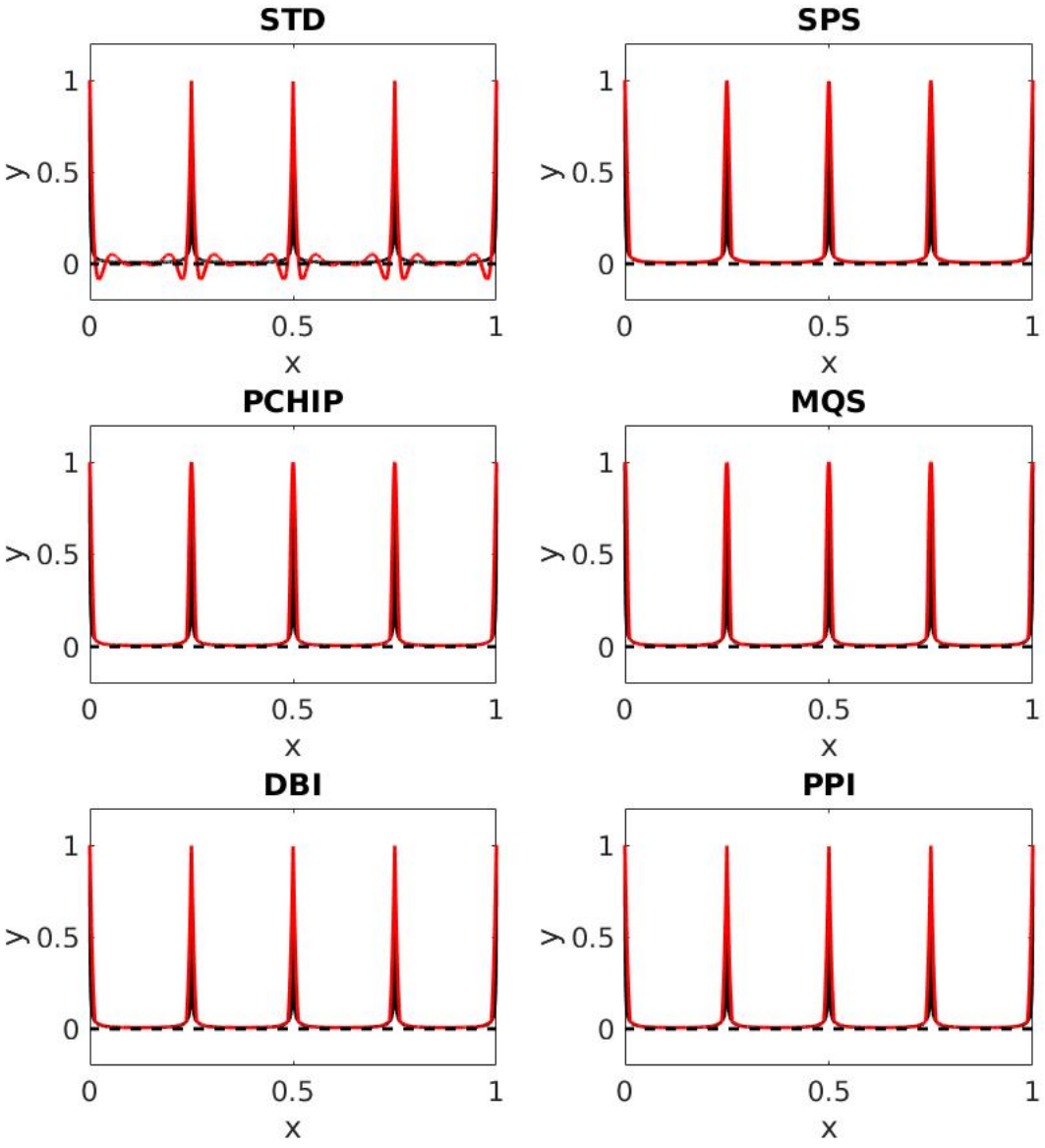

Figure 8: Approximation of $f_4(x)$, with $N = 33$ points. The interval $[0, 1]$ is divided into four elements, and 9 quadrature points are used in each interval.

3.5 Example V $f_5(x)$

This example is constructed using the *tanh* function and by introducing C^0-continuities at the elements boundaries. The constructed function is defined as follows:

$$f_5(x) = \begin{cases} tanh(xk) & \text{if } x \in [a, a+h] \\ 2tanh(xk) - tanh((a+h)k) & \text{if } x \in [a+h, a+2h] \\ 3tanh(xk) - tanh((a+h)k) - tanh((a+2h)k) & \text{if } x \in [a+2h, a+3h] \\ \quad \vdots & \end{cases} \tag{5}$$

where the overall interval is $[-2, 0]$ with $a = -2$ and $k = 10$. h represents the size of each element. $f_5(x)$ depends on the element size h and, therefore, on the number of elements in a given interval. This example is built to mirror the C^0-continuity at the elements boundaries in the spectral element method used in NEPTUNE. In this example, the gradients at the elements boundaries are always positive, and are not as large as the ones in $f_4(x)$ from Example IV. The approximations shown in Figures 9 and 10 use 17 points, four elements, and five points inside each element. The plots in Figures 9 and 10 show that the standard interpolation method does not preserve positivity and that the PCHI, MQS, DBI, and PPI can be used to enforce positivity as required.

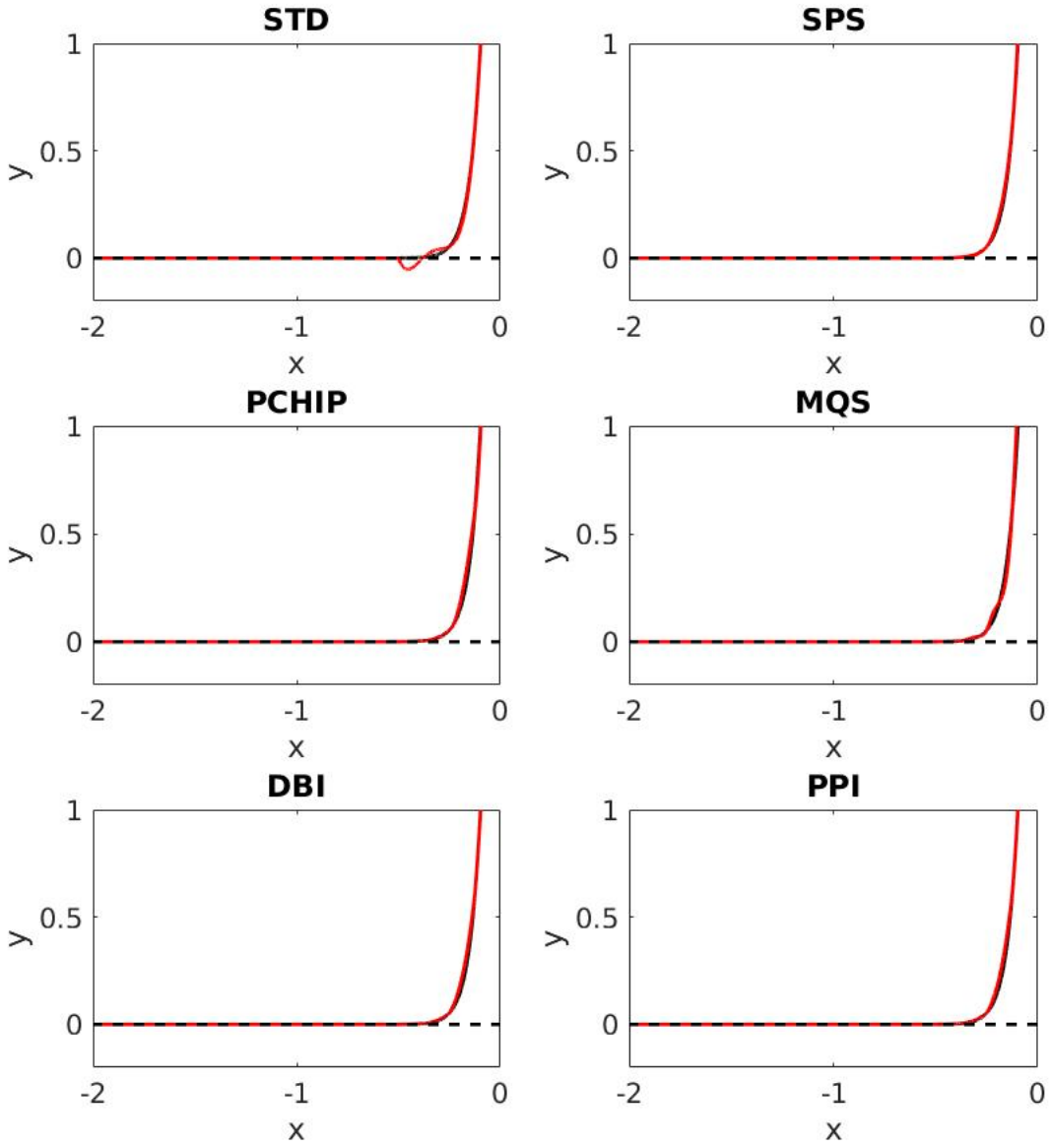

Figure 9: Approximation of $f_5(x)$, with $N = 17$ points. The points are uniformly distributed and the target polynomial degree for the DBI and PPI is $d = 4$.

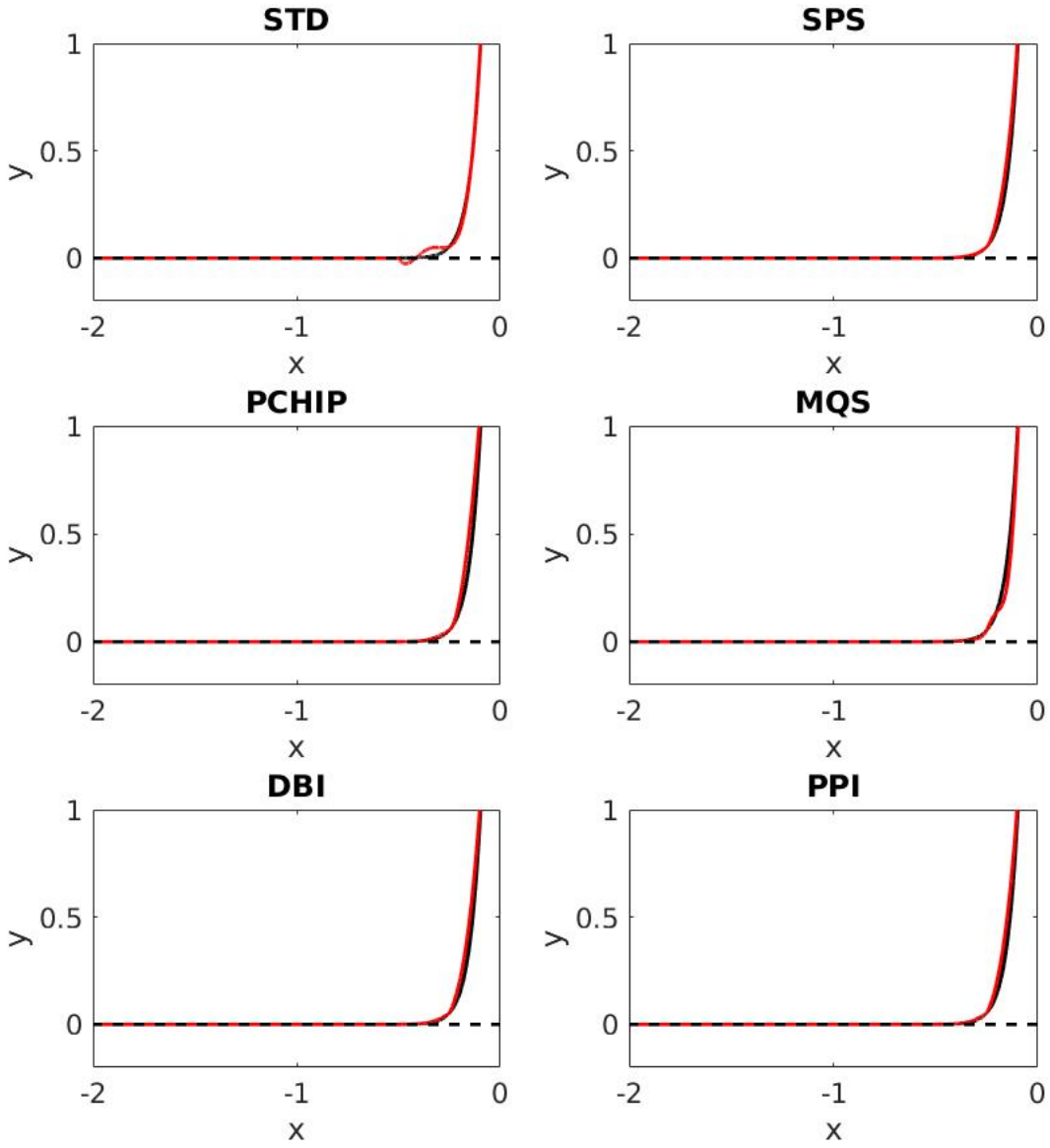

Figure 10: Approximation of $f_5(x)$, with $N = 17$ points. The interval $[-2, 0]$ is divided in four elements and 5 quadrature points are used in each interval.

4 Convergence

This section focuses on the second comparison criterion, which consists of evaluating the convergence of the different methods when applied to a smooth function. As NUMA [5], the dynamics part of NEPTUNE, uses a spectral element method that has high-order accuracy, especially in smooth regions, it is important when interpolating solution values between dynamics and physics

meshes for the interpolation scheme to not degrade the accuracy obtained from the spectral element method.

The test function

$$f_6(x) = 1 + sin(x), x \in [0, \pi] \tag{6}$$

is used to study the convergence of the different methods. $f_6(x)$ is infinitely smooth with no sharp gradients or discontinuities. These characteristics make $f_6(x)$ a suitable test function for evaluating which approach is a good choice for representing smooth functions. These experiments focus on the accuracy of the approximation as the resolution and the polynomial degree both increase.

Table 1 shows L^2-errors when approximating $f_6(x)$ using the different interpolation methods. In all cases the L^2-error is estimated by sampling the error at 10000 equally spaced points in the interval and using trapezoidal quadrature.

Table 2 shows the ratio, $e_{N_i}/e_{N_{i+1}}$ of the L^2-errors in Table 1 as the resolution increases. The DBI method lead to higher L^2-errors compared to the other methods. The average polynomial degree used for each interpolant is almost linear, as shown in the sixth column in Table 1. These result indicate that the bound of Berzins [4] on the ratio of divided differences for enforcing data-boundedness is too restrictive when it comes to enforcing positivity. The SPS method shows smaller errors than DBI but larger than the remaining methods. Furthermore, as the polynomial degree increases, the accuracy of the approximation decreases. These results are consistent with those in [27, 28]. Costantini [29, 27] demonstrated that the SPS method is bounded by $O(h^4)$ and in the limit (as the spline degree increases) the spline tends to a linear interpolation. Both the SPS and DBI methods have about the same rate of convergence, as observed in Table 2. The B-spline and PPI methods have smaller L^2-errors compared to the other methods, and their accuracy improves as the polynomial degree increases. Table 2 shows that both methods have better convergence rates compared to PCHIP, MQS, and SPS. The PPI method lead to slightly smaller errors compared to the unmodified B-spline approach. For $\mathcal{P}_8$ and $\mathcal{P}_{16}$ the approximation errors are close to machine precision, which explains the slow rate of convergence observed for B-spline and PPI in Table 2.

N_i	PCHIP	MQS	SPS	B-spline	DBI		PPI	
	L^2-error	L^2-error	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
					$\mathcal{P}_1$			
17	–	–	–	–	4.41E-3	1	4.41E-3	1
33	–	–	–	–	1.10E-3	1	1.10E-3	1
65	–	–	–	–	2.76E-4	1	2.76E-4	1
129	–	–	–	–	6.89E-5	1	6.89E-5	1
257	–	–	–	–	1.72E-5	1	1.72E-5	1
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$					
17	7.96E-4	3.16E-5	4.84E-4	4.52E-6	4.41E-3	1.13	4.56E-6	4
33	1.39E-4	5.25E-6	1.20E-4	2.07E-7	1.10E-3	1.06	1.25E-7	4
65	2.45E-5	9.25E-7	3.01E-5	1.22E-8	2.76E-4	1.03	3.50E-9	4
129	4.34E-6	1.63E-7	7.52E-6	7.56E-10	6.89E-5	1.01	1.02E-10	4
257	7.69E-7	2.89E-8	1.88E-6	4.72E-11	1.72E-5	1.01	3.07E-12	4
			$\mathcal{P}_8$					
17	–	–	2.00E-3	2.45E-9	4.41E-3	1.13	1.87E-9	8
33	–	–	4.96E-4	3.47E-12	1.10E-3	1.06	3.25E-12	8
65	–	–	1.24E-4	6.11E-15	2.76E-4	1.03	6.12E-15	8
129	–	–	3.10E-5	3.23E-15	6.89E-5	1.01	1.56E-15	8
257	–	–	7.74E-6	2.93E-15	1.72E-5	1.01	1.52E-15	8
			$\mathcal{P}_{16}$					
17	–	–	3.10E-3	5.61E-15	4.41E-3	1.13	4.08E-15	16
33	–	–	7.75E-4	4.35E-13	1.10E-3	1.06	2.36E-15	16
65	–	–	1.94E-4	2.75E-13	2.76E-4	1.03	5.21E-15	16
129	–	–	4.84E-5	9.00E-14	6.89E-5	1.01	4.74E-15	16
257	–	–	1.21E-5	6.83E-14	1.72E-5	1.01	1.82E-15	16

Table 1: L^2-errors when using the PCHIP, MQS, SPS, B-splines, DBI, and PPI methods to approximate the function $f_6(x)$. N_i represents the number of input points used to build the approximation. P_j represents the space of polynomials of degree j, with j being the target degree for each interval. The seventh and ninth columns show the average polynomial degree used for the DBI and PPI methods, respectively. The input points are uniformly distributed over the interval $[0, \pi]$.

$e_{N_i}/e_{N_{i+1}}$	PCHIP	MQS	SPS	B-spline	DBI	PPI
					$\mathcal{P}_1$	
e_{17}/e_{33}	–	–	–	–	4.01	4.01
e_{33}/e_{65}	–	–	–	–	3.99	3.99
e_{65}/e_{129}	–	–	–	–	4.01	4.01
e_{129}/e_{257}	–	–	–	–	4.01	4.01
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$			
e_{17}/e_{33}	5.73	6.01	4.03	21.84	4.01	36.48
e_{33}/e_{65}	5.67	5.68	3.99	16.97	3.99	35.71
e_{65}/e_{129}	5.65	5.65	4.00	16.13	4.01	34.31
e_{129}/e_{257}	5.64	5.64	4.00	16.02	4.01	33.22
			$\mathcal{P}_8$			
e_{17}/e_{33}	–	–	4.03	706.05	4.01	575.38
e_{33}/e_{65}	–	–	4.00	567.92	3.99	532.05
e_{65}/e_{129}	–	–	4.00	1.89	4.01	3.92
e_{129}/e_{257}	–	–	4.01	1.10	4.01	1.02
			$\mathcal{P}_{16}$			
e_{17}/e_{33}	–	–	4.01	0.01	4.01	1.73
e_{33}/e_{65}	–	–	3.99	1.58	3.99	0.45
e_{65}/e_{129}	–	–	4.01	3.06	4.01	1.10
e_{129}/e_{257}	–	–	4.00	1.32	4.01	2.60

Table 2: Ratio of L^2-errors from Table 1 ($e_{N_i}/e_{N_{i+1}}$). N_i represents the number of input points used to build the approximation. P_j represents the space of polynomials of degree j, with j being the target degree for each interval.

5 Results

In this section, the different interpolation methods are used to approximate functions with steep gradients, C^0-continuity, and discontinuities. These experiments focus on the third criterion, which consists of evaluating the ability of the different methods to represent nonsmooth functions. The data points for the interpolation are sampled from 1D and 2D functions. Two types of meshes are used for the various experiments. The first type of mesh uses uniform elements and uniformly spaced nodes within each element. The second type of mesh uses uniform elements and Legendre Gauss-Lobatto (LGL) quadrature nodes [37] within each element. The experimental results compare the PPI method against the SPS, DBI, PCHIP [17], and MQS [23] methods.

The MQS algorithm is designed for monotonically increasing data. In order to use the MQS approach with the different 1D examples, we divide the data into monotonically increasing and decreasing regions. For the monotonically increasing data, the MQS algorithm is applied directly.

For the monotonically decreasing data, we uses the reflection of the data about a vertical axis and applied the MQS algorithm. Because of the data transformation involved, the MQS method is used only for the 1D examples.

Tables 3 – 8 and 14 – 17 show L^2-errors when using the different methods to approximate the 1D and 2D functions, respectively. The 'avg. deg.' columns in these tables represent the average polynomial degree used in the DBI and PPI method.

5.1 Example I $f_1(x)$

This example is the 1D Runge function [38] defined in Equation 1. Tables 3 and 4 demonstrate that the PPI method gives smaller approximation errors when compared to the other approaches. The requirements of data boundedness in the DBI method are restrictive compared to the positivity requirements in PPI. These restrictions lead to lower average polynomial degrees for DBI compared to PPI, as shown in the sixth and eighth columns in Tables 3 and 4. As the average degree used by DBI increases the L^2-errors remain the same. The approximation error using the DBI method does not improve as the average degree increases because the global error is dominated by the local error of those subintervals with low degree interpolants. The polynomial degree of the interpolants used for these intervals remains the same as the average polynomial degree of the interpolant increases elsewhere. In the worst case scenario, the L^2-errors from DBI and linear interpolation are the same. The PPI methods uses higher order interpolants compared to the SPS, DBI, PCHIP, and MQS methods in both the uniform and LGL meshes. The uniform mesh leads to slightly more accurate results than the LGL mesh. These results show that the PPI method is a suitable approach for interpolating data from one mesh to another in cases where the underlying function is similar to the Runge function.

N_i	PCHIP	MQS	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
				$\mathcal{P}_1$			
17	–	–	–	2.16E-2	1	2.16E-2	1
33	–	–	–	6.02E-3	1	6.02E-3	1
65	–	–	–	1.52E-3	1	1.52E-3	1
129	–	–	–	3.82E-4	1	3.82E-4	1
257	–	–	–	9.56E-5	1	9.56E-5	1
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$				
17	7.15E-3	4.04E-3	4.83E-3	2.16E-2	2.63	8.51E-3	4
33	1.91E-3	2.69E-5	7.98E-4	5.64E-3	2.94	7.00E-4	4
65	3.70E-4	4.56E-5	1.70E-4	1.42E-3	3.09	2.55E-5	4
129	6.79E-5	3.72E-6	4.19E-5	3.60E-4	3.13	8.01E-7	4
257	1.22E-5	4.82E-7	1.05E-5	9.00E-5	3.15	2.55E-8	4
			$\mathcal{P}_8$				
17	–	–	1.21E-2	2.16E-2	2.75	5.89E-3	7.13
33	–	–	2.74E-3	5.64E-3	3.88	1.61E-4	8
65	–	–	6.86E-4	1.42E-3	4.59	1.16E-6	8
129	–	–	1.72E-4	3.60E-4	4.84	3.47E-9	8
257	–	–	4.30E-5	9.00E-5	4.98	7.55E-12	8
			$\mathcal{P}_{16}$				
17	–	–	1.64E-2	2.16E-2	2.75	5.76E-3	7.31
33	–	–	4.25E-3	5.64E-3	3.88	8.57E-5	15.59
65	–	–	1.07E-3	1.42E-3	4.84	6.22E-8	16
129	–	–	2.69E-4	3.57E-4	5.40	4.65E-12	16
257	–	–	6.71E-5	9.00E-5	5.38	2.45E-16	16

Table 3: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the Runge function $f_1(x) = \frac{1}{1+25x^2}, \quad x \in [-1, 1]$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The sixth and eighth columns show the average polynomial degree used for the DBI and PPI methods, respectively. The input points are uniformly distributed over the interval $[-1, 1]$.

N_i	PCHIP	MQS	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
				$\mathcal{P}_1$			
17	–	–	–	2.16E-2	1	2.16E-2	1
33	–	–	–	6.02E-3	1	6.02E-3	1
65	–	–	–	1.52E-3	1	1.52E-3	1
129	–	–	–	3.82E-4	1	3.82E-4	1
257	–	–	–	9.56E-5	1	9.56E-5	1
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$				
17	1.02E-2	4.44E-3	9.73E-3	2.53E-2	2.50	8.37E-3	4
33	1.86E-3	6.46E-4	1.63E-3	5.13E-3	2.94	8.84E-4	4
65	3.68E-4	5.99E-5	2.24E-4	1.97E-3	3.06	4.39E-5	4
129	7.20E-5	4.41E-6	6.03E-5	5.04E-4	3.12	1.30E-6	4
257	1.52E-5	4.05E-7	1.48E-5	1.27E-4	3.15	3.96E-8	4
			$\mathcal{P}_8$				
17	–	–	8.44E-3	1.59E-2	2.38	3.22E-3	6.75
33	–	–	2.69E-3	5.28E-3	3.94	1.72E-4	8
65	–	–	7.59E-4	1.57E-3	4.53	2.87E-6	8
129	–	–	2.61E-4	5.50E-4	4.83	9.44E-9	8
257	–	–	6.85E-5	1.42E-4	4.98	4.23E-11	8
			$\mathcal{P}_{16}$				
17	–	–	2.29E-2	3.93E-2	3	2.15E-2	11
33	–	–	3.26E-3	3.10E-3	3.63	9.83E-6	15.06
65	–	–	1.11E-3	1.13E-3	4.81	9.65E-8	16
129	–	–	3.12E-4	4.15E-4	5.30	5.25E-11	16
257	–	–	1.08E-4	1.45E-4	5.16	3.32E-15	16

Table 4: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the Runge function $f_1(x) = \frac{1}{1+25x^2}, \quad x \in [-1, 1]$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The sixth and eight columns show the average polynomial degree used for the DBI and PPI methods respectively. The interval [-1,1] is divided into $(N_i - 1)/j$ and $j + 1$ LGL quadrature points are used in each element.

5.2 Example II $f_2(x)$

The second example uses the analytic approximation of the Heaviside function defined in Equation 2. As mentioned in Section 3.2, this function, $f_2(x)$, is challenging because of the sharp gradient around $x = 0$. For polynomial degree five or less, the results from Tables 5 and 6 suggest that the MQS method leads to slighter better approximations than PPI for $f_2(x)$. The fifth columns

in Tables 5 and 6 indicate that the average degree used by the DBI approach increases but the L^2-errors remain the same. The global error of the DBI approach is dominated by the local error of the subintervals with low degree interpolants, and the interpolants used for these sub-intervals remain the same as we increase the target degree. Overall, the results from Tables 5 and 6 indicate, the PPI method has smaller L^2-errors compared to the other methods. Approximating $f_2(x)$ from data on a uniform mesh leads to slightly better results compared to LGL mesh data. For smooth data with a large gradient, these results indicate that the PPI approach is suitable for interpolating from one mesh to another.

N_i	PCHIP	MQS	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
				$\mathcal{P}_1$			
17	–	–	–	2.89E-2	1	2.89E-2	1
33	–	–	–	7.69E-3	1	7.69E-3	1
65	–	–	–	1.80E-3	1	1.80E-3	1
129	–	–	–	4.58E-4	1	4.58E-4	1
257	–	–	–	1.15E-4	1	1.15E-4	1
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$				
17	2.02E-02	1.67E-2	1.82E-2	2.89E-2	2.94	2.41E-02	3.56
33	3.38E-03	4.16E-3	3.72E-3	7.41E-3	3.47	6.26E-03	3.70
65	3.59E-04	2.29E-4	3.40E-4	1.10E-3	3.67	3.95E-04	3.84
129	4.21E-05	7.48E-6	5.36E-5	2.69E-4	3.76	1.55E-05	3.99
257	5.12E-06	2.16E-7	1.27E-5	6.78E-5	3.65	5.21E-07	3.97
			$\mathcal{P}_8$				
17	–	–	3.75E-3	2.89E-2	3.69	2.41E-02	5.20
33	–	–	5.24E-3	7.38E-3	5.84	5.93E-03	6.50
65	–	–	8.71E-4	1.10E-3	6.61	1.52E-04	7.39
129	–	–	2.08E-4	2.69E-4	6.70	1.56E-06	7.77
257	–	–	5.17E-5	6.78E-5	6.52	5.07E-09	7.89
			$\mathcal{P}_{16}$				
17	–	–	5.90E-3	2.89E-2	3.69	2.41E-02	5.19
33	–	–	6.34E-3	7.38E-3	7.16	5.92E-03	9.41
65	–	–	1.30E-3	1.10E-3	10.70	1.47E-04	13.36
129	–	–	3.23E-4	2.69E-4	10.69	1.96E-07	15.05
257	–	–	8.08E-5	6.78E-5	10.06	3.19E-11	15.31

Table 5: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_2(x) = \frac{1}{1+e^{-2kx}}, \quad k = 100$, and $x \in [-0.2, 0.2]$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The sixth and eighth columns show the average polynomial degree used for the DBI and PPI methods respectively. The input points are uniformly distributed over the interval $[-1, 1]$.

N_i	PCHIP	MQS	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
				$\mathcal{P}_1$			
17	–	–	–	2.89E-2	1	2.89E-2	1
33	–	–	–	7.69E-3	1	7.69E-3	1
65	–	–	–	1.80E-3	1	1.80E-3	1
129	–	–	–	4.58E-4	1	4.58E-4	1
257	–	–	–	1.15E-4	1	1.15E-4	1
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$				
17	8.60E-3	7.38E-3	7.15E-3	1.52E-2	2.94	1.42E-2	3.38
33	2.50E-3	2.50E-3	8.04E-4	3.31E-3	3.47	2.99E-3	3.69
65	6.36E-4	2.11E-4	4.18E-4	9.15E-4	3.67	3.86E-4	3.83
129	1.02E-4	1.01E-5	9.07E-5	4.56E-4	3.70	1.62E-5	3.95
257	1.83E-5	2.93E-7	1.83E-5	9.62E-5	3.63	6.87E-7	3.96
			$\mathcal{P}_8$				
17	–	–	4.43E-3	5.32E-3	3.75	4.49E-3	5.06
33	–	–	2.51E-3	1.25E-3	5.63	8.67E-4	6.50
65	–	–	1.00E-3	1.07E-3	6.56	7.95E-5	7.28
129	–	–	3.65E-4	5.02E-4	6.49	2.21E-6	7.75
257	–	–	9.11E-5	1.30E-4	6.45	1.75E-8	7.90
			$\mathcal{P}_{16}$				
17	–	–	4.52E-2	4.91E-2	3.69	4.90E-2	4.69
33	–	–	2.03E-3	1.45E-3	6.28	1.83E-4	9.19
65	–	–	9.55E-4	2.25E-4	10.20	6.80E-6	13.30
129	–	–	4.16E-4	2.95E-4	10.39	9.99E-8	15.47
257	–	–	1.51E-4	7.88E-5	9.59	1.30E-10	15.26

Table 6: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_2(x) = \frac{1}{1+e^{-2kx}}, \quad k = 100$, and $x \in [-0.2, 0.2]$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The sixth and eighth columns show the average polynomial degree used for the DBI and PPI methods respectively. The interval $[-0.2, 0.2]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

5.3 Example III $f_3(x)$

The third example uses the modified function introduced in Equation 3. The function $f_3(x)$ is particularly challenging because of the discontinuity at $x = -0.5$. The results from Tables 7 and 8 show that the L^2-errors from the four interpolation methods have the same order of accuracy. The PPI method gives slightly better approximation results compared to the other methods. The

average polynomial degrees for the PPI approach show that high-order polynomials are used. This suggest that in the smooth regions away from the discontinuity the PPI approach lead to high-order accuracy. However, at the discontinuity, the PPI and other methods struggle to represent the underlying function. This example shows that the PPI method is an appropriate approach for interpolating from one mesh to another, because around the discontinuity, the method is as accurate as the other ones, and in smooth regions the method gives better approximation results than the other approaches.

N_i	PCHIP	MQS	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
				$\mathcal{P}_1$			
17	–	–	–	2.07E-1	1	2.07E-1	1
33	–	–	–	1.45E-1	1	1.45E-1	1
65	–	–	–	1.02E-1	1	1.02E-1	1
129	–	–	–	7.25E-2	1	7.25E-2	1
257	–	–	–	5.15E-2	1	5.15E-2	1
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$				
17	2.33E-1	1.59E-1	2.32E-1	1.68E-1	1.63	1.60E-1	3.06
33	1.58E-1	1.11E-1	1.56E-1	1.19E-1	1.69	1.13E-1	3.63
65	1.10E-1	7.90E-2	1.09E-1	8.40E-2	1.72	8.02E-2	3.81
129	7.71E-2	5.63E-2	7.69E-2	5.96E-2	1.73	5.69E-2	3.91
257	5.45E-2	4.04E-2	5.45E-2	4.25E-2	1.74	4.05E-2	3.95
			$\mathcal{P}_8$				
17	–	–	2.25E-1	1.68E-1	1.63	1.44E-1	5.06
33	–	–	1.53E-1	1.19E-1	2	1.02E-1	7.13
65	–	–	1.07E-1	8.40E-2	2.38	7.22E-2	7.56
129	–	–	7.58E-2	5.96E-2	2.56	5.13E-2	7.78
257	–	–	5.37E-2	4.25E-2	2.65	3.66E-2	7.89
			$\mathcal{P}_{16}$				
17	–	–	2.25E-1	1.68E-1	1.63	1.36E-1	6.63
33	–	–	1.50E-1	1.19E-1	2	9.29E-2	12.13
65	–	–	1.05E-1	8.40E-2	2.94	6.59E-2	15.06
129	–	–	7.45E-2	5.96E-2	3.24	4.69E-2	15.53
257	–	–	5.29E-2	4.25E-2	2.94	3.35E-2	15.77

Table 7: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_3(x)$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The fifth and seventh columns show the average polynomial degree used for the DBI and PPI methods, respectively. The input points are uniformly distributed over the interval $[-1, 1]$.

N_i	PCHIP	MQS	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
				$\mathcal{P}_1$			
17	–	–	–	2.07E-1	1	2.07E-1	1
33	–	–	–	1.45E-1	1	1.45E-1	1
65	–	–	–	1.02E-1	1	1.02E-1	1
129	–	–	–	7.25E-2	1	7.25E-2	1
257	–	–	–	5.15E-2	1	5.15E-2	1
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_4$				
17	1.89E-1	1.39E-1	1.87E-1	1.49E-1	1.63	1.43E-1	3.44
33	1.30E-1	9.79E-2	1.29E-1	1.05E-1	1.69	1.01E-1	3.63
65	9.09E-2	6.94E-2	9.04E-2	7.45E-2	1.72	7.19E-2	3.81
129	6.41E-2	4.96E-2	6.39E-2	5.29E-2	1.73	5.11E-2	3.91
257	4.54E-2	3.58E-2	4.54E-2	3.77E-2	1.74	3.64E-2	3.95
			$\mathcal{P}_8$				
17	–	–	2.84E-1	1.98E-1	1.63	1.59E-1	5.50
33	–	–	9.61E-2	8.19E-2	2	7.64E-2	7.56
65	–	–	6.79E-2	5.81E-2	2.38	5.42E-2	7.34
129	–	–	4.82E-2	4.14E-2	2.56	3.87E-2	7.67
257	–	–	3.44E-2	2.97E-2	2.66	2.78E-2	7.84
			$\mathcal{P}_{16}$				
17	–	–	1.11E-1	9.75E-2	1.88	9.91E-2	7.75
33	–	–	1.86E-1	1.45E-1	2	1.05E-1	12.8
65	–	–	4.91E-2	4.30E-2	2.94	3.98E-2	15.06
129	–	–	3.51E-2	3.08E-2	3.12	2.86E-2	15.06
257	–	–	2.53E-2	2.23E-2	2.84	2.08E-2	15.53

Table 8: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_3(x)$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The sixth and eighth columns show the average polynomial degree used for the DBI and PPI methods respectively. The interval $[-1, 1]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

The results from Tables 7 and 8 show that the L^2-error from the four interpolation methods have the same order.

5.4 Example IV $f_4(x)$

The fourth example uses the function $f_4(x)$ defined in Equation 4. $f_4(x)$ depends on the size h of each element, and as the number of element changes, so does the element size h and the function $f_4(x)$.

At the element boundaries $f_4(x)$ is only C^0-continuous with large gradients of opposite signs. The results from Tables 9 and 10 show that all the methods all the methods struggle to approximate the underlying function. With the exception of using a uniform mesh with PCHIP and SPS, the remaining results from Tables 9 and 10 show that all the methods have the same order of accuracy for both uniform and LGL meshes. The PPI and DBI methods give slightly smaller L^2-errors compared to the other approaches.

N_i	PCHIP	MQS	$\mathcal{P}_j$	SPS	DBI		PPI	
	L^2-error	L^2-error		L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
	$ne = 4$							
17	3.73E-1	3.95E-1	$\mathcal{P}_4$	3.72E-1	2.95E-1	2	2.95E-1	2.25
33	2.47E-1	2.59E-1	$\mathcal{P}_8$	2.46E-1	1.58E-1	4	1.58E-1	5.37
65	8.32E-1	1.63E-1	$\mathcal{P}_{16}$	1.54E-1	7.59E-2	8	7.59E-2	12.88
	$ne = 8$							
33	3.84E-1	3.94E-1	$\mathcal{P}_4$	3.83E-1	2.95E-1	2	2.95E-1	2.25
65	2.54E-1	2.59E-1	$\mathcal{P}_8$	2.52E-1	1.58E-1	4	1.58E-1	5.37
129	1.60E-1	8.23E-2	$\mathcal{P}_{16}$	1.57E-1	7.66E-2	7.94	7.66E-2	12.85
	$ne = 16$							
65	3.90E-1	3.93E-1	$\mathcal{P}_4$	3.89E-1	2.95E-1	2	2.95E-1	2.25
129	2.58E-1	2.58E-1	$\mathcal{P}_8$	2.55E-1	1.58E-1	4	1.58E-1	5.38
257	1.63E-1	8.06E-2	$\mathcal{P}_{16}$	1.58E-1	7.68E-2	7.94	7.68E-2	12.85

Table 9: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the Runge function $f_4(x)$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The value ne represents the number of elements. The seventh and ninth columns show the average polynomial degree used for the DBI and PPI methods, respectively. The input points are uniformly distributed over the interval $[0, 1]$.

N_i	PCHIP	MQS	$\mathcal{P}_j$	SPS	DBI		PPI	
	L^2-error	L^2-error		L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
	$ne = 4$							
17	3.02E-1	3.18E-1	$\mathcal{P}_4$	3.02E-1	2.54E-1	2	2.54E-1	2
33	1.33E-1	1.39E-1	$\mathcal{P}_8$	1.33E-1	9.93E-2	4	9.93E-2	4.75
65	3.80E-2	3.92E-2	$\mathcal{P}_{16}$	3.77E-2	2.15E-2	8	2.15E-2	10.23
	$ne = 8$							
33	3.10E-1	3.17E-1	$\mathcal{P}_4$	3.10E-1	2.54E-1	2	2.54E-1	2
65	1.37E-1	1.39E-1	$\mathcal{P}_8$	1.36E-1	9.93E-2	4	9.93E-2	4.75
129	3.98E-2	3.72E-2	$\mathcal{P}_{16}$	3.87E-2	2.15E-2	8	2.15E-2	10.20
	$ne = 16$							
65	3.14E-1	3.16E-1	$\mathcal{P}_4$	3.14E-1	2.54E-1	2	2.54E-1	2
129	1.39E-1	1.38E-1	$\mathcal{P}_8$	1.37E-1	9.93E-2	4	9.93E-2	4.75
257	4.07E-2	3.37E-2	$\mathcal{P}_{16}$	3.90E-2	2.15E-2	7.94	2.15E-2	10.23

Table 10: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the Runge function $f_4(x)$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The value ne represents the number of elements. The seventh and ninth columns show the average polynomial degree used for the DBI and PPI methods respectively. The interval $[0, 1]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

5.5 Example V $f_5(x)$

The fifth experiment uses the function $f_5(x)$ defined in Equation 5. $f_5(x)$ depends on the size h of each element and as the number of elements changes, so does the element size h and $f_5(x)$. Similarly to $f_4(x)$, $f_5(x)$ is only C^0-continuous at the element boundaries. However, the gradients remain positive over the entire interval. This example is constructed to reflect the C^0-continuity observed in the spectral element method used in NEPTUNE. Tables 11 and 12 show the approximation errors from PCHIP, MQS, SPS, and DBI methods improve slowly compared to the PPI methods, as we increase the polynomial degree and the number of points. The PCHIP, SPS, and MQS methods use approximations of the first derivatives and enforce C^1-continuity at the element boundaries. The L^2-errors from the DBI approach are dominated by the local error of the intervals in which low-degree polynomials are used. Overall, the results from Tables 11 and 12 show that the PPI method has smaller L^2-errors compared to the remaining methods.

N_i	PCHIP	MQS	$\mathcal{P}_j$	SPS	DBI		PPI	
	L^2-error	L^2-error		L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
	$ne = 4$							
17	1.68E-2	4.50E-2	$\mathcal{P}_4$	1.23E-2	2.36E-2	3.63	2.36E-2	3.62
33	9.95E-3	6.04E-3	$\mathcal{P}_8$	1.36E-2	1.12E-2	5.88	4.20E-4	7.25
65	1.67E-3	3.82E-4	$\mathcal{P}_{16}$	5.77E-3	4.65E-3	6.36	3.65E-5	13.59
	$ne = 8$							
33	1.99E-2	1.20E-2	$\mathcal{P}_4$	1.29E-2	2.25E-2	3.69	1.66E-2	3.84
65	3.35E-3	7.63E-4	$\mathcal{P}_8$	7.42E-3	9.30E-3	4.61	6.15E-4	7.55
129	3.70E-4	4.44E-5	$\mathcal{P}_{16}$	2.89E-3	2.42E-3	5.32	1.84E-6	13.75
	$ne = 16$							
33	6.73E-3	2.46E-3	$\mathcal{P}_4$	4.57E-3	1.87E-2	3.27	1.18E-2	3.95
65	8.22E-4	4.53E-4	$\mathcal{P}_8$	3.61E-3	4.85E-3	3.94	5.27E-5	7.52
256	1.53e-4	1.59E-4	$\mathcal{P}_{16}$	1.42E-3	1.28E-3	5.02	4.27E-11	14.20

Table 11: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_5(x)$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The value ne is the number of elements used. The seventh and ninth columns show the average polynomial degree used for the DBI and PPI methods respectively. The input points are uniformly distributed over the interval $[-2, 0]$.

N_i	PCHIP	MQS	$\mathcal{P}_j$	SPS	DBI		PPI	
	L^2-error	L^2-error		L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
	$ne = 4$							
17	4.64E-2	3.10E-2	$\mathcal{P}_4$	1.23E-2	3.76E-2	3.62	3.76E-2	3.63
33	7.43E-3	4.29E-3	$\mathcal{P}_8$	1.36E-2	1.93E-2	4.78	1.36E-3	6.88
65	9.48E-4	1.58E-4	$\mathcal{P}_{16}$	5.77E-3	5.27E-3	6.30	3.05E-6	14.05
	$ne = 8$							
33	2.57E-2	9.45E-3	$\mathcal{P}_4$	1.29E-2	3.23E-2	3.68	1.56E-2	3.81
65	3.18E-3	9.15E-4	$\mathcal{P}_8$	7.42E-3	1.80E-2	3.78	9.54E-5	7.48
129	4.07E-4	2.71E-5	$\mathcal{P}_{16}$	2.89E-3	2.82E-3	5.60	5.46E-9	14.19
	$ne = 16$							
33	8.03E-3	3.77E-3	$\mathcal{P}_4$	4.57E-3	3.26E-2	3.09	1.91E-3	3.81
65	9.93E-4	2.22E-4	$\mathcal{P}_8$	3.61E-3	9.31E-3	3.89	2.23E-6	7.63
256	1.23E-4	3.30E-5	$\mathcal{P}_{16}$	1.42E-3	2.48E-3	5.19	2.63E-12	14.30

Table 12: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_5(x)$. N_i represents the number of input points used to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree for each interval. The value of ne is the number of elements used. The seventh and ninth columns show the average polynomial degree used for the DBI and PPI methods respectively. The interval $[-2, 0]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

5.6 Example VI BOMEX

The 1D Barbados Oceanographic and Meteorological Experiment (BOMEX) [40] is a simulation developed to measure and study the rate of the properties of heat, moisture, and momentum. In this case, the dynamics solution is computed using LGL mesh points and the physics routines using uniform mesh points. The dynamics (advection part) uses a third-order positivity-preserving finite difference scheme for space and a third-order Runge-Kutta [41] time integrator. This experiment compares the standard polynomial interpolation method against the PPI method for the mapping between dynamics and physics. In this case, the mapping between dynamics and physics may lead to an over-production of the cloud water mixing ratio and the introduction of negative values of the cloud water mixing ratio.

Using linear interpolants for the mapping between dynamics and physics ensures positivity but leads to no formation of the cloud water mixing ratio. The linear interpolation method is highly diffusive which eliminates away the cloud water mixing ratio. In addition, the linear interpolation method does not take advantage of the high-order accuracy coming from the dynamics' solution.

As in [42], let q_c be the cloud water mixing profile in the different experiments. The result in Figure 11 is used as the target solution because: (1) the cloud water mixing ratio shown in Figure 11 uses the same mesh for both physics and dynamics and so does not require any mapping

between the physics and dynamics routines; (2) as we increase the spatial and temporal resolutions, the solution converges to the results in Figure 11.

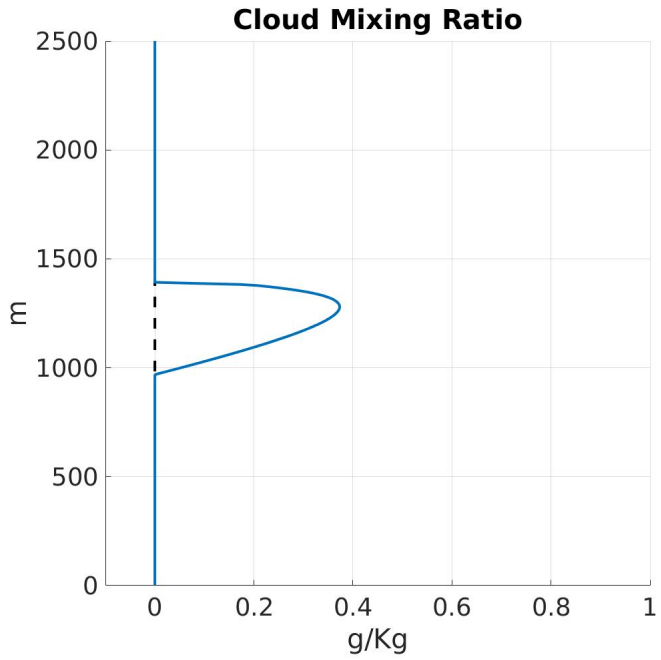

Figure 11: The cloud water mixing ratio with the same mesh used for both physics and dynamics. The dynamics uses a third-order scheme in space and time with a $CFL = 0.1$ and 200 elements are used.

Figure 12 shows the cloud water mixing ratio results in the case when the mapping between physics and dynamics is done by using a standard polynomial interpolant for each element and then evaluating the interpolant at the required physics points. This approach may lead to negative quantities, large oscillations, and positive bias in the cloud water mixing ratio prediction. The total amount of the cloud water mixing ratio in the different experiments is estimated by approximating the integral of q_c from 0 to 3000 m. The amount of cloud water mixing ratio in Figure 12 is about 65.9% more than the target solution in Figure 11. The peak of the cloud water mixing ratio in this case is $max(q_c) = 0.52g/Kg$. This peak is larger than the peak value $max(q_c) = 0.37g/Kg$ observed in the target solution.

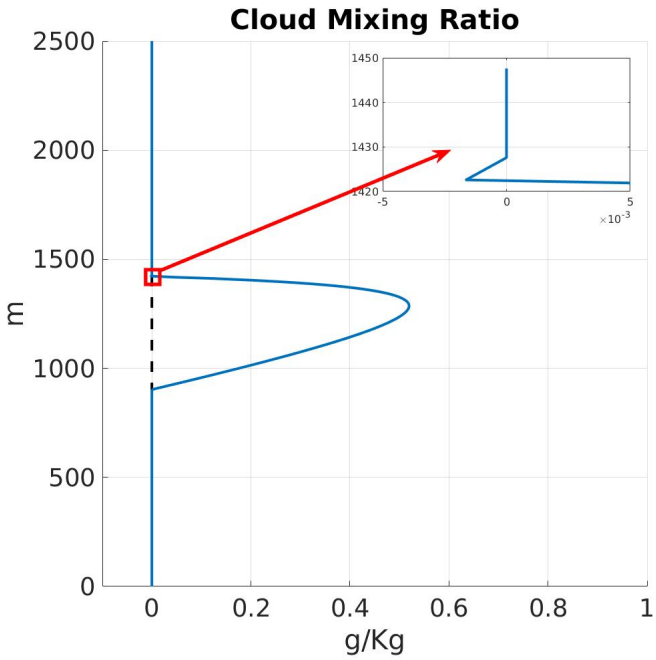

Figure 12: The cloud water mixing ratio with the staggered grid. The dynamics uses a third-order scheme in space and time with a $CFL = 0.1$ and 200 elements are used. The mapping between physics and dynamics is done by using a standard polynomial interpolant for each element and then evaluating the interpolant at the required physics points.

In the physics routines used here, clipping, which consists of setting the negative values produced by the physics to zero, may not address the impact of any negative values of cloud water mixing ratio that are inputs to the physics routines to begin with. For example, using clipping to enforce positivity in the mapping between physics and dynamics, as shown in Figure 13, leads to a small blip that is inconsistent with the other solutions.

In addition, enforcing positivity via clipping does not address the positive bias in the cloud water mixing ratio. The amount of cloud water mixing ratio in Figure 13 is 64.8% more than the target solution, and the peak of the cloud water mixing ratio is $max(q_c) = 0.51g/Kg$.

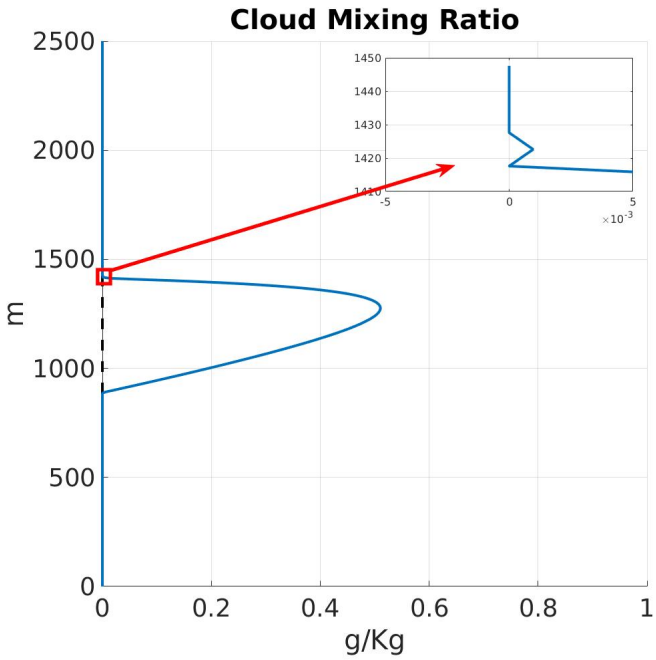

Figure 13: The cloud water mixing ratio with the staggered grid. The dynamics uses a third-order scheme in space and time with a $CFL = 0.1$ and 200 elements are used. The mapping between physics and dynamics is done by using a standard polynomial interpolant of for each element and then evaluating the interpolant at the required physics points. In addition, the negative tracer values are set to zero to ensure positivity.

Figure 14 shows the cloud water mixing ratio when the PCHIP method is used for the mapping between the physics and dynamics grids. These results show that using PCHIP reduces the positive bias in cloud water mixing ratio prediction and maintains positivity when mapping between physics and dynamics. In addition, the cloud water mixing ratio peak is $max(q_c) = 0.36g/kg$, and the amount of the cloud water mixing ratio is about 8.25% more than the target solution. These results are closer to the target solution compared to the standard interpolation method. In the case of NEPTUNE where a spectral method is used, low-order interpolation methods such as PCHIP may degrade the higher order accuracy from dynamics. This limitation can be addressed by using high-order interpolation approaches such PPI.

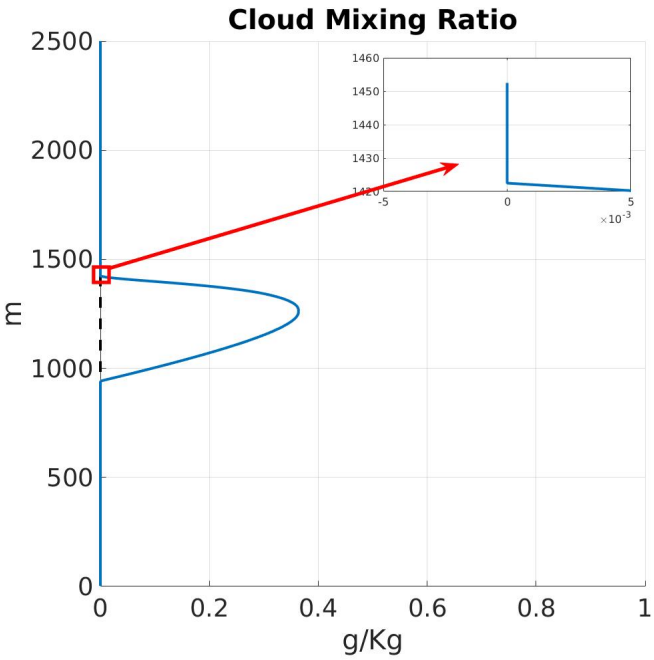

Figure 14: The cloud water mixing ratio with the staggered grid. The dynamics uses a third order scheme in space and time with a $CFL = 0.1$ and 200 elements are used. The mapping between physics and dynamics is done using the PCHIP method.

Figure 15 shows the cloud water mixing ratio when the PPI method is used to perform the mapping between the physics and dynamics grids. No negative quantities are observed, and the positive bias is significantly reduced. These results indicate that using PPI reduces the positive bias in cloud water mixing ratio prediction and maintains positivity in the mapping between physics and dynamics. In addition, the cloud water mixing ratio peak when using PPI is $max(q_c) = 0.36g/kg$, and the amount of cloud water mixing ratio is about 9.4% more than the target solution. These results are closer to those of the target solution than those obtained with the other approaches used. When the PPI method is used, the cloud water mixing ratio has a slightly larger peak and covers a wider area compared to the target solution, as shown in Figure 16. These differences account for the 9.4% increase observed in the total amount of cloud water mixing ratio.

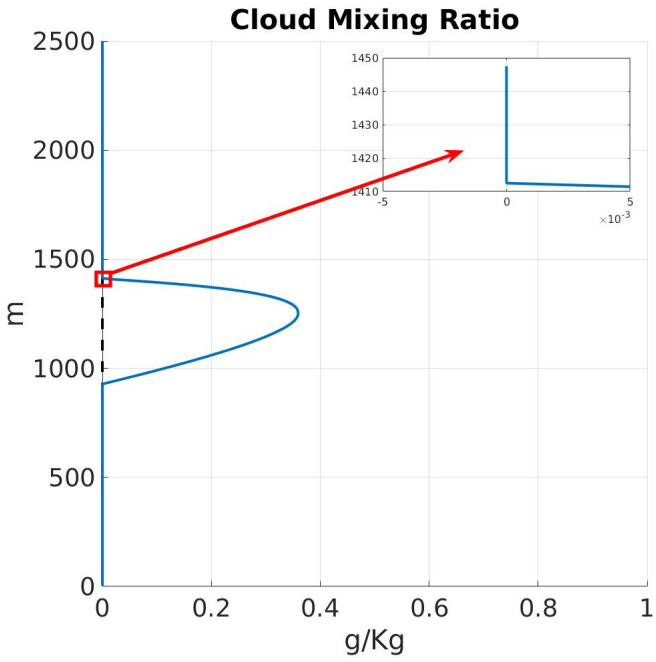

Figure 15: The cloud water mixing ratio with the staggered grid. The dynamics uses a third-order scheme in space and time with a $CFL = 0.1$ and 200 elements are used. The mapping between physics and dynamics is done using the PPI method.

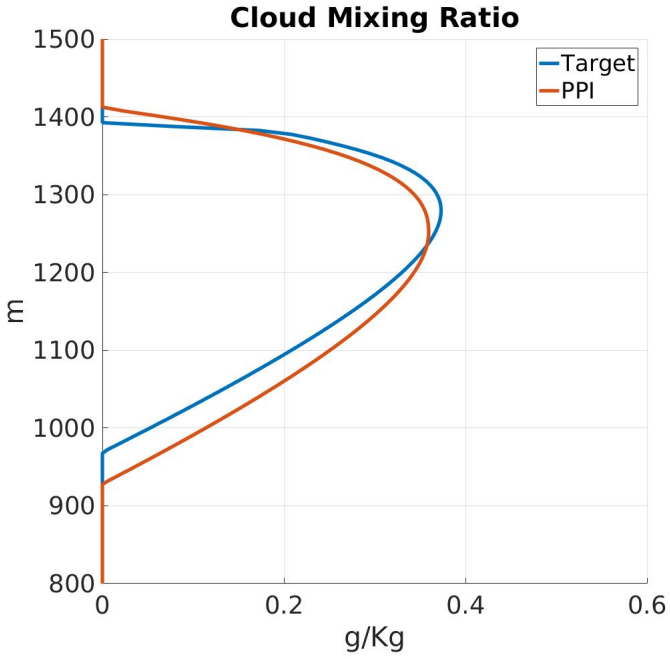

Figure 16: The target and approximation cloud water mixing ratio using the PPI method. The dynamics uses a third-order scheme in space and time with a $CFL = 0.1$ and 200 elements are used.

Figure 17 and Table 13 summarize results from the different approaches used for the BOMEX test case. In Figure 17, the target solution is shown in blue, the results from the PPI method in red, the PCHIP method in yellow, the interpolation method with clipping in purple, and the standard polynomial interpolation method in black. Table 13 shows the peak values of the cloud water mixing ratio and the total amount of the cloud mixing ratio from the different methods. These results indicate using the PCHIP and PPI methods lead to results that are closer to the target solution as compared against the other methods.

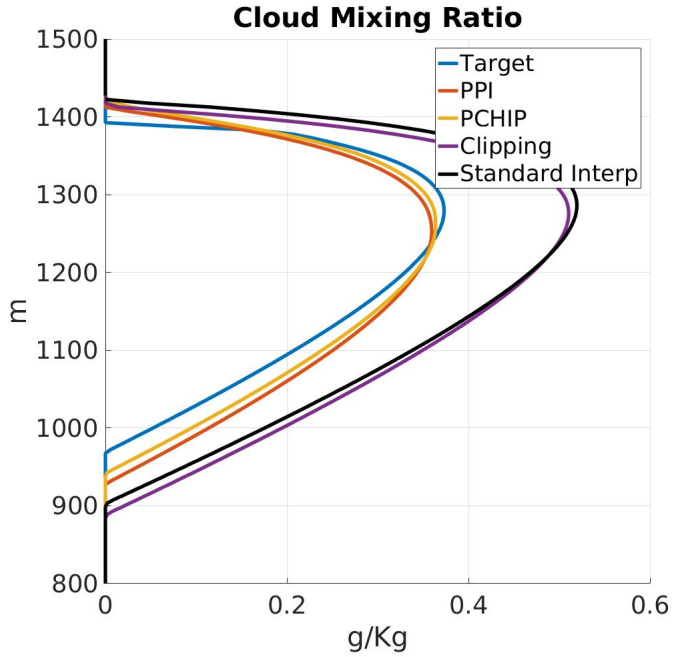

Figure 17: The cloud water mixing ratio with the staggered grid. The dynamics uses a third-order scheme in space and time with a $CFL = 0.1$ and 200 elements are used.

	No limiter	Clipping	PCHIP	PPI	Target
maximum q_c	0.52	0.51	0.36	0.36	0.37
Total q_c	171.8	170.58	112.07	113.26	103.5

Table 13: The maximum values and total amount of q_c in the different approaches. The maximum values of cloud water mixing ratio q_c is calculated by $max(q_c)$ and the total amount of q_c is calculated by estimating the integral of q_c. The units of the cloud water mixing ratio are represented in g/Kg.

5.7 Example VII $f_7(x)$

This example uses an extended version of the 1D Runge function defined in Equation 1 from Section 3.1 to a 2D function:

$$f_7(x, y) = \frac{1}{1 + 25(x^2 + y^2)}, \quad x, y \in [-1, 1] \tag{7}$$

The results from Tables 14 and 15 show that the PPI method gives smaller approximation errors compared to the PCHIP, SPS, and DBI methods. In this case, the PPI method uses higher order polynomial interpolants for each interval. These higher order interpolants help improve the approximation compared to the PCHIP, SPS, and DBI methods which use lower order interpolants.

N_i	PCHIP	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	1.60E-2	1	1.60E-2	1
33	–	–	4.42E-3	1	4.42E-3	1
65	–	–	1.12E-3	1	1.12E-3	1
129	–	–	2.82E-4	1	2.82E-4	1
257	–	–	7.11E-5	1	7.11E-5	1
	$\mathcal{P}_3$	$\mathcal{P}_4$				
17	5.01E-3	3.61E-3	1.61E-2	1.67	6.06E-3	4
33	1.23E-3	5.44E-4	4.41E-3	1.81	4.86E-4	4
65	2.33E-4	1.23E-4	1.12E-3	1.88	1.76E-5	4
129	4.26E-5	3.07E-5	2.80E-4	1.92	5.54E-7	4
257	7.67E-6	3.34E-6	7.01E-5	1.93	1.73E-8	4
		$\mathcal{P}_8$				
17	–	8.55E-3	1.61E-2	1.70	4.14E-3	7.84
33	–	1.99E-3	4.41E-3	1.97	1.09E-4	8
65	–	4.97E-4	1.12E-3	2.17	7.41E-7	8
129	–	1.25E-4	2.80E-4	2.27	2.17E-9	8
257	–	3.16E-5	7.01E-5	2.32	4.65E-12	8
		$\mathcal{P}_{16}$				
17	–	1.19E-2	1.61E-2	1.70	4.07E-3	11.43
33	–	3.10E-3	4.41E-3	1.97	5.34E-5	15.94
65	–	7.82E-4	1.12E-3	2.19	3.70E-8	16
129	–	1.96E-4	2.80E-4	2.32	2.62E-12	16
257	–	4.93E-5	7.01E-5	2.36	1.99E-15	16

Table 14: $L^2 - errors$ when approximating $f_7(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods, respectively. The mesh points are uniformly distributed on each dimension.

N_i	PCHIP	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	1.60E-2	1	1.60E-2	1
33	–	–	4.42E-3	1	4.42E-3	1
65	–	–	1.12E-3	1	1.12E-3	1
129	–	–	2.82E-4	1	2.82E-4	1
257	–	–	7.11E-5	1	7.11E-5	1
	$\mathcal{P}_3$	$\mathcal{P}_4$				
17	5.01E-3	5.41E-3	1.61E-2	1.54	4.26E-3	4
33	1.23E-3	1.05E-3	4.41E-3	1.62	5.25E-4	4
65	2.33E-4	1.61E-4	1.12E-3	1.68	2.96E-5	4
129	4.26E-5	4.31E-5	2.80E-4	1.70	8.50E-7	4
257	7.67E-6	1.12E-5	7.01E-5	1.71	2.58E-8	4
		$\mathcal{P}_8$				
17	–	5.89E-3	1.61E-2	1.68	1.90E-3	7.72
33	–	1.72E-3	4.41E-3	1.74	1.11E-4	8
65	–	5.44E-4	1.12E-3	1.74	1.68E-6	8
129	–	1.90E-4	2.80E-4	1.77	5.41E-9	8
257	–	4.91E-5	7.01E-5	1.79	2.41E-11	8
		$\mathcal{P}_{16}$				
17	–	1.88E-2	1.61E-2	1.96	1.67E-2	13.45
33	–	2.11E-3	4.41E-3	2.12	5.90E-6	15.83
65	–	6.96E-4	1.11E-3	1.90	4.75E-8	16
129	–	2.25E-4	2.80E-4	1.80	2.48E-11	16
257	–	7.93E-5	7.01E-5	1.76	2.01E-15	16

Table 15: $L^2 - errors$ when approximating $f_7(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods respectively. For each dimension, the interval $[-1, 1]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

5.8 Example VIII $f_8(x)$

This example uses a 2D function that is used to study positive and monotonic splines [43, 44, 45]. The function is defined as follows:

$$f_8(x,y) = \begin{cases} 2(y-x) & \text{if } 0 \leq y-x \leq 0.5 \\ 1 & \text{if } y-x \geq 0.5 \\ cos\left(4\pi\sqrt{(x-1.5)^2+(y-0.5)^2}\right) & \text{if } (x-1.5)^2+(y-0.5)^2 \leq \frac{1}{16} \\ 0 & otherwise \end{cases} \tag{8}$$

As in Example V, the function $f_8(x)$ is C^0-continuous and the underlying mesh used for the approximations does not capture the sharp corners. The L^2-errors from the DBI and PPI method are dominated by the local errors of the intervals with low degree polynomial interpolants. Tables 16 and 17 show that the L^2-errors from the three methods have the same order, with PPI having slightly smaller errors than the other approaches. In the cases where the underlying function is C^0, the results from PPI are comparable to the other approaches. Furthermore, the results from PPI can be improved by using a mesh that captures C^0-continuity, as is the case with the spectral element methods in NEPTUNE.

N_i	PCHIP	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	2.70E-2	1	2.70E-2	1
33	–	–	9.51E-3	1	9.51E-3	1
65	–	–	3.40E-3	1	3.40E-3	1
129	–	–	1.20E-3	1	1.20E-3	1
257	–	–	4.30E-4	1	4.30E-4	1
	$\mathcal{P}_3$	$\mathcal{P}_4$				
17	1.91E-2	1.87E-2	2.91E-02	3.26	1.99E-2	3.68
33	6.92E-3	6.11E-3	9.56E-03	3.59	7.88E-3	3.91
65	2.47E-3	2.69E-3	3.18E-03	3.66	2.66E-3	3.96
129	8.99E-4	7.71E-4	1.12E-03	3.68	9.98E-4	3.98
257	3.23E-4	2.77E-4	3.92E-04	3.69	3.43E-4	3.99
		$\mathcal{P}_8$				
17	–	1.91E-2	2.92E-02	5.05	2.03E-2	6.07
33	–	6.46E-3	9.57E-03	6.62	7.92E-3	7.42
65	–	2.24E-3	3.17E-03	7.11	2.67E-3	7.84
129	–	8.12E-4	1.12E-03	7.23	1.00E-3	7.93
257	–	2.92E-4	3.91E-04	7.27	3.45E-4	7.97
		$\mathcal{P}_{16}$				
17	–	2.19E-2	2.92E-02	6.69	2.05E-2	7.81
33	–	7.57E-3	9.58E-03	10.72	8.14E-3	12.07
65	–	2.68E-3	3.18E-03	13.43	2.79E-3	14.94
129	–	9.63E-4	1.12E-03	14.23	1.04E-3	15.72
257	–	3.45E-4	3.92E-04	14.38	3.59E-4	15.88

Table 16: $L^2 - errors$ when approximating $f_8(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods, respectively. The points are uniformly distributed in each dimension.

N_i	PCHIP		DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	2.70E-2	1	2.70E-2	1
33	–	–	9.51E-3	1	9.51E-3	1
65	–	–	3.40E-3	1	3.40E-3	1
129	–	–	1.20E-3	1	1.20E-3	1
257	–	–	4.30E-4	1	4.30E-4	1
	$\mathcal{P}_3$		$\mathcal{P}_4$			
17	1.91E-2	2.55E-2	2.92E-2	3.07	2.32E-2	3.69
33	6.92E-3	5.76E-3	9.57E-3	3.38	7.75E-3	3.90
65	2.47E-3	2.11E-3	3.19E-3	3.47	3.04E-3	3.96
129	8.99E-4	8.08E-4	1.12E-3	3.49	1.10E-3	3.98
257	3.23E-4	2.92E-4	3.93E-4	3.49	3.97E-4	3.99
			$\mathcal{P}_8$			
17	–	4.06E-2	2.93E-2	4.60	5.61E-2	5.83
33	–	9.69E-3	9.58E-3	5.96	1.12E-2	7.44
65	–	2.46E-3	3.17E-3	6.42	3.08E-3	7.83
129	–	9.83E-4	1.12E-3	6.57	1.19E-3	7.93
257	–	3.63E-4	3.93E-4	6.57	4.37E-4	7.97
			$\mathcal{P}_{16}$			
17	–	4.36E-2	2.92E-2	6.91	5.08E-2	8.32
33	–	1.53E-2	9.58E-3	9.68	1.60E-2	11.86
65	–	4.31E-3	3.18E-3	11.82	4.29E-3	14.91
129	–	1.13E-3	1.12E-3	12.75	1.08E-3	15.69
257	–	4.38E-4	3.92E-4	12.82	4.38E-4	15.87

Table 17: $L^2 - errors$ when approximating $f_8(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods respectively. For each dimension, the interval $[-1, 1]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

5.9 Example IX $f_9(x)$

This example is used therein to study shape-preserving (monotonicity and convexity) splines [46].

$$f_9(x, y) = max\Big(0, sin(\pi x)sin(\pi y)\Big) \quad x, y \in [-1, 1] \tag{9}$$

The function $f_9(x, y)$ is a C^0-continuous function. Tables 18 and 17 show L^2-errors when approximating $f_9(x, y)$ with the PCHIP, SPS, DBI, and PPI methods. The underlying mesh is such that the C^0-continuities are at the elements boundaries except for $\mathcal{P}_{16}$ and $N = 17$. The PCHIP and SPS methods struggle to capture the C^0-continuities because both methods enforce C^1-continuity. The L^2-error from DBI is dominated by the local error from the intervals with low-degree interpolants and so as the average polynomial degree increases the L^2-errors do not improve. The L^2-error for $\mathcal{P}_{16}$ and $N = 17$ is larger compared to the other cases when the PPI method is used. For $\mathcal{P}_{16}$ and $N = 17$, there is no mesh point at the points of C^0-continuity and so the L^2-error is dominated by the local error from those intervals where low degree interpolants are used. Overall, the results from Tables 18 and 17 demonstrate that the PPI method leads to smaller approximation errors than the PCHIP, SPS, and DBI methods in this case.

N_i	PCHIP	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	5.80E-2	1	5.80E-2	1
33	–	–	1.88E-2	1	1.88E-2	1
65	–	–	6.27E-3	1	6.27E-3	1
129	–	–	2.17E-3	1	2.17E-3	1
257	–	–	7.87E-4	1	7.87E-4	1
	$\mathcal{P}_3$	$\mathcal{P}_4$				
17	1.91E-2	1.80E-2	1.85E-2	2.50	1.24E-4	3.91
33	6.77E-3	6.21E-3	4.74E-3	2.49	3.88E-6	3.95
65	2.39E-3	2.18E-3	1.19E-3	2.50	1.06E-7	3.98
129	8.47E-4	7.70E-4	2.98E-4	2.50	9.83E-9	3.99
257	3.00E-4	2.74E-4	7.45E-5	2.50	6.50E-9	3.99
		$\mathcal{P}_8$				
17	–	1.47E-2	1.85E-2	3.92	1.88E-7	7.37
33	–	4.57E-3	4.74E-3	4.40	1.91E-8	7.89
65	–	1.51E-4	1.19E-3	4.45	1.35E-8	7.95
129	–	5.16E-4	2.98E-4	4.47	9.19E-9	7.97
257	–	1.84E-4	7.45E-5	4.49	5.98E-9	7.99
		$\mathcal{P}_{16}$				
17	–	1.50E-2	1.85E-2	4.15	2.49E-3	8.09
33	–	4.03E-3	4.74E-3	7.77	1.85E-8	15.31
65	–	1.15E-3	1.19E-3	8.35	1.28E-8	15.88
129	–	3.50E-4	2.98E-4	8.42	8.64E-9	15.94
257	–	1.17E-4	7.45E-5	8.46	5.57E-9	15.97

Table 18: $L^2 - errors$ when approximating $f_9(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods respectively. The points are uniformly distributed on each dimension.

N_i	PCHIP	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	5.80E-2	1	5.80E-2	1
33	–	–	1.88E-2	1	1.88E-2	1
65	–	–	6.27E-3	1	6.27E-3	1
129	–	–	2.17E-3	1	2.17E-3	1
257	–	–	7.87E-4	1	7.87E-4	1
	$\mathcal{P}_3$	$\mathcal{P}_4$				
17	1.91E-2	1.05E-2	1.85E-2	2.52	1.46E-4	3.91
33	6.77E-3	3.61E-3	4.74E-3	2.61	4.77E-6	3.95
65	2.39E-3	1.26E-3	1.19E-3	2.62	1.43E-7	3.98
129	8.47E-4	4.44E-4	2.98E-4	2.62	9.16E-9	3.99
257	3.00E-4	1.61E-4	7.45E-5	2.62	5.52E-9	3.99
		$\mathcal{P}_8$				
17	–	1.61E-2	1.85E-2	4.04	6.01E-8	7.36
33	–	3.34E-3	4.74E-3	4.57	1.37E-8	7.89
65	–	8.74E-4	1.19E-3	4.57	9.43E-9	7.95
129	–	2.40E-4	2.98E-4	4.56	6.15E-9	7.97
257	–	7.32E-5	7.45E-5	4.56	3.57E-9	7.99
		$\mathcal{P}_{16}$				
17	–	2.72E-2	1.85E-2	4.29	2.35E-3	8.75
33	–	6.65E-3	4.74E-3	7.93	9.88E-9	15.31
65	–	1.33E-3	1.19E-3	8.57	6.44E-9	15.88
129	–	3.35E-4	2.98E-4	8.54	3.78E-9	15.94
257	–	8.41E-5	7.45E-5	8.55	1.75E-9	15.97

Table 19: $L^2 - errors$ when approximating $f_9(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods respectively. For each dimension, the interval $[-1, 1]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

5.10 Example X $f_{10}(x)$

This example uses a 2D extension of the 1D approximation of the Heaviside function $f_2(x)$ defined in Equation 2 which is defined as follows:

$$f_{10}(x, y) = \frac{1}{1 + e^{-\sqrt{2}k(x+y)}}, \quad x, y \in [-0.2, 0.2] \tag{10}$$

The function $f_{10}(x, y)$ is challenging because of the large gradient at $y = -x$. Tables 20 and 21 show L^2-errors when approximating $f_{10}(x)$ using PCHIP, SPS, DBI, and PPI. As the average polynomial degree increases the accuracy of the DBI method does not improve. The L^2-error from the DBI method is dominated by the local error from the intervals with low-degree interpolants. For the PPI method with $n = 65$, when going from $\mathcal{P}_8$ to $\mathcal{P}_{16}$, while average polynomial degree increases the L^2-error increases slightly. As the average polynomial degree increases more oscillation are introduced around the gradient because the PPI method allows the interpolant to grow beyond the data values while remaining positive. The L^2-error in this case is dominated by the local error of the region with oscillations. Overall, the results from the Tables 20 and 21 show that the PPI method leads to smaller L^2-errors compared to the other methods.

N_i	PCHIP	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	1.50E-2	1	1.50E-2	1
33	–	–	4.57E-3	1	4.57E-3	1
65	–	–	1.26E-3	1	1.26E-3	1
129	–	–	3.23E-4	1	3.23E-4	1
257	–	–	8.15E-5	1	8.15E-5	1
	$\mathcal{P}_3$	$\mathcal{P}_4$				
17	8.07E-3	1.99E-3	1.45E-2	3.08	1.20E-2	3.54
33	1.26E-3	2.43E-4	3.58E-3	3.52	1.93E-3	3.84
65	1.44E-4	4.92E-5	9.73E-4	3.62	1.16E-4	3.96
129	1.63E-5	1.20E-5	2.09E-4	3.66	4.00E-6	3.99
257	1.94E-6	3.05E-6	4.74E-5	3.67	1.28E-7	3.99
		$\mathcal{P}_8$				
17	–	1.80E-1	1.45E-2	4.50	1.16E-2	5.34
33	–	1.22E-1	3.58E-3	5.95	1.75E-3	7.03
65	–	8.48E-2	9.73E-4	6.49	6.82E-5	7.80
129	–	1.04E-1	2.09E-4	6.54	2.64E-7	7.92
257	–	8.02E-2	4.74E-5	6.48	5.47E-10	7.95
		$\mathcal{P}_{16}$				
17	–	4.72E-3	1.45E-2	5.06	1.16E-2	6.05
33	–	1.22E-3	3.58E-3	8.53	1.78E-3	10.77
65	–	3.11E-4	9.73E-4	10.23	1.25E-4	14.66
129	–	7.80E-5	2.09E-4	10.25	2.40E-7	15.59
257	–	1.95E-5	4.74E-5	9.57	3.32E-10	15.62

Table 20: $L^2 - errors$ when approximating $f_{10}(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods, respectively. The points are uniformly distributed on each dimension.

N_i	PCHIP	SPS	DBI		PPI	
	L^2-error	L^2-error	L^2-error	avg. deg.	L^2-error	avg. deg.
			$\mathcal{P}_1$			
17	–	–	1.50E-2	1	1.50E-2	1
33	–	–	4.57E-3	1	4.57E-3	1
65	–	–	1.26E-3	1	1.26E-3	1
129	–	–	3.23E-4	1	3.23E-4	1
257	–	–	8.15E-5	1	8.15E-5	1
	$\mathcal{P}_3$	$\mathcal{P}_4$				
17	8.07E-3	2.79E-3	1.45E-2	3.08	1.49E-2	3.55
33	1.26E-3	3.49E-4	3.58E-3	3.22	2.94E-3	3.82
65	1.44E-4	6.87E-5	9.73E-4	3.02	1.56E-4	3.94
129	1.63E-5	1.64E-5	2.09E-4	2.76	5.65E-6	3.98
257	1.94E-6	4.12E-6	4.74E-5	2.67	1.83E-7	3.99
		$\mathcal{P}_8$				
17	–	5.82E-3	1.45E-2	4.13	1.61E-2	5.48
33	–	1.26E-3	3.58E-3	5.08	3.54E-3	6.98
65	–	3.00E-4	9.73E-4	4.93	1.45E-4	7.80
129	–	7.58E-5	2.09E-4	4.47	4.65E-7	7.94
257	–	1.90E-5	4.74E-5	4.25	1.30E-9	7.93
		$\mathcal{P}_{16}$				
17	–	8.05E-3	1.45E-2	4.75	1.74E-2	6.68
33	–	2.06E-3	3.58E-3	7.12	4.29E-3	10.92
65	–	5.14E-4	9.73E-4	7.46	2.30E-4	14.67
129	–	1.26E-4	2.09E-4	6.85	9.61E-8	15.64
257	–	3.18E-5	4.74E-5	5.99	3.28E-11	15.68

Table 21: $L^2 - errors$ when approximating $f_{10}(x, y)$ with $N_i \times N_i$ points. N_i represents the number of input points used in each dimension to build the approximation. P_j represents the use of polynomials of degree j, with j being the target degree. The fourth and sixth columns show the average polynomial degree used for the DBI and PPI methods respectively. For each dimension, the interval $[-0.2, 0.2]$ is divided into $(N_i - 1)/j$ elements and $j + 1$ LGL quadrature points are used in each element.

6 Discussion and Conclusion

In this report, a representative sample of existing methods is compared against our new approach on a number of different test functions, including smooth, C^0, discontinuous, and steep functions. The comparison undertaken here focuses on how accurately the different methods are able to represent this underlying set of test functions. Overall, the new method both performs well and is suited to

the C^0 continuity of the spectral element methods in NEPTUNE. The experiments in this section suggest that the PPI method is a suitable approach for interpolating smooth functions and C^0 continuous functions while enforcing positivity. In detail the conclusions are that:

- The results in Section 3 Examples I, II and III show that the new PPI approach preserves positivity exactly as the proof in [9] indicates;
- The results in Section 4 and Sections 5.1, 5.2 and 5.3 show that new PPI approach gives a much higher level of accuracy than the DBI method by allowing the solution to be outside the local bounds while remaining positive. The PPI method also appears to give better results than the SPS method in line with the studies in [29] and [27] which demonstrate that the SPS method does not achieve high-order accuracy; and
- In the cases when steep gradients or discontinuities force the use of low-order approximations, the PPI method competes against the well-known cubic spline method PCHIP and the higher order MQS and the SPS spline methods.

Overall it would seem that when it is possible to use higher-order polynomial approximations the PPI method appears to give levels of accuracy that compete with standard unmodified high-order spline methods while at the same time preserving positivity.

Acknowledgement

This work has been supported by US Naval Research Laboratory (559000669), the National Science Foundation (1521748), and the Intel Graphics and Visualization Institute at the University of Utah's Scientific Computing and Imaging (SCI) Institute (29715). The authors would like like to thank Dr. Alex Reinecke of Naval Research Laboratory for his constant support and help.

References

[1] D. Sahasrabudhe, M. Berzins, and J. Schmidt. Node failure resiliency for uintah without checkpointing. *Concurrency and Computation: Practice and Experience*, page e5340, 2019.

[2] M. Berzins. Nonlinear data-bounded polynomial approximations and their applications in eno methods. *Numerical Algorithms*, 55(2):171, 2010.

[3] M. Berzins. Data and range-bounded polynomials in eno methods. *Journal of Computational Science*, 4(1-2):62–70, 2013.

[4] M. Berzins. Adaptive Polynomial Interpolation on Evenly Spaced Meshes. *SIAM Review*, 49(4):604–627, 2007.

[5] F. X. Giraldo, J. F. Kelly, and E. M. Constantinescu. Implicit-explicit formulations of a three-dimensional nonhydrostatic unified model of the atmosphere (numa). *SIAM Journal on Scientific Computing*, 35(5):B1162–B1194, 2013.

[6] K.C. Viner, J.D. Doyle P.A. Reinecke, M. Martini S. Gabersek, J. Michalakes D.D. Flagg, and F.X. Giraldo D.R. Ryglicki. Next Generation NWP Using a Spectral Element Dynamical Core. *AGU Fall Meeting Abstracts*, pages A34A–02, December 2016.

[7] William C. Skamarock and Morris L. Weisman. The Impact of Positive-Definite Moisture Transport on NWP Precipitation Forecasts. *Monthly Weather Review*, 137(1):488–494, 2009.

[8] T.A.J. Ouermi, Robert Kirby, and Martin Berzins. High-Order Positivity-Preserving Interpolation for Physics-Dynamics Coupling. *Submitted to Monthly Weather Review*, X(x):xxx–yyy, 20xx.

[9] T.A.J. Ouermi, Robert Kirby, and Martin Berzins. High-Order Data-Bounded and Positivity-Preserving Interpolation. *Submitted to Mathematical Modeling and Numerical Analysis (M2AN)*, X(x):xxx–yyy, 20xx.

[10] J. W. Schmidt and W. Heß. Positive interpolation with rational quadratic splines. *Computing*, 38(3):261–267, Sep 1987.

[11] Schmidt, Jochen W. and Heß , Walter. Positivity of Cubic Polynomials on Intervals and Positive Spline Interpolation. *BIT Numerical Mathematics*, 28(2):340–352, Jun 1988.

[12] Walter Heß and Jochen W. Schmidt. Positive quartic, monotone quintic c2-spline interpolation in one and two dimensions. *Journal of Computational and Applied Mathematics*, 55(1):51 – 67, 1994.

[13] Samsul Ariffin Abdul Karim and Kong Pang Pang. Shape preserving interpolation using rational cubic spline. *Journal of Applied Mathematics*, 2016, 2016.

[14] Abdul Karim, Samsul Ariffin, Kong Voon Pang, and Azizan Saaban. Positivity Preserving Interpolation using Rational Bicubic Spline. *Journal of Applied Mathematics*, 2015, 2015.

[15] Malik Zawwar Hussain and Muhammad Sarfraz. Positivity-Preserving Interpolation of Positive Data by Rational Cubics. *Journal of Computational and Applied Mathematics*, 218(2):446 – 458, 2008. The Proceedings of the Twelfth International Congress on Computational and Applied Mathematics.

[16] S. Butt and K.W. Brodlie. Preserving Positivity Using Piecewise Cubic Interpolation. *Computers and Graphics*, 17(1):55 – 64, 1993.

[17] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[18] Rida T. Farouki, Carla Manni, and Alessandra Sestini. Shape-preserving interpolation by G1 and G2 PH quintic splines. *IMA Journal of Numerical Analysis*, 23(2):175–195, 04 2003.

[19] Rida T. Farouki, Carla Manni, Maria Lucia Sampoli, and Alessandra Sestini. Shape-preserving interpolation of spatial data by Pythagorean-hodograph quintic spline curves. *IMA Journal of Numerical Analysis*, 35(1):478–498, 02 2014.

[20] Rida T. Farouki, Carlotta Giannelli, Carla Manni, and Alessandra Sestini. Identification of spatial ph quintic Hermite interpolants with near-optimal shape measures. *Computer Aided Geometric Design*, 25(4):274 – 297, 2008. Pythagorean-Hodograph Curves and Related Topics.

[21] Malik Zawwar Hussain, Maria Hussain, and Zahra Yameen. A C2-continuous rational quintic interpolation scheme for curve data with shape control. *Journal of The National Science Foundation of Sri Lanka*, 46:341, 2018.

[22] Maria Hussain, Malik Zawwar Hussain, and Robert J Cripps. C2 rational quintic function. *Journal of Prime Research in Mathematics*, 5:115–123, 2009.

[23] Thomas C. H. Lux, Layne T. Watson, and Tyler H. Chang. An algorithm for contructing monotone quintic interpolating splines. In *SpringSim '20: Proceedings of the 2020 Spring Simulation Conference, May 2020*, pages 1–12, 2019.

[24] Jochen W. Schmidt and Walter Heß. Positivity of cubic polynomials on intervals and positive spline interpolation. *BIT*, 28(2):340–352, feb 1988.

[25] Gary Ulrich and Layne T. Watson. Positivity conditions for quartic polynomials. *SIAM Journal on Scientific Computing*, 15(3):528–544, 1994.

[26] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988.

[27] P. Costantini. Boundary-valued shape-preserving interpolating splines. *ACM Trans. Math. Softw.*, 23(2):229–251, jun 1997.

[28] P. Costantini. Algorithm 770: Bvspis a package for computing boundary-valued shape-preserving interpolating splines. *ACM Trans. Math. Softw.*, 23(2):252–254, jun 1997.

[29] P. Costantini. On some recent methods for bivariate shape-preserving interpolation. In W. Haußmann and K. Jetter, editors, *Multivariate Approximation and Interpolation: Proceedings of an International Workshop held at the University of Duisburg, August 14–18, 1989*, pages 55–68, Basel, 1990. Birkhäuser Basel.

[30] Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

[31] Xiangxiong Zhang. On positivity-preserving high order discontinuous Galerkin schemes for compressible NavierStokes equations. *Journal of Computational Physics*, 328:301 – 343, 2017.

[32] Xiangxiong Zhang, Yinhua Xia, and Chi-Wang Shu. Maximum-Principle-Satisfying and Positivity-Preserving High Order Discontinuous Galerkin Schemes for Conservation Laws on Triangular Meshes. *Journal of Scientific Computing*, 50(1):29–62, Jan 2012.

[33] Xiangxiong Zhang and Chi-Wang Shu. Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 467(2134):2752–2776, 2011.

[34] Devin Light and Dale Durran. Preserving Nonnegativity in Discontinuous Galerkin Approximations to Scalar Transport via Truncation and Mass Aware Rescaling (TMAR). *Monthly Weather Review*, 144(12):4771–4786, 2016.

[35] Ami Harten, Bjorn Engquist, Stanley Osher, and Sukumar R. Chakravarthy. Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III. *Journal of Computational Physics*, 131(1):3 – 47, 1997.

[36] Xiangxiong Zhang and Chi-Wang Shu. Positivity-preserving high order finite difference weno schemes for compressible euler equations. *J. Comput. Phys.*, 231(5):2245–2258, mar 2012.

[37] Nicholas Hale and Alex Townsend. Fast and accurate computation of gauss–legendre and gauss–jacobi quadrature nodes and weights. *SIAM Journal on Scientific Computing*, 35(2):A652–A674, 2013.

[38] James F. Epperson. On the Runge Example. *The American Mathematical Monthly*, 94(4):329–341, 1987.

[39] Eitan Tadmor and Jared Tanner. Adaptive mollifiers for high resolution recovery of piecewise smooth data from its spectral information. *Foundations of Computational Mathematics*, 2(2):155–189, Jan 2002.

[40] Howard A Friedman, Gerald Conrad, and James D McFadden. ESSA Research Flight Facility Aircraft Participation in the Barbados Oceanographic and Meteorological Experiment. *Bulletin of the American Meteorological Society*, 51(9):822–834, 1970.

[41] W. Hundsdorfer, B. Koren, M. vanLoon, and J.G. Verwer. A positive finite-difference advection scheme. *Journal of Computational Physics*, 117(1):35 – 46, 1995.

[42] Leon D. Rotstayn, Brian F. Ryan, and Jack J. Katzfey. A scheme for calculation of the liquid fraction in mixed-phase stratiform clouds in large-scale models. *Monthly Weather Review*, 128(4):1070–1088, 04 2000.

[43] E.S. Chan and B.H. Ong. Range restricted scattered data interpolation using convex combination of cubic bzier triangles. *Journal of Computational and Applied Mathematics*, 136(1):135 – 147, 2001.

[44] Abd. Rahni Mt. Piah, Tim N. T. Goodman, and Keith Unsworth. Positivity-preserving scattered data interpolation. In Ralph Martin, Helmut Bez, and Malcolm Sabin, editors, *Mathematics of Surfaces XI*, pages 336–349, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[45] Peter Lancaster and Kestutis Šalkauskas. Curve and surface fitting: an introduction. *csfa*, 1986.

[46] Paolo Costantini and Ferruccio Fontanella. Shape-preserving bivariate interpolation. *SIAM Journal on Numerical Analysis*, 27(2):488–506, 1990.