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`

Revised version June 15, 2022

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 for this work was a numerical weather prediction (NWP) code that uses a spectral element mesh discretization for its dynamics core and a cartesian tensor product mesh for the evaluation of its physics routines. This coupling of spectral element mesh, which uses nonuniformly spaced quadrature/collocation points, and uniformly-spaced cartesian mesh combined with the desire to maintain positivity when moving between these meshes 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 when mapping solution values between structured meshes. The theoretical basis for the algorithm builds on the data-bounded work of Berzins [4] to develop a new data-bounded and positivity-preserving methods for both evenly- and unevenly-spaced structure meshes. The new data-bounded interpolation (DBI) method in [9] relaxes conditions for data-boundedness, which gives greater accuracy than the conditions used in [4]. Ouermi et al. [9] further extended the DBI method to give a new positivity-preserving interpolation (PPI) method. The application of these new methods to numerical weather prediction examples 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 approaches 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. Overall, it will be shown that the new methods are well suited for function approximation and mapping data values between meshes for numerical weather examples. The generality of this approach sug-

gests that these methods also have application to other problems for which preserving positivity is important.

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. Both methods 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 non-oscillatory (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.

A DBI method was developed by Berzins using evenly spaced meshes from ENO methods [4]. This method was extended by the authors in [9, 8] to work for both evenly and unevenly spaced meshes and, more importantly, to the PPI method. Ouermi et al. [9] relaxed the conditions for data-boundedness which, gives greater accuracy compared to the conditions used in [4]. Both the improved DBI and the new PPI methods are used in this report. 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 until the required bounds are violated or $d+1$ points are selected, with d being the target degree of the interpolant. In addition to enforcing data-boundedness and positivity, the algorithm in [8] uses a user-supplied parameter st to guide the stencil construction procedure. When adding the next point to both the right or left of the current stencil meets the requirements for data-boundedness or positivity, the algorithm makes the selection based on the three cases below.

- If $st = 1$, the algorithm chooses the point with the smallest divided difference, as in the ENO stencil.

- If $st = 2$, the point to the left of the current stencil is selected if the number of points to the left of x_i is smaller than the number of points to right. Similarly, the point to the right is selected if the number of points to the right of x_i is smaller than the number of points to the left. When both the number of points to right and left are the same, the algorithm chooses the point with the smallest ratio of divided differences.

- If $st = 3$, the algorithm chooses the point that is closest to the starting interval I_i.

Enforcing positivity alone may still lead to undesirables oscillations. To address this limitation the algorithm, provides the parameters ϵ_0 and ϵ_1 that are used to impose an upper and lower bound for each interpolant. For each interval I_i, the bounds are constructed using the parameters ϵ_0 and

ϵ_1, and the data values u_i and u_{i+1}. Both the DBI and PPI methods and the algorithm details are described in [8] with numerical examples pertaining to NWP.

2 Comparison Methodology

2.1 Compared Methods

The numerical experiments in this report use the PCHIP [17], MQS [23], SPS [27, 28], B-splines [30], the improved DBI [4], and the new PPI methods [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`.

HPPIS: The DBI and PPI methods have been developed based on the theory and algorithm in [9, 8]. The software and implementation details can be found in [8].

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, SPS, DBI, and PPI against a standard interpolation method using five 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:

1. a uniform mesh that is constructed using uniformly spaced points. In this mesh, all the elements have the same size and the nodes are uniformly spaced inside each element.

2. an LGL mesh that consists of uniform elements with Legendre Gauss-Lobatto (LGL) quadrature nodes [37]. For example, the global interval is divided into $(N_i - 1)/j$ and $(N_i - 1)^2/j^2$ elements for the 1D and 2D examples, respectively. N_i is the total number of points used for the approximation and j is the target polynomial degree. For the 1D examples, $j + 1$ LGL quadrature nodes are placed inside each element. The 2D examples use a tensor product of $(j + 1) \times (j + 1) = (j + 1)^2$ LGL nodes in each element. This LGL mesh is different than the one used in [9] where the node distribution in each element remains fixed ($(N_i - 1)/8$ and $(N_i - 1)^2/16$) as the target polynomial changes. This fixed number of nodes in each element enables the study of the DBI and PPI methods in the case of a fixed LGL mesh with varying polynomial degrees.

In the figures presented in this section, the black and red plots represent the underlying function and its approximation using the different interpolation methods. Both the DBI and PPI methods use a mesh point selection method that favors a symmetric stencil about x_i by setting $st = 1$ with $\epsilon_0 = 0.01$ and $\epsilon_1 = 1.0$. The results in Figures 1 to 6 below demonstrate that the PCHIP, MQS, DBI, SPS, 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.
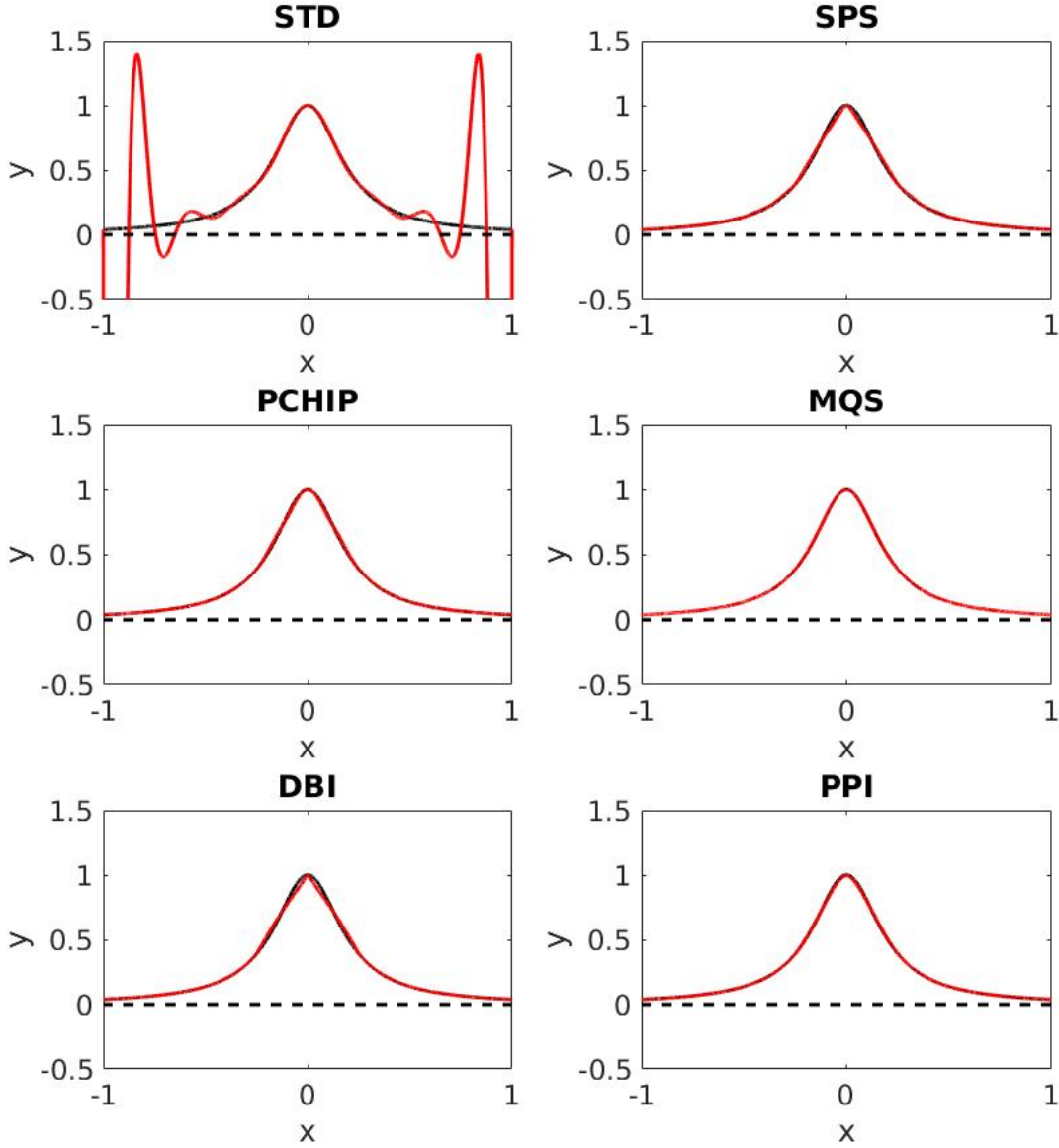
Figure 1: Approximation of the Runge function with the $N = 17$ points that are uniformly distributed on the interval $[-1, 1]$.
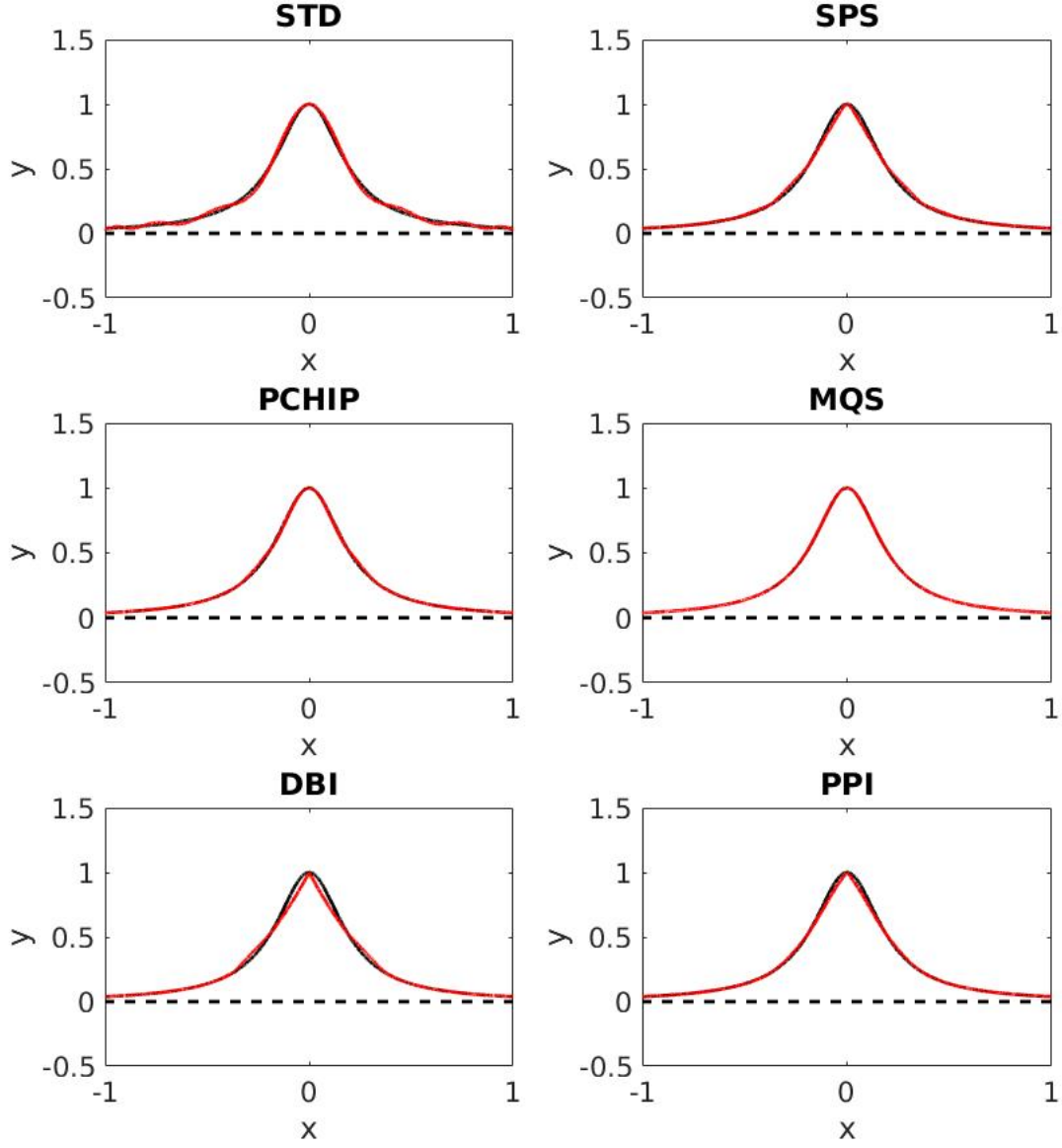
Figure 2: Approximation of the Runge function with the $N = 17$ LGL quadrature points distributed on the interval $[-1, 1]$.

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 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 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.
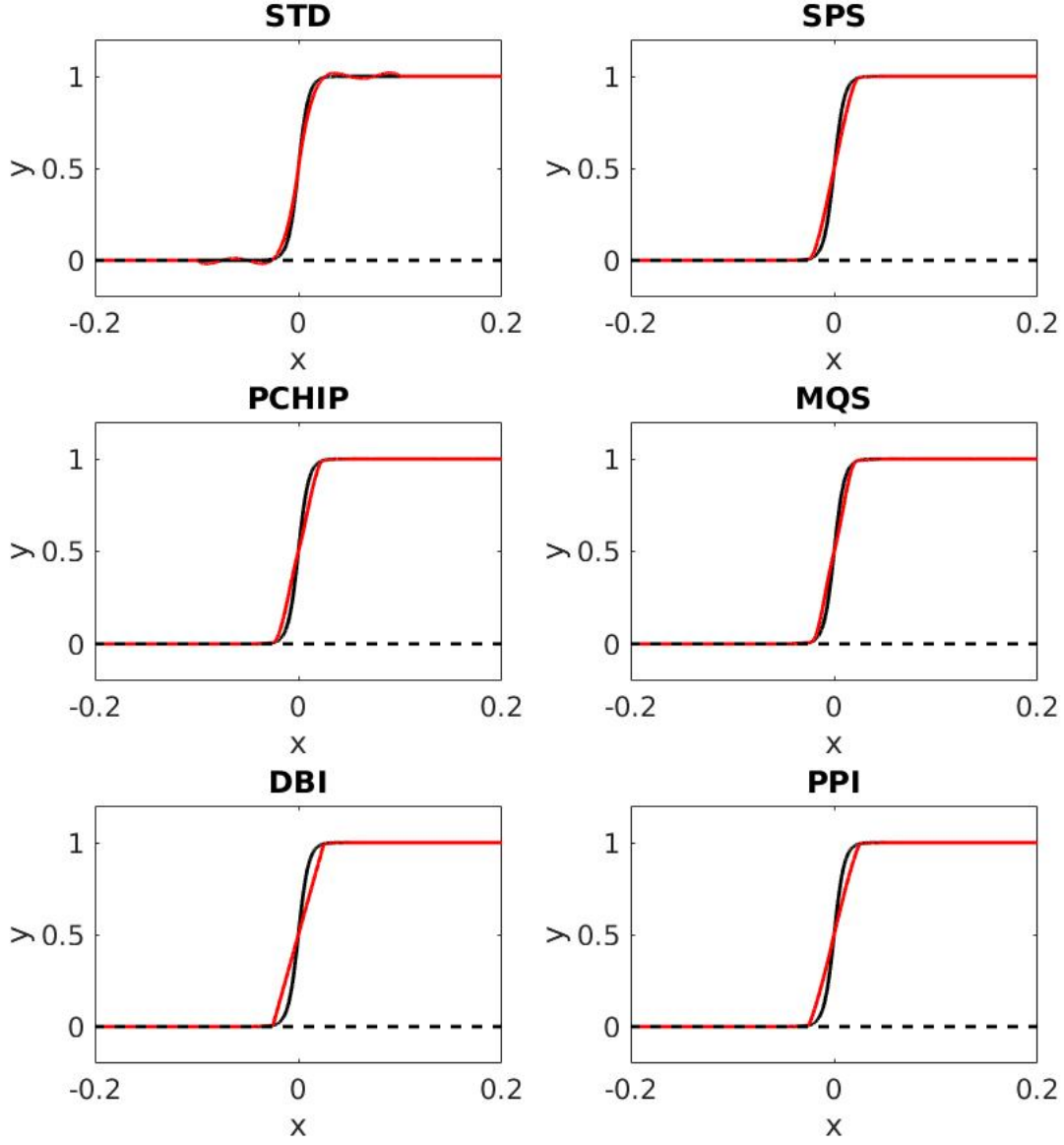
Figure 3: Approximation of $f_2(x) = \frac{1}{1+e^{-2kx}}$, $k = 100$, and $x \in [-0.2, 0.2]$, with $N = 17$ points. The points are uniformly distributed and the target polynomial degree for the DBI and PPI is $d = 8$.
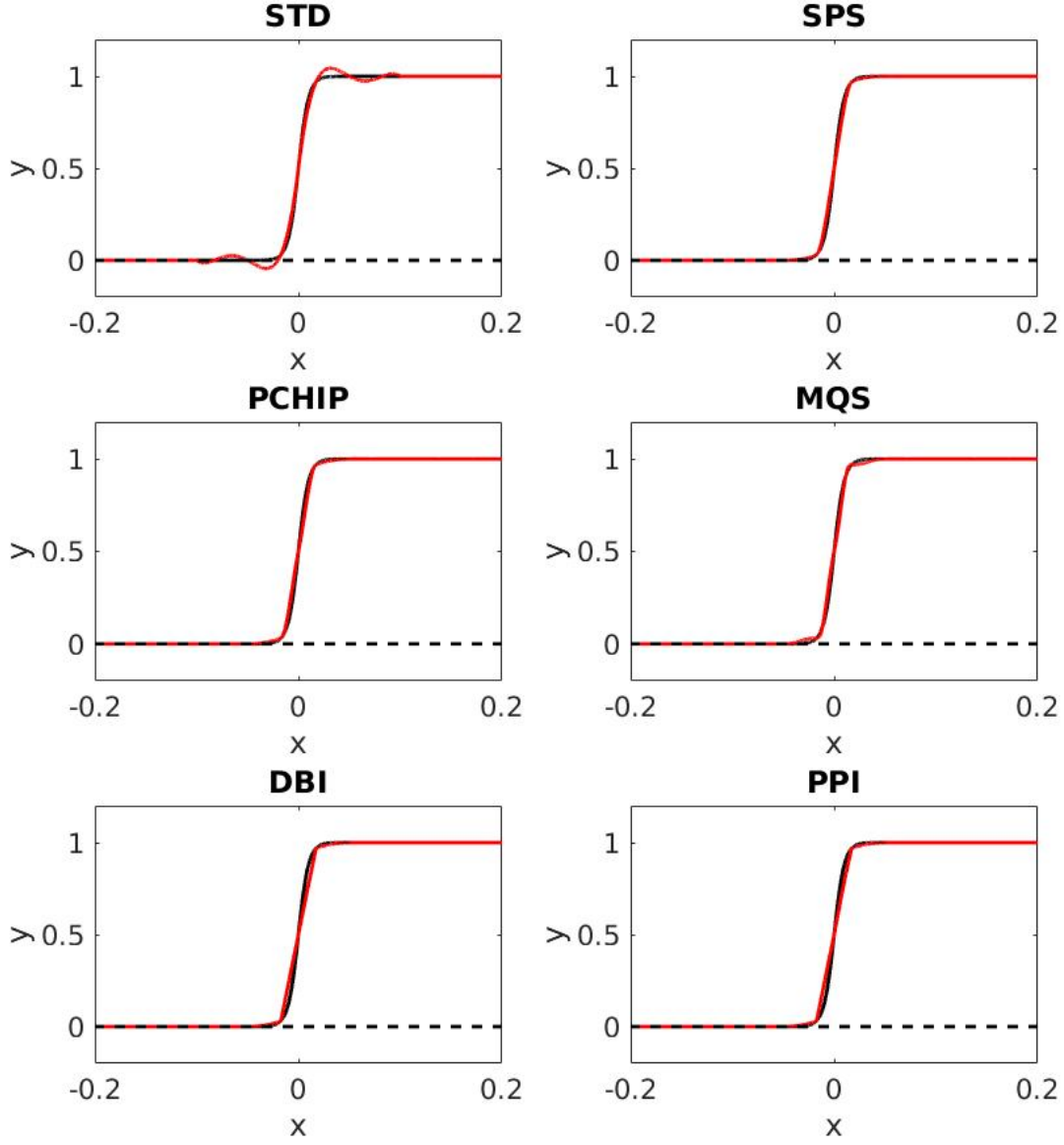
Figure 4: Approximation of $f_2(x) = \frac{1}{1+e^{-2kx}}$, $\quad k = 100$, and $x \in [-0.2, 0.2]$, with $N = 17$ points. The interval $[-0.2, 0.2]$ is divided in two elements and 9 LGL quadrature points are used in each interval.

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

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

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 methods is $d = 4$. Figures 5 and 6 demonstrate that the interpolants built using the PCHIP, MQS, SPS, 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.
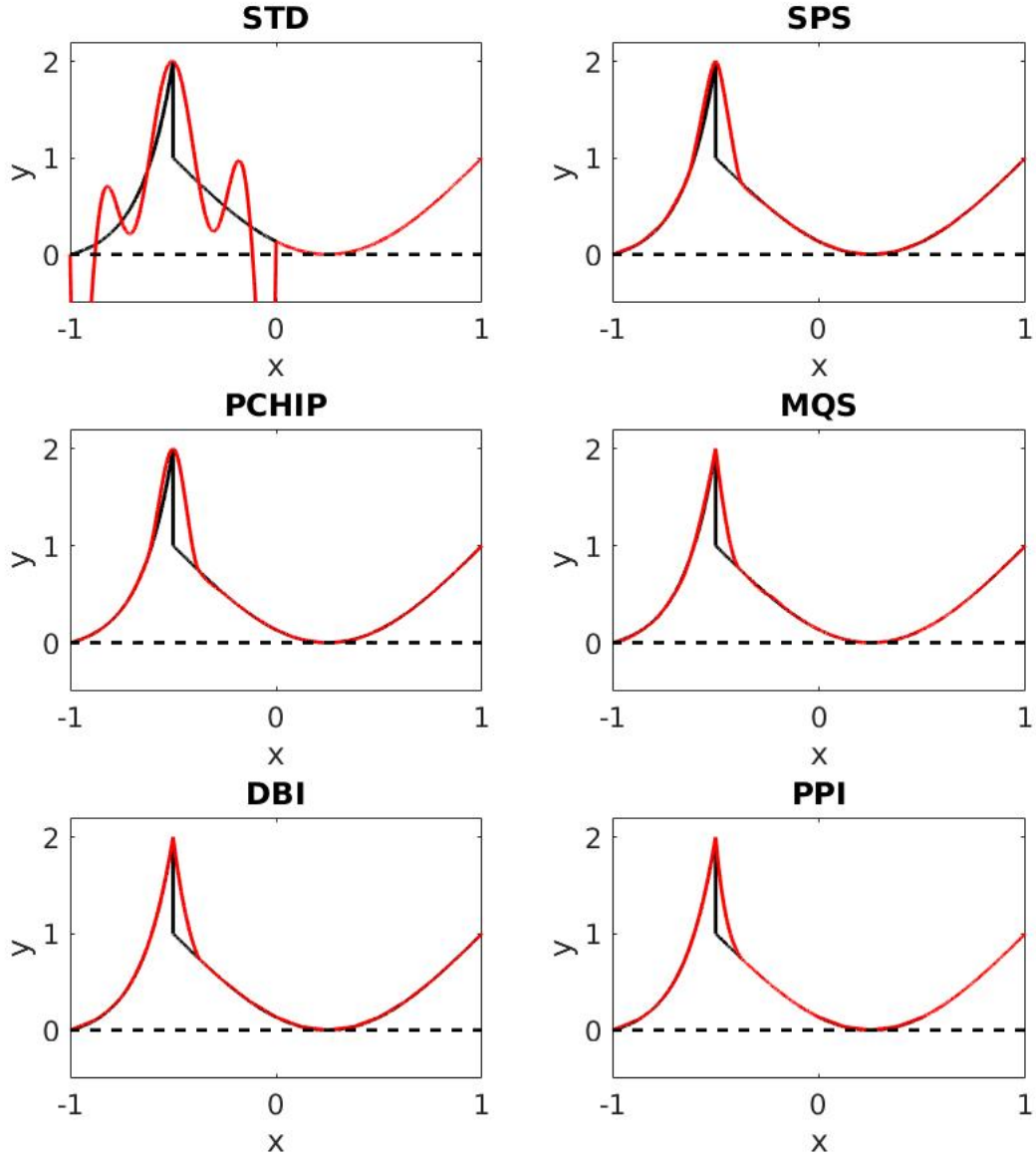
Figure 5: Approximation of $f_3(x)$ with $N = 17$ points. The points are distributed uniformly over the interval $[-1, 1]$.

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, SPS, DBI, and PPI methods preserve positivity and remove the oscillations.
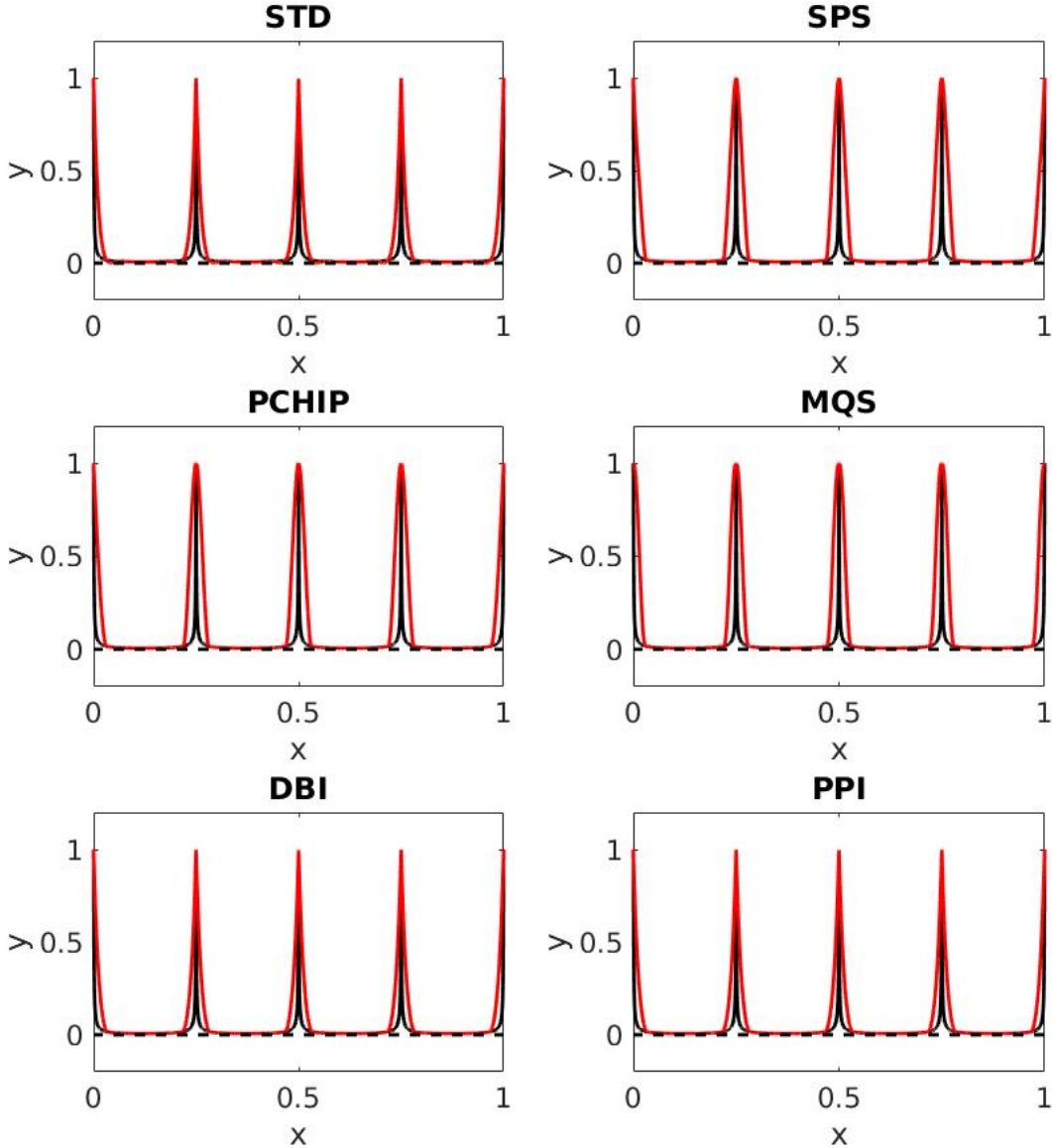
Figure 7: Approximation of $f_4(x)$, with $N = 33$ points. The points are uniformly distributed, and the target polynomial degree for the DBI and PPI is $d = 8$.
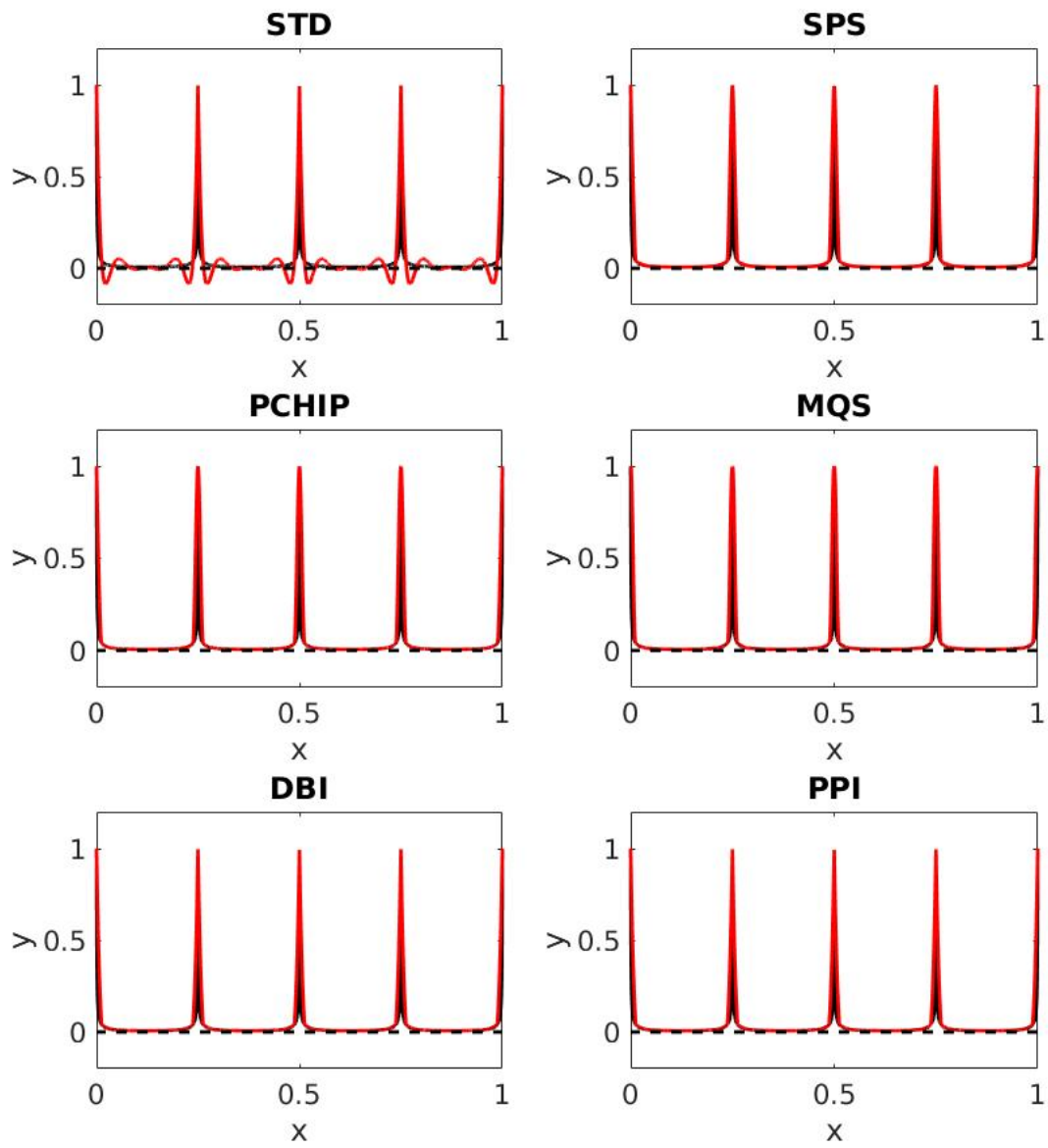
Figure 8: Approximation of $f_4(x)$, with $N = 33$ points. The interval $[0,1]$ is divided into four elements, and 9 quadrature points are used in each interval.

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:

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

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 PCHIP, MQS, SPS, DBI, and PPI can be used to enforce positivity as required.
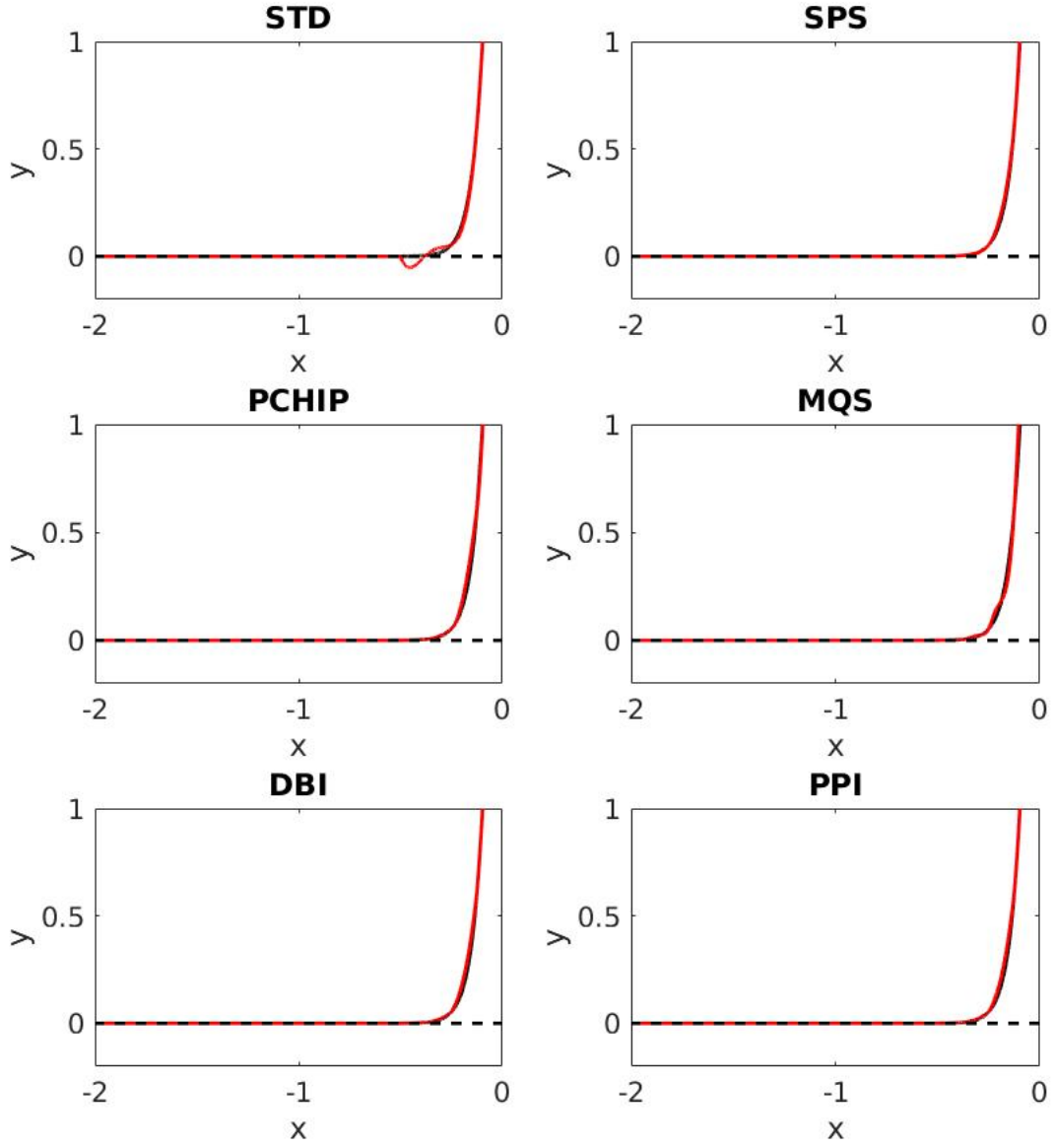
Figure 9: Approximation of $f_5(x)$, with $N = 17$ points. The points are uniformly distributed and the target polynomial degree for the DBI and PPI is $d = 4$.
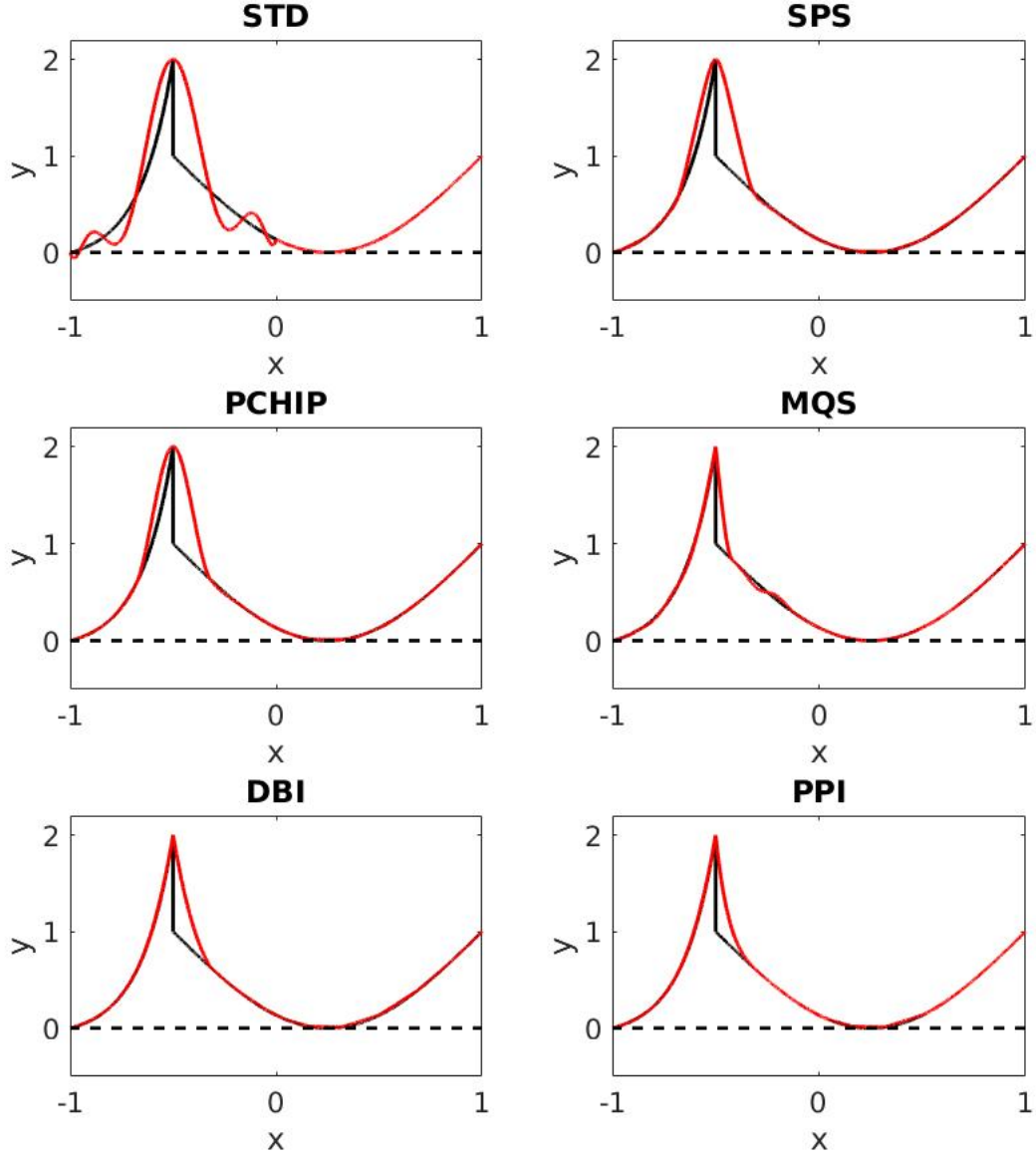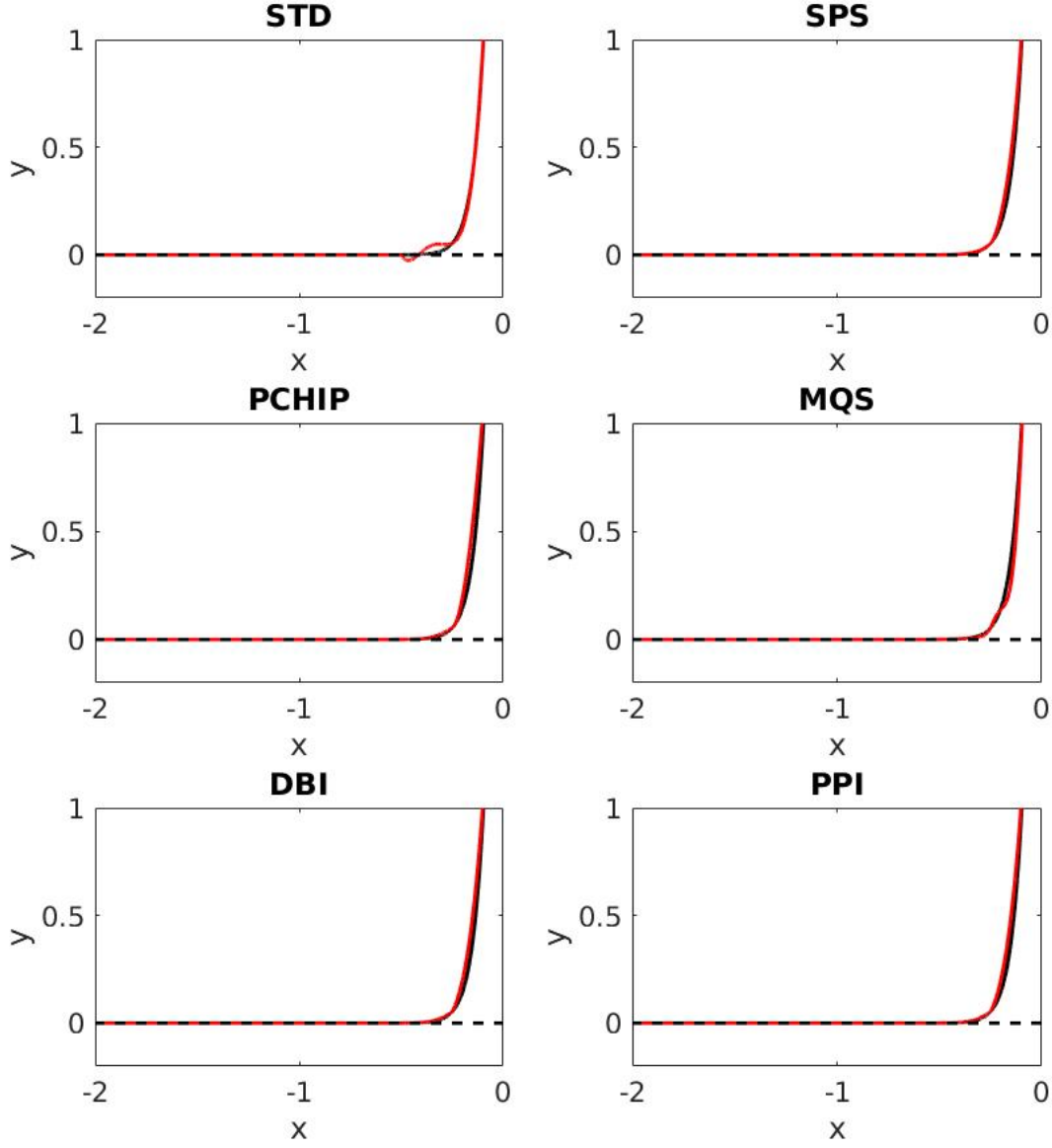
Figure 10: Approximation of $f_5(x)$, with $N = 17$ points. The interval $[-2, 0]$ is divided in four elements and 5 quadrature points are used in each interval.

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 this experiment, the parameters ϵ_0, ϵ_1, and st are chosen to be 0.01, 1, and 1, respectively. 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 and PPI methods lead to smaller error compared to the PCHIP, MQS, and SPS methods. As the average polynomial degree increases, the approximation using the DBI method does not improve because the global error is dominated by the local error from the intervals using lower degree interpolants compared to PPI. These results show that the conditions for data-boundedness may be more restrictive when it comes to enforcing positivity. The SPS method shows smaller errors compared to the other 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. 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 leads 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	–	–	–	–	2.49E-3	1	2.49E-3	1
33	–	–	–	–	6.22E-4	1	6.22E-4	1
65	–	–	–	–	1.56E-4	1	1.56E-4	1
129	–	–	–	–	3.89E-5	1	3.89E-5	1
257	–	–	–	–	9.72E-6	1	9.72E-6	1
	$\mathcal{P}_3$	$\mathcal{P}_5$			$\mathcal{P}_4$			
17	4.49E-4	4.47E-5	4.84E-4	4.52E-6	6.70E-06	3.94	2.52E-06	4
33	7.83E-5	7.42E-6	1.20E-4	2.07E-7	2.85E-07	3.97	6.94E-08	4
65	1.38E-5	1.31E-6	3.01E-5	1.22E-8	1.24E-08	3.98	1.96E-09	4
129	2.45E-6	2.31E-7	7.52E-6	7.56E-10	5.43E-10	3.99	5.73E-11	4
257	4.34E-7	4.09E-8	1.88E-6	4.72E-11	2.39E-11	4.00	1.73E-12	4
					$\mathcal{P}_8$			
17	–	–	2.00E-3	2.45E-9	6.21E-06	7.69	1.06E-09	8
33	–	–	4.96E-4	3.47E-12	2.76E-07	7.84	1.83E-12	8
65	–	–	1.24E-4	6.11E-15	1.22E-08	7.92	3.44E-15	8
129	–	–	3.10E-5	3.23E-15	5.40E-10	7.96	1.00E-15	8
257	–	–	7.74E-6	2.93E-15	2.39E-11	7.98	9.64E-16	8
					$\mathcal{P}_{16}$			
17	–	–	3.10E-3	5.61E-15	6.21E-06	15.19	3.98E-15	16
33	–	–	7.75E-4	4.35E-13	2.76E-07	15.59	1.95E-15	16
65	–	–	1.94E-4	2.75E-13	1.22E-08	15.80	4.65E-15	16
129	–	–	4.84E-5	9.00E-14	5.40E-10	15.90	2.33E-15	16
257	–	–	1.21E-5	6.83E-14	2.39E-11	15.95	1.10E-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	4
e_{33}/e_{65}	–	–	–	–	4	4
e_{65}/e_{129}	–	–	–	–	4	4
e_{129}/e_{257}	–	–	–	–	4	4
	$\mathcal{P}_3$	$\mathcal{P}_5$		$\mathcal{P}_4$		
e_{17}/e_{33}	5.73	6.02	4.03	21.84	24	36
e_{33}/e_{65}	5.67	5.67	3.99	16.97	23	35
e_{65}/e_{129}	5.63	5.66	4.00	16.13	23	34
e_{129}/e_{257}	5.64	5.66	4.00	16.02	23	33
				$\mathcal{P}_8$		
e_{17}/e_{33}	–	–	4.03	706.05	22	576
e_{33}/e_{65}	–	–	4.00	567.92	23	533
e_{65}/e_{129}	–	–	4.00	1.89	23	3
e_{129}/e_{257}	–	–	4.01	1.10	23	1
				$\mathcal{P}_{16}$		
e_{17}/e_{33}	–	–	4.01	0.01	22	2
e_{33}/e_{65}	–	–	3.99	1.58	23	0
e_{65}/e_{129}	–	–	4.01	3.06	23	2
e_{129}/e_{257}	–	–	4.00	1.32	23	2

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 DBI and PPI methods against the SPS, 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 15 – 18 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 with $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. 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. In the case of the uniform mesh, 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. 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	5.72E-3	8.34E-3	8.34E-3	4	7.02E-3	4
33	1.91E-3	3.95E-4	5.91E-4	5.91E-4	4	5.91E-4	4
65	3.70E-4	6.44E-5	4.26E-5	4.26E-5	3.98	2.39E-5	4
129	6.79E-5	5.27E-6	2.68E-6	2.68E-6	3.98	8.00E-7	4
257	1.22E-5	6.83E-7	8.63E-8	8.63E-8	4.00	2.55E-8	4
				$\mathcal{P}_8$			
17	–	–	1.21E-2	4.61E-3	7.88	3.11E-3	7.88
33	–	–	2.74E-3	4.43E-4	7.88	1.51E-4	8
65	–	–	6.86E-4	3.67E-5	7.92	1.05E-6	8
129	–	–	1.72E-4	2.56E-6	7.92	3.10E-9	8
257	–	–	4.30E-5	8.24E-8	7.97	6.80E-12	8
				$\mathcal{P}_{16}$			
17	–	–	1.64E-2	4.34E-3	11.31	3.44E-3	11.75
33	–	–	4.25E-3	4.21E-4	15.62	4.85E-5	16
65	–	–	1.07E-3	3.67E-5	15.69	5.92E-8	16
129	–	–	2.69E-4	2.56E-6	15.80	4.21E-12	16
257	–	–	6.71E-5	8.24E-8	15.91	2.18E-16	16

Table 3: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the Runge function $f_1(x) = \frac{1}{1+25x^2}$, $x \in [-1, 1]$. 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	6.29E-3	9.73E-3	8.63E-3	4	8.39E-3	4
33	1.86E-3	9.13E-4	1.63E-3	7.95E-4	4	7.80E-4	4
65	3.68E-4	8.47E-5	2.24E-4	4.76E-5	3.98	4.64E-5	4
129	7.20E-5	6.23E-6	6.03E-5	1.49E-6	3.98	1.27E-6	4
257	1.52E-5	5.72E-7	1.48E-5	4.68E-8	4	3.95E-8	4
				$\mathcal{P}_8$			
17	–	–	8.44E-3	3.49E-3	8.00	4.40E-3	8
33	–	–	2.69E-3	1.76E-4	7.88	1.76E-4	8
65	–	–	7.59E-4	3.25E-6	7.92	3.01E-6	8
129	–	–	2.61E-4	5.64E-8	7.94	8.82E-9	8
257	–	–	6.85E-5	3.51E-9	7.96	3.96E-11	8
				$\mathcal{P}_{16}$			
17	–	–	2.29E-2	9.12E-3	12.19	1.25E-2	12.62
33	–	–	3.26E-3	5.87E-5	15.28	5.86E-5	16
65	–	–	1.11E-3	1.41E-7	15.62	1.17E-7	16
129	–	–	3.12E-4	3.52E-9	15.90	4.44E-11	16
257	–	–	1.08E-4	1.56E-10	15.91	2.88E-15	16

Table 4: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the Runge function $f_1(x) = \frac{1}{1+25x^2}$, $x \in [-1, 1]$. 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 with $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. 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 DBI and PPI for $f_2(x)$. Overall, the results from Tables 5 and 6 indicate that the DBI and PPI methods have

26

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 both the DBI and PPI approaches are 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	2.23E-2	2.75	2.23E-2	3.38
33	3.38E-03	4.16E-3	3.72E-3	4.09E-3	3.62	4.10E-3	3.72
65	3.59E-04	2.29E-4	3.40E-4	3.05E-4	3.86	3.05E-4	3.86
129	4.21E-05	7.48E-6	5.36E-5	1.35E-5	3.88	1.35E-5	3.88
257	5.12E-06	2.16E-7	1.27E-5	4.71E-7	3.85	4.71E-7	3.86
				$\mathcal{P}_8$			
17	–	–	3.75E-3	2.08E-2	3.25	2.08E-2	5.50
33	–	–	5.24E-3	3.36E-3	3.88	3.33E-3	5.72
65	–	–	8.71E-4	1.38E-4	7.59	1.38E-4	7.59
129	–	–	2.08E-4	1.22E-6	7.68	1.22E-6	7.73
257	–	–	5.17E-5	4.44E-9	7.61	4.44E-9	7.67
				$\mathcal{P}_{16}$			
17	–	–	5.90E-3	2.00E-2	4.25	2.00E-2	6.62
33	–	–	6.34E-3	2.93E-3	4.38	2.91E-3	9.72
65	–	–	1.30E-3	9.17E-5	14.64	9.17E-5	14.86
129	–	–	3.23E-4	1.70E-7	15.15	1.70E-7	15.41
257	–	–	8.08E-5	2.64E-11	15.05	2.64E-11	15.30

Table 5: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_2(x) = \frac{1}{1+e^{-2kx}}$, $k = 100$, 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	1.26E-2	2.88	1.25E-2	3.44
33	2.50E-3	2.50E-3	8.04E-4	3.11E-3	3.03	2.83E-3	3.44
65	6.36E-4	2.11E-4	4.18E-4	3.28E-4	3.81	3.72E-4	3.84
129	1.02E-4	1.01E-5	9.07E-5	1.55E-5	3.88	1.55E-5	3.88
257	1.83E-5	2.93E-7	1.83E-5	6.29E-7	3.85	6.29E-7	3.86
				$\mathcal{P}_8$			
17	–	–	4.43E-3	4.87E-3	3.50	4.68E-3	5.00
33	–	–	2.51E-3	8.71E-4	4.34	7.84E-4	5.75
65	–	–	1.00E-3	7.57E-5	6.64	1.24E-4	7.28
129	–	–	3.65E-4	2.17E-6	7.65	2.17E-6	7.73
257	–	–	9.11E-5	1.95E-8	7.55	1.95E-8	7.73
				$\mathcal{P}_{16}$			
17	–	–	4.52E-2	3.77E-2	3.81	3.73E-2	7.25
33	–	–	2.03E-3	2.53E-4	5.56	5.23E-4	9.84
65	–	–	9.55E-4	1.37E-5	10.53	6.95E-5	12.56
129	–	–	4.16E-4	2.19E-7	15.16	2.19E-7	15.30
257	–	–	1.51E-4	1.56E-10	14.96	1.56E-10	15.30

Table 6: L^2-errors when using the PCHIP, MQS, SPS, DBI, and PPI methods to approximate the function $f_2(x) = \frac{1}{1+e^{-2kx}}$, $k = 100$, 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 with $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. 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 DBI and PPI methods give slightly better approximation results compared to the other methods. The average polynomial degrees for the DBI and PPI

approaches show that high-order polynomials are used. This suggest that in the smooth regions away from the discontinuity the DBI and PPI approaches lead to high-order accuracy. However, at the discontinuity, the DBI and PPI and other methods struggle to represent the underlying function. This example shows that both the DBI and PPI methods are appropriate approaches for interpolating from one mesh to another, because around the discontinuity, the methods are as accurate as the other ones, and in smooth regions the method gives better approximation results than the other approaches. The results from Tables 7 and 8 show that the L^2-error from the four interpolation methods have the same order.

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	–	–	–	1.82E-1	1	1.82E-1	1
33	–	–	–	1.39E-1	1	1.39E-1	1
65	–	–	–	1.01E-1	1	1.01E-1	1
129	–	–	–	7.16E-2	1	7.16E-2	1
257	–	–	–	5.05E-2	1	5.05E-2	1
	$\mathcal{P}_3$	$\mathcal{P}_5$		$\mathcal{P}_4$			
17	1.77E-1	1.59E-1	2.32E-1	1.82E-1	3.62	1.71E-1	3.81
33	1.39E-1	1.11E-1	1.56E-1	1.39E-1	3.97	1.29E-1	3.97
65	1.03E-1	7.90E-2	1.09E-1	9.38E-2	3.98	9.38E-2	3.98
129	7.42E-2	5.63E-2	7.69E-2	6.69E-2	3.99	6.70E-2	3.99
257	5.28E-2	4.04E-2	5.45E-2	4.73E-2	4	4.74E-2	4
				$\mathcal{P}_8$			
17	–	–	2.25E-1	1.83E-1	6.62	1.70E-1	7.06
33	–	–	1.53E-1	1.36E-1	7.81	1.31E-1	7.81
65	–	–	1.07E-1	9.62E-2	7.92	9.65E-2	7.92
129	–	–	7.58E-2	6.90E-2	7.96	6.93E-2	7.96
257	–	–	5.37E-2	4.90E-2	7.98	4.92E-2	7.98
				$\mathcal{P}_{16}$			
17	–	–	2.25E-1	1.82E-1	12.06	1.66E-1	13.06
33	–	–	1.50E-1	1.37E-1	14.09	1.33E-1	14.19
65	–	–	1.05E-1	9.75E-2	15.78	9.81E-2	15.78
129	–	–	7.45E-2	7.02E-2	15.90	7.07E-2	15.90
257	–	–	5.29E-2	4.99E-2	15.95	5.03E-2	15.95

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	–	–	–	1.82E-1	1	1.82E-1	1
33	–	–	–	1.39E-1	1	1.39E-1	1
65	–	–	–	1.01E-1	1	1.01E-1	1
129	–	–	–	7.16E-2	1	7.16E-2	1
257	–	–	–	5.05E-2	1	5.05E-2	1
	$\mathcal{P}_3$	$\mathcal{P}_5$		$\mathcal{P}_4$			
17	1.64E-1	1.39E-1	1.87E-1	1.61E-1	3.81	1.58E-1	3.81
33	1.20E-1	9.79E-2	1.29E-1	1.18E-1	3.97	1.18E-1	3.97
65	8.70E-2	6.94E-2	9.04E-2	8.32E-2	3.98	8.53E-2	3.98
129	6.21E-2	4.96E-2	6.39E-2	5.93E-2	3.99	6.08E-2	3.99
257	4.39E-2	3.58E-2	4.54E-2	4.19E-2	4.00	4.29E-2	4
				$\mathcal{P}_8$			
17	–	–	2.84E-1	1.85E-1	7.38	1.81E-01	7.50
33	–	–	9.61E-2	9.38E-2	7.62	1.27E-01	7.66
65	–	–	6.79E-2	6.75E-2	7.92	9.33E-02	7.92
129	–	–	4.82E-2	4.79E-2	7.96	6.72E-02	7.96
257	–	–	3.44E-2	3.37E-2	7.98	4.77E-02	7.98
				$\mathcal{P}_{16}$			
17	–	–	1.11E-1	1.08E-1	11.62	1.51E-1	12.12
33	–	–	1.86E-1	1.66E-1	14.31	1.55E-1	14.88
65	–	–	4.91E-2	5.06E-2	15.56	8.28E-2	15.58
129	–	–	3.51E-2	3.56E-2	15.90	5.92E-2	15.90
257	–	–	2.53E-2	2.48E-2	15.94	4.19E-2	15.94

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.

5.4 Example IV $f_4(x)$

The fourth example uses the function $f_4(x)$ defined in Equation 4 with $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. $f_4(x)$ 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	3.74E-01	3.95E-1	$\mathcal{P}_4$	3.72E-1	3.49E-1	2.62	3.49E-1	2.88
33	2.47E-01	2.59E-1	$\mathcal{P}_8$	2.46E-1	2.19E-1	5.19	2.19E-1	6.19
65	1.55E-01	1.63E-1	$\mathcal{P}_{16}$	1.54E-1	1.32E-1	13.34	1.32E-1	15.34
				$ne = 8$				
33	3.84E-01	3.94E-1	$\mathcal{P}_4$	3.83E-1	4.04E-1	2.69	4.04E-1	2.94
65	2.54E-01	2.59E-1	$\mathcal{P}_8$	2.52E-1	2.60E-1	5.34	2.74E-1	6.34
129	1.61E-01	8.23E-2	$\mathcal{P}_{16}$	1.57E-1	1.67E-1	13.55	1.78E-1	15.30
				$ne = 16$				
65	3.90E-01	3.93E-1	$\mathcal{P}_4$	3.89E-1	3.61E-1	2.72	3.61E-1	2.97
129	2.58E-01	2.58E-1	$\mathcal{P}_8$	2.55E-1	2.26E-1	5.42	2.26E-1	6.42
257	1.63E-01	8.06E-2	$\mathcal{P}_{16}$	1.58E-1	1.36E-1	13.65	1.36E-1	15.65

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	2.32E-1	2.75	2.32E-1	3.00
33	1.33E-1	1.39E-1	$\mathcal{P}_8$	1.33E-1	9.60E-2	5.25	9.60E-2	6.25
65	3.80E-2	3.92E-2	$\mathcal{P}_{16}$	3.77E-2	2.11E-2	11.88	2.11E-2	13.31
				$ne = 8$				
33	3.10E-1	3.17E-1	$\mathcal{P}_4$	3.10E-1	3.42E-1	2.75	3.42E-1	3.00
65	1.37E-1	1.39E-1	$\mathcal{P}_8$	1.36E-1	1.59E-1	5.25	1.65E-1	6.12
129	3.98E-2	3.72E-2	$\mathcal{P}_{16}$	3.87E-2	5.33E-2	11.88	5.60E-2	13.31
				$ne = 16$				
65	3.14E-1	3.16E-1	$\mathcal{P}_4$	3.14E-1	2.32E-1	2.75	2.32E-1	3.00
129	1.39E-1	1.38E-1	$\mathcal{P}_8$	1.37E-1	9.60E-2	5.25	9.60E-2	6.25
257	4.07E-2	3.37E-2	$\mathcal{P}_{16}$	3.90E-2	2.11E-2	11.88	2.11E-2	13.31

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 with $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 1$. $f_5(x)$ 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 shows that the approximation errors from PCHIP, MQS, and SPS methods improve slowly compared to the DBI and 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. Overall, the results from Tables 11 and 12 show that the DBI and PPI methods 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	3.56	2.36E-2	3.56
33	9.95E-3	6.04E-3	$\mathcal{P}_8$	1.36E-2	4.20E-4	7.59	4.20E-4	7.62
65	1.67E-3	3.82E-4	$\mathcal{P}_{16}$	5.77E-3	3.65E-5	14.98	3.65E-5	14.98
				$ne = 8$				
33	1.99E-2	1.20E-2	$\mathcal{P}_4$	1.29E-2	1.65E-2	3.91	1.65E-2	3.91
65	3.35E-3	7.63E-4	$\mathcal{P}_8$	7.42E-3	2.17E-4	7.67	2.17E-4	7.67
129	3.70E-4	4.44E-5	$\mathcal{P}_{16}$	2.89E-3	5.01E-5	14.84	5.01E-5	14.88
				$ne = 16$				
33	6.73E-3	2.46E-3	$\mathcal{P}_4$	4.57E-3	1.18E-3	3.86	1.18E-3	3.86
65	8.22E-4	4.53E-4	$\mathcal{P}_8$	3.61E-3	5.27E-5	7.51	5.27E-5	7.51
256	1.53e-4	1.59E-4	$\mathcal{P}_{16}$	1.42E-3	4.27E-11	14.65	4.27E-11	14.70

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	5.09E-2	3.06	5.09E-2	3.06
33	7.43E-3	4.29E-3	$\mathcal{P}_8$	1.36E-2	1.47E-3	6.34	1.47E-3	6.44
65	9.48E-4	1.58E-4	$\mathcal{P}_{16}$	5.77E-3	3.15E-6	14.83	3.15E-6	14.84
				$ne = 8$				
33	2.57E-2	9.45E-3	$\mathcal{P}_4$	1.29E-2	1.52E-2	3.84	1.52E-2	3.84
65	3.18E-3	9.15E-4	$\mathcal{P}_8$	7.42E-3	2.66E-4	7.66	2.66E-4	7.67
129	4.08E-4	2.71E-5	$\mathcal{P}_{16}$	2.89E-3	3.75E-4	14.88	4.53E-4	14.88
				$ne = 16$				
33	8.03E-3	3.77E-3	$\mathcal{P}_4$	4.57E-3	1.91E-3	3.81	1.91E-3	3.81
65	9.93E-4	2.22E-4	$\mathcal{P}_8$	3.61E-3	2.23E-6	7.51	2.23E-6	7.51
256	1.23E-4	3.30E-5	$\mathcal{P}_{16}$	1.42E-3	2.63E-12	14.37	2.63E-12	14.50

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 single column test case that was developed to measure and study changes in the properties of heat, moisture, and momentum. In this example, the dynamics and physics results are calculated on different meshes. The dynamics uses uniformly spaced points that indicate the boundary of each level in the vertical column. The physics mesh is constructed using the mid-point of each level. The advections in the dynamics are approximated using a fifth order weighted essentially non-oscillatory (WENO) and third-order Runge-Kutta methods [41]. At each time step, the dynamics are calculated on the dynamics mesh, and the results are interpolated to the physics mesh for the use of the physics routines. The physics terms are calculated using the physics mesh, and the results are interpolated back to the dynamics mesh.

As in [42], let q_c be the cloud water mixing ratio profile in the different experiments. Figures 11-15 show the cloud mixing ratio profile q_c at $t = 5h$ that is used as input for the physics routines. The physics calculations require positive input values for q_c. Figure 11 shows the target profile for q_c. This target profile is obtained by using the same mesh for both dynamics and physics calculations where mapping is not required and q_c remains positive during the simulation. In addition, as the temporal and spatial resolution increases, q_c converges to the profile shown in Figure 11. Figures 12-15 are used to investigate different interpolation methods for mapping the

solution values between meshes in the case where the dynamics and physics are calculated using different meshes.
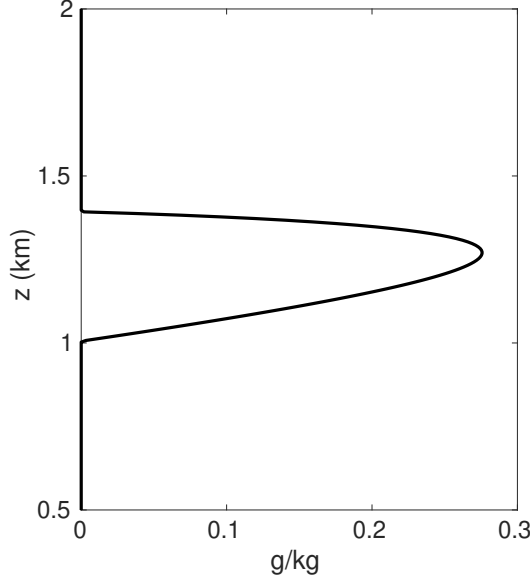

Figure 11: Target cloud mixing ratio q_c profile from BOMEX test case at $t = 5h$ with $nz = 600$ points. A fifth-order WENO and third-order Runge-Kutta schemes with $CFL = 0.1$ are used for the dynamics (advection). The same mesh is used for the dynamics and physics calculations.

Figure 12 shows the cloud mixing ratio profiles q_c for the target and approximated solution at $t = 5h$. In the case of the approximated solution, a fifth-order standard polynomial interpolation is used when mapping between dynamics and physics meshes. For a given interval I_i, the polynomial interpolant is constructed using the stencil $\mathcal{V}_4 = \{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}, x_{i+3}, \}$. At the boundary and nearby boundary intervals, the stencil $\mathcal{V}_4$ is biased toward the interior of the domain. The results in Figure 12 demonstrate that using the standard polynomial interpolation lead to oscillations, negatives values, and an overestimation of the peak and total cloud mixing ratio of the profile q_c. Using standard polynomial interpolation leads to an overproduction of the total cloud mixing ratio by 93.45%. The peak is $max(q_c) = 0.46g/kg$, which is larger than the target peak $max(q_c) = 0.28g/kg$.
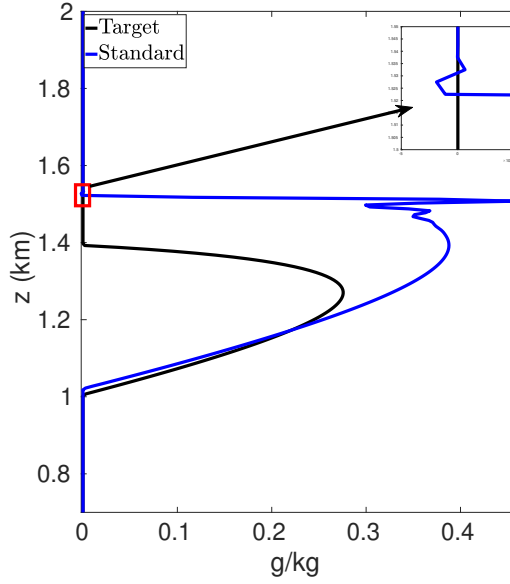
Figure 12: Cloud mixing ratio q_c profile from BOMEX test case at $t = 5h$ with $nz = 600$ points. The profile in black is the target solution. The profile in blue is obtained using a standard interpolation method when mapping solution values between dynamics and physics meshes. A fifth-order WENO and third-order Runge-Kutta schemes with $CFL = 0.1$ are used for the dynamics (advection).

The negative values in Figure 12 can be removed via "clipping". "Clipping" is a procedure that consists of removing the negative values by setting them to zero. Figure 13 shows the cloud mixing ratio profiles for the target solution and an approximated solution that uses "clipping" to remove the negative values at each time step. The approximated solution uses a standard interpolation to map the data values from one mesh to another. The interpolant for each interval is constructed using the stencil $\mathcal{V}_4 = \{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}, x_{i+3}, \}$ with a fifth order polynomial. Once the interpolation is completed, "clipping" is used to remove the negative values. Figure 13 shows that using "clipping" still allows for oscillations and a positive bias in the prediction of cloud mixing ratio q_c. The total cloud mixing ratio is 2.09 greater than the target solution and the peak $max(q_c) = 0.46g/kg$ is larger than the target peak $max(q_c) = 0.28g/kg$.
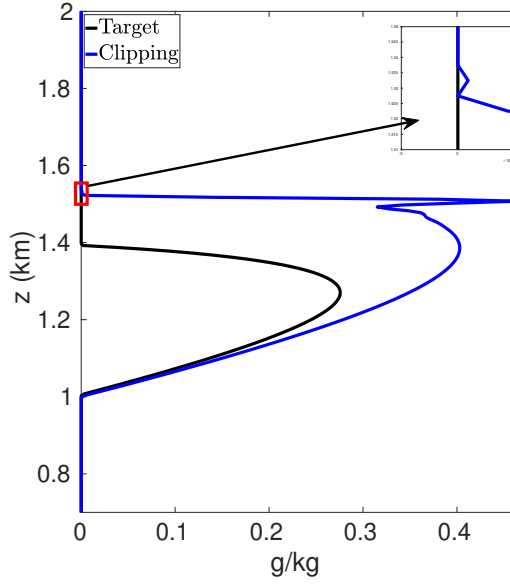
Figure 13: Cloud mixing ratio q_c profile from BOMEX test case at $t = 5h$ with $nz = 601$ points. The profile in black is the target solution. The profile in blue is obtained using a standard interpolation method when mapping solution values between dynamics and physics meshes. "Clipping is used after the interpolation to remove any negative value. A fifth-order WENO and third-order Runge-Kutta schemes with $CFL = 0.1$ are used for the dynamics (advection).

Using PCHIP for the mapping between the dynamics and physics meshes eliminates the negative values, remove oscillations, and reduces the positive bias in the cloud mixing ratio prediction compared to the standard interpolation with and without "clipping". Figure 14 shows the target profile q_c and an approximated profile that uses PCHIP for mapping solution values between dynamics and physics meshes. The total cloud mixing ratio is now 27.21% less than the target with a peak $max(q_c) = 0.21g/kg$. In the case of the BOMEX test case, NEPTUNE, and similar codes, using PCHIP for mapping data values from one mesh to another can degrade the high-order accuracy obtained from the high-order methods used for the dynamics calculations. PCHIP is only third-order whereas the dynamics calculations use a fifth order method. This limitation can be addressed via high-order data-bounded or positivity-preserving methods.
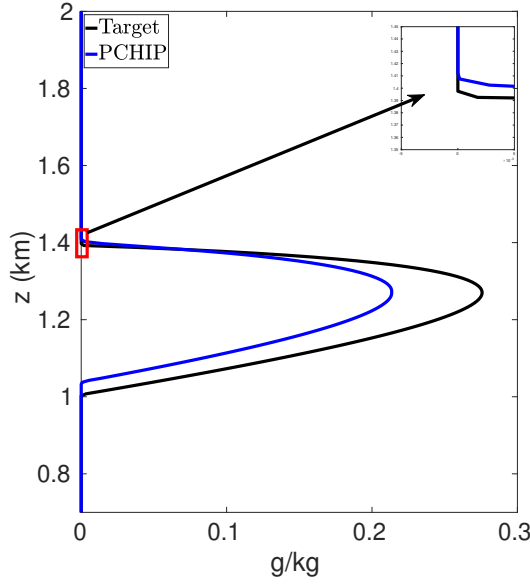
Figure 14: Cloud mixing ratio q_c profile from BOMEX test case at $t = 5h$ with $nz = 600$ points. The profile in black is the target solution. The profile in blue is obtained using the PCHIP method when mapping solution values between dynamics and physics meshes. A fifth-order WENO and third-order Runge-Kutta schemes with $CFL = 0.1$ are used for the dynamics (advection).

Figure 15 shows cloud mixing ratio profiles for the target and approximated solutions that use the DBI and PPI methods to map the solution values between meshes. The maximum polynomial degree for the DBI and PPI methods is set to 5 and 7, and the parameters ϵ_0 and ϵ_1 are both set a value of 10^{-5}. For larger values of ϵ_0 and ϵ_1, the PPI approach introduces oscillations that lead to positive bias prediction of the cloud mixing ratio. These oscillation are caused by the relaxed nature of the PPI approach, which still allows the interpolants to oscillate while remaining positive. The positive bias and oscillations can be removed using the DBI or PPI method with small values for ϵ_0 and ϵ_1. When using the PPI method for mapping, the total amount of the cloud mixing ratio is less than the target for $st = 1$ and more than the target for $st = 2$ and $st = 3$. The parameter st is described in Section 1.4. Figure 15 shows that using the DBI and PPI methods with $\epsilon_0 = \epsilon_1 = 10^{-5}$ to map data values between the dynamics and physics meshes eliminates the negative values, removes the oscillations, and significantly reduces the positive bias in the cloud mixing ratio prediction. Using the DBI and PPI methods leads to better approximation of the peak value of the total cloud mixing ratio compared to using the standard interpolation and PCHIP approaches. The best approximation of the total amount of the cloud mixing ratio is with the DBI method, which is 7.57% more than the target with a peak of $max(q_c) = 0.28g/kg$.
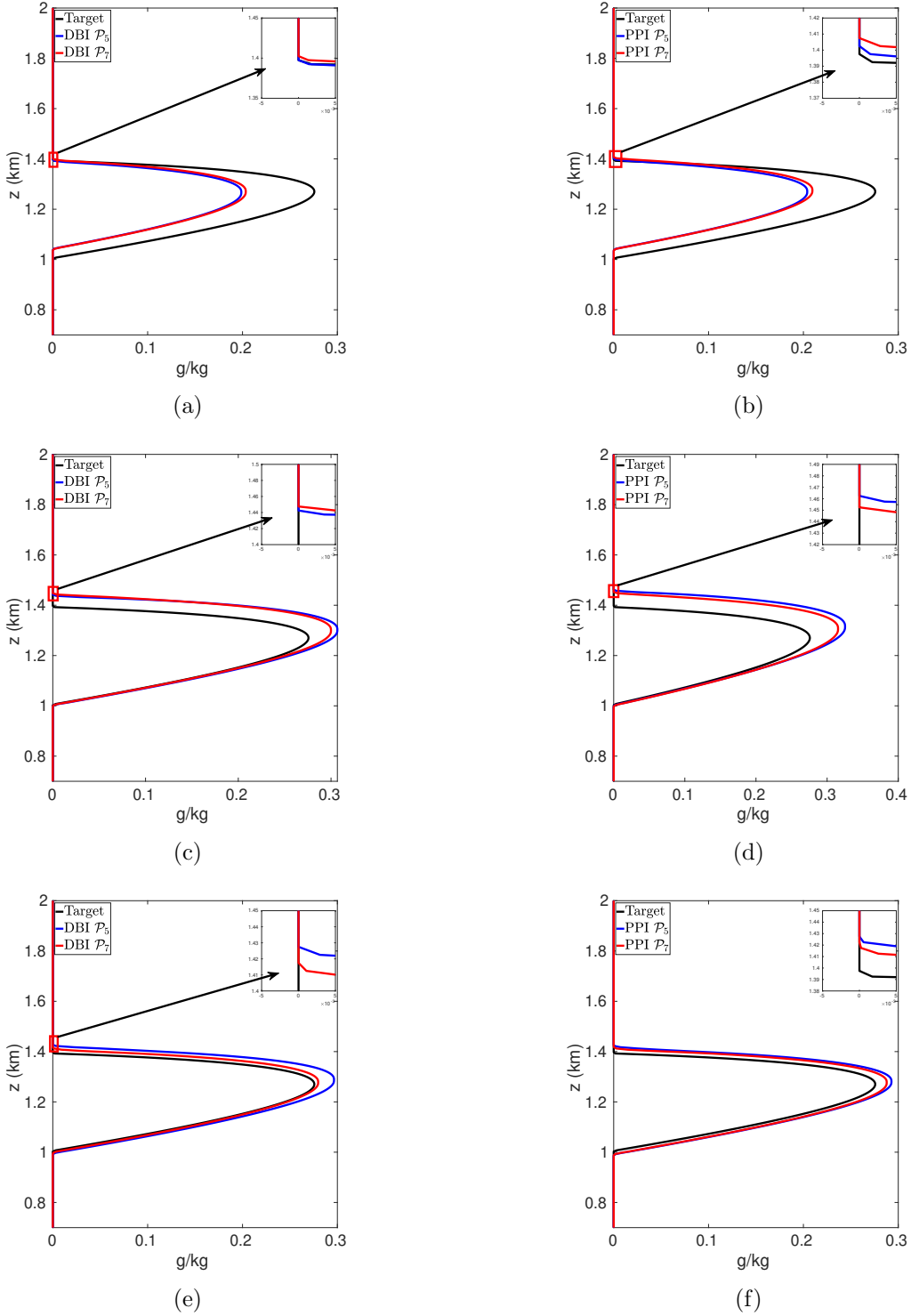
Figure 15: Cloud mixing ratio q_c profile from BOMEX test case at $t = 5h$ with $nz = 600$ points with $\epsilon_0 = \epsilon_1 = 10^{-5}$. The profile in black is the target solution. The profiles on the left and right are obtained using the DBI and PPI methods, respectively, to map solution values between meshes. The maximum polynomial degrees are set to 5 and 7 for the blue and red plots, respectively. A fifth-order WENO and third-order Runge-Kutta schemes with $CFL = 0.1$ are used for the dynamics (advection).

In summary, using DBI and PPI methods to map data values between both dynamics and physics meshes produces better approximation results compared to the standard interpolation and PCHIP methods. Tables 13 and 14 provide a summary of the maximum values and the total amount of the cloud mixing ratios for each case. The DBI and PPI methods with a target polynomial set to $d = 7$ lead to a better approximation of the peak and the total the cloud mixing ratios compared to the standard interpolation and PCHIP approaches. The results from Tables 13 and 14 indicate that the DBI method is the most suitable approach to map data values between meshes for the BOMEX test case. This study provided an example demonstrating how to use the DBI and PPI methods for mapping data values between meshes in the context of NWP. The BOMEX example also demonstrated that positivity alone may not be sufficient to remove oscillations in the solution, and the interpolants may need to be constrained to be between the data values for better approximation.

	Target	STD	Clipping	PCHIP
maximum q_c	0.28	0.46	0.46	0.21
total q_c	69.82	135.07	145.89	50.82

Table 13: Maximum values of q_c and the total amount of the cloud mixing ratio at $t = 5h$ with $nz = 600$ points. The total amount of the cloud mixing ratio is calculated by estimating the integral q_c. The units of q_c are is g/kg.

	$st = 1$	$st = 2$	$st = 3$	$st = 1$	$st = 2$	$st = 3$	
		$\mathcal{P}_5$			$\mathcal{P}_7$		Target
			DBI				
maximum q_c	0.20	0.20	0.31	0.30	0.30	0.28	0.28
total q_c	45.91	47.74	87.98	86.57	82.67	75.11	69.82
			PPI				
maximum q_c	0.20	0.21	0.33	0.32	0.29	0.29	0.28
total q_c	47.87	50.09	97.60	92.54	81.44	78.85	69.82

Table 14: Maximum values of q_c and the total amount of the cloud mixing ratio at $t = 5h$ with $nz = 600$ points and $\epsilon_0 = \epsilon_1 = 10^{-5}$. The total amount of the cloud mixing ratio is calculated by estimating the integral q_c. The units of q_c are is 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:

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

41

For the DBI and PPI algorithm, $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. The results from Tables 15 and 16 show that the DBI and PPI methods give smaller approximation errors compared to the PCHIP and SPS methods. In this case, the DBI and PPI methods use higher order polynomial interpolants for each interval. These higher order interpolants help improve the approximation compared to the PCHIP and SPS.

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	–	–	7.06E-5	1	7.06E-5	1
	$\mathcal{P}_3$		$\mathcal{P}_4$			
17	5.01E-3	3.61E-3	5.16E-3	3.97	4.28E-3	4
33	1.23E-3	5.44E-4	3.51E-4	3.98	3.31E-4	4
65	2.33E-4	1.23E-4	2.55E-5	3.98	1.31E-5	4
129	4.27E-5	3.07E-5	1.20E-6	3.99	4.36E-7	4
257	7.72E-6	3.34E-6	4.96E-8	4	1.39E-8	4
			$\mathcal{P}_8$			
17	–	8.55E-3	3.19E-3	7.75	1.84E-3	7.99
33	–	1.99E-3	2.78E-4	7.82	7.86E-5	8
65	–	4.97E-4	2.31E-5	7.90	5.14E-7	8
129	–	1.25E-4	1.13E-6	7.95	1.49E-9	8
257	–	3.16E-5	4.78E-8	7.98	3.25E-12	8
			$\mathcal{P}_{16}$			
17	–	1.19E-2	3.49E-3	13.15	2.83E-3	14.26
33	–	3.10E-3	2.74E-4	15.43	2.68E-5	16
65	–	7.82E-4	2.30E-5	15.69	2.63E-8	16
129	–	1.96E-4	1.13E-6	15.84	1.77E-12	16
257	–	4.93E-5	4.76E-8	15.92	1.89E-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. 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	6.31E-03	5.41E-3	5.54E-3	3.97	5.33E-3	4
33	4.15E-04	1.05E-3	4.59E-4	3.98	4.53E-4	4
65	1.07E-04	1.61E-4	2.67E-5	3.98	2.54E-5	4
129	2.46E-05	4.31E-5	7.97E-7	3.99	6.86E-7	4
257	5.20E-06	1.12E-5	2.79E-8	4.00	2.15E-8	4
			$\mathcal{P}_8$			
17	–	5.89E-3	3.02E-3	7.75	2.81E-3	7.99
33	–	1.72E-3	9.41E-5	7.82	9.34E-5	8
65	–	5.44E-4	1.78E-6	7.90	1.51E-6	8
129	–	1.90E-4	4.31E-8	7.95	4.53E-9	8
257	–	4.91E-5	1.73E-9	7.98	1.87E-11	8
			$\mathcal{P}_{16}$			
17	–	1.88E-2	6.31E-3	13.15	8.93E-3	14.26
33	–	2.11E-3	2.93E-5	15.43	2.91E-5	16
65	–	6.96E-4	8.12E-7	15.69	5.41E-8	16
129	–	2.25E-4	2.27E-8	15.84	1.97E-11	16
257	–	7.93E-5	9.21E-10	15.92	7.22E-15	16

Table 16: $L^2 - errors$ 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:

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

For the DBI and PPI algorithm, $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. 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 methods are dominated by the local errors of the intervals with C^0-continuity and low degree polynomial interpolants. Tables 17 and 18 show that the L^2-errors from the three methods have the same order, with DBI and PPI having slightly smaller errors than the other approaches. In the cases where the underlying function is C^0, the results from DBI and PPI are comparable to the other approaches. Furthermore, the results from DBI and 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	1.77E-2	2.04	1.73E-2	2.06
33	6.92E-3	6.11E-3	6.22E-3	1.93	6.21E-3	1.95
65	2.47E-3	2.69E-3	2.24E-3	1.89	2.24E-3	1.90
129	8.99E-4	7.71E-4	8.17E-4	1.88	8.16E-4	1.88
257	3.23E-4	2.77E-4	2.95E-4	1.87	2.94E-4	1.87
			$\mathcal{P}_8$			
17	–	1.91E-2	1.73E-2	3.19	1.69E-2	3.31
33	–	6.46E-3	6.20E-3	3.09	6.19E-3	3.16
65	–	2.24E-3	2.21E-3	3.04	2.20E-3	3.09
129	–	8.12E-4	7.98E-4	3.03	7.97E-4	3.04
257	–	2.92E-4	2.87E-4	3.01	2.87E-4	3.02
			$\mathcal{P}_{16}$			
17	–	2.19E-2	2.02E-2	4.58	2.34E-2	4.94
33	–	7.57E-3	6.14E-3	5.13	6.17E-3	5.35
65	–	2.68E-3	2.25E-3	5.25	2.26E-3	5.38
129	–	9.63E-4	8.07E-4	5.29	8.08E-4	5.35
257	–	3.45E-4	2.89E-4	5.29	2.89E-4	5.32

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. The points are uniformly distributed in each dimension.

N_i	PCHIP L^2-error	L^2-error	DBI L^2-error	avg. deg.	PPI 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	2.18E-2	2.04	2.17E-2	2.06
33	6.92E-3	5.76E-3	7.22E-3	1.93	7.18E-3	1.95
65	2.47E-3	2.11E-3	2.74E-3	1.89	2.72E-3	1.90
129	8.99E-4	8.08E-4	9.93E-4	1.88	9.87E-4	1.88
257	3.23E-4	2.92E-4	3.61E-4	1.87	3.59E-4	1.87
			$\mathcal{P}_8$			
17	–	4.06E-2	3.42E-2	3.19	3.19E-2	3.31
33	–	9.69E-3	8.68E-3	3.09	8.67E-3	3.16
65	–	2.46E-3	2.76E-3	3.04	2.82E-3	3.09
129	–	9.83E-4	1.09E-3	3.03	1.12E-3	3.04
257	–	3.63E-4	3.85E-4	3.01	3.99E-4	3.02
			$\mathcal{P}_{16}$			
17	–	4.36E-2	3.35E-2	4.58	2.81E-2	4.94
33	–	1.53E-2	1.06E-2	5.13	1.06E-2	5.35
65	–	4.31E-3	3.42E-3	5.25	3.46E-3	5.38
129	–	1.13E-3	9.84E-4	5.29	1.29E-3	5.35
257	–	4.38E-4	3.95E-4	5.29	5.07E-4	5.32

Table 18: $L^2 - errors$ 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 herein to study shape-preserving (monotonicity and convexity) splines [46].

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

46

For the DBI and PPI algorithm, $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. The function $f_9(x, y)$ is a C^0-continuous function. Tables 19 and 18 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 19 and 18 demonstrate that the DBI and PPI methods lead to smaller approximation errors than the PCHIP and SPS 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	–	–	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	1.20E-2	2.56	1.20E-2	2.66
33	6.77E-3	6.21E-3	4.25E-3	2.53	4.25E-3	2.58
65	2.39E-3	2.18E-3	1.50E-3	2.52	1.50E-3	2.54
129	8.47E-4	7.70E-4	5.30E-4	2.51	5.30E-4	2.52
257	3.01E-4	2.74E-4	1.88E-4	2.50	1.88E-4	2.51
			$\mathcal{P}_8$			
17	–	1.47E-2	1.27E-2	4.22	1.27E-2	4.72
33	–	4.57E-3	4.48E-3	4.37	4.48E-3	4.61
65	–	1.51E-4	1.58E-3	4.44	1.58E-3	4.56
129	–	5.16E-4	5.60E-4	4.47	5.60E-4	4.53
257	–	1.84E-4	1.98E-4	4.48	1.98E-4	4.51
			$\mathcal{P}_{16}$			
17	–	1.50E-2	2.22E-2	5.92	2.74E-2	7.11
33	–	4.03E-3	4.79E-3	8.05	4.79E-3	8.67
65	–	1.15E-3	1.69E-3	8.28	1.69E-3	8.58
129	–	3.50E-4	5.98E-4	8.39	5.98E-4	8.54
257	–	1.17E-4	2.12E-4	8.44	2.12E-4	8.52

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. 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	3.56E-02	1.05E-2	1.85E-2	2.52	1.46E-4	3.91
33	1.79E-03	3.61E-3	4.74E-3	2.61	4.77E-6	3.95
65	2.53E-04	1.26E-3	1.19E-3	2.62	1.43E-7	3.98
129	9.94E-05	4.44E-4	2.98E-4	2.62	9.16E-9	3.99
257	3.26E-05	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 20: $L^2 - errors$ 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:

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

For the DBI and PPI algorithm, $\epsilon_0 = 0.01$, $\epsilon_1 = 1$ and $st = 2$. The function $f_{10}(x, y)$ is challenging because of the large gradient at $y = -x$. Tables 21 and 22 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 and PPI methods improves. In this case, the L^2-error is dominated by the local error of the region with the steep gradient. The errors for the DBI and PPI methods are similar because the stencils used for both methods are the same in the region with the large gradient. Overall, the results from the Tables 21 and 22 show that the DBI and PPI methods lead 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	9.45E-3	2.75	9.44E-3	3.23
33	1.26E-3	2.43E-4	1.33E-3	3.53	1.29E-3	3.65
65	1.44E-4	4.92E-5	9.29E-5	3.82	9.29E-5	3.82
129	1.63E-5	1.20E-5	3.67E-6	3.81	3.67E-6	3.81
257	1.94E-6	3.05E-6	1.21E-7	3.79	1.21E-7	3.79
			$\mathcal{P}_8$			
17	–	1.80E-1	8.05E-3	3.15	8.67E-3	5
33	–	1.22E-1	1.03E-3	5.78	9.05E-4	6.55
65	–	8.48E-2	4.83E-5	7.54	4.99E-5	7.57
129	–	1.04E-1	2.57E-7	7.53	2.57E-7	7.55
257	–	8.02E-2	5.27E-10	7.49	5.27E-10	7.52
			$\mathcal{P}_{16}$			
17	–	4.72E-3	7.39E-3	3.57	1.86E-2	7.45
33	–	1.22E-3	1.02E-3	6.71	2.33E-3	10.23
65	–	3.11E-4	2.12E-4	14.76	2.41E-4	14.94
129	–	7.80E-5	1.03E-6	14.93	1.03E-6	15.05
257	–	1.95E-5	4.41E-11	14.83	4.41E-11	14.97

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. 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	1.32E-2	2.79E-3	1.12E-2	2.75	1.11E-2	3.23
33	2.75E-3	3.49E-4	1.73E-3	3.53	1.68E-3	3.65
65	3.57E-4	6.87E-5	1.28E-4	3.82	1.28E-4	3.82
129	4.09E-5	1.64E-5	5.47E-6	3.81	5.47E-6	3.81
257	5.04E-6	4.12E-6	1.77E-7	3.79	1.77E-7	3.79
			$\mathcal{P}_8$			
17	–	5.82E-3	1.22E-2	3.15	1.20E-2	5.00
33	–	1.26E-3	1.82E-3	5.78	1.67E-3	6.55
65	–	3.00E-4	4.98E-5	7.54	4.98E-5	7.57
129	–	7.58E-5	4.03E-7	7.53	4.03E-7	7.55
257	–	1.90E-5	1.21E-9	7.49	1.21E-9	7.52
			$\mathcal{P}_{16}$			
17	–	8.05E-3	1.34E-2	3.57	1.31E-2	7.45
33	–	2.06E-3	2.04E-3	6.71	1.92E-3	10.23
65	–	5.14E-4	3.81E-5	14.76	3.84E-5	14.94
129	–	1.26E-4	6.20E-8	14.93	6.20E-8	15.05
257	–	3.18E-5	6.20E-12	14.83	6.20E-12	14.97

Table 22: $L^2 - errors$ 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 approaches 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 DBI and PPI methods perform well

and are suited to the C^0 continuity of the spectral element methods in NEPTUNE. The experiments show that the DBI and PPI methods are suitable approaches for interpolating smooth functions and C^0 continuous functions while enforcing positivity. In detail the summary is that:

- The results in Section 3 Examples I, II, and III show that the improved DBI and new PPI approaches preserve positivity exactly as the proofs in [9] indicate;

- The results in Section 4 and Sections 5.1, 5.2, and 5.3 show that the DBI and PPI approaches give much higher levels 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 Examples V and VI, better approximations are obtained using $st = 1$ and $st = 3$, respectively, for the PPI and DBI methods. These results show that in some cases the appropriate choice of the parameter st can further improve the approximation in addition to preserving data-boundedness or positivity. Additional studies evaluating the different choices of ϵ_0, ϵ_1, and st are presented in [8].

- In the cases when steep gradients or discontinuities force the use of low-order approximations, the DBI and PPI methods compete 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.

Acknowledgements

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 M. Kirby, and Martin Berzins. HPPIS: A high-order positivity-preserving mapping software for structured meshes. *Manuscript in preparation*, 2022.

[9] Timbwaoga AJ Ouermi, Robert M Kirby, and Martin Berzins. Eno-based high-order data-bounded and constrained positivity-preserving interpolation. *Accepted by Numerical Algorithms*, June 2022. https://doi.org/10.48550/arXiv.2204.06168.

[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] Chi-Wang Shu. High-order finite difference and finite volume weno schemes and discontinuous galerkin methods for cfd. *International Journal of Computational Fluid Dynamics*, 17(2):107–118, 2003.

[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.