Noname manuscript No.
(will be inserted by the editor)

ENO-Based High-Order Data-Bounded and Constrained Positivity-Preserving Interpolation

T. A. J. Ouermi
Robert M. Kirby
Martin Berzins

Received: date / Accepted: date

Abstract A number of key scientific computing applications that are based upon tensor-product grid constructions, such as numerical weather prediction (NWP) and combustion simulations, require *property-preserving* interpolation. Essentially Non-Oscillatory (ENO) interpolation is a classic example of such interpolation schemes. In the aforementioned application areas, property preservation often manifests itself as a requirement for either data boundedness or positivity preservation. For example, in NWP, one may have to interpolate between the grid on which the dynamics is calculated to a grid on which the physics is calculated (and back). Interpolating density or other key physical quantities without accounting for property preservation may lead to negative values that are nonphysical and result in inaccurate representations and/or interpretations of the physical data.

Property-preserving interpolation is straightforward when used in the context of low-order numerical simulation methods. High-order property-preserving

T. A. J. Ouermi
University of Utah School of Computing
U of U Scientific Computing and Imaging Institute
Salt Lake City, Utah, USA
E-mail: touermi@cs.utah.edu

Robert M. Kirby
University of Utah School of Computing
U of U Scientific Computing and Imaging Institute
Salt Lake City, Utah, USA
E-mail: kiby@cs.utah.edu

Martin Berzins
University of Utah School of Computing
U of U Scientific Computing and Imaging Institute
Salt Lake City, Utah, USA
E-mail: mb@sci.utah.edu

interpolation is, however, nontrivial, especially in the case where the interpolation points are not equispaced. In this paper, we demonstrate that it is possible to construct high-order interpolation methods that ensure either data boundedness or constrained positivity preservation. A novel feature of the algorithm is that the positivity-preserving interpolant is constrained; that is, the amount by which it exceeds the data values may be strictly controlled. The algorithm we have developed comes with theoretical estimates that provide sufficient conditions for data boundedness and constrained positivity preservation. We demonstrate the application of our algorithm on a collection of 1D and 2D numerical examples, and show that in all cases property preservation is respected.

Keywords Data-Bounded Interpolation · Positivity-Preserving Interpolation · Newton Polynomial · Essentially Non-Oscillatory Methods · Property-Preserving

Mathematics Subject Classification (2020) MSC 65D05 · MSC 65D15

1 Introduction

A number of key scientific computing applications that are based upon high-order methods over tensor-product grid constructions, such as numerical weather prediction (NWP) and combustion simulations, require *property-preserving* interpolation. In the aforementioned application areas, property preservation often manifests itself as a requirement for either data boundedness or positivity preservation. The particular application motivating this work is the Navy Environmental Prediction System Using the NUMA Core (NEPTUNE). NEPTUNE is a next-generation global NWP system being developed at the Naval Research Laboratory (NRL) and the Naval Postgraduate School (NPS) [32]. NEPTUNE makes use of the Nonhydrostatic Unified Model of the Atmosphere (NUMA) [10] three-dimensional spectral element dynamical core, but currently uses physics routines that were developed assuming uniform grid spacing on the elements. At least two options are available for combining these two NWP building blocks: either (1) evaluate the physics routines at the (nonuniformly-spaced) quadrature points on the spectral element with acknowledgment that a modeling 'crime' has been committed or (2) interpolate between the grid (quadrature points) on which the dynamics is calculated to a grid on which the physics is calculated (and back), and hence incur an interpolation error. Since there is a long-standing history of using the validated physics routines designed for use on uniformly-spaced grids, there is a strong incentive to apply the second option. However, interpolating density or other key physical quantities without accounting for property preservation may lead to negative values that are nonphysical and result in inaccurate representations and/or interpretations of the physical data. For example, Skamrock et al. [28] demonstrated that not preserving positivity may lead to a positive bias in a predicted physical quantity of interest (e.g., prediction of moisture). The

second option mentioned above of moving information from nonuniform to uniform and back via ENO-type interpolation schemes, explored in [21] in the context of high-order methods for numerical weather prediction, is the main motivation for this work.

Property-preserving interpolation is straightforward when used in the context of low-order numerical simulation methods. High-order property-preserving interpolation is, however, nontrivial, especially when the interpolation points are not uniformly-spaced. In this paper, we demonstrate that it is possible to adaptively construct high-order interpolation methods over unevenly-spaced tensor product grids in a way that ensures either data boundedness or positivity preservation (within user-supplied bounds). The algorithm we have developed comes with theoretical estimates, presented herein, that provide sufficient conditions for data boundedness and positivity preservation.

1.1 Previous Work

In this section, we provide an overview of various numerical approaches to data bounded and positivity preservation. This overview is not meant to be exhaustive, but instead to summarize the various ways by which researchers have attempted to tackle this challenging problem.

Introduced by Harten et al. [15], Essentially Non-Oscillatory (ENO) schemes were developed to solve problems with sharp gradients and discontinuities while achieving high-order accuracy in both smooth and non-smooth regions. As with many finite-difference based methods, the backbone of these schemes is interpolation methods. In the context of this paper, which is to propose ENO-like interpolation schemes that are property preserving, we briefly review ENO methods.

In the context of finite volume schemes, Fjordholm et al. [8] demonstrated that ENO schemes are stable, in the sense that the jump of the reconstructed value at each cell interface has the same sign as the jump in the underlying cell average. Building on the work [29] and [8], Fjordholm et al. [7] developed a high-order entropy stable ENO scheme for conservation laws. This approach consists of using entropy conservative flux based on [29], adding a numerical diffusion to obtain a stable scheme, and obtaining the high-order accuracy via ENO reconstruction.

Harten [12,13] developed an ENO scheme for subcell resolution in the cases where a discontinuity lies inside a given cell. Weighted Essentially Non-Oscillatory (WENO) schemes were later proposed by Liu et al. [18] to address some of the shortcomings of the ENO schemes. Shu [27] provided a comprehensive overview of different applications and problems in which ENO and WENO schemes are used. Shen et al. [25] proposed an adaptive mesh refinement method (AMR) based on WENO schemes for hyperbolic conservation laws. In this approach, high-order WENO interpolation is used for the prolongation. A generalization of the AMR-WENO in [33] was used to solve a multi-dimension detonation problem.

Another body of literature sometimes considered around property-preserving methods is computer-aided design and visualization. Although different from the finite difference (stencil) methods that we seek, we briefly review this literature. In that literature, "shape" preservation is often used to describe the preservation of properties like monotonicity and convexity, and may include positivity and data boundedness [4] and [3]. We only briefly review this literature as the additional smoothness constraints at the stencil points enforced by these methods introduce a level of complexity not needed for our application domain. Our focus is finite difference ENO-type schemes. Perhaps the most widely used approach for preserving monotonicity in many applications is PCHIP by Fritch and Carlson [9], 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. Lux et al. [19] proposed a monotone quintic spline (MQS) algorithm that relies on the results of Heß and Schmidt [23] and Ulrich and Watson [31]. This method is dependent on the value of the first and second derivatives at the node. The algorithm uses the sufficient conditions from [23] to check for monotonicity. When the conditions are not met, the method in [31] is used to modify the values of the first and second derivatives to ensure monotonicity. The work of Dougherty et al. [5] extends these ideas to preserving convexity and concavity and also to quintic splines.

A second area in which one often finds the development of methods for property preservation is numerical methods for partial differential equations (PDEs). Various methods have been developed to enable, for example, positivity-preserving approximations. To preserve positivity in discontinuous Galerkin (dG) schemes, Zhang et al. [36,39,37] introduced a linear rescaling of polynomials that ensures that the evaluation of the polynomial at the quadrature points remains positive. In addition, the linear rescaling of the polynomial conserves mass. Light et al. [17] developed a similar approach with a more involved linear polynomial rescaling that preserves positivity at the quadrature nodes and conserves mass. The polynomial rescaling does not address the case of interpolating between different meshes, which is the primary focus of this work. Harten et al. [15] developed an Essentially Non-Oscillatory (ENO) piecewise polynomial reconstruction that enables interpolation between different meshes. The ENO method adaptively chooses stencil points for the interpolation and helps remove Gibbs-like effects but does not guarantee positivity. As previously mentioned, extensions of these ideas to a Weighted ENO (WENO) combination of these schemes have been proposed by Zhang et al. [38] and others. Finally, Zala et al. [34,35] developed a nonlinear filtering operator for property-preservation by casting it as an optimization problem in which the desired "structures" (properties) are encoded as constraints.

The data-bounded interpolation (DBI) method of Berzins [1] builds on three ideas from these ENO and WENO algorithms in the area of the numerical solution of advection equations: adaptively selecting stencils as in the ENO methods to reduce oscillations [15]; altering the polynomial approximation so

that any discontinuities in higher derivatives are removed [14]; and altering the polynomial degree and/or terms so that the ratio of successive divided differences in the series is strictly limited to enforce the boundedness of the interpolation [1]. The work in [2] extends the earlier proof to 1D unevenly-spaced points where, in addition to the interval points, all the remaining points used to build the interpolant are to the right or left of the interval of interest. In addition, the work in [2] recognizes that switching off data boundedness when extrema are present is important for maintaining accuracy. Positivity is important in interpolation cases in which extrema lie between data points and where the data-bounded interpolant will "clip" the function, resulting in a loss of accuracy. A novel feature of the approach addresses the fact that preserving positivity alone may still produce undesirable oscillations that lead to an inaccurate representation and/or interpretation of the underlying data. These oscillations are removed here by imposing strict user-supplied bounds on the positive interpolants as a way of limiting oscillations and correspondingly improving accuracy.

This work extends the ideas in [1] by addressing data boundedness and positivity (within user-supplied bounds) in the same framework and by allowing meshes of unevenly-spaced points. The DBI method presented in this paper introduces more relaxed conditions for data-boundedness which give greater accuracy than the conditions used in [1]. Thus, these new proofs provide the previously missing theoretical underpinning for complex interpolation cases such as those like the NWP case described above. The new approach used here both generalizes the DBI method to unevenly-spaced structured meshes and extends the approach to preserve positivity (positivity-preserving interpolation (PPI)) rather than the more restrictive data-bounded approach in [1] and [2].

1.2 Outline

The paper proceeds as follows. In Section 2, we provide a review of Newton interpolation with particular emphasis on the properties on Newton polynomials required in this work. In Section 3, we present our *first major contribution*: theoretical guarantees for adaptive high-order data bounded polynomial interpolation on nonuniformly-spaced points. In Section 4, we extend the ideas presented in Section 3 to positivity. We present our *second major contribution*: theoretical guarantees for adaptive high-order positivity-preserving polynomial interpolation on nonuniformly-spaced points. In Section 4, we also address the case of hidden extrema with the new limiting approach and provided an algorithm for the DBI and PPI methods. In Section 5, we provide 1D and 2D results demonstrating the properties of our proposed algorithms. We summarize and conclude the paper in Section 6.

2 Background

The approach introduced in this work relies on the Newton polynomial [30,16] representation to build interpolants that are positive or bounded by the data values. The ability to adaptively select the divided differences or the stencil as in ENO methods [15] is central to the data-bounded and positivity-preserving interpolation approaches presented in this paper.

Consider a 1D mesh defined as follows:

$$\mathcal{M} = \{x_{i-J}, \cdots, x_i, x_{i+1}, \cdots, x_{i+L}\}, \tag{1}$$

where $x_{i-J} < \cdots < x_i < x_{i+1} < \cdots < x_{i+L}$, and $\{u_{i-J}, \cdots, u_{i+L}\}$ is the set of data values associated with the mesh points. In the definition of the mesh $\mathcal{M}$, the subscripts J, L, $i, \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$, and x_k, $u_k \in \mathbb{R}$ for $i - J \leq k \leq i + L$. For the given mesh $\mathcal{M}$, the Newton divided differences are recursively defined as follows:

$$\begin{cases} U[x_i] = u_i \\ U[x_i, \cdots, x_{i+j}] = \frac{U[x_{i+1}, \cdots, x_{i+j}] - U[x_i, \cdots, x_{i+j-1}]}{x_{i+j} - x_i}. \end{cases} \tag{2}$$

The ENO procedure starts by setting the initial stencil $\mathcal{V}_0$:

$$\mathcal{V}_0 = \{x_i, x_{i+1}\} = \{x_0^l, x_0^r\}. \tag{3}$$

The stencil $\mathcal{V}_0$ is expanded by successively appending a point to right or left of $\mathcal{V}_j$ to form $\mathcal{V}_{j+1}$. The point appended is selected by picking the smallest divided difference at each step.

Given $\mathcal{V}_j$, let x_j^l and x_j^r be the leftmost and rightmost stencil points, respectively. In addition, let x_p and x_q be the stencil points immediately to the left and right of $\mathcal{V}_j$. The stencil is expanded from $\mathcal{V}_j$ to $\mathcal{V}_{j+1}$ based on the following rules:

- if $|U[x_p, x_j^l, \cdots, x_j^r]| < |U[x_j^l, \cdots, x_j^r, x_q]|$ then
  $\mathcal{V}_{j+1} = \{x_p, \mathcal{V}_j\}$ with $x_{j+1}^l = x_p$ and $x_{j+1}^r = x_j^r$.
- otherwise
  $\mathcal{V}_{j+1} = \{\mathcal{V}_j, x_q\}$ with $x_{j+1}^l = x_j^l$ and $x_{j+1}^r = x_q$.

Let

$$I_i = [x_i, x_{i+1}], \quad \text{for } 0 \leq i \leq n-1. \tag{4}$$

Once the final stencil $\mathcal{V}_{n-1}$ is obtained, the interpolant of degree n defined on I_i can be written as

$$\begin{aligned} U_n(x) = \quad & u_i + U[x_0^l, x_0^r]\pi_{0,i}(x) + U[x_1^l, \cdots, x_1^r]\pi_{1,i}(x) + \cdots \\ & \cdots + U[x_{n-1}^l, \cdots, x_{n-1}^r]\pi_{n-1,i}(x), \end{aligned} \tag{5}$$

where $\pi_{0,i}(x) = (x - x_i), \pi_{1,i}(x) = (x - x_i)(x - x_1^e), \cdots$ are the Newton basis functions. x_j^e is the point added to expand the stencil $\mathcal{V}_{j-2}$ to $\mathcal{V}_{j-1}$ and can

be explicitly expressed as

$$\begin{cases} x_0^e = x_i, \\ x_1^e = x_{i+1}, \\ x_j^e = \mathcal{V}_{j-1} \setminus \mathcal{V}_{j-2}, \quad 2 \leq j \leq n-1. \end{cases} \tag{6}$$

The first step in developing the DBI and PPI methods consists of reorganizing the terms in the polynomial $U_n(x)$ defined in Equation (5) to expose the features used to enforce data boundedness and positivity. The reorganization begins by defining λ_j as follows:

$$\lambda_j = \begin{cases} 1, \quad j = 0 \\ \frac{U[x_j^l, \cdots, x_j^r]}{U[x_{j-1}^l, \cdots, x_{j-1}^r]}(x_j^r - x_j^l), \quad 1 \leq j \leq n-1. \end{cases} \tag{7}$$

Expressing $U_n(x)$ in terms of λ_j, for $j > 0$ gives

$$\begin{aligned} U_n(x) = u_i + (u_{i+1} - u_i)\frac{x - x_0^e}{x_0^r - x_0^l}\Bigg(1 + \frac{(x - x_1^e)}{(x_1^r - x_1^l)}\lambda_1 \\ \Bigg(1 + \frac{(x - x_2^e)}{(x_2^r - x_2^l)}\lambda_2\Bigg(\cdots \lambda_{n-2}\Bigg(1 + \frac{(x - x_{n-1}^e)}{(x_{n-1}^r - x_{n-1}^l)}\lambda_{n-1}\Bigg)\cdots\Bigg). \end{aligned} \tag{8}$$

For $x \in I_i$, s, t_j, and d_j are defined as follows:

$$0 \leq s = \frac{x - x_i}{x_{i+1} - x_i} = \frac{x - x_0^e}{x_0^r - x_0^l} \leq 1, \tag{9}$$

$$t_j = -\frac{x_i - x_j^e}{x_0^r - x_0^l}, \text{ and} \tag{10}$$

$$0 \leq d_j = \frac{x_j^r - x_j^l}{x_0^r - x_0^l}. \tag{11}$$

s and d_j are defined such that $s \in [0, 1]$ and $d_j \geq 0$. Expressing $\frac{x - x_j^e}{x_j^r - x_j^l}$ in terms of s, t_j, and d_j gives

$$\frac{x - x_j^e}{x_j^r - x_j^l} = \frac{\frac{x - x_i}{x_0^r - x_0^l} + \frac{x_i - x_j^e}{x_0^r - x_0^l}}{\frac{x_j^r - x_j^l}{x_0^r - x_0^l}} = \frac{s - t_j}{d_j}. \tag{12}$$

Using the results from Equation (12), the polynomial $U_n(x)$ as expressed in Equation (8) can be written as

$$U_n(x) = u_i + (u_{i+1} - u_i)S_n(x) \tag{13}$$

with $S_n(x)$ defined as

$$S_n(x) = s\Bigg(1 + \frac{(s-1)}{d_1}\lambda_1\Bigg(1 + \frac{(s - t_2)}{d_2}\lambda_2\Bigg(\cdots\Bigg(1 + \frac{(s - t_{n-1})}{d_{n-1}}\lambda_{n-1}\Bigg)\cdots\Bigg). \tag{14}$$

For future use below, $S_n(x)$ can be compactly represented by introducing δ_j defined as

$$\begin{cases} \delta_n = 1 \\ \delta_j = 1 + \frac{s-t_j}{d_j}\lambda_j\delta_{j+1} \quad 2 \leq j \leq n-1 \\ \delta_1 = s + \frac{s(s-1)}{d_1}\delta_2 = S_n(x). \end{cases} \tag{15}$$

Together, $U_n(x)$ and $S_n(x)$ in Equations (13) and (14) are reorganizations needed to construct the DBI and PPI algorithm. The general approach is to first bound the quadratic term in $S_n(x)$ and then to increase the order to cubic, quartic, and higher order polynomials. This iterative procedure is used to define computational bounds on the values of $\bar{\lambda}_j = \prod_{k=0}^{j} \lambda_k$. $\bar{\lambda}_j$ can be explicitly written as

$$\bar{\lambda}_j = \lambda_j\bar{\lambda}_{j-1} = \prod_{k=1}^{j} \lambda_k = \begin{cases} 1 \quad j = 0, \\ \frac{U[x_j^l,\cdots,x_j^r]}{U[x_0^l,x_0^r]} \prod_{k=1}^{j}(x_k^r - x_k^l), \quad 1 \leq j \leq n-1. \end{cases} \tag{16}$$

3 Data-Bounded Interpolation

The DBI method builds on three ideas from algorithms in the area of the numerical solution of advection equations: adaptively selecting stencils as in the ENO methods to reduce oscillations [15]; altering the polynomial approximation so that any discontinuities in higher derivatives are removed [14]; and altering the polynomial degree and/or terms so that the ratio of successive divided differences in the series is strictly limited to enforce the boundedness of the interpolation [1]. In the DBI method introduced here, more relaxed bounds on $\bar{\lambda}_j$ defined in Equation (16) are derived which gives greater accuracy than those in [1]. The work in [1] requires that the absolute values of $\bar{\lambda}_j$ decrease as more terms are added ($|\bar{\lambda}_j| > |\bar{\lambda}_{j+1}|$) and $|\bar{\lambda}_j| < 1$ which are more restrictive than the bounds in Equation (27). For a given set of mesh points and the data values associated with those mesh points, we approximate the data with a $\mathbf{C}^0$ continuous function that is built by fitting a polynomial in each subinterval I_i. The fitted polynomial is constructed in such a way that it is bounded by u_i and u_{i+1}. Given that this work concerns itself with locally fitting a polynomial in the interval I_i, let us assume, for the remaining parts of this paper, that $x \in I_i$ and that building the interpolant always starts with the stencil $\mathcal{V}_0 = \{x_i, x_{i+1}\}$.

Let $U^l(x)$ be the limited polynomial defined as in Equation (13) and bounded by u_i and u_{i+1}. For the polynomial $U^l(x)$ to be bounded by u_i and u_{i+1}, it follows that for $x \in I_i$

$$0 \leq S_n(x) \leq 1, \tag{17}$$

with $S_n(x)$ defined in Equation (14). The reconstruction procedure begins by considering the linear and quadratic terms from $S_n(x)$ in Equation (14), and

imposing the following bounds:

$$0 \leq s\big(1 + \frac{s-1}{d_1}\bar{\lambda}_1\big) \leq 1. \tag{18}$$

As $s \in [0,1]$ and isolating $\bar{\lambda}_1$ in Equation (18) gives

$$-\frac{d_1}{s} \leq \bar{\lambda}_1 \leq \frac{d_1}{1-s}, \text{ and} \tag{19}$$

$$-d_1 \leq \bar{\lambda}_1 \leq d_1. \tag{20}$$

The bounds from Equation (20) are extended to bound the cubic form by requiring that what multiplies $\bar{\lambda}_1$ must fit into the inequality in Equation (20). Thus, for the cubic case Equation (20) becomes

$$-d_1 \leq \bar{\lambda}_1\big(1 + \frac{(s-t_2)}{d_2}\lambda_2\big) \leq d_1. \tag{21}$$

Subtracting $\bar{\lambda}_1$ from this inequality gives

$$-d_1 - \bar{\lambda}_1 \leq \frac{(s-t_2)}{d_2}\bar{\lambda}_2 \leq d_1 - \bar{\lambda}_1. \tag{22}$$

In the case when t_2 is negative, $s - t_2$ has a maximum value at $s = 1$ and a minimum value at $s = 0$. $\bar{\lambda}_2$ is then bounded by

$$\frac{d_2}{(1-t_2)}(-d_1 - \bar{\lambda}_1) \leq \bar{\lambda}_2 \leq (d_1 - \bar{\lambda}_1)\frac{d_2}{(1-t_2)}. \tag{23}$$

When t_2 positive, $\frac{1}{1-t_2}$ is substituted by $\frac{1}{-t_2}$ and the inequalities $\leq$ with $\geq$ and vice versa are swapped. In the quartic case, we require that

$$\frac{d_2}{1-t_2}(-d_1 - \bar{\lambda}_1) \leq \bar{\lambda}_2\left(1 + \frac{(s-t_3)}{d_3}\lambda_3\right) \leq \frac{d_2}{1-t_2}(d_1 - \bar{\lambda}_1). \tag{24}$$

If we assume that t_3 is negative

$$\frac{d_3}{1-t_3}\left(\frac{d_2}{1-t_2}(-d_1 - \bar{\lambda}_1) - \bar{\lambda}_2\right) \leq \bar{\lambda}_3 \leq \frac{d_3}{1-t_3}\left(\frac{d_2}{1-t_2}(d_1 - \bar{\lambda}_1) - \bar{\lambda}_2\right). \tag{25}$$

This reconstruction procedure can be continued to higher orders provided that care is taken to correctly manage the impact of the signs of t_j. For the boundary and nearby boundary intervals, fewer choices are available, and the final stencil is biased towards the interior of the domain because there are no points to choose from beyond the boundaries. In the process of constructing $\mathcal{V}_{n-1}$, when the left or right boundary are reached, the remaining mesh points are obtained from the side that is towards the interior of the domain.

For a more formal and complete expression of this recursive procedure, the bounds on $\bar{\lambda}_j$ can be defined as follows:

$$B_j^- = \begin{cases} -d_1 & j = 0 \\ (B_{j-1}^- - \bar{\lambda}_{j-1})\frac{d_j}{1-t_j}, & t_j \in (-\infty, 0] \quad j > 1 \\ (B_{j-1}^+ - \bar{\lambda}_{j-1})\frac{d_j}{-t_j}, & t_j \in (0, +\infty) \quad j > 1, \end{cases} \tag{26a}$$

and

$$B_j^+ = \begin{cases} d_1, & j = 1 \\ (B_{j-1}^+ - \bar{\lambda}_{j-1})\frac{d_j}{1-t_j}, & t_j \in (-\infty, 0] \quad j > 1 \\ (B_{j-1}^- - \bar{\lambda}_{j-1})\frac{d_j}{-t_j}, & t_j \in (0, +\infty) \quad j > 1. \end{cases} \tag{26b}$$

The sign of t_j is incorporated into the definitions of B_j^- and B_j^+ in Equations (26a) and (26b), respectively. The sufficient conditions for data boundedness such as Equations (20), (23) and (25) can now be written as

$$B_j^- \leq \bar{\lambda}_j \leq B_j^+, \text{ for } j \geq 0. \tag{27}$$

Lemma 1 *Let us assume that for $x \in I_i$, B_j^- and B_j^+ are defined as in Equations (26b) and (26a), respectively. In addition, let δ_j be defined as in Equation (15). If for $x \in I_i$, B_j^- is negative, B_j^+ is positive, and $B_j^- \leq \bar{\lambda}_j \delta_{j+1} \leq B_j^+$, then*

$$B_{j-1}^- \leq \bar{\lambda}_{j-1}\delta_j \leq B_{j-1}^+.$$

Proof The proof is split into two cases that take into consideration the different possible values of t_j, and in each case we consider the left and right side of the inequality separately.

(I) $t_j \in (-\infty, 0]$

Let us start with the left side of the inequality (i.e., $B_{j-1}^- \leq \bar{\lambda}_{j-1}\delta_j$). Noting that $\frac{1-t_j}{s-t_j} \geq 1$ for $s \in [0, 1]$, and using $B_j^- \leq 0$ and $B_j^- \leq \bar{\lambda}_j \delta_{j+1}$, we have

$$\begin{aligned} (B_{j-1}^- - \bar{\lambda}_{j-1})\frac{d_j}{s-t_j} =& \frac{1-t_j}{s-t_j} B_j^- \\ \leq& B_j^- \\ \leq& \bar{\lambda}_j \delta_{j+1}. \end{aligned} \tag{28}$$

Isolating B_{j-1}^- in Equation (28) and using Equations (15) and (16) leads to

$$\begin{aligned} B_{j-1}^- \leq& \bar{\lambda}_{j-1} + \frac{s-t_j}{d_j}\bar{\lambda}_j \delta_{j+1} \\ \leq& \bar{\lambda}_{j-1}\left(1 + \frac{s-t_j}{d_j}\lambda_j \delta_{j+1}\right) \\ =& \bar{\lambda}_{j-1}\delta_j. \end{aligned} \tag{29}$$

Now, let us focus on the right side of the inequality (i.e., $B_{j-1}^{+} \geq \bar{\lambda}_{j-1}\delta_j$) Again, observing that $\frac{1-t_j}{s-t_j} \geq 1$ for $s \in [0,1]$ and using $B_j^{+} \geq 0$ and $B_j^{+} \geq \bar{\lambda}_j\delta_{j+1}$ yields

$$\begin{aligned}(B_{j-1}^{+} - \bar{\lambda}_{j-1})\frac{d_j}{s-t_j} =& \frac{1-t_j}{s-t_j}B_j^{+} \\ \geq& B_j^{+} \\ \geq& \bar{\lambda}_j\delta_{j+1}.\end{aligned} \tag{30}$$

Isolating B_{j-1}^{+} in Equation (30) yields

$$\begin{aligned}B_{j-1}^{+} \geq& \bar{\lambda}_{j-1} + \frac{s-t_j}{d_j}\bar{\lambda}_j\delta_{j+1} \\ \geq& \bar{\lambda}_{j-1}\left(1 + \frac{s-t_j}{d_j}\lambda_j\delta_{j+1}\right) \\ =& \bar{\lambda}_{j-1}\delta_j.\end{aligned} \tag{31}$$

(II) $t_j \in (0, +\infty)$

Let us consider the left side of the inequality (i.e., $B_{j-1}^{-} \leq \bar{\lambda}_{j-1}\delta_j$). Multiplying B_j^{-} by $\frac{-t_j}{s-t_j}$ yields

$$(B_{j-1}^{+} - \bar{\lambda}_{j-1})\frac{d_j}{s-t_j} = \frac{-t_j}{s-t_j}B_j^{-}. \tag{32}$$

Given that $B_j^{-} \leq 0$ and $B_j^{-} \leq \bar{\lambda}_j\delta_{j+1}$, and noting that $\frac{-t_j}{s-t_j} \geq 1$ for $s \in [0,1]$, the right side of Equation (32) can be bounded by B_j^{-} to give

$$\begin{aligned}(B_{j-1}^{+} - \bar{\lambda}_{j-1})\frac{d_j}{s-t_j} \leq& B_j^{-} \\ \leq& \bar{\lambda}_j\delta_{j+1}.\end{aligned} \tag{33}$$

Isolating B_{j-1}^{+} in Equation (33) leads to

$$\begin{aligned}B_{j-1}^{+} \geq& \bar{\lambda}_{j-1} + \frac{s-t_j}{d_j}\bar{\lambda}_j\delta_{j+1} \\ \geq& \bar{\lambda}_{j-1}\left(1 + \frac{s-t_j}{d_j}\lambda_j\delta_{j+1}\right) \\ =& \bar{\lambda}_{j-1}\delta_j.\end{aligned} \tag{34}$$

For the right side of the inequality (i.e. $B_{j-1}^{-} \leq \bar{\lambda}_{j-1}\delta_j$), $\frac{-t_j}{s-t_j} \geq 1$ for $s \in [0,1]$, and using $B_j^{-} \leq 0$ and $B_j^{-} \leq \bar{\lambda}_j\delta_{j+1}$ yields

$$\begin{aligned}(B_{j-1}^{-} - \bar{\lambda}_{j-1})\frac{d_j}{s-t_j} =& \frac{-t_j}{s-t_j}B_j^{+} \\ \geq& B_j^{-} \\ \geq& \bar{\lambda}_j\delta_{j+1}.\end{aligned} \tag{35}$$

Isolating B_{j-1}^{-} in Equation (35) yields

$$\begin{aligned} B_{j-1}^{-} \leq & \bar{\lambda}_{j-1} + \frac{s-t_j}{d_j}\bar{\lambda}_j\delta_{j+1} \\ \leq & \bar{\lambda}_{j-1}\left(1 + \frac{s-t_j}{d_j}\lambda_j\delta_{j+1}\right) \\ = & \bar{\lambda}_{j-1}\delta_j. \end{aligned} \tag{36}$$

The results from Equations (29), (31), (29), and (31) can be summarized as

$$B_{j-1}^{-} \leq \bar{\lambda}_{j-1}\delta_j \leq B_{j-1}^{+}.$$

Theorem 1 *Assuming that for $x \in I_i$, the polynomial $S_n(x)$ of degree n is built starting from the stencil $\mathcal{V}_0 = \{x_i, x_{i+1}\}$, and then by successively appending mesh points from the left and/or right of the interval I_i to obtain the final stencil $\mathcal{V}_{n-1}$. The construction of $\mathcal{V}_{n-1}$ does not require the points to be added in a symmetric fashion alternating from left to right. If for $x \in I_i$, B_j^{-} defined in Equation (26a) is negative, B_j^{+} defined in Equation (26b) is positive, and $B_j^{-} \leq \bar{\lambda}_j \leq B_j^{+}$ then for $x \in I_i$*

$$0 \leq S_N(x) \leq 1.$$

Proof This proof builds on the results from Lemma 1 and starts by using $B_j^{-} \leq \bar{\lambda}_j \leq B_j^{+}$ to bound $\bar{\lambda}_{n-1}$ as follows:

$$B_{n-1}^{-} \leq \bar{\lambda}_{n-1} \leq B_{n-1}^{+}. \tag{37}$$

By Lemma 1, Equation (37) then leads to

$$B_{n-2}^{-} \leq \bar{\lambda}_{n-2}\delta_{n-1} \leq B_{n-2}^{+}. \tag{38}$$

Successively, using the results from Lemma 1 to bound $\bar{\lambda}_{n-2}\delta_{n-1}$, $\bar{\lambda}_{n-3}\delta_{n-2}$, $\cdots$, $\bar{\lambda}_1\delta_2$, yields

$$B_1^{-} \leq \bar{\lambda}_1\delta_2 \leq B_1^{+}, \tag{39}$$

where δ_j is defined in Equation (15). The results from Equation (39) may now be used to derive the target bounds (i.e., $0 \leq S_N(x) \leq 1$). Considering the left side of Equation (39) (i.e., $B_1^{-} \leq \bar{\lambda}_1\delta_2$), and noting that $\frac{(s-1)}{s(s-1)} \geq 1$, gives

$$\begin{aligned} -\frac{(s-1)}{s(s-1)}d_1 = & B_1^{-}\frac{(s-1)}{s(s-1)} \\ \leq & B_1^{-} \\ \leq & \bar{\lambda}_1\delta_2. \end{aligned} \tag{40}$$

Isolating δ_1 from Equation (40) gives

$$1 \geq s + \frac{s(1-s)}{d_1}\bar{\lambda}_1\delta_2 = \delta_1 = S_n(x). \tag{41}$$

Considering the right side of Equation (39) (i.e. $B_1^+ \geq \bar{\lambda}_1 \delta_2$), and noting that $\frac{(-s)}{s(s-1)} \geq 1$, gives

$$\begin{aligned} \frac{(-s)}{s(s-1)} d_1 =& B_1^+ \frac{(-s)}{s(s-1)} \\ \geq& B_1^+ \\ \geq& \bar{\lambda}_1 \delta_2. \end{aligned} \tag{42}$$

Isolating δ_1 from Equation (42) gives

$$0 \leq s + \frac{s(1-s)}{d_1} \bar{\lambda}_1 \delta_2 = \delta_1 = S_n(x). \tag{43}$$

The proof concludes by combining the results from Equations (41) and (43) to obtain

$$0 \leq s + \frac{s(1-s)}{d_1} \bar{\lambda}_1 \delta_2 = \delta_1 = S_n(x) \leq 1. \tag{44}$$

4 Constrained Positivity-Preserving Interpolation

In many cases, it is sufficient to preserve positivity through interpolation and not to enforce the stricter requirement of data boundedness. As mentioned in the introduction, the case of unknown extrema between data points is an important example. Let $U^p(x)$ be a positive polynomial of degree n defined over the interval I_i as in Equation (13). For $x \in I_i$, the polynomial $U^p(x)$ is allowed to grow beyond u_i and u_{i+1} but must remain positive. For the polynomial to be positive, one requires that

$$U^p(x) \geq 0. \tag{45}$$

However, in practice, enforcing positivity alone may still result in large oscillations and in extrema that degrade the approximation. We observe this behavior because enforcing positivity alone does not restrict how much the polynomial is allowed to grow beyond the data values. In addition to enforcing positivity, it is important to remove the undesirable oscillations and extrema as much as possible. Let us define u_{min} and u_{max} as

$$u_{min} = \mathbf{min}(u_i, u_{i+1}) - \Delta_{min}, \tag{46}$$

and

$$u_{max} = \mathbf{max}(u_i, u_{i+1}) + \Delta_{max}, \tag{47}$$

where Δ_{min} and Δ_{max} are user-defined parameters used to bound the positive polynomial $U^p(x)$. To allow the polynomial to grow beyond the data values but not produce extrema that are too large, we bound $U^p(x)$ as follows:

$$u_{min} \leq U^p(x) = u_i + (u_{i+1} - u_i) S_n(x) \leq u_{max}. \tag{48}$$

The interpolant $U^p(x)$ is now positive and bounded by u_{min} and u_{max}. Equation (48) is equivalent to bounding $S_n(x)$ as follows:

$$m_\ell \leq S_n(x) \leq m_r, \tag{49}$$

where the factors m_ℓ and m_r are expressed as

(I) : $u_{i+1} > u_i$

$$m_\ell = \mathbf{min}\left(0, \frac{u_{min} - u_i}{u_{i+1} - u_i}\right), \text{ and } m_r = \mathbf{max}\left(1, \frac{u_{max} - u_i}{u_{i+1} - u_i}\right) \tag{50}$$

(II) : $u_{i+1} < u_i$

$$m_\ell = \mathbf{min}\left(0, \frac{u_{max} - u_i}{u_{i+1} - u_i}\right), \text{ and } m_r = \mathbf{max}\left(1, \frac{u_{min} - u_i}{u_{i+1} - u_i}\right). \tag{51}$$

We note that if we set $\Delta_{min} = 0$ and $\Delta_{max} = 0$, we recover Equation (45).

The PPI method is constructed by relaxing the bounds imposed on $\bar{\lambda}_1$ as follows:

$$\Big(- 4(m_r - 1) - 1\Big)d_1 \leq \bar{\lambda}_1 \leq \Big(- 4m_\ell + 1\Big)d_1. \tag{52}$$

Let us demonstrate how the PPI method is constructed in the case of a quadratic interpolant. Starting from the DBI results in the Theorem 1, it follows that

$$0 \leq s + \frac{s(s-1)}{d_1}\bar{\lambda}_1 \leq 1. \tag{53}$$

Relaxing the left and right bounds in Equation (53) by m_ℓ and m_r, respectively leads to

$$m_\ell \leq s + \frac{s(s-1)}{d_1}\bar{\lambda}_1 \leq m_r. \tag{54}$$

Isolating $\bar{\lambda}_1$ from Equation (54) leads to

$$\frac{m_r - s}{s(s-1)}d_1 \leq \bar{\lambda}_1 \leq \frac{m_\ell - s}{s(s-1)}d_1. \tag{55}$$

Equation (55) can be reorganized to obtain

$$\left(\frac{m_r - 1}{s(s-1)} + \frac{1-s}{s(s-1)}\right)d_1 \leq \bar{\lambda}_1 \leq \left(\frac{m_\ell}{s(s-1)} - \frac{s}{s(s-1)}\right)d_1 \tag{56}$$

and then

$$\left(\frac{m_r - 1}{s(s-1)} - \frac{1}{s}\right)d_1 \leq \bar{\lambda}_1 \leq \left(\frac{m_\ell}{s(s-1)} - \frac{1}{(s-1)}\right)d_1. \tag{57}$$

Noting that $\frac{1}{s(s-1)} \leq -4$, $\frac{1}{s} \geq 1$, and $\frac{1}{s-1} \leq -1$, we obtain

$$\Big(- 4(m_r - 1) - 1\Big)d_1 \leq \bar{\lambda}_1 \leq \Big(- 4m_\ell + 1\Big)d_1. \tag{58}$$

Once the bounds on $\bar{\lambda}_1$ and the quadratic interpolant are determined, the extension to cubic, quartic, and higher order interpolants follows the same reconstruction procedure used in the DBI method and outlined from Equation (21) to (25). As in the case of the DBI method, fewer choices are available for $\mathcal{V}_{n-1}$ at the boundary and nearby boundary intervals because there are no points to choose from beyond the boundaries. When a boundary is reached during the process of constructing the stencil $\mathcal{V}_{n-1}$, the remaining mesh points are picked from the side that is towards the interior of the domain. The final stencil at the boundary and nearby the boundary intervals are biased towards the interior of the domain. The recursive expression for the bounds on $\bar{\lambda}_j$ for the PPI method becomes

$$B_j^- = \begin{cases} (-4(m_r-1)-1)d_1 & j=1 \\ (B_{j-1}^- - \bar{\lambda}_{j-1})\frac{d_j}{1-t_j}, \text{ if } t_j \in (-\infty, 0] & j>1 \\ (B_{j-1}^+ - \bar{\lambda}_{j-1})\frac{d_j}{-t_j}, \text{ if } t_j \in (0,1)\cup(1,+\infty) & j>1, \end{cases} \tag{59a}$$

and

$$B_j^+ = \begin{cases} (-4m_\ell+1)d_1, & j=1 \\ (B_{j-1}^+ - \bar{\lambda}_{j-1})\frac{d_j}{1-t_j}, \text{ if } t_j \in (-\infty, 0] & j>1 \\ (B_{j-1}^- - \bar{\lambda}_{j-1})\frac{d_j}{-t_j}, \text{ if } t_j \in (0,+\infty) & j>1. \end{cases} \tag{59b}$$

The difference between the DBI and PPI methods is highlighted in how the bounds B_1^- and B_1^+ are calculated. More precisely, B_1^- and B_1^+ are defined as $-d_1$ and d_1 for the DBI method, whereas for the PPI method, they are defined as $(-4(m_r-1)-1)d_1$ and $(-4m_\ell+1)d_1$, respectively. In addition, the DBI method can be recovered from the PPI methods by setting $m_\ell = 0$ and $m_r = 1$. For example, in the case of the right boundary Equations (20) and (58) can be written as

$$-d_1 \le \bar{\lambda}_1 = \frac{U[x_{N-2}, x_{N-1}, x_N]}{U[x_{N-1}, x_N]}(x_N - x_{N-1}) \le d_1, \text{ and} \tag{60}$$

$$\Big(-4(m_r-1)-1\Big)d_1 \le \bar{\lambda}_1 = \frac{U[x_{N-2}, x_{N-1}, x_N]}{U[x_{N-1}, x_N]}(x_N - x_{N-1}) \le \Big(-4m_\ell+1\Big)d_1, \tag{61}$$

where x_N is the mesh point at the right boundary, $m_\ell \le 0$, $m_r \ge 1$, and

$$d_1 = \frac{x_N - x_{N-2}}{x_N - x_{N-1}}. \tag{62}$$

From Equations (50) and (51), $m_r = 18.94$ and $m_\ell = -18.94$ for the right boundary of the Runge example in Figure 1 below. Equations (60) and (61) show the bounds on $\bar{\lambda}_1$ for data-boundedness and positivity, respectively. Given that $(-4(m_r-1)-1) \le 0$ and $(-4m_\ell+1) \ge 1$, the bounds for positivity are more relaxed than data-boundedness. Thus, enabling the use of higher degree polynomials for the PPI method than for the DBI method.

Theorem 2 *Let us assume that for $x \in I_i$, the polynomials $U_n(x)$ and $S_n(x)$ of degree n are defined as in Equations (13) and (14), respectively. Both polynomials are built starting from the stencil $\mathcal{V}_0 = \{x_i, x_{i+1}\}$, and then by successively appending mesh points from the left and/or right of the interval I_i to obtain the final stencil $\mathcal{V}_{n-1}$. The construction of $\mathcal{V}_{n-1}$ does not require the points to be added in a symmetric fashion alternating from left to right. If for $x \in I_i$, B_j^- defined in Equation (26a) is negative, B_j^+ defined in Equation (26b) is positive, and $B_j^- \leq \bar{\lambda}_j \leq B_j^+$ then for $x \in I_i$*

$$m_\ell \leq S_n(x) \leq m_r,$$

where m_ℓ and m_r are provided in Equations (50) and (51).

Proof As in Theorem 1, the proof begins by using the results from Lemma 1 and the expression $B_j^- \leq \bar{\lambda}_j \leq B_j^+$ to bound $\bar{\lambda}_{n-2}\delta_{n-1}$, $\bar{\lambda}_{n-3}\delta_{n-2}$, $\cdots$, $\bar{\lambda}_1\delta_2$ and so to obtain the result

$$B_1^- \leq \bar{\lambda}_1\delta_2 \leq B_1^+. \tag{63}$$

Equation (63) is then used to derive the target bounds. Starting with the left side of the inequality (i.e., $B_1^- \leq \bar{\lambda}_1\delta_2$) and noting that $\frac{1}{s(s-1)} \leq -4$ and $-\frac{1}{s} \leq -1$, yields

$$\begin{aligned}
\frac{m_r - s}{s(s-1)}d_1 =& \left(\frac{m_r - 1}{s(s-1)} + \frac{1-s}{s(s-1)}\right)d_1 \\
=& \left(\frac{m_r - 1}{s(s-1)} - \frac{1}{s}\right)d_1 \\
\leq& \left(-4(m_r - 1) - 1\right)d_1 \\
=& B_1^- \\
\leq& \bar{\lambda}_1\delta_2.
\end{aligned} \tag{64}$$

Isolating m_r, leads to the desired result

$$m_r \geq s + \frac{s(s-1)}{d_1}\bar{\lambda}_1\delta_2 = \delta_1 = S_n(x). \tag{65}$$

Now, addressing the right side of the inequality (i.e. $B_1^+ \geq \bar{\lambda}_1\delta_2$) and noting that $\frac{1}{s(s-1)} \leq -4$ and $-\frac{1}{s-1} \geq 1$, gives

$$\begin{aligned}
\frac{m_\ell - s}{s(s-1)}d_1 =& \left(\frac{m_\ell}{s(s-1)} - \frac{s}{s(s-1)}\right)d_1 \\
=& \left(\frac{m_\ell}{s(s-1)} - \frac{1}{(s-1)}\right)d_1 \\
\geq& \left(-4m_\ell + 1\right)d_1 \\
=& B_1^+ \\
\geq& \lambda_1\delta_2.
\end{aligned} \tag{66}$$

Isolating m_ℓ leads to the desired bound

$$m_\ell \leq s + \frac{s(s-1)}{d_1}\bar{\lambda}_1\delta_2 = \delta_1 = S_n(x). \tag{67}$$

The proof is concluded by combining Equations (65) and (67) to obtain

$$m_\ell \leq s + \frac{s(s-1)}{d_1}\bar{\lambda}_1\delta_2 = \delta_1 = S_n(x) \leq m_r. \tag{68}$$

At the boundary intervals both the DBI and PPI methods construct the interpolants using a left- or right-biased stencil. For the left boundary, the final stencil is built by successively appending mesh points from the right side of the of the interval I_i. In the same way, the final stencil for the right boundary interval is obtained by successively appending the mesh points from the left side. For the nearby boundary intervals, the stencil points selection process could reach the boundary before completing the final stencil. In such a case, the remaining points are selected from the right if the left boundary is reached and from the left is the right boundary is reached.

4.1 Hidden Local Extrema

The interval I_i may contain a hidden extremum when two of three divided differences $U[x_{i-1}, x_i]$, $U[x_{i+1}, x_i]$ and $U[x_{i+1}, x_{i+2}]$ of the neighboring intervals are of opposite signs. In this case, the PCHIP and DBI algorithms truncate the extremum whereas the relaxed nature of the PPI algorithm allows for a better approximation of the extremum. In [2], when an extremum is detected, the ENO approach is used to construct the interpolant. The ENO approach may fail to recover the extremum or result in oscillations that violate the requirements for positivity and reduce the accuracy. The data-bounded method in [2] is much more restrictive and does not address positivity. These limitations can be addressed by using a bounded positive interpolant.

To simplify the notation, let us defined σ_{i-1}, σ_i and σ_{i+1} such that

$$\sigma_{i-1} = U[x_{i-1}, x_i],\ \sigma_i = U[x_{i+1}, x_i],\ \text{and}\ \sigma_{i+1} = U[x_{i+1}, x_{i+2}]. \tag{69}$$

As in [2] and [24], we assume that there exists an extremum in I_i if

$$\sigma_{i-1}\sigma_{i+1} < 0,\ \text{or}\ \sigma_{i-1}\sigma_i < 0. \tag{70}$$

To address the cases with and without extremum, we choose the parameters Δ_{min} and Δ_{max} according to

$$\Delta_{min} = \begin{cases} \left|\mathbf{min}\big(u_i, u_{i+1}\big)\right| & \text{if } \sigma_{i-1}\sigma_{i+1} < 0 \text{ and } \sigma_{i-1} < 0 \\ & \text{or } \sigma_{i-1}\sigma_{i+1} \geq 0 \text{ and } \sigma_{i-1}\sigma_i < 0 \\ \epsilon\left|\mathbf{min}\big(u_i, u_{i+1}\big)\right| & \text{otherwise,} \end{cases} \tag{71}$$

and

$$\Delta_{max} = \begin{cases} \left|\mathbf{max}\left(u_i, u_{i+1}\right)\right| & \text{if } \sigma_{i-1}\sigma_{i+1} < 0 \text{ and } \sigma_{i-1} > 0 \\ & \text{or } \sigma_{i-1}\sigma_{i+1} \geq 0 \text{ and } \sigma_{i-1}\sigma_i < 0 \\ \epsilon\left|\mathbf{max}\left(u_i, u_{i+1}\right)\right| & \text{otherwise.} \end{cases} \tag{72}$$

ϵ is a parameter introduced to adjust Δ_{min} and Δ_{max} when no extremum is detected. In Equation (71), the interval I_i has a local maximum if $\sigma_{i-1}\sigma_{i+1} < 0$ and $\sigma_{i-1} < 0$. Correspondingly, in Equation (72), the interval I_i has a local minimum if $\sigma_{i-1}\sigma_{i+1} < 0$ and $\sigma_{i-1} > 0$. In both Equations (71) and 72, the type of extremum is ambiguous if $\sigma_{i-1}\sigma_{i+1}$, and $\sigma_{i-1}\sigma_i < 0$. When an extremum is identified, Δ_{min} and/or Δ_{max} are chosen to be sufficiently large to allow the interpolant $U^p(x)$ to grow beyond the data as needed to approximate the extremum without violating the requirement for positivity. In the case where no extremum is identified, the parameter ϵ is used to adjust Δ_{min} and/or Δ_{max} to be sufficiently large to allow higher degree interpolants compared to the DBI method, but sufficiently small to not allow for large oscillations that will degradate the accuracy of the approximation.

In Figure 1, we approximate the Runge function with $N = 17$ LGL points and different values of ϵ, and the target polynomial degree is set to $d = 16$ for each interval. For $\epsilon > 0.01$, the PPI method leads to oscillations, whereas for $\epsilon \leq 0.01$ the oscillations are removed. Similar oscillations are seen when using high-order Chebyshev polynomials. The cutoff for the positive parameter ϵ depends on the underlying function and the input data. For the Runge example with $N = 17$ uniformly-spaced points, the spurious oscillations are removed for $\epsilon \leq 0.05$. With the same Runge example with $N = 129$ and $d = 16$, the unconstrained approximation does not produce oscillations and ϵ can be set to any value in $[0, 1]$. In the case of the smoothed Heaviside examples, setting $\epsilon = 0.05$ with $N = 17$ uniformly-spaced points lead to large oscillations that degrade the approximations. However, for $\epsilon \leq 0.01$ with $N = 17$, the oscillations are significantly reduced, and the approximation improved, as shown on the bottom part of Figure 1. Setting $\epsilon = 0.0$ will completely eliminate the oscillations. Overall, using $\epsilon \leq 0.01$ is sufficient to remove or significantly reduce the oscillations and improve the approximation. For an interval I_i with no extremum, as ϵ approaches zero and both Δ_{min} and Δ_{max} get smaller, the approximation method becomes closer to the DBI approach. As for the DBI approach, the PPI method may become restrictive for higher degree polynomial interpolants as ϵ approaches zero. This approach is also further explored for a variety of practical applications [21].

The right part of Figure 1 shows the interpolants used at the right boundaries in both the Runge and smoothed Heaviside examples. At the right boundary of the Runge example, the stencil $\{x_{N-12} \cdots x_N\}$ is used to build the data-bounded interpolant and the stencil $\{x_{N-16}, \cdots, x_N\}$ is used for the positive interpolant with $\epsilon = 1$. As the positive parameter ϵ gets smaller the upper and lower bounds for the interpolant gets tighter and converges to the DBI bounds. The stencil used for both the DBI and PPI are the same for $\epsilon \leq 0.01$. At the

boundary intervals the PPI method allows for higher degree interpolants compared to the DBI method. However, these higher degree interpolants while positive may introduce oscillations that can be removed using the parameter ϵ.

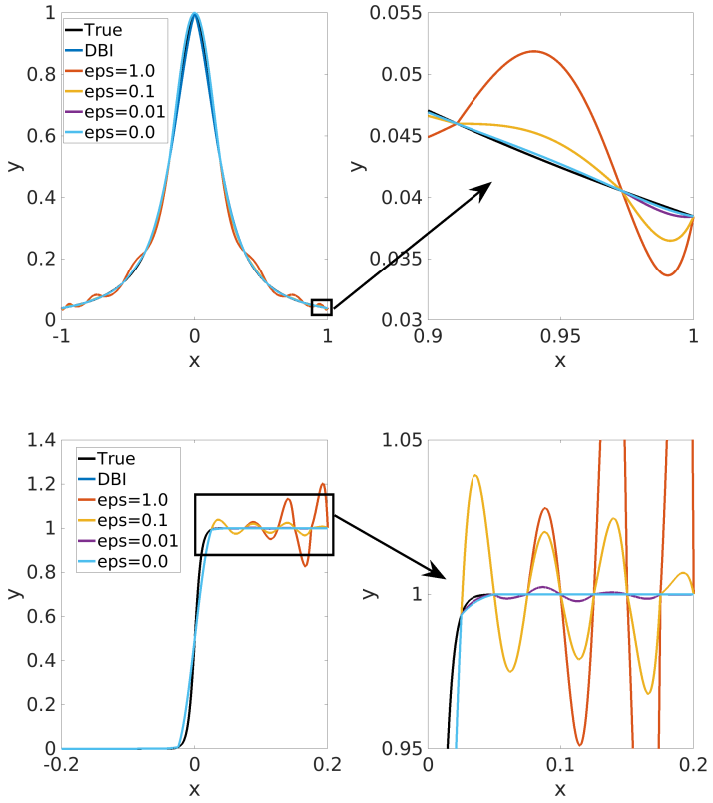

Fig. 1: The top row shows an approximation of $f_1(x)$ from $N = 17$ LGL points using DBI and PPI with different values of ϵ. The bottom row shows an approximation of $f_2(x)$ from $N = 17$ uniformly-spaced points using DBI and PPI with different values of ϵ. The target polynomial degree is set to $d = 16$ for both $f_1(x)$ and $f_2(x)$.

For $u_i = u_{i+1}$, m_ℓ, m_r and $U_n(x)$ as written in Equations (50), (51) and (13) are not defined. The PPI algorithm addresses this limitation by re-writing $U_n(x)$ as

$$U_n(x) = u_i + U[x_1^l, \cdots, x_1^r](x_{i+1} - x_i)(x_1^r - x_1^l)S_n(x), \tag{73}$$

where $S_n(x)$ is expressed as follows:

$$S_n(x) = \sum_{j=1}^{n-1} \bar{s}_j. \tag{74}$$

The summation starts at $j = 1$ because the linear term $\frac{u_{i+1}-u_i}{x_{i+1}-x_i}(x - x_i) = 0$. Let

$$w = U[x_1^l, \cdots, x_1^r](x_{i+1} - x_i)(x_1^r - x_1^l). \tag{75}$$

$\bar{\lambda}_j$ in this context is defined as

$$\bar{\lambda}_j = \frac{U[x_j^l, \cdots, x_j^r]}{w} \prod_{k=0}^{j} (x_k^r - x_k^l). \tag{76}$$

For $u_i = u_{i+1}$, the parameters m_ℓ and m_r are then defined according to

(I) : $U[x_1^l, \cdots, x_1^r] > 0$

$$m_\ell = \mathbf{min}\left(0, \frac{u_{min} - u_i}{w}\right), \text{ and } m_r = \mathbf{max}\left(1, \frac{u_{max} - u_i}{w}\right) \tag{77}$$

(II) : $U[x_1^l, \cdots, x_1^r] < 0$

$$m_\ell = \mathbf{min}\left(0, \frac{u_{max} - u_i}{w}\right), \text{ and } m_r = \mathbf{max}\left(1, \frac{u_{min} - u_i}{w}\right). \tag{78}$$

For $U[x_i, x_{i+1}] = U[x_1^l, \cdots, x_1^r] = 0$, the data u_{i-1}, u_i, u_{i+1}, and u_{i+2} have the same value ($u_{i-1} = u_i = u_{i+1} = u_{i+2}$). In this case, the algorithm approximates the function in the interval I_i with a linear interpolant. For both cases $U[x_1^l, \cdots, x_1^r] < 0$ and $U[x_1^l, \cdots, x_1^r] > 0$, B_j^+ and B_j^- remain defined as previously in Equations (59b) and (59a). Lemma 1 and Theorem 2 still hold and remain unchanged.

Figure 2 shows an example with $u_i = u_{i+1}$ and a hidden local extremum at $x = 0$. In Figure 2, we approximate the Runge function $f_1(x)$ using the PCHIP, DBI, and PPI methods from 16 uniformly-spaced data points. The PPI method is able to better capture the peak compared to the DBI and PCHIP methods.

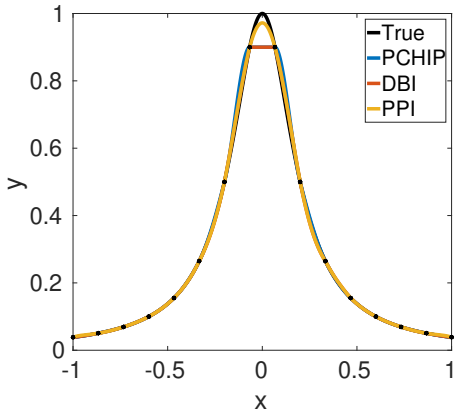

Fig. 2: Approximation of $f_1(x)$ with $N = 16$ points using PCHIP, DBI and PPI. The interpolants from DBI and PPI are in $\mathcal{P}_8$, where 8 is the target polynomial degree.

4.2 Algorithm

The ENO reconstruction can result in a stencil that is biased to the left or right. Rogerson et al. [22] demonstrated that a biased ENO stencil may lead to some stability issues when used to solve hyperbolic equations, and a refined resolution may lead to even larger errors. To address this limitation, Shu [26] developed a modified ENO reconstruction that uses a bias coefficient to target a preferred final stencil. Furthermore, a left- and right-biased stencil may fail to recover hidden local extrema. For instance, if $U[x_{i-1}, x_i] > 0$, $U[x_i, x_{i+1}] < 0$, and $U[x_{i+1}, x_{i+2}] > 0$, the interval I_i has an extremum. In such a case, if the points in the final stencil are all to the right or left of x_i, the interpolant may fail to recover the extremum. The points x_{i-1} and x_{i+2} are important for identifying and reconstructing a hidden local extremum. However, the right-biased stencils does not include x_{i-1}, and the left-biased stencil does not include x_{i+2}. To resolve these issues due to biased stencils, the algorithm introduced here favors a symmetric stencil over the ENO stencil in addition to enforcing the requirements for data boundedness or positivity preservation. A symmetric stencil centered around x_i includes x_{i-1} and x_{i+2} and better approximates a hidden local extremum compared to a biased stencil.

Before we present the algorithm for the DBI and PPI method, let us define $\bar{\lambda}^-_{j+1}$ and $\bar{\lambda}^+_{j+1}$. At any given step j, the next point inserted into $\mathcal{V}_j$ can be to the right or left. $\bar{\lambda}^-_{j+1}$ and $\bar{\lambda}^+_{j+1}$ correspond to the case where the stencil inserted is to the left and right, respectively.

$$\begin{cases} \bar{\lambda}^-_{j+1} = \bar{\lambda}_{j+1} & \text{with } \mathcal{V}_{j+1} = \{x_p\} \cup \mathcal{V}_j \\ \bar{\lambda}^+_{j+1} = \bar{\lambda}_{j+1} & \text{with } \mathcal{V}_{j+1} = \mathcal{V}_j \cup \{x_q\}. \end{cases} \tag{79}$$

As a reminder, x_p and x_q are the mesh points immediately to the left and right of $\mathcal{V}_j$. Given $\mathcal{V}_j$, let μ^l_j be the number of points to the left of x_i and μ^r_j

the number of points to the right. Below we introduce an algorithm for DBI and PPI based on the procedures introduced above.
Input: $\{x_i\}_{i=0}^{n}$, $\{u_i\}_{i=0}^{n}$, $\{\tilde{x}_i\}_{i=0}^{\tilde{n}}$, ϵ and d. **Output:** $\{\tilde{u}_i\}_{i=0}^{\tilde{n}}$.

1. Select an interval $[x_i, x_{i+1}]$. Let $\mathcal{V}_0 = \{x_i, x_{i+1}\} = \{x_0^l, x_0^r\}$.
2. If $\sigma_{i-1}\sigma_{i+1} < 0$ or $\sigma_{i-1}\sigma_i < 0$, then the interval I_i has a hidden local extremum. For the boundary intervals, we assume that the divided differences to the left and right have the same sign.
3. Compute u_{min} and u_{max} using Equations (46) and (47).
4. Compute m_r and m_ℓ based on Equations (50) and (51) or Equations (72) and (73). For DBI, set $m_r = 1$ and $m_\ell = 0$.
5. Given a stencil $\mathcal{V}_j$,
   - if $B_{j+1}^- \le \bar{\lambda}_{j+1}^+ \le B_{j+1}^+$ and $B_{j+1}^- \le \bar{\lambda}_{j+1}^- \le B_{j+1}^+$
     - if $\mu_j^l < \mu_j^r$ then insert a new stencil point to the left;
     - else if $\mu_j^l > \mu_j^r$ then insert a new stencil point to the right;
     - else insert a new stencil point to the right if $|\bar{\lambda}_{j+1}^l| \ge |\bar{\lambda}_{j+1}^r|$, otherwise insert a new point to left;
   - else if $B_{j+1}^- \le \bar{\lambda}_{j+1}^- \le B_{j+1}^+$, then insert a new stencil point to the left;
   - else if $B_{j+1}^- \le \bar{\lambda}_{j+1}^+ \le B_{j+1}^+$, then insert a new stencil point to the right;
6. This process (Steps 3) iterates until the halting criterion that the ratio of divided differences lies outside the required bounds stated above or the stencil has $d+1$ points, with d being the target degree for the interpolant.
7. Evaluate the final interpolant $U^l(x)$ (for DBI) or $U^p(x)$ (for PPI) at the output points $\tilde{x}_i$ that are in I_i.
8. Repeat Steps 1–7 for each interval in the input 1D mesh.

At the left and right boundary intervals there are no mesh points beyond the boundaries to calculate σ_{i-1} and σ_{i+1}, respectively. At both boundaries σ_{i-1} is set to σ_{i+1} ($\sigma_{i-1} = \sigma_{i+1}$) to ensure that no new extrema are introduced. At the boundary and nearby boundary intervals the algorithm allows for hidden local extrema to be recovered. For example, if the right boundary interval has a hidden extremum $\sigma_{i-1}\sigma_i < 0$ (from Step 2) then the algorithm will relax the bounds on the interpolant and allow for the extremum to be recovered.

5 Numerical Experiments

In this section, we present both 1D and 2D numerical experiments that demonstrate the properties of our proposed methods. These experimental studies use the PCHIP, DBI, and PPI methods. The test functions used here are taken from test problems 1, 2, 7, and 10 in [20]. A full suite of test problems has been undertaken by the authors in [20]. In that study, nine test problems are used with both uniform and nonuniform Legendre-Gauss-Lobatto (LGL) meshes. The Legendre-Gauss-Lobatto mesh consists of uniform elements with eight LGL quadrature nodes [11] inside each element. The number of elements is determined by $(N-1)/8$ and $(N-1)^2/16$ for the 1D and 2D examples. The

integrals in the L^2-norm calculation are approximated using the trapezoid rule with 10^4 and $10^3 \times 10^3$ uniformly-spaced points for the 1D and 2D examples, respectively. The parameter ϵ is set to 0.01 and this choice is to allow the interpolant in each interval to grow beyond the data in a bounded way.

For various problems, including all the examples below, a standard Lagrange interpolant leads to large oscillations and negative values. While the ENO and WENO methods reduce the oscillations, they do not address the issue of preserving data boundedness or positivity. The DBI and PPI methods resolve both issues. The numerical experiments compare the DBI and PPI methods against the widely used PCHIP method, and show approximation errors using the algorithm described in Section 4.2.

5.1 1D Example: Runge Function

Our first example uses the Runge [6] function, defined as follows:

$$f_1(x) = \frac{1}{1+25x^2}, \quad x \in [-1,1]. \tag{80}$$

Approximating the Runge function via a standard global polynomial using the set of points provided for the experiment leads to large oscillations and negative values.

Tables 1 and 2 show L^2-errors and convergence rates when approximating the Runge function $f_1(x)$ using the uniform and LGL meshes. For the approximations in Table 1, we use the PCHIP, DBI, and PPI methods with a target polynomial degree $d = 3$; whereas in Table 2, we use the DBI and PPI methods with the target polynomial degree varying from $d = 1$ to $d = 16$. The results in Table 1 show that the DBI and PPI methods lead to smaller errors and larger convergence rates compared to PCHIP for N larger than 17 in both the uniform and LGL mesh examples. For $N = 17$, the PCHIP approach leads to smaller errors. For higher polynomial degrees, the PPI method gives better results compared to the DBI and PCHIP, as demonstrated in Table 2. These results demonstrate that the PPI method is a suitable approach for interpolating data from one mesh to another when the underlying function is similar to the Runge function. For $N = 17$ in this example, the higher order terms added when going from $\mathcal{P}_8$ to $\mathcal{P}_{16}$ increase the L^2-error norms. These results indicate that resolution for $N = 17$ is not sufficient to see polynomial convergence when going from $\mathcal{P}_8$ to $\mathcal{P}_{16}$. The L^2-errors norms decrease with larger values of N.

Figure 3 shows the errors found when approximating the Runge function $f_1(x)$ with PCHIP, DBI, and PPI. The top and bottom plots in Figure 3 show the absolute errors when approximating the Runge example using $N = 33$ and $N = 129$ uniformly-spaced points, respectively. The target polynomial degree is set to $d = 8$ for both the DBI and PPI methods and $\epsilon = 0.01$. The errors around the middle of the domain dominate the overall error. The relaxed nature of the PPI method allows for higher degree interpolants compared to

the DBI and PCHIP which leads to better approximations, as shown in the bottom plots in Figure 3.

N	PCHIP	Rate	DBI	Rate	PPI	Rate
Uniform Mesh						
17	7.15E-03	–	1.01E-02	–	1.01E-02	–
33	1.91E-03	1.99	1.21E-03	3.20	1.59E-03	2.78
65	3.70E-04	2.42	9.64E-05	3.73	1.12E-04	3.92
129	6.79E-05	2.47	6.29E-06	3.98	6.29E-06	4.20
257	1.22E-05	2.49	3.94E-07	4.02	3.94E-07	4.02
LGL Mesh						
17	4.75E-03	–	8.36E-03	–	8.38E-03	–
33	1.30E-03	1.96	1.84E-03	2.28	1.84E-03	2.28
65	2.86E-04	2.23	2.05E-04	3.24	2.05E-04	3.24
129	5.81E-05	2.32	1.17E-05	4.17	1.17E-05	4.17
257	1.15E-05	2.35	1.04E-06	3.51	1.04E-06	3.51

Table 1: L^2-errors and rates of convergence when using the PCHIP, DBI, and PPI methods to approximate the function $f_1(x)$. N represents the number of input points used to build the approximation. The approximation functions for the DBI and PPI methods are cubic interpolants.

	Uniform Mesh				LGL Mesh			
N	DBI		PPI		DBI		PPI	
	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate
	$\mathcal{P}_1$							
17	2.16E-02	–	2.16E-02	–	1.69E-02	–	1.69E-02	–
33	6.02E-03	1.92	6.02E-03	1.92	5.84E-03	1.60	5.84E-03	1.60
65	1.52E-03	2.03	1.52E-03	2.03	1.66E-03	1.86	1.66E-03	1.86
129	3.82E-04	2.02	3.82E-04	2.02	5.80E-04	1.53	5.80E-04	1.53
257	9.56E-05	2.01	9.56E-05	2.01	1.52E-04	1.94	1.52E-04	1.94
	$\mathcal{P}_4$							
17	8.34E-03	–	7.02E-03	–	6.55E-03	–	6.54E-03	–
33	5.91E-04	3.99	5.91E-04	3.73	7.62E-04	3.24	7.62E-04	3.24
65	4.26E-05	3.88	2.39E-05	4.73	5.30E-05	3.93	5.29E-05	3.94
129	2.68E-06	4.03	8.00E-07	4.95	3.44E-06	3.99	3.44E-06	3.99
257	8.63E-08	4.99	2.55E-08	5.00	8.88E-08	5.31	8.87E-08	5.31
	$\mathcal{P}_8$							
17	4.61E-03	–	3.11E-03	–	3.49E-03	–	4.40E-03	–
33	4.43E-04	3.53	1.51E-04	4.56	1.76E-04	4.50	1.76E-04	4.85
65	3.67E-05	3.67	1.05E-06	7.33	3.25E-06	5.89	3.01E-06	6.00
129	2.56E-06	3.88	3.10E-09	8.50	5.64E-08	5.91	8.82E-09	8.51
257	8.24E-08	4.99	6.80E-12	8.88	3.51E-09	4.03	3.96E-11	7.84
	$\mathcal{P}_{16}$							
17	4.34E-03	–	3.44E-03	–	4.89E-03	–	5.01E-03	–
33	4.21E-04	3.52	4.85E-05	6.43	1.18E-04	5.62	1.17E-04	5.67
65	3.67E-05	3.60	5.92E-08	9.89	1.22E-06	6.75	9.40E-08	10.51
129	2.56E-06	3.88	4.21E-12	13.94	5.57E-08	4.50	1.02E-11	13.32
257	8.24E-08	4.99	2.18E-16	14.32	3.51E-09	4.01	5.04E-16	14.38

Table 2: L^2-errors and rates of convergence when using the DBI and PPI methods to approximate the function $f_1(x)$. N represents the number of input points used to build the approximation. The interpolants are in $\mathcal{P}_j$, where j is the target polynomial degree.

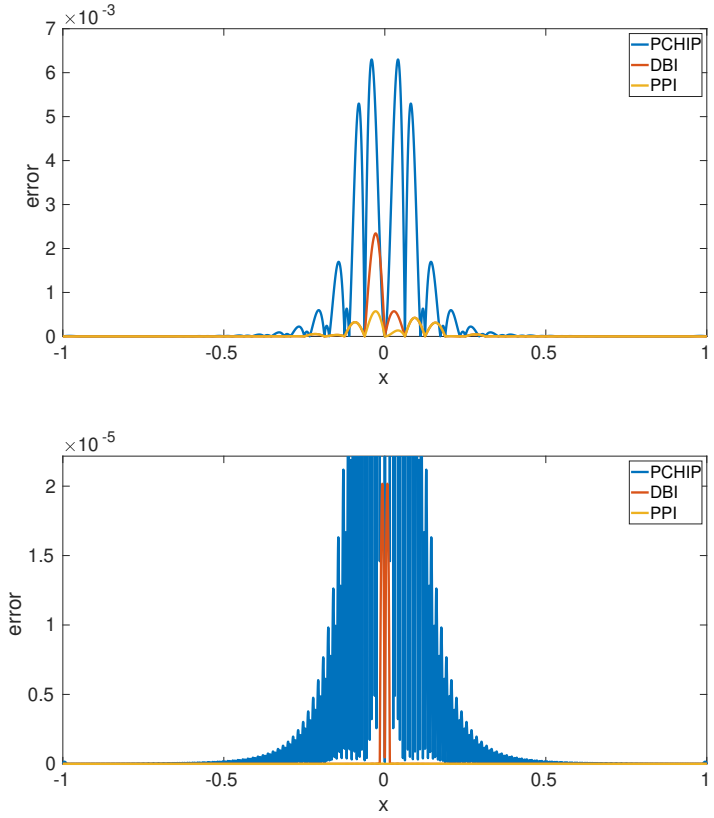

Fig. 3: Error plots when approximating $f_1(x)$. The top and bottom error plots are obtained from approximating $f_1(x)$ with $N = 33$ and $N = 129$ uniformly-spaced points, respectively. The target polynomial degree is set to $d = 8$ and $\epsilon = 0.01$.

5.2 1D Example: Smoothed Heaviside Function

This 1D example uses an analytic approximation of the Heaviside function defined as

$$f_2(x) = \frac{1}{1 + e^{-2kx}}, \quad k = 100, \text{ and } x \in [-0.2, 0.2]. \tag{81}$$

A polynomial approximation of $f_2(x)$ is challenging because of the large solution gradient around $x = 0$. Attempts to use a standard polynomial approximation for this function result in oscillations and negative values.

Tables 3 and 4 show L^2-errors and convergence rates when approximating the smoothed Heaviside function $f_2(x)$ using the uniform and LGL meshes. Table 4 shows that for a target polynomial of degree $d = 3$, the errors for

PCHIP, DBI, and PPI are comparable. When the target degree increases from $d = 1$ to $d = 16$, the errors for the DBI and PPI methods decrease, as shown in Table 4. Overall, the errors from the DBI and PPI methods are comparable with DBI yielding slightly smaller errors than PPI. The uniform mesh leads to better approximation results compared to the LGL mesh. These results demonstrate that the DBI and PPI methods are both suitable for mapping data between different meshes when the underlying function is similar to the smoothed Heaviside function.

Figure 4 provides examples of error plots for approximating the smoothed Heaviside function $f_2(x)$ with PCHIP, DBI, and PPI. The top and bottom plots in Figure 4 show the absolute error when approximating the smoothed Heaviside function $f_2(x)$ using $N = 33$ and $N = 129$ uniformly-spaced points, respectively. The global error is dominated by the errors in the region with the steep gradient around $x = 0$. The error from DBI and PPI are identical for $N = 129$ because the stencil selected by both methods are the same around the region with the steep gradients. Away from the steep gradient the DBI and PPI methods use different stencils but the errors in those regions are negligible compared to the errors around $x = 0$.

N	PCHIP	Rate	DBI	Rate	PPI	Rate
	Uniform Mesh					
17	2.02E-02	–	1.97E-02	–	1.97E-02	–
33	3.38E-03	2.70	3.53E-03	2.59	3.54E-03	2.59
65	3.59E-04	3.31	5.00E-04	2.88	5.00E-04	2.89
129	4.21E-05	3.13	4.51E-05	3.51	4.51E-05	3.51
257	5.12E-06	3.06	3.01E-06	3.93	3.01E-06	3.93
	LGL Mesh					
17	3.65E-03	–	5.38E-03	–	5.38E-03	–
33	1.45E-03	1.39	1.55E-03	1.88	1.56E-03	1.86
65	4.07E-04	1.87	6.49E-04	1.28	6.49E-04	1.30
129	8.85E-05	2.23	9.77E-05	2.76	9.77E-05	2.76
257	1.38E-05	2.70	9.06E-06	3.45	9.06E-06	3.45

Table 3: L^2-errors and rates of convergence when using the PCHIP, BDI, and PPI methods to approximate the function $f_2(x)$. N represents the number of input points used to build the approximation. The approximation functions for the DBI and PPI methods are cubic interpolants.

	Uniform Mesh				LGL Mesh			
N	DBI		PPI		DBI		PPI	
	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate
				$\mathcal{P}_1$				
17	2.89E-02	–	2.89E-02	–	8.58E-03	–	8.58E-03	–
33	7.69E-03	1.99	7.69E-03	1.99	5.24E-03	0.74	5.24E-03	0.74
65	1.80E-03	2.14	1.80E-03	2.14	2.20E-03	1.28	2.20E-03	1.28
129	4.58E-04	2.00	4.58E-04	2.00	8.08E-04	1.47	8.08E-04	1.47
257	1.15E-04	2.00	1.15E-04	2.00	2.01E-04	2.01	2.01E-04	2.01
				$\mathcal{P}_4$				
17	2.23E-02	–	2.23E-02	–	5.24E-03	–	5.24E-03	–
33	4.09E-03	2.56	4.10E-03	2.56	1.10E-03	2.36	1.11E-03	2.34
65	3.05E-04	3.83	3.05E-04	3.84	3.06E-04	1.88	3.07E-04	1.89
129	1.35E-05	4.55	1.35E-05	4.55	3.32E-05	3.24	3.32E-05	3.24
257	4.71E-07	4.87	4.71E-07	4.87	1.17E-06	4.85	1.17E-06	4.85
				$\mathcal{P}_8$				
17	2.08E-02	–	2.08E-02	–	4.87E-03	–	4.68E-03	–
33	3.36E-03	2.75	3.33E-03	2.76	8.71E-04	2.59	7.84E-04	2.69
65	1.38E-04	4.70	1.38E-04	4.69	7.57E-05	3.60	1.24E-04	2.72
129	1.22E-06	6.90	1.22E-06	6.90	2.17E-06	5.19	2.17E-06	5.90
257	4.44E-09	8.15	4.44E-09	8.15	1.95E-08	6.83	1.95E-08	6.83
				$\mathcal{P}_{16}$				
17	2.00E-02	–	2.00E-02	–	4.83E-03	–	4.64E-03	–
33	2.93E-03	2.90	2.91E-03	2.91	7.38E-04	2.83	7.27E-04	2.80
65	9.17E-05	5.11	9.17E-05	5.10	7.60E-05	3.35	9.41E-05	3.02
129	1.70E-07	9.17	1.70E-07	9.17	2.88E-07	8.14	2.88E-07	8.45
257	2.64E-11	12.73	2.64E-11	12.73	5.39E-11	12.45	5.39E-11	12.45

Table 4: L^2-errors and rates of convergence when using the DBI and PPI methods to approximate the function $f_2(x)$. N represents the number of input points used to build the approximation. The interpolants are in $\mathcal{P}_j$, where j is the target polynomial degree.

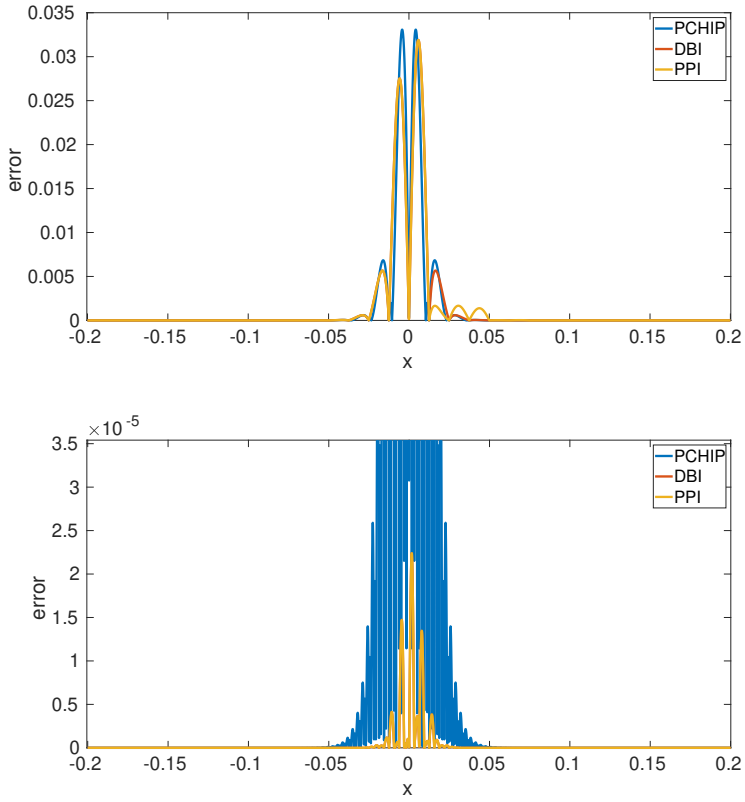

Fig. 4: Error plots when approximating $f_2(x)$. The top and bottom error plots are obtained from approximating $f_1(x)$ with $N = 33$ and $N = 129$ uniformly-spaced points, respectively. The target polynomial degree is set to $d = 8$ and $\epsilon = 0.01$.

5.3 2D Example: Runge Function

The 2D example uses an extended version of the 1D Runge function defined in Equation (80) to give

$$f_7(x, y) = \frac{1}{1 + 25(x^2 + y^2)}, \quad x, y \in [-1, 1]. \tag{82}$$

Tables 5 and 6 show L^2-errors and convergence rates when approximating the 2D Runge function $f_7(x, y)$ using the uniform and LGL meshes. Table 5 compares PCHIP against DBI, and PPI with a target degree $d = 3$. Table 6 focuses on high-order interpolants using the DBI and PPI methods. Both the DBI and PPI methods have smaller errors compared to the PCHIP approach.

As the target polynomial degree increases, the PPI method gives better approximation results compared to DBI and PCHIP.

The errors in the DBI method drop more slowly than PPI when more mesh points are used because the PPI method uses higher degree interpolants compared to the DBI method. The bounds on the interpolants and $\bar{\lambda}_j$ for data-boundedness (DBI) are more restrictive than the bounds for positivity (PPI). In Tables 6 and 2, when going from $d = 8$ to 16 with $N = 65$, 129, and 257 the relaxed nature of the PPI method allows for more stencil points to be used to construct the final interpolant for each interval. However, the conditions for data-boundedness are more restrictive and do not allow for more stencil points to be added when going from $d = 8$ to 16.

N^2	PCHIP	Rate	DBI	Rate	PPI	Rate
Uniform Mesh						
17^2	5.01E-03	–	7.14E-03	–	7.28E-03	–
33^2	1.23E-03	2.12	7.82E-04	3.33	8.55E-04	3.23
65^2	2.33E-04	2.45	5.65E-05	3.88	5.59E-05	4.02
129^2	4.27E-05	2.48	3.59E-06	4.02	3.63E-06	3.99
257^2	7.72E-06	2.48	2.24E-07	4.03	2.27E-07	4.02
LGL Mesh						
17^2	3.26E-03	–	5.62E-03	–	5.60E-03	–
33^2	8.58E-04	2.01	1.09E-03	2.48	1.09E-03	2.47
65^2	1.88E-04	2.24	1.17E-04	3.29	1.17E-04	3.29
129^2	3.75E-05	2.35	7.05E-06	4.09	7.05E-06	4.09
257^2	7.32E-06	2.37	6.07E-07	3.56	6.07E-07	3.56

Table 5: L^2-errors and rates of convergence when using the PCHIP, DBI, and PPI methods to approximate the function $f_7(x, y)$. N^2 represents the number of input points used to build the approximation. The approximation functions for the DBI and PPI methods are cubic interpolants.

	Uniform Mesh				LGL Mesh			
N^2	DBI		PPI		DBI		PPI	
	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate
				$\mathcal{P}_1$				
17^2	1.60E-02	–	1.60E-02	–	1.10E-02	–	1.10E-02	–
33^2	4.42E-03	1.94	4.42E-03	1.94	3.62E-03	1.68	3.62E-03	1.68
65^2	1.12E-03	2.02	1.12E-03	2.02	1.20E-03	1.62	1.20E-03	1.62
129^2	2.82E-04	2.02	2.82E-04	2.02	4.27E-04	1.51	4.27E-04	1.51
257^2	7.06E-05	2.01	7.06E-05	2.01	1.11E-04	1.96	1.11E-04	1.96
				$\mathcal{P}_4$				
17^2	5.07E-03	–	4.63E-03	–	4.02E-03	–	4.05E-03	–
33^2	3.71E-04	3.94	3.60E-04	3.85	4.45E-04	3.32	4.45E-04	3.33
65^2	2.62E-05	3.91	1.31E-05	4.89	3.08E-05	3.94	3.08E-05	3.94
129^2	1.23E-06	4.46	4.36E-07	4.96	1.88E-06	4.08	1.88E-06	4.08
257^2	4.96E-08	4.66	1.39E-08	5.00	4.80E-08	5.32	4.79E-08	5.32
				$\mathcal{P}_8$				
17^2	3.24E-03	–	3.41E-03	–	3.56E-03	–	3.47E-03	–
33^2	2.88E-04	3.65	1.95E-04	4.31	9.39E-05	5.48	9.34E-05	5.45
65^2	2.35E-05	3.70	5.14E-07	8.76	1.80E-06	5.83	1.51E-06	6.08
129^2	1.16E-06	4.39	1.49E-09	8.53	4.43E-08	5.41	4.53E-09	8.48
257^2	4.78E-08	4.63	3.25E-12	8.89	1.73E-09	4.71	1.87E-11	7.96
				$\mathcal{P}_{16}$				
17^2	3.69E-03	–	3.89E-03	–	4.18E-03	–	4.18E-03	–
33^2	2.85E-04	3.86	1.85E-04	4.59	5.62E-05	6.50	5.68E-05	6.48
65^2	2.35E-05	3.68	2.63E-08	13.07	1.19E-06	5.69	4.28E-08	10.61
129^2	1.16E-06	4.38	1.77E-12	14.01	5.42E-08	4.51	4.31E-12	13.43
257^2	4.76E-08	4.64	1.89E-15	9.93	2.02E-09	4.77	1.02E-14	8.77

Table 6: L^2-errors and rates of convergence when using the DBI and PPI methods to approximate the function $f_7(x, y)$. N^2 represents the number of input points used to build the approximation. The interpolants are in $\mathcal{P}_j$, where j is the target polynomial degree.

5.4 2D Example: Smoothed Heaviside Function

This 2D example uses an extension of the 1D approximation of the Heaviside function $f_2(x)$ defined in Equation (81). The extended version is defined as follows:

$$f_{10}(x, y) = \frac{1}{1 + e^{-\sqrt{2}k(x+y)}}, \quad k = 100, \text{ and } x, y \in [-0.2, 0.2]. \tag{83}$$

The function $f_{10}(x, y)$ is challenging because of the large gradient at $y = -x$.

Tables 7 and 8 show L^2-errors and convergence rates when approximating the smoothed Heaviside function $f_{10}(x, y)$ using the uniform and LGL meshes. For a target polynomial of degree $d = 3$, the errors for PCHIP, DBI, and PPI are comparable. As the target degree increases, the errors for the DBI and PPI decrease, as shown in Table 4. Overall, the errors from the DBI and PPI approaches are similar.

In Tables 8 and 4, the errors for the DBI and PPI methods are the same because the example used has no extrema and the interpolants used in the regions with steep gradients are the same for both the DBI and PPI methods. The global errors in both examples are dominated by errors in the regions

with steep gradients. These regions are around $x = 0$ and $y = -x$ for the 1D and 2D examples, respectively. Away from the steep gradients DBI and PPI use different interpolants and the errors are small compared to errors around $x = 0$ and $y = -x$.

N^2	PCHIP	Rate	DBI	Rate	PPI	Rate
Uniform Mesh						
17^2	8.07E-03	–	1.04E-02	–	1.05E-02	–
33^2	1.26E-03	2.80	2.06E-03	2.44	2.05E-03	2.47
65^2	1.44E-04	3.20	2.38E-04	3.18	2.38E-04	3.17
129^2	1.63E-05	3.18	1.64E-05	3.90	1.64E-05	3.90
257^2	1.94E-06	3.08	1.05E-06	3.99	1.05E-06	3.99
LGL Mesh						
17^2	1.23E-02	–	1.54E-02	–	1.56E-02	–
33^2	2.51E-03	2.39	3.86E-03	2.09	3.83E-03	2.11
65^2	3.37E-04	2.96	5.53E-04	2.87	5.53E-04	2.86
129^2	4.19E-05	3.04	4.09E-05	3.80	4.09E-05	3.80
257^2	5.96E-06	2.83	2.50E-06	4.05	2.50E-06	4.05

Table 7: L^2-errors and rates of convergence when using the PCHIP, DBI, and PPI methods to approximate the function $f_{10}(x, y)$. N^2 represents the number of input points used to build the approximation. The approximation functions for the DBI and PPI methods are cubic interpolants.

	Uniform Mesh				LGL Mesh			
N^2	DBI		PPI		DBI		PPI	
	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate	L^2-error	Rate
				$\mathcal{P}_1$				
17^2	1.50E-02	–	1.50E-02	–	2.05E-02	–	2.05E-02	–
33^2	4.57E-03	1.79	4.57E-03	1.79	6.79E-03	1.66	6.79E-03	1.66
65^2	1.26E-03	1.90	1.26E-03	1.90	1.89E-03	1.89	1.89E-03	1.89
129^2	3.23E-04	1.98	3.23E-04	1.98	4.86E-04	1.98	4.86E-04	1.98
257^2	8.15E-05	2.00	8.15E-05	2.00	1.24E-04	1.98	1.24E-04	1.98
				$\mathcal{P}_4$				
17^2	9.45E-03	–	9.42E-03	–	1.37E-02	–	1.36E-02	–
33^2	1.33E-03	2.95	1.31E-03	2.98	2.72E-03	2.43	2.71E-03	2.44
65^2	9.29E-05	3.93	9.29E-05	3.90	2.39E-04	3.59	2.39E-04	3.58
129^2	3.67E-06	4.71	3.67E-06	4.71	1.10E-05	4.49	1.10E-05	4.49
257^2	1.21E-07	4.95	1.21E-07	4.95	3.90E-07	4.84	3.90E-07	4.84
				$\mathcal{P}_8$				
17^2	8.04E-03	–	8.00E-03	–	1.22E-02	–	1.21E-02	–
33^2	1.03E-03	3.10	9.30E-04	3.25	1.76E-03	2.91	1.75E-03	2.92
65^2	4.83E-05	4.51	4.89E-05	4.35	4.98E-05	5.26	4.98E-05	5.25
129^2	2.57E-07	7.64	2.57E-07	7.66	4.03E-07	7.03	4.03E-07	7.03
257^2	5.27E-10	8.98	5.27E-10	8.98	1.21E-09	8.42	1.21E-09	8.42
				$\mathcal{P}_{16}$				
17^2	7.32E-03	–	7.31E-03	–	1.17E-02	–	1.16E-02	–
33^2	1.03E-03	2.96	8.90E-04	3.17	1.46E-03	3.14	1.44E-03	3.15
65^2	2.13E-04	2.32	2.09E-04	2.13	1.83E-04	3.06	1.64E-04	3.20
129^2	1.03E-06	7.78	1.03E-06	7.76	2.15E-07	9.85	2.15E-07	9.69
257^2	4.41E-11	14.59	4.41E-11	14.59	9.37E-12	14.57	9.37E-12	14.57

Table 8: L^2-errors and rates of convergence when using the DBI and PPI methods to approximate the function $f_{10}(x, y)$. N^2 represents the number of input points used to build the approximation. The interpolants are in $\mathcal{P}_j$, where j is the target polynomial degree.

5.5 Hidden Local Extrema Examples

This numerical study demonstrates the ability of the PPI method to recover hidden extrema. The study uses the Runge functions $f_1(x)$ and $f_7(x, y)$ with ~~a~~ uniform meshes. The uniformly-spaced mesh points are constructed such that the extremum at $x = 0$ lies inside of an interval. Tables 9 and 10 show L^2-error norms and convergence rates when approximating $f_1(x)$ and $f_7(x, y)$ from Equations (80) and (82). The results from both tables show that the PPI method leads to smaller errors and larger convergence rates compared to the DBI method. The DBI approach uses a bounded interpolant that fails to represent the extremum at $x = 0$, whereas the relaxed nature of the PPI approach allows for a more accurate representation of the extremum. In the case of DBI, as the target polynomial degree increases from $\mathcal{P}_4$ to $\mathcal{P}_{16}$, the errors and convergence rates do not improve because the global error is dominated by the local error in the interval with the hidden extremum. The DBI approach only achieves an $O(h^{2.5})$ accuracy as opposed to the PPI method, that achieves the same high accuracy regardless of whether or not the extremal values are data points. These results highlight the advantage of the PPI method over the DBI method for recovering hidden extrema from data. Overall, the PPI method

achieves high-order accuracy when approximating the Runge functions from data with and without hidden extrema.

N	DBI		PPI	
	L^2-error	Rate	L^2-error	Rate
		$\mathcal{P}_1$		
16	2.81E-02	–	2.81E-02	–
32	6.41E-03	2.13	6.41E-03	2.13
64	1.57E-03	2.03	1.57E-03	2.03
128	3.88E-04	2.02	3.88E-04	2.02
256	9.63E-05	2.01	9.63E-05	2.01
		$\mathcal{P}_4$		
16	2.81E-02	–	1.37E-02	–
32	4.72E-03	2.57	6.85E-04	4.32
64	8.14E-04	2.54	2.57E-05	4.73
128	1.42E-04	2.52	8.32E-07	4.95
256	2.49E-05	2.51	2.60E-08	5.00
		$\mathcal{P}_8$		
16	2.74E-02	–	1.07E-02	–
32	4.69E-03	2.55	2.06E-04	5.70
64	8.14E-04	2.53	1.19E-06	7.43
128	1.42E-04	2.52	3.32E-09	8.49
256	2.49E-05	2.51	7.04E-12	8.88
		$\mathcal{P}_{16}$		
16	2.75E-02	–	1.02E-02	–
32	4.69E-03	2.55	1.43E-04	6.16
64	8.14E-04	2.53	7.18E-08	10.96
128	1.42E-04	2.52	4.74E-12	13.89
256	2.49E-05	2.51	2.77E-16	14.06

Table 9: L^2-errors and rates of convergence when using the DBI and PPI methods to approximate the function $f_1(x)$. The uniform mesh used to build the approximation is constructed with N points . The interpolants are in $\mathcal{P}_j$, where j is the target polynomial degree.

N^2	DBI		PPI	
	L^2-error	Rate	L^2-error	Rate
		$\mathcal{P}_1$		
16^2	1.97E-02	–	1.97E-02	–
32^2	4.71E-03	2.07	4.71E-03	2.07
64^2	1.16E-03	2.02	1.16E-03	2.02
128^2	2.86E-04	2.02	2.86E-04	2.02
256^2	7.11E-05	2.01	7.11E-05	2.01
		$\mathcal{P}_4$		
16^2	1.91E-02	–	7.97E-03	–
32^2	3.27E-03	2.55	3.83E-04	4.38
64^2	5.43E-04	2.59	1.41E-05	4.76
128^2	9.18E-05	2.56	4.53E-07	4.96
256^2	1.59E-05	2.53	1.41E-08	5.00
		$\mathcal{P}_8$		
16^2	1.90E-02	–	6.03E-03	–
32^2	3.27E-03	2.53	1.05E-04	5.84
64^2	5.43E-04	2.59	5.83E-07	7.49
128^2	9.18E-05	2.56	1.59E-09	8.51
256^2	1.59E-05	2.53	3.36E-12	8.89
		$\mathcal{P}_{16}$		
16^2	1.91E-02	–	6.06E-03	–
32^2	3.28E-03	2.54	8.11E-05	6.22
64^2	5.43E-04	2.60	3.19E-08	11.31
128^2	9.18E-05	2.56	2.00E-12	13.96
256^2	1.59E-05	2.53	2.83E-15	9.46

Table 10: L^2-errors and rates of convergence when using the DBI and PPI methods to approximate the function $f_7(x, y)$. The uniform mesh used to build the approximation is constructed with N^2 points. The interpolants are in $\mathcal{P}_j$, where j is the target polynomial degree.

6 Summary and Conclusions

In this paper, we present both an algorithm and theoretical foundations for sufficient conditions to ensure data boundedness and positivity on any set of mesh points via a Newton polynomial formulation. The one-dimensional PPI and DBI methods analyzed herein are building blocks that have been extended to multidimensional PPI and DBI methods using tensor-products. This extension consists of successively applying the one-dimensional PPI or DBI method on each dimension to generate the multidimensional results.

The DBI method imposes restrictions on the ratio of divided differences to ensure that the interpolants are bounded by the input data. The proof of the DBI approach presents new challenges because the configuration of mesh points may not exhibit a regular structure. The PPI method starts from the DBI method and relaxes the bounds on the ratio of divided differences, thereby allowing the interpolants to grow beyond the data as needed while remaining positive. The positive interpolant is further bounded by the parameters u_{min} and u_{max} to remove undesirable oscillations that may potentially degrade the approximation. The proofs of both the DBI and PPI approaches rely on the results from Lemma 1, which consist of using the definition of B_j^+, B_j^- to

arrive at the bounds $B_{j-1}^{-} \leq \bar{\lambda}_{j-1}\delta_j \leq B_{j-1}^{+}$. The proofs from Theorems 1 and 2 use Lemma 1 to show that $0 \leq S_n(x) \leq 1$ for the DBI method and $m_\ell \leq S_n(x) \leq m_r$ for the PPI method.

Note that one observation we have made is that the PPI method uses higher order interpolants compared to the DBI method. Relaxing the bounds on the ratio of divided differences increases the range of polynomial degrees that meet the desired requirement. The 1D and 2D numerical results, in Tables 1-8, indicate that the DBI or PPI methods provided herein are appropriate for ensuring data boundedness or positivity preservation, and both methods converge as the interpolant degree and resolution increase. Figure 1 demonstrates that enforcing positivity alone may not be sufficient to remove large oscillations. We resolve this issue by bounding the positive polynomial with u_{min} and u_{max}, which are determined based on user-supplied values, such as $\epsilon = 0.01$ for the numerical examples in Section 5. In addition, Figure 2 demonstrates that for an interval I_i where there exists a local extremum, the PCHIP and DBI methods truncate the extremum whereas the PPI method leads to a better approximation of the extremum. The different results demonstrated that the PPI method is able to produce high-order accurate approximations in examples with and without a hidden extremum.

As this work continues, we plan to investigate different methods for accelerating the algorithm. The performance optimization will focus on different strategies to enable data locality and vectorization of the PPI and DBI algorithm to better take advantage of different computational architectures. In addition, we will evaluate the use of both DBI and PPI methods for various practical applications. This work is ongoing [21].

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

References

1. Berzins, M.: Adaptive polynomial interpolation on evenly spaced meshes. SIAM Review **49**(4), 604–627 (2007). DOI https://doi.org/10.1137/050625667
2. Berzins, M.: Nonlinear data-bounded polynomial approximations and their applications in ENO methods. Numer. Algor. **55**(2), 171–189 (2010). DOI https://doi.org/10.1007/s11075-010-9395-8
3. Costantini, P.: On some recent methods for bivariate shape-preserving interpolation. In: W. Haußmann, K. Jetter (eds.) Multivariate Approximation and Interpolation: Proceedings of an International Workshop held at the University of Duisburg, August 14–18, 1989, pp. 55–68. Birkhäuser Basel, Basel (1990). DOI https://doi.org/10.1007/978-3-0348-5685-0_4
4. Costantini, P.: Boundary-valued shape-preserving interpolating splines. ACM Trans. Math. Softw. **23**(2), 229–251 (1997). DOI https://doi.org/10.1145/264029.264050
5. Dougherty, R.L., Edelman, A., Hyman, J.M.: Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic hermite interpolation. Math. Comput. **52**(186), 471–494 (1989). DOI https:doi.org/10.1090/S0025-5718-1989-0962209-1

6. Epperson, J.F.: On the runge example. Amer. Math. Monthly **94**(4), 329–341 (1987). DOI https://doi.org/10.1080/00029890.1987.12000642
7. Fjordholm, U.S., Mishra, S., Tadmor, E.: Arbitrarily high-order accurate entropy stable essentially nonoscillatory schemes for systems of conservation laws. SIAM J. Numer. Anal. **50**(2), 544–573 (2012)
8. Fjordholm, U.S., Mishra, S., Tadmor, E.: ENO reconstruction and ENO interpolation are stable. Found. Comput. Math. **13**(2), 139–159 (2013). DOI https://doi.org/10.1007/s10208-012-9117-9
9. Fritsch, F.N., Carlson, R.E.: Monotone piecewise cubic interpolation. SIAM J. Numer. Anal. **17**(2), 238–246 (1980). DOI https://doi.org/10.1137/0717021
10. Giraldo, F.X., Kelly, J.F., Constantinescu, E.M.: Implicit-explicit formulations of a three-dimensional nonhydrostatic unified model of the atmosphere (numa). SIAM J. Sci. Comput. **35**(5), B1162–B1194 (2013). DOI https://doi.org/10.1137/120876034
11. Hale, N., Townsend, A.: Fast and accurate computation of gauss–legendre and gauss–jacobi quadrature nodes and weights. SIAM J. Sci. Comput. **35**(2), A652–A674 (2013). DOI https://doi.org/10.1137/120889873
12. Harten, A.: ENO schemes with subcell resolution. J. Comput. Phys. **83**(1), 148–184 (1989). DOI https://doi.org/10.1016/0021-9991(89)90226-X
13. Harten, A.: Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. Comm. Pure Appl. Math. **48**(12), 1305–1342 (1995). DOI https://doi.org/10.1002/cpa.3160481201
14. Harten, A.: Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. Communications on Pure and Applied Mathematics **48**(12), 1305–1342 (1995). DOI https://doi.org/10.1002/cpa.3160481201
15. Harten, A., Engquist, B., Osher, S., Chakravarthy, S.R.: Uniformly high order accurate essentially non-oscillatory schemes, iii. J. Comput. Phys. **131**(1), 3 – 47 (1997). DOI https://doi.org/10.1006/jcph.1996.5632
16. Krogh, F.T.: Efficient algorithms for polynomial interpolation and numerical differentiation. Math. Comput. **24**(109), 185–190 (1970). DOI https://doi.org/10.2307/2004888
17. Light, D., Durran, D.: Preserving nonnegativity in discontinuous galerkin approximations to scalar transport via truncation and mass aware rescaling (TMAR). Mon. Weather Rev. **144**(12), 4771–4786 (2016). DOI https://doi.org/10.1175/MWR-D-16-0220.1
18. Liu, X.D., Osher, S., Chan, T.: Weighted essentially non-oscillatory schemes. J. Comput. Phys. **115**(1), 200–212 (1994). DOI https://doi.org/10.1006/jcph.1994.1187
19. Lux, T.C.H., Watson, L.T., Chang, T.H.: An algorithm for constructing monotone quintic interpolating splines. In: 2020 Spring Simulation Conference (SpringSim), pp. 1–12 (2020). DOI https://doi.org/10.22360/SpringSim.2020.HPC.003
20. Ouermi, T.A.J., Kirby, R.M., Berzins, M.: Numerical testing of a new positivity-preserving interpolation algorithm (2020). DOI https://doi.org/10.48550/arxiv.2009.08535
21. Ouermi, T.A.J., Kirby, R.M., Berzins, M.: HPPIS: A high-order positivity-preserving mapping software for structured meshes. Manuscript in preparation (20xx)
22. Rogerson, A.M., Meiburg, E.: A numerical study of the convergence properties of ENO schemes. J. Sci. Comput. **5**(2), 151–167 (1990). DOI https://doi.org/10.1007/BF01065582
23. Schmidt, J.W., Heß, W.: Positivity of cubic polynomials on intervals and positive spline interpolation. BIT Numer. Math. **28**(2), 340–352 (1988). DOI https://doi.org/10.1007/BF01934097
24. Sekora, M., Colella, P.: Extremum-preserving limiters for muscl and ppm (2009). DOI https://doi.org/10.48550/arXiv.0903.4200
25. Shen, C., Qiu, J.M., Christlieb, A.: Adaptive mesh refinement based on high order finite difference WENO scheme for multi-scale simulations. J. Comput. Phys. **230**(10), 3780–3802 (2011). DOI https://doi.org/10.1016/j.jcp.2011.02.008
26. Shu, C.W.: Numerical experiments on the accuracy of ENO and modified ENO schemes. J. Sci. Comput. **5**(2), 127–149 (1990). DOI https://doi.org/10.1007/BF01065581
27. Shu, C.W.: Essentially non-oscillatory and weighted essentially non-oscillatory schemes. Acta Numer. **29**, 701762 (2020). DOI https://doi.org/10.1017/S0962492920000057

28. Skamarock, W.C., Weisman, M.L.: The Impact of Positive-Definite Moisture Transport on NWP Precipitation Forecasts. Mon. Weather Rev. **137**(1), 488–494 (2009). DOI https://doi.org/10.1175/2008MWR2583.1
29. Tadmor, E.: Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. Acta Numer. **12**, 451512 (2003). DOI https://doi.org/10.1017/S0962492902000156
30. Tal-Ezer, H.: High degree polynomial interpolation in Newton form. SIAM J. Sci. Statist. Comput. **12**(3), 648–667 (1991). DOI https://doi.org/10.1137/0912034
31. Ulrich, G., Watson, L.T.: Positivity conditions for quartic polynomials. SIAM J. Sci. Comput. **15**(3), 528–544 (1994). DOI https://doi.org/10.1137/0915035
32. Viner, K., Reinecke, P., Doyle, J., Gabersek, S., Martini, M., Flagg, D., Michalakes, J., Ryglicki, D., Giraldo, F.: Next generation NWP using a spectral element dynamical core. AGU Fall Meeting Abstracts A34A-02 (2016)
33. Wang, C., Dong, X., Shu, C.W.: Parallel adaptive mesh refinement method based on WENO finite difference scheme for the simulation of multi-dimensional detonation. J. Comput. Phys. **298**, 161–175 (2015). DOI https://doi.org/10.1016/j.jcp.2015.06.001
34. Zala, V., Kirby, M., Narayan, A.: Structure-preserving function approximation via convex optimization. SIAM J. Sci. Comput. **42**(5), A3006–A3029 (2020). DOI https://doi.org/10.1137/19M130128X
35. Zala, V., Kirby, R.M., Narayan, A.: Structure-preserving nonlinear filtering for continuous and discontinuous galerkin spectral/hp element methods. SIAM J. Sci. Comput. **43**(6), A3713–A3732 (2021). DOI https://doi.org/10.1137/20M1337223
36. Zhang, X.: On positivity-preserving high order discontinuous galerkin schemes for compressible navier–stokes equations. J. Comput. Phys. **328**, 301 – 343 (2017). DOI https://doi.org/10.1016/j.jcp.2016.10.002
37. Zhang, X., Shu, C.W.: Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments. Proc. Math. Phys. Eng. Sci. **467**(2134), 2752–2776 (2011). DOI https://doi.org/10.1098/rspa.2011.0153
38. Zhang, X., Shu, C.W.: Positivity-preserving high order finite difference WENO schemes for compressible euler equations. J. Comput. Phys. **231**(5), 2245–2258 (2012). DOI https://doi.org/10.1016/j.jcp.2011.11.020
39. Zhang, X., Xia, Y., Shu, C.W.: Maximum-principle-satisfying and positivity-preserving high order discontinuous galerkin schemes for conservation laws on triangular meshes. J. Sci. Comput. **50**(1), 29–62 (2012). DOI https://doi.org/10.1007/s10915-011-9472-8