Algorithm xxxx: HiPPIS A High-Order Positivity-Preserving Mapping Software for Structured Meshes

TIMBWOGA A. J. OUERMI, ROBERT M. KIRBY, and MARTIN BERZINS, University of Utah Scientific Computing Imaging Institute, USA

Polynomial interpolation is an important component of many computational problems. In several of these computational problems, failure to preserve positivity when using polynomials to approximate or map data values between meshes can lead to negative unphysical quantities. Currently, most polynomial-based methods for enforcing positivity are based on splines and polynomial rescaling. The spline-based approaches build interpolants that are positive over the intervals in which they are defined and may require solving a minimization problem and/or system of equations. The linear polynomial rescaling methods allow for high-degree polynomials but enforce positivity only at limited locations (e.g., quadrature nodes). This work introduces open-source software (HiPPIS) for high-order data-bounded interpolation (DBI) and positivity-preserving interpolation (PPI) that addresses the limitations of both the spline and polynomial rescaling methods. HiPPIS is suitable for approximating and mapping physical quantities such as mass, density, and concentration between meshes while preserving positivity. This work provides Fortran and Matlab implementations of the DBI and PPI methods, presents an analysis of the mapping error in the context of PDEs, and uses several 1D and 2D numerical examples to demonstrate the benefits and limitations of HiPPIS.

CCS Concepts: • **Mathematics of computing → Computations on polynomials**.

Additional Key Words and Phrases: positivity-preserving, data-bounded, polynomial interpolation, vectorization

ACM Reference Format:
Timbwoga A. J. Ouermi, Robert M. Kirby, and Martin Berzins. 2023. Algorithm xxxx: HiPPIS A High-Order Positivity-Preserving Mapping Software for Structured Meshes. 1, 1 (October 2023), 31 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Mapping data values from one grid to another is a fundamental part of many computational problems. Preserving certain properties such as positivity when interpolating solution values between meshes is important. In many applications [1, 24, 33, 40, 41, 45], failure to preserve the positivity of quantities such as mass, density, and concentration results in negative values that are unphysical. These negative values may propagate to other calculations and corrupt other quantities. Many polynomial-based methods have been developed to address these limitations.

Positivity-preserving methods based on linear polynomial rescaling are introduced in [15, 23, 24, 45, 46]. These polynomial rescaling methods are often used in the context of hyperbolic PDEs, in numerical weather prediction (NWP) [23], combustion simulation [15, 24], and other applications. These methods introduce rescaling parameters obtained from quadrature weights that are used to linearly rescale the polynomial to ensure positivity at the quadrature

Authors' address: Timbwoga A. J. Ouermi, touermi@cs.utah.edu; Robert M. Kirby, kirby@cs.utah.edu; Martin Berzins, mb@sci.utah.edu, University of Utah Scientific Computing Imaging Institute, 72 Central campus Drive, Salt Lake City, Utah, USA, 84112.

nodes and conserve mass. These approaches ensure positivity only at the set of mesh points used for the simulation but do not address the case of mapping data values between different meshes, which is the focus of HiPPIS.

Other approaches for preserving positivity that are based on splines can be found in computer-aided design (CAD), graphics, and visualization [10, 18, 20, 34–36]. Several positivity- and monotonicity-preserving cubic splines have been developed. A widely used example of such an approach is the piecewise cubic Hermite interpolation (PCHIP) [10], which is available as open-source code in [27]. In addition, quartic and quintic spline-based approaches have been introduced in [14, 16, 17, 25, 26]. These methods impose some restrictions on the first and second derivatives to ensure monotonicity, positivity, and continuity. For instance, the monotonic quintic spline interpolation (MQSI) methods in [26] and [25] use the sufficient conditions stated in [36] and [44] to check for monotonicity and iteratively adjust the first and second derivative values to enforce monotonicity.

Positivity can also be enforced using ENO-type methods [2, 3, 29, 33], which enforce data-boundedness and positivity by adaptively selecting mesh points to build the stencil used to construct the positive interpolant for each interval. ENO-type methods use divided differences to develop a sufficient condition for data-boundedness or positivity that is used to guide the stencil selection process. The software introduced in this work is based on the high-order ENO-type data-bounded interpolation (DBI) and positivity-preserving interpolation (PPI) methods in [29]. The work in [29] provides a positivity-preserving method that uses higher degree polynomials compared to the other ENO-type methods in [2, 3, 33] and the spline-based methods.

Given that polynomial interpolation is well-established and widely used, several implementations of the different polynomial approximation algorithms are available. For example, FunC by Green et al. [12] uses polynomial interpolation with a lookup table for faster approximation of a given function compared to direct evaluation. However, most of these implementations, including FunC, do not preserve data-boundedness and positivity. The implementations available for positivity preservation are based on splines [10, 14] and polynomial rescaling [23, 45]. The spline-based approaches often require solving a linear system of equations to ensure continuity and an optimization problem in the case of quartic and quintic splines. These spline approaches are often limited to fifth-order polynomials and can be computationally expensive in cases where solving a global optimization problem is required. A full suite of test problems comparing the DBI and PPI methods against different spline-based methods including PCHIP [10], MQSI [26], and shape-preserving splines (SPS) [6] has been undertaken by the authors in [28]. The different polynomial rescaling methods allow for polynomial degrees higher than five and are built as part of larger partial differential equation (PDE) solvers [23, 45]. As previously mentioned, the polynomial rescaling approaches guarantee positivity only at a given set of points, not over the entire domain. The present work provides an implementation of a high-order software (HiPPIS) based on [29] that guarantees positivity over the entire domain where the interpolant is defined. In addition, this work evaluates the use of HiPPIS in the context of function approximation and mapping between different meshes. This evaluation provides an analysis of the mapping error in the case of PDEs and numerical examples demonstrating the benefits and limitations of HiPPIS.

The remaining parts of the paper are organized as follows: Section 2 presents the background for the mathematical framework required for the DBI and PPI methods. Section 3 provides the algorithms used to build the software, and the descriptions of the different components of HiPPIS. Section 4 provides several 1D and 2D numerical examples, while Section 5 conducts an analysis and evaluation of the mapping error in the context of time-dependent PDEs. A discussion and concluding remarks are presented in Section 6.

2 MATHEMATICAL FRAMEWORK

This section provides a summary and the theoretical background of both the DBI and PPI methods.

2.1 Adaptive Polynomial Construction

Both the DBI and PPI methods rely on the Newton polynomial [21, 43] representation to build interpolants that are positive or bounded by the data values. The ability to adaptively choose stencil points to construct the interpolation, as in ENO methods [13], is the key feature employed to develop the data-bounded and positivity-preserving interpolants.

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 Equation (1). The subscripts J, L, $i, \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$, and x_k, $u_k \in \mathbb{R}$ for $i - J \leq k \leq i + L$. The DBI and PPI procedure starts by setting the initial stencil $\mathcal{V}_0$,

$$\mathcal{V}_0 = \{x_i, x_{i+1}\} = \{x_0^l, x_0^r\}. \tag{2}$$

The stencil $\mathcal{V}_0$ in Equation (2) is expanded by successively appending a point to the right or left of $\mathcal{V}_j$ to form $\mathcal{V}_{j+1}$. Once the final stencil $\mathcal{V}_{n-1}$ is obtained, the interpolant of degree n defined on $I_i = \{x_i, x_{i+1}\}$ can be written as

$$U_n(x) = \quad u_i + U[x_0^l, x_0^r]\pi_{0,i}(x) + U[x_1^l, \cdots, x_1^r]\pi_{1,i}(x) + \cdots + U[x_{n-1}^l, \cdots, x_{n-1}^r]\pi_{n-1,i}(x), \tag{3}$$

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

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

The divided differences are recursively defined as follows:

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

The polynomial $U_n(x)$ can be compactly expressed as

$$U_n(x) = u_i + (u_{i+1} - u_i)S_n(x). \tag{4}$$

$S_n(x)$ in Equation (4) is defined as

$$S_n(x) = s\left(1 + \frac{(s-1)}{d_1}\lambda_1\left(1 + \frac{(s-t_2)}{d_2}\lambda_2\left(\cdots\left(1 + \frac{(s-t_{n-1})}{d_{n-1}}\lambda_{n-1}\right)\cdots\right)\right)\right), \tag{5}$$

where s, t_j, and d_j are expressed as follows:

$$0 \leq s = \frac{x - x_i}{x_{i+1} - x_i} = \frac{x - x_0^e}{x_0^r - x_0^l} \leq 1, \tag{6}$$

$$t_j = -\frac{x_i - x_j^e}{x_0^r - x_0^l}, \text{ and} \tag{7}$$

$$0 \leq d_j = \frac{x_j^r - x_j^l}{x_0^r - x_0^l}. \tag{8}$$

s and d_j in Equations (5), (6), and (8) are defined such that $s \in [0,1]$ and $d_j \geq 0$. The positivity-preserving and data-bounded interpolants are obtained by imposing some bounds on $\bar{\lambda}_j$, defined as

$$\bar{\lambda}_j = \prod_{k=1}^{j} \lambda_j = \lambda_j \bar{\lambda}_{j-1} = \prod_{k=1}^{j} \lambda_k = \begin{cases} 1 \quad j=0 \\ \frac{U[x_j^l,\cdots,x_j^r]}{U[x_0^l,x_0^r]} \prod_{k=1}^{j}(x_k^r - x_k^l), \quad 1 \leq j \leq n-1. \end{cases} \tag{9}$$

2.2 Positivity-Preserving and Data-Bounded Interpolation

For a given interval inside a mesh, the DBI and PPI polynomial interpolant is constructed by adaptively selecting points near the target interval to build an interpolation stencil and polynomial that together meet the requirements for positivity or data-boundedness. Requiring positivity alone can lead to large oscillations and extrema that degrade the approximation. Positivity alone does not restrict how much the interpolant is allowed to grow beyond the data values. The large oscillations can be removed with the PCHIP, MQSI, and DBI methods. However, in the case where a given interval I_i has a hidden extremum, PCHIP, MQSI, and DBI will truncate the extremum. As in [3, 37], the interval I_i has an extremum when two of the three divided differences $\sigma_{i-1} = U[x_{i-1}, x_i]$, $\sigma_i = U[x_i, x_{i+1}]$, and $\sigma_{i+1} = U[x_{i+1}, x_{i+2}]$ of neighboring intervals are of opposite signs. The constrained PPI algorithm addresses these limitations by allowing the constructed interpolant to grow beyond the data values but not produce extrema that are too large.

The positive polynomial interpolant is constrained as follows:

$$u_{min} \leq U^p(x) = u_i + (u_{i+1} - u_i)S_n(x) \leq u_{max}. \tag{10}$$

The bounds u_{min} and u_{max} in Equation (10) are defined as

$$\begin{cases} u_{min} = min(u_i, u_{i+1}) - \Delta_{min}, \\ u_{max} = max(u_i, u_{i+1}) + \Delta_{max}. \end{cases} \tag{11}$$

The parameters Δ_{min} and Δ_{max} in Equation (11) are positive, and the data-bounded interpolant is obtained for Δ_{min}=Δ_{max} = 0.0. These parameters are chosen according to

$$\Delta_{min} = \begin{cases} \epsilon_1 \big|min(u_i, u_{i+1})\big| & \text{if } \sigma_{i-1}\sigma_{i+1} < 0 \text{ and } \sigma_{i-1} < 0 \text{ or } \sigma_{i-1}\sigma_{i+1} \geq 0 \text{ and } \sigma_{i-1}\sigma_i < 0 \\ \epsilon_0 \big|min(u_i, u_{i+1})\big| & \text{otherwise,} \end{cases} \tag{12}$$

and

$$\Delta_{max} = \begin{cases} \epsilon_1 \big|max(u_i, u_{i+1})\big| & \text{if } \sigma_{i-1}\sigma_{i+1} < 0 \text{ and } \sigma_{i-1} > 0 \text{ or } \sigma_{i-1}\sigma_{i+1} \geq 0 \text{ and } \sigma_{i-1}\sigma_i < 0 \\ \epsilon_0 \big|max(u_i, u_{i+1})\big| & \text{otherwise.} \end{cases} \tag{13}$$

The positive parameters ϵ_0 and ϵ_1, used for intervals with and without extrema, respectively, are introduced to adjust Δ_{min} and Δ_{max}. This work extends the bounds in [29] by introducing the parameter ϵ_1 to allow for more flexibility on how to bound the interpolants in cases where an extremum is detected. The choice for the positive parameters ϵ_0 and ϵ_1 depends on the underlying function and the input data used for the approximation. As both ϵ_0 and ϵ_1 get smaller, the upper and lower bounds get tighter and the PPI method converges to the DBI method. The choices for ϵ_0 and ϵ_1 are further discussed in Section 3.2. In Equation (12), the interval I_i has a local maximum if $\sigma_{i-1}\sigma_{i+1} < 0$ and $\sigma_{i-1} < 0$.

Correspondingly, in Equation (13), the interval I_i has a local minimum if $\sigma_{i-1}\sigma_{i+1} < 0$ and $\sigma_{i-1} > 0$. In both Equations (12) and (13), the type of extremum is ambiguous if $\sigma_{i-1}\sigma_{i+1}$, and $\sigma_{i-1}\sigma_i < 0$.

Equation (10) is equivalent to bounding $S_n(x)$ as follows:

$$m_\ell \le S_n(x) \le m_r, \tag{14}$$

where the factors m_ℓ and m_r in Equation (14) are expressed as

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

$$m_\ell = min\left(0, \frac{u_{min} - u_i}{u_{i+1} - u_i}\right), \text{ and } m_r = max\left(1, \frac{u_{max} - u_i}{u_{i+1} - u_i}\right) \tag{15}$$

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

$$m_\ell = min\left(0, \frac{u_{max} - u_i}{u_{i+1} - u_i}\right), \text{ and } m_r = max\left(1, \frac{u_{min} - u_i}{u_{i+1} - u_i}\right). \tag{16}$$

The DBI method can be recovered from the PPI methods by setting $m_\ell = 0$ and $m_r = 1$. For $u_i = u_{i+1}$, m_ℓ, m_r and $U_n(x)$ as written in Equations (15), (16), and (4) are not defined. This limitation is addressed by re-writing $U_n(x)$ as

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

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

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

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

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

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

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

For $u_i = u_{i+1}$, the parameters m_ℓ and m_r in Equation (14) are then defined according to

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

$$m_\ell = min\left(0, \frac{u_{min} - u_i}{w}\right), \text{ and } m_r = max\left(1, \frac{u_{max} - u_i}{w}\right) \tag{17}$$

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

$$m_\ell = min\left(0, \frac{u_{max} - u_i}{w}\right), \text{ and } m_r = max\left(1, \frac{u_{min} - u_i}{w}\right). \tag{18}$$

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.

The positivity-preserving result in Equation (10) is obtained by successively imposing bounds on the quadratic, cubic, and higher order terms in the expression of $S_n(x)$ defined in Equation (5). The reconstruction procedure begins by considering the linear and quadratic terms from $S_n(x)$ in Equation (5) and imposing the following bounds:

$$m_\ell \le s + \frac{s(s-1)}{d_1}\bar{\lambda}_1 \le m_r. \tag{19}$$

Equation (19) can be reorganized to obtain

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

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

$$\left(-4(m_r-1)-1\right)d_1 \le \bar{\lambda}_1 \le \left(-4m_\ell+1\right)d_1. \tag{20}$$

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

$$\left(-4(m_r-1)-1\right)d_1 \le \bar{\lambda}_1\left(1+\frac{(s-t_2)}{d_2}\lambda_2\right) \le d_1\left(-4m_\ell+1\right).$$

When t_2 defined in Equation (7) 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

$$\frac{d_2}{(1-t_2)}\left(\left(-4(m_r-1)-1\right)d_1-\bar{\lambda}_1\right) \le \bar{\lambda}_2 \le \left(d_1\left(-4m_\ell+1\right)-\bar{\lambda}_1\right)\frac{d_2}{(1-t_2)}.$$

When t_2 is positive, $\frac{1}{1-t_2}$ is substituted by $\frac{1}{-t_2}$ and the inequalities $\le$ with $\ge$ and vice versa are swapped.

This procedure is continued to quartic and higher order interpolants to produce the recursive expression for the bounds on $\bar{\lambda}_j$ for the PPI and DBI methods as follows:

$$B_j^- = \begin{cases} (-4(m_r-1)-1)d_1 & j=1 \\ (B_{j-1}^- - \bar{\lambda}_{j-1})\frac{d_j}{1-t_j}, \text{ if } t_j \in (-\infty,0] & j>1 \\ (B_{j-1}^+ - \bar{\lambda}_{j-1})\frac{d_j}{-t_j}, \text{ if } t_j \in (0,+\infty) & j>1, \end{cases} \tag{21a}$$

and

$$B_j^+ = \begin{cases} (-4m_\ell+1)d_1, & j=1 \\ (B_{j-1}^+ - \bar{\lambda}_{j-1})\frac{d_j}{1-t_j}, \text{ if } t_j \in (-\infty,0] & j>1 \\ (B_{j-1}^- - \bar{\lambda}_{j-1})\frac{d_j}{-t_j}, \text{ if } t_j \in (0,+\infty) & j>1. \end{cases} \tag{21b}$$

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. We refer the reader to Theorems 1 and 2 in [29] for more details on the mathematical foundation used to build the positivity-preserving software.

3 ALGORITHMS AND SOFTWARE

This section describes the algorithms and different components used in the data-bounded and positivity-preserving software. The software developed in this work provides 1D, 2D, and 3D implementations of the DBI and PPI methods for uniform and nonuniform structured meshes. The 1D implementation is constructed based on the mathematical framework provided in Section 2. The 2D and 3D implementations are obtained via a tensor product of the 1D version.

3.1 Algorithms

The algorithms provide the necessary elements to construct the data-bounded or positive interpolants. Rogerson and Meiburg [31] showed that the ENO reconstruction can lead to a left- or right-biased stencil that causes stability issues when used to solve hyperbolic equations. Shu [38] addressed this limitation by introducing a bias coefficient used to target a preferred stencil. As indicated in [29], the left- and right-biased stencil can fail to recover hidden extrema. For a given interval I_i, the left- and right-biased stencil does not include the points x_{i-1} or x_{i+1}, respectively. **Algorithm I** addresses these limitations by extending the algorithm in [29] to introduce more options for the adaptive stencil selection process described below. In addition to the symmetry-based points selection in [29], **Algorithm I** includes ENO-type and locality-based point selection processes.

At any given step j, the next point inserted into $\mathcal{V}_j$ can be to the left or right. Let x_p and x_q be the mesh points immediately to the left and right of $\mathcal{V}_j$, respectively.

We define $\bar{\lambda}^-_{j+1}$ and $\bar{\lambda}^+_{j+1}$ as follows:

$$\begin{cases} \bar{\lambda}^-_{j+1} = \bar{\lambda}_{j+1} & \text{with } \mathcal{V}_{j+1} = \{x_p\} \cup \mathcal{V}_j \\ \bar{\lambda}^+_{j+1} = \bar{\lambda}_{j+1} & \text{with } \mathcal{V}_{j+1} = \mathcal{V}_j \cup \{x_q\}. \end{cases} \tag{22}$$

The terms $\bar{\lambda}^-_{j+1}$ and $\bar{\lambda}^+_{j+1}$ in Equation 22 correspond to the case where the stencil inserted is to the left and right, respectively. 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. **Algorithm I** extends the algorithm in [29] by introducing a user-supplied parameter st used to guide the procedure for stencil construction. In the cases where adding both x_p (to the left) or x_q (to the right) are valid, the algorithm makes the selection based on the three cases below:

- If $st = 1$ (default), 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 the 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 the right and left are the same, the algorithm chooses the point with the smallest $\bar{\lambda}_{j+1}$.
- If $st = 3$, the algorithm chooses the point that is closest to the starting interval I_i. It is important to prioritize the closest points in cases where the intervals surrounding I_i vary significantly in size. These variations are found in computational problems for which different resolutions are used for different parts of the domain.

Algorithm II describes the 1D DBI and PPI methods built using the mathematical framework in Section 2 and **Algorithm I**. **Algorithm II** further extends the constraints in [29] by introducing the user-supplied positive parameter ϵ_1 that is used to impose upper and lower bounds on the interpolants according to Equations (12) and (13). The positive parameters ϵ_0 and ϵ_1 are used for intervals without and with an extremum, respectively. The user-supplied parameter im is used to choose between the DBI and PPI methods. **Algorithm III** and **Algorithm IV** describe the extension from 1D to 2D and 3D, respectively. Both **Algorithm III** and **IV** are constructed by successively applying **Algorithm II** to each dimension. Given that the DBI and PPI methods are nonlinear, the order in which **Algorithm II** is used can lead to different approximation results. In this paper and in [29], the 1D DBI and PPI are first applied to x, then the y dimension, and finally the z dimensions, as indicated in **Algorithm III** and **IV**. In **Algorithm III** and **IV**, the input and intermediate data values are modified only by **Algorithm I**, which preserves data-boundedness or positivity. Therefore, the resulting solutions from the 2D and 3D extensions will preserve data-boundedness and positivity. Similar to **Algorithm II**, the choices of parameters st, ϵ_0, and ϵ_1 influence the quality of the approximation in **Algorithm III** and **IV**. In the 1D, 2D, and 3D cases, the choice for parameters st, ϵ_0, and ϵ_1 is dependent on the data. In the case of 2D and 3D, applying the 1D PPI method (**Algorithm II**) along the x and/or y dimensions may introduce oscillations that could be amplified when applying the 1D PPI in the subsequent dimensions. These oscillations can be significantly reduced with small values for ϵ_0 and ϵ_1. The parameters ϵ_0 and ϵ_1 should be chosen to be small enough such that hidden extrema are recovered without introducing new large oscillations. In **Algorithm II** and **IV**, $\mathcal{M}_{\square x \times \square y}$ and $\mathcal{M}_{\square x \times \square y \square z}$ represent 2D and 3D meshes obtained by taking the tensor product of the 1D mesh along the x and y dimensions for the 2D mesh, and along the x, y, and z dimensions for the 3D mesh. The square $\square$ represents n or m.

Algorithm I

Input: μ_j^l, μ_j^l, x_p, x_i, x_q, x_{i+1}, $U[x_p, \cdots, x_j^r]$, $U[x_j^l, \cdots, x_q]$ $\bar{\lambda}_{j+1}^-$, $\bar{\lambda}_{j+1}^+$, and st.

(1) if $st = 1$
- if $|U[x_p \cdots, x_j^r]| < |U[x_j^l, \cdots, x_q]|$, then insert a new stencil point to the left;
- else if $|U[x_p \cdots, x_j^r]| > |U[x_j^l, \cdots, x_q]|$, then insert a new stencil point to the right;
- else insert a new stencil point to the right if $|\bar{\lambda}_{j+1}^-| \geq |\bar{\lambda}_{j+1}^+|$; otherwise, insert a new point to left;

(2) if $st = 2$
- 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 $|\bar{\lambda}_{j+1}^-| \geq |\bar{\lambda}_{j+1}^+|$; otherwise, insert a new point to left;

(3) else $st = 3$
- if $|x_p - x_i| < |x_q - x_{i+1}|$, then insert a new stencil point to the left;
- else if $|x_p - x_i| > |x_q - x_{i+1}|$, then insert a new stencil point to the right;
- else insert a new stencil point to the right if $|\bar{\lambda}_{j+1}^-| \geq |\bar{\lambda}_{j+1}^+|$; otherwise, insert a new point to left;

Algorithm II (1D)

Input: $\{x_i\}_{i=0}^n$, $\{u_i\}_{i=0}^n$, $\{\tilde{x}_i\}_{i=0}^{\tilde{n}}$, d, st ϵ_0, im, and ϵ_1. **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 (11), (12), and (13).
(4) Compute m_r and m_ℓ based on Equations (15) and (16) or Equations (17) and (18). For DBI, set $m_r = 1$ and $m_\ell = 0$.
(5) Given a stencil $\mathcal{V}_j$,
- if $B_{j+1}^- \leq \bar{\lambda}_{j+1}^+ \leq B_{j+1}^+$ and $B_{j+1}^- \leq \bar{\lambda}_{j+1}^- \leq B_{j+1}^+$, choose the point to add based on **Algorithm I**
- else if $B_{j+1}^- \leq \bar{\lambda}_{j+1}^- \leq B_{j+1}^+$, then insert a new stencil point to the left;
- else if $B_{j+1}^- \leq \bar{\lambda}_{j+1}^+ \leq B_{j+1}^+$, 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.

Algorithm III (2D)

Input:$\{x_i\}_{i=0}^{nx}$,$\{y_i\}_{i=0}^{ny}$ $\{u_{i,j}\}_{i,j=0}^{nx,ny}$, d, st ϵ_0, $\{\tilde{x}_i\}_{i=0}^{mx}$, $\{\tilde{y}_j\}_{j=0}^{my}$, im, and ϵ_1. **Output:** $\{\tilde{u}_{i,j}\}_{i,j=0}^{mx,my}$.

(1) Interpolate from the mesh $\mathcal{M}_{nx \times ny}$ to $\mathcal{M}_{mx \times ny}$ by applying the **Algorithm II (1D)** along the x dimension for each index j along the y direction. More precisely, for each index j $(0 \leq j \leq nx)$, **Algorithm II (1D)** uses $\{x_i\}_{i=0}^{nx}$, $\{u_{i,j}\}_{i=0}^{nx}$ to interpolate from $\{x_i, y_j\}_{i=0}^{nx}$ to $\{\tilde{x}_i, y_j\}_{i=0}^{mx}$. The interpolation solution is saved in $\{q_{i,j}\}_{i,j=0}^{mx,ny}$
(2) Use the solution$\{q_{i,j}\}_{i,j=0}^{mx,ny}$ from step (1) to interpolate from $\mathcal{M}_{mx \times ny}$ to the final output mesh $\mathcal{M}_{mx \times my}$ by applying the **Algorithm II (1D)** along the y dimension for each index i along the x direction. For each index

i $(0 < i < mx)$, **Algorithm II (1D)** uses $(\{y_j\}_{j=0}^{ny}, \{q_{i,j}\}_{j=0}^{ny})$ to interpolate from $\{\tilde{x}_i, y_j\}_{j=0}^{ny}$ to $\{\tilde{x}_i, \tilde{y}_j\}_{j=0}^{my}$. The interpolation final results are saved in $\{\tilde{u}_{i,j}\}_{i,j}^{mx,my}$

Algorithm IV (3D)

Input:$\{x_i\}_{i=0}^{nx}$,$\{y_i\}_{i=0}^{ny}$, $\{z_k\}_{k=0}^{nz}$, $\{u_{i,j,k}\}_{i,j,k=0}^{nx,ny,nz}$, d, st ϵ_0, $\{\tilde{x}_i\}_{i=0}^{mx}$, $\{\tilde{y}_j\}_{j=0}^{my}$, $\{\tilde{z}_k\}_{k=0}^{mz}$, im, and ϵ_1. **Output:** $\{\tilde{u}_{i,j,k}\}_{i,j=0}^{mx,my,mz}$.

(1) Interpolate from the mesh $\mathcal{M}_{nx\times ny\times nz}$ to $\mathcal{M}_{mx\times ny\times nz}$ by applying the **Algorithm II (1D)** along the x dimension for each pair j, k. More precisely, for each each pair j, K $(0 \le j \le nx$ and $0 \le k \le nz)$, **Algorithm II (1D)** uses $\{x_i\}_{i=0}^{nx}$, $\{u_{i,j,k}\}_{i=0}^{nx}$ to interpolate from $\{x_i, y_j, z_k\}_{i=0}^{nx}$ to $\{\tilde{x}_i, y_j, z_k\}_{i=0}^{mx}$. The interpolation solution is saved in $\{q_{i,j,k}\}_{i,j,k=0}^{mx,ny,nz}$

(2) Use the solution$\{q_{i,j,k}\}_{i,j,k=0}^{mx,ny,nz}$ from step (1) to interpolate from $\mathcal{M}_{mx\times ny\times nz}$ to $\mathcal{M}_{mx\times my\times nz}$ by applying the **Algorithm II (1D)** along the y dimension. For each pair i, k $(0 \le i \le mx$ and $0 \le k \le nz)$, **Algorithm II (1D)** uses $(\{y_j\}_{j=0}^{ny}, \{q_{i,j,k}\}_{j=0}^{ny})$ to interpolate from $\{\tilde{x}_i, y_j, z_k\}_{j=0}^{ny}$ to $\{\tilde{x}_i, \tilde{y}_j, z_k\}_{j=0}^{my}$. The interpolation results are saved in $\{\tilde{g}_{i,j,k}\}_{i,j}^{mx,my,nz}$.

(3) Use the solution$\{q_{i,j,k}\}_{i,j,k=0}^{mx,ny,nz}$ from step (1) to interpolate from $\mathcal{M}_{mx\times ny\times nz}$ to $\mathcal{M}_{mx\times my\times nz}$ by applying the **Algorithm II (1D)** along the y dimension. For each pair i, j $(0 \le i \le mx$ and $0 \le j \le my)$, **Algorithm II (1D)** uses $(\{z_k\}_{k=0}^{nz}, \{g_{i,j,k}\}_{k=0}^{nz})$ to interpolate from $\{\tilde{x}_i, \tilde{y}_j, z_k\}_{k=0}^{nz}$ to $\{\tilde{x}_i, \tilde{y}_j, \tilde{z}_k\}_{k=0}^{mz}$. The interpolation final results are saved in $\{\tilde{u}_{i,j,k}\}_{i,j}^{mx,my,mz}$.

3.2 Software Description

The DBI and PPI software implementation is guided by the algorithms described above. HiPPIS is available at https://github.com/ouermijudicael/HiPPIS. The software can be organized into four major parts: (1) computation of divided differences, (2) calculations of upper and lower bounds for each interval, (3) a stencil construction procedure, and (4) 1D, 2D, and 3D DBI and PPI implementations.

The divided differences are essential to the DBI and PPI methods because they are used in the calculations of $\bar{\lambda}_j$ and the stencil selection process. The divided differences are computed using the standard recurrence form in Equation (2.1) and stored in a table of dimension $n \times (d + 1)$ where d is the maximum polynomial degree for each interpolant. Given that the maximum degree is d, it is sufficient to consider the $d + 1$ divided differences for the stencil selection process and the construction of the final polynomial interpolant for each interval.

The bounds on each interpolant are obtained from Equation (11), (12), and (13) where the positive parameters ϵ_0 and ϵ_1 are user-supplied values used to adjust the bounds for the interval with and without extremum, respectively. The adjustment focuses on removing large oscillations as much as possible while still allowing high-degree polynomial interpolants that meet the positivity requirements.

The stencil selection process requires the computation of B_j^+ and B_j^-, which are both dependent on d_j, t_j, and $\bar{\lambda}_j$. The stencil $\mathcal{V}_j$ is constructed from $\mathcal{V}_{j-1}$ by appending a point to the left or right of $\mathcal{V}_{j-1}$. When appending to either the right or left meets the requirements for positivity, the software offers three possible options for choosing from both points that can be set by the user. The first and default option $(st = 1)$ chooses the stencil with the smallest divided difference, similar to the ENO-like approach. The second option $(st = 2)$ prioritizes the choice that makes the stencil more symmetric around x_i. The third option $(st = 3)$ chooses the point closest to the starting interval I_i, thus prioritizing locality.

The 1D DBI and PPI methods use (1), (2), and (3) as building blocks to construct the final approximation. Once the final stencil has been selected, the interpolant is built using a Newton polynomial representation and then evaluated at the corresponding output points. The Newton polynomial is used here because its coefficients/divided differences are available. The 2D and 3D implementations successively use the 1D version along each dimension to construct the final approximation on uniform and nonuniform structured meshes.

The interfaces for the 1D, 2D, and 3D DBI and PPI subroutines are designed to be similar to widely used interfaces for polynomial interpolation such as PCHIP, and can be incorporated into larger application codes. The interfaces require

- the input mesh points and the data values associated with those points,
- the maximum polynomial degree to be used for each interpolant,
- the interpolation method to be used (DBI or PPI), and
- the output mesh points.

Listing 1 shows examples of how to use the 1D, 2D, and 3D interfaces for DBI and PPI in HiPPIS. The variables *x, y,* and *z* are 1D arrays used to define the input meshes, and *xout, yout,* and *zout* are used to define the output meshes. The variables *v, v2D,* and *v3D* correspond to the input data values associated with the input meshes. The parameters d and im (1, or 2) are used to indicate the target polynomial degree and the interpolation method to be used. For DBI and PPI, the parameter im is set to 1 and 2, respectively. The parameters st, ϵ_0, and ϵ_1 are optional parameters that are set to 3, 0.01, and 1 by default, as explained below. The choice of the optional parameters depends on the underlying function and the input data.

In problems for which different resolutions are used for different parts of the computational domain, *st=3* is a preferable choice. The algorithm prioritizes the closest points to the starting interval I_i if *st=3.* This choice is particularly important in regions where the size of the intervals varies significantly. For cases when smoothness is the primary goal, *st=1* is a suitable choice. For *st=1*, the **Algorithm I** prioritizes smoothness by choosing the points with the smallest divided differences during the stencil construction process. Both the *st=1* and *st=3* can lead to a left- or right-biased stencil. In these instances, *st=2* can be used to remove the bias. For *st=2*, the algorithm prioritizes a symmetric stencil. The default value of *st* is set to 3 because the examples in this study indicate that st=3 leads to better approximations compared to st=1 or 2 and locality is often a highly desired property in many computational problems.

The positive parameters ϵ_0 and ϵ_1 are used to bound the interpolants for the intervals with and without extrema, respectively. The configurations in [29] and [28] correspond to setting the parameters ϵ_0 and ϵ_1 to the default values of 0.01 and 1, respectively. The values of ϵ_0 and ϵ_1 are chosen such that the lower and upper bounds on each interpolant are relaxed enough to allow for a high-order polynomial that does not introduce undesirable oscillations. For profiles that are prone to oscillation such as the logistic functions, it is important to choose small values for ϵ_0 and ϵ_1. For $N \times N = 17 \times 17$, the approximation leads to large oscillations if ϵ_0 and ϵ_1 are greater than 10^{-4}. For intervals without extrema, it is important to keep ϵ_0 small to not introduce new extrema. For the intervals with extrema, ϵ_1 needs to be large enough to allow for recovery of hidden extrema but small enough to not cause undesired large oscillations. This is very challenging given that the sizes of the peaks are not known a priori. The default values of $\epsilon_1 = 1$ are such that the interpolant maximum value is twice $max(u_i, u_{I+1})$. This default value of one is sufficient for the modified Runge and TWP-ICE examples. However, in the case of BOMEX, smaller values of $\epsilon_1 \leq 10^{-5}$ are required to remove undesired oscillations. In practice, it is prudent to start with a small value for ϵ_0 and ϵ_1 and increase them as needed if the approximation fails to recover hidden extrema or uses low-degree polynomial interpolants.

Figure 1 is a diagram of the different components of the main module of HiPPIS. The function *divdiff(...)* is used to calculate the divided differences needed for **Algorithms I** and **II**. Once the final stencil is constructed, the function *newtonPolyVal(...)* is used to build and evaluate the positive interpolant at the corresponding output points. The major part of the data-boundedness and positivity preservation including **Algorithms I** and **II** is in the function *adaptiveInterpolation1D(...)*. This function is used for the 1D approximation or mapping problems and depends on the function *divdiff(...)* and *newtonPolyVal(...)*. The functions *adaptiveIterpolation2D(...)* and *adaptiveInterpolation3D(...)* use *adaptiveInterpolation1D(..)* to construct the data-bounded or positive polynomial approximations on 2D and 3D structured tensor product meshes, respectively. The interfaces for the 1D, 2D, and 3D interpolations, in bold, require the parameter *im*, which is used to indicate the interpolation method chosen. For the DBI and PPI methods, the parameter *im* is set using 1 and 2, respectively. HiPPIS does not allow for any other choices for the parameter *im*.

```
% 1D example
vout = adaptiveInterpolation1D(x, v, xout, d, im, st, ϵ0, ϵ1 );

%2D example
vout2D = adaptiveInterpolation2D(x, y, v2D, xout,yout, d, im, st, ϵ0, ϵ1 );

%3D example
vout3D = adaptiveInterpolation3D(x, y, z, v3D, xout, yout, zout, d, im, st, ϵ0, ϵ1 );

```

Listing 1. Interface examples

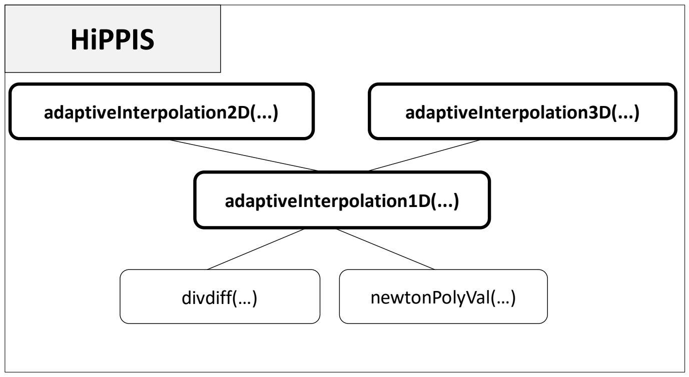

HiPPIS
adaptiveInterpolation2D(...)
adaptiveInterpolation3D(...)
adaptiveInterpolation1D(...)
divdiff(...)
newtonPolyVal(...)

Fig. 1. Diagram showing the components of the main module used to build the HiPPIS software.

4 NUMERICAL EXAMPLES FOR FUNCTION APPROXIMATION

This section provides 1D and 2D numerical examples used to evaluate the PCHIP, MQSI, DBI, and PPI methods. The results are based on the Fortran implementation of the software. These examples include a subset of the full suite of test problems considered in [28]. The interpolation methods are used to approximate positive functions from provided data values that are obtained by evaluating the 1D and 2D functions on a given set of mesh points. The function approximations in 2D examples use the 2D extension of the DBI and PPI methods described in **Algorithm III**. The 2D PCHIP and MQSI methods are obtained by successively applying the 1D PCHIP and MQSI algorithms to each dimension, similar to the steps described in **Algorithm III**. The 1D PCHIP and MQSI are first applied to x, then the y dimension, and finally the z dimension. Similar to 2D and 3D DBI and PPI, the order in which 1D PCHIP or MQSI is applied can lead to different results, as both methods are nonlinear and non-commutative. Using a standard polynomial interpolation to approximate the different functions leads to negative values and oscillations. In this section, the L^2-norms in the tables below are approximated using the trapezoidal rule with, 10^4 and $10^3 \times 10^3$ uniformly spaced points for the 1D and 2D examples, respectively. For the numerical examples in Sections 4.1 - 5.3, the errors from using $st = 1, 2,$ and 3 are similar, with $st = 3$ leading to slightly smaller errors compared to $st = 1$ and $st = 2$. Given that the results are similar, Tables 1 - 8 show errors with the parameter st set to 3. For the BOMEX example, the errors from the three choices are significantly different. Therefore, the results from all three choices are included. More test examples can be found in [28].

4.1 Example I: Modified Runge Function

This example uses a modified version of the canonical Runge function defined as

$$f_1(x) = \frac{0.1}{0.1 + 25x^2}, \quad x \in [-1, 1]. \tag{23}$$

Approximating the modified Runge function in Equation (23) with a global standard polynomial leads to large oscillations. Table 1 shows the L^2-errors norms when using the PCHIP, MQSI, DBI, and PPI methods to approximate $f_1(x)$. The DBI and PPI methods lead to better approximation results compared to the PCHIP and MQSI methods. The errors from PCHIP and MQSI are comparable. In this example, the algorithms used in MQSI to approximate and adjust the derivatives to enforce monotonicity do not produce more accurate results compared to PCHIP. As the target polynomial degree increases from $d = 4$ to $d = 8$, the DBI approximation does not improve significantly compared to the PPI method. The relaxed nature of the PPI method allows for higher degree polynomial interpolants compared to DBI, PCHIP, and MQSI which lead to better approximations.

N	PCHIP	MQSI	DBI			PPI		
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$
17	3.99E-2	3.63E-2	5.10E-2	2.91E-2	4.61E-2	5.10E-2	2.91E-2	4.61E-2
33	4.52E-3	4.32E-3	6.31E-3	9.57E-3	3.05E-3	6.31E-3	9.57E-3	3.05E-3
65	2.79E-3	2.67E-3	2.44E-3	2.49E-3	1.33E-3	2.44E-3	2.49E-3	9.92E-4
129	6.23E-4	6.71E-4	2.22E-4	1.21E-4	1.05E-4	2.22E-4	1.21E-4	2.43E-5
257	1.17E-4	9.89E-5	1.51E-5	1.15E-5	1.07E-5	1.51E-5	4.68E-6	9.89E-8

Table 1. L^2-errors when using the PCHIP, MQSI, DBI, and PPI methods to approximate the function $f_1(x)$. N represents the number of input points used to build the approximation. The parameters ϵ_0, ϵ_1, and st are set to 0.01, 1.0, and 3, respectively.

4.2 Example II: 1D Logistic Function

This test case uses a logistic function defined as

$$f_2(x) = \frac{1}{1 + e^{-2kx}}, \quad k = 100, \text{ and } x \in [-0.2, 0.2]. \tag{24}$$

This function is a smoothed analytical approximation of the Heaviside step function. The logistic function in Equation (24) is challenging because of the steep gradient at about $x = 0$. Approximating $f_2(x)$ with a standard polynomial interpolation leads to large oscillations to the left and right of the gradient. In addition, the oscillations to the left produce negative values.

Table 2 shows L^2-error norms when using the PCHIP, MQSI, DBI, and PPI methods to approximate the smoothed logistic function $f_2(x)$. The MQSI method has larger errors compared to the other methods. In this case, the algorithms employed by MQSI to approximate and adjust derivatives values used to construct the monotonic quintic splines are less accurate than the one used in PCHIP. For a target polynomial degree $d = 3$, the approximation errors using PCHIP, DBI, and PPI are comparable. Increasing the target polynomial degree improves the approximations for DBI and PPI, as shown in Table 2. The errors from both the DBI and PPI methods are similar because the logistic example has no hidden extrema, and the stencils used for both methods are the same, around $x = 0$. The global error is dominated by the local errors in the region with steep gradients around $x = 0$.

Figure 2 shows approximation plots of $f_2(x)$ using $N = 17$ uniformly spaced points with different values of ϵ_0 and $\epsilon_1 = 1$. The target polynomial degree is set to $d = 8$. For $\epsilon_0 = 1$, we observe oscillations, as shown in the right part of Figure 2. As ϵ_0 decreases, the oscillations decrease. For $\epsilon_0 \leq 0.01$, the errors and oscillations are negligible compared to errors in the region with the steep gradient. The oscillations are completely removed for $\epsilon_0 = 0.0$.

N	PCHIP	MQSI	DBI			PPI		
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$
17	2.02E-2	1.92E-2	2.41E-2	2.41E-2	2.08E-2	2.41E-2	2.41E-2	2.08E-2
33	3.38E-3	3.72E-3	4.89E-3	4.86E-3	3.59E-3	4.90E-3	4.86E-3	3.57E-3
65	3.59E-4	1.61E-3	4.17E-4	1.89E-4	1.47E-4	4.17E-4	1.89E-4	1.47E-4
129	4.21E-5	1.71E-4	3.09E-5	1.55E-5	1.70E-6	3.09E-5	1.55E-5	1.70E-6
257	5.12E-6	1.75E-5	2.04E-6	5.31E-7	5.22E-9	2.04E-6	5.31E-7	5.22E-9

Table 2. L^2-errors when using the PCHIP, MQSI , DBI, and PPI methods to approximate the function $f_2(x)$. N represents the number of input points used to build the approximation. The parameters ϵ_0, ϵ_1, and st are set to 0.01, 1, and 3, respectively.

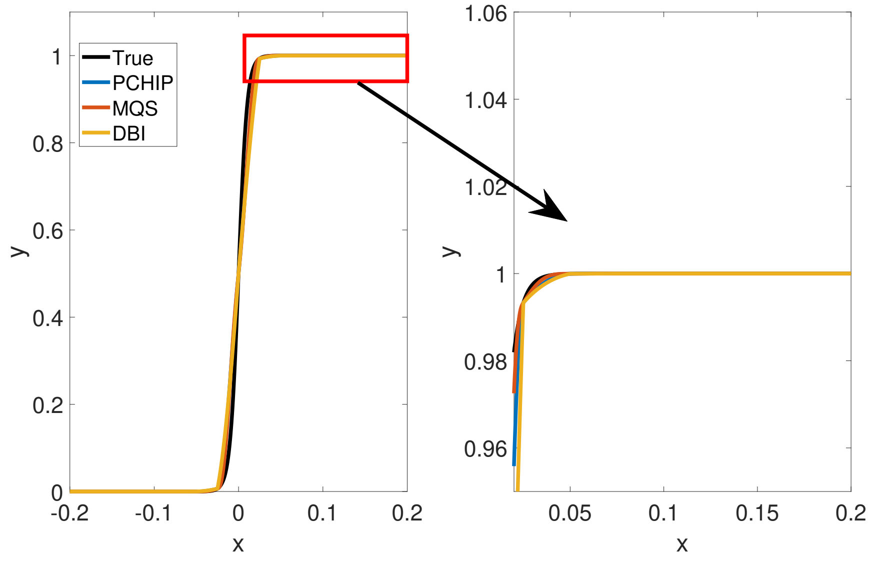
4.3 Example III: 1D Discontinuous Function

This example uses a modified version of a function introduced by Tadmor and Tanner [42] and used by Berzins [2] in the context of bounded interpolation methods. The value one is added to the original function in [42] to ensure that the function is positive over the interval [-1,1]. The modified function is defined as

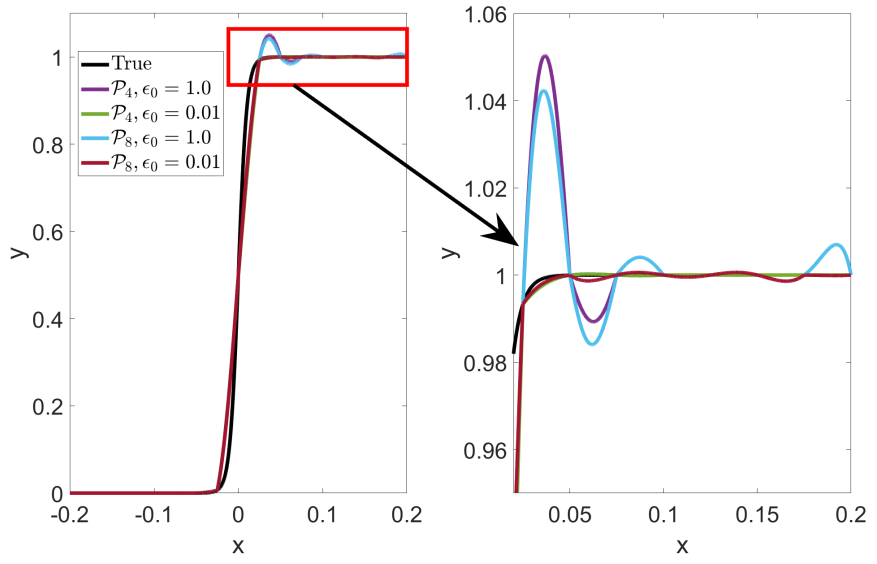
$$f_3(x) = \begin{cases} 1 + \frac{2e^{2\pi x} - 1 - e^{\pi}}{e^{\pi} - 1}, & x \in [-1, -0.5) \\ \\ 1 - sin\left(\frac{2\pi x}{3} + \frac{\pi}{3}\right), & x \in [-0.5, 1]. \end{cases} \tag{25}$$

Table 3 shows L^2-error norms when using the PCHIP, MQSI, DBI, and PPI methods to approximate the function in Equation (25). Approximating $f_3(x)$ is challenging because $f_3(x)$ is a piecewise function with a discontinuity at $x = 0.5$. The global error is dominated by the local errors around the discontinuity. The PCHIP, MQSI, DBI, and PPI approximation results are comparable. Increasing the target polynomial degree does not decrease the L^2-error norms. The approximations in the smooth regions improve as we increase the target polynomial degree, but the global error is dominated by the error around the discontinuity. The error around the discontinuity does not decrease with higher polynomial degrees.

Figure 3 shows approximation plots of $f_3(x)$ using $N = 17$ uniformly spaced points with different values of ϵ_0. The target polynomial degree is set to $d = 4, 8$. The bottom right part of Figure 3 shows oscillations at the left boundary



(a) PCHIP, MQSI, and DBI with $d = 8$.



(b) PPI with $d = 4$ and $d = 8$.

Fig. 2. Approximation of $f_2(x)$ from $N = 17$ uniformly spaced points with different interpolation methods. The top row (Figure 2a) shows approximation results using PCHIP, MQSI, and DBI. The bottom row (Figure 2b) shows approximation results using PPI with $d = 4, 8$ and $\epsilon_0 = 1.0, 0.01$. An enlarged version of the region in the red rectangle is shown on the right of each row. The value of ϵ_1 is set to 1.0.

for $\epsilon_0 = 1$. The oscillations are removed for $\epsilon_0 \leq 0.1$. As expected, all the interpolation methods have difficulties approximating the function around the discontinuity, as shown in Figure 3.

N	PCHIP	MQSI		DBI			PPI	
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$
17	1.77E-1	1.76E-1	1.82E-1	1.83E-1	1.82E-1	1.73E-1	1.72E-1	1.70E-1
33	1.39E-1	1.41E-1	1.35E-1	1.39E-1	1.36E-1	1.35E-1	1.39E-1	1.36E-1
65	1.03E-1	1.06E-1	9.95E-2	1.04E-1	1.02E-1	9.95E-2	1.04E-1	1.02E-1
129	7.42E-2	7.63E-2	7.12E-2	7.54E-2	7.35E-2	7.15E-2	7.55E-2	7.38E-2
257	5.28E-2	5.43E-2	5.06E-2	5.38E-2	5.24E-2	5.07E-2	5.39E-2	5.26E-2

Table 3. L^2-errors when using the PCHIP, MQSI DBI, and PPI methods to approximate the function $f_3(x)$. N represents the number of input points used to build the approximation. The parameters ϵ_0, ϵ_1, and st are set to 0.01, 1, and 3, respectively.

4.4 Example IV: 2D Modified Runge Function

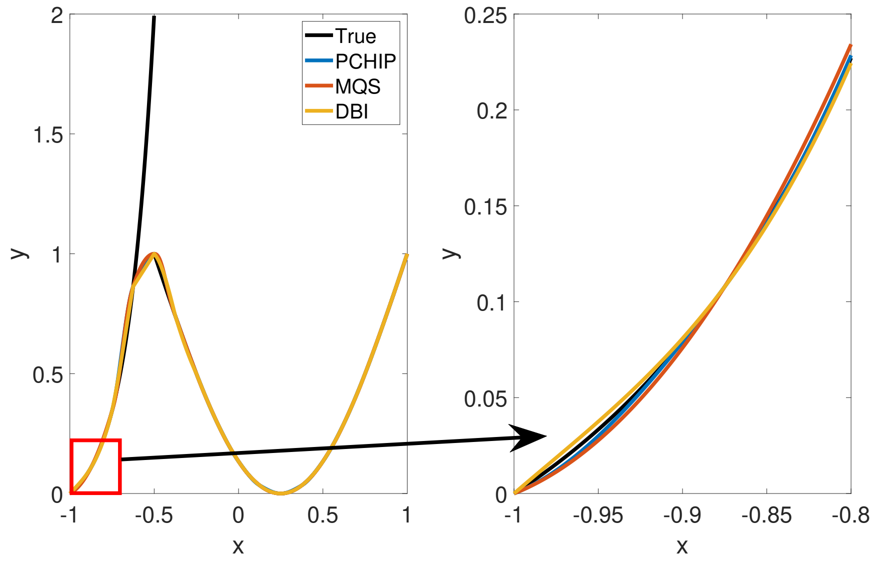
This example extends the previously modified 1D Runge function to 2D as follows:

$$f_4(x,y) = \frac{0.1}{0.1 + 25(x^2 + y^2)}, \quad x, y \in [-1, 1]. \tag{26}$$

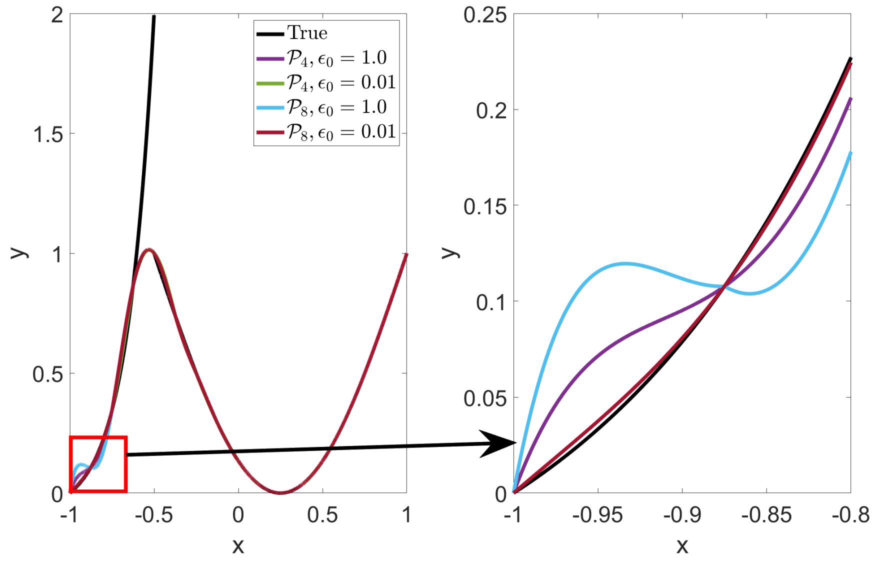
Table 4 shows L^2-error norms when using the PCHIP, MQSI, DBI, and PPI methods to approximate the 2D modified Runge function in Equation (26). In this example, as in Example I, the MQSI and PCHIP methods have comparable errors. The approaches used to approximate the derivatives and construct the quintic splines are not more accurate than the approximation obtained from PCHIP. The DBI and PPI methods with $d = 3$ lead to better approximation results compared to the PCHIP and MQSI. As the target polynomial degree d increases, the approximation errors from PPI decrease much faster than from DBI. The relaxed nature of the PPI methods allows for higher degree polynomials compared to DBI. The bounds for data-boundedness, which are based on Equation (11) with $\Delta_{min} = \Delta_{max} = 0.0$, are more restrictive than positivity for which $\Delta_{min} > 0.0$ and $\Delta_{max} > 0.0$. In addition, the approximation does not lead to oscillations for ϵ_0 and $\epsilon_1 \in [0, 1]$

N^2	PCHIP	MQSI		DBI			PPI	
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$
17	1.76E-2	1.67E-2	2.12E-2	9.09E-3	1.91E-2	2.12E-2	9.09E-3	1.91E-2
33	2.05E-3	2.84E-3	2.45E-3	4.61E-3	1.25E-3	2.45E-3	4.61E-3	1.24E-3
65	1.05E-3	8.01E-4	8.59E-4	9.33E-4	4.99E-4	8.59E-4	9.33E-4	3.51E-4
129	2.23E-4	2.57E-4	7.47E-5	4.76E-5	4.12E-5	7.47E-5	4.64E-5	7.16E-6
257	4.19E-5	3.53E-5	5.05E-6	4.20E-6	3.80E-6	5.05E-6	1.62E-6	2.91E-8

Table 4. L^2-errors when using the PCHIP, MQSI, DBI, and PPI methods to approximate the function $f_4(x, y)$. N represents the number of input points used to build the approximation. The parameters ϵ_0, ϵ_1, and st are set to 0.01, 1, and 3, respectively.



(a) PCHIP, MQSI, and DBI with $d = 8$.



(b) PPI with $d = 4$ and $d = 8$.

Fig. 3. Approximation of $f_3(x)$ from $N = 17$ uniformly spaced points with different interpolation methods. The top row (Figure 3a) shows approximation results using PCHIP, MQSI, and DBI. The bottom row (Figure 3b) shows approximation results using PPI with $d = 4, 8$ and $\epsilon_0 = 1.0, 0.01$. An enlarged version of the region in the red rectangle is shown on the right of each row. The value of ϵ_1 is set to 1.0.

4.5 Example V: 2D Logistic Function

The test case extends the 1D logistic (smoothed Heaviside) function from Example II to 2D.

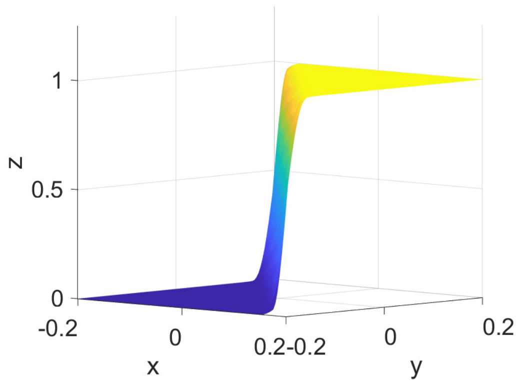
$$f_5(x, y) = \frac{1}{1 + e^{-\sqrt{2}k(x+y)}}, \quad x, y \in [-0.2, 0.2]. \tag{27}$$

The function $f_5(x, y)$ is challenging because of the large gradient at $y = -x$. Approximating $f_5(x, y)$ with a standard polynomial interpolation leads to oscillations and negative values that violate the desired property of positivity.

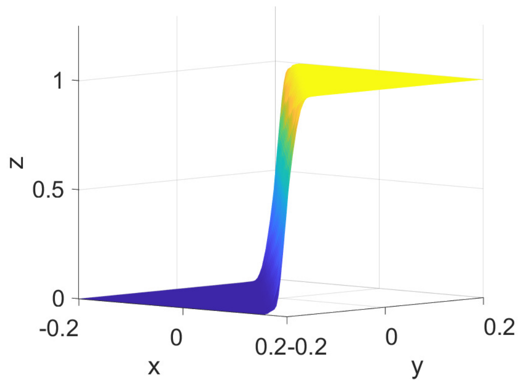
Table 5 shows L^2-error norms when using the PCHIP, MQSI, DBI, and PPI methods to approximate the 2D logistic function $f_5(x, y)$ defined in Equation (27). The errors from MQSI are larger compared to the other methods. In this example, the approximated derivatives and quintic splines from MQSI are less accurate than the approximations from PCHIP. The DBI and PPI methods lead to better approximation results compared to the PCHIP and MQSI approaches. Increasing the target polynomial degree improves the approximations for DBI and PPI, as shown in Table 5. The global error is dominated by the local around the steep gradients at $y = -x$. For $N^2 = 33^2$ points and above with $\epsilon_0 = 0.01$, $\epsilon_1 = 1.0$, and $st = 3$, the DBI and PPI choose the identical stencils for the intervals around the steep gradient. Therefore, the global error, which is dominated by the local error around the steep gradient, is the same for both DBI and PPI. However, for $N^2 = 17^2$ points with $\epsilon_0 = 0.01$, $\epsilon_1 = 1.0$, and $st = 3$, the DBI and PPI choose different stencils that lead to different errors, as indicated in the row with $N^2 = 17^2$ of Table 5. Figure 4 shows approximation plots of $f_5(x, y)$ using $N \times N = 17 \times 17$ uniformly spaced points with PCHIP, MQSI, and PPI. The approximations in Figures 4c and 4e are obtained using PPI with $\epsilon_0 = \epsilon_1 = 1$. The PPI method with $\epsilon_0 = \epsilon_1 = 10^{-4}$ is used for the approximations in Figures 4d and 4f. The target polynomial degrees for the second and third rows are set to $d = 4$ and $d = 8$, respectively. The oscillations increase when going from $\mathcal{P}_4$ to $\mathcal{P}_8$ with $\epsilon_1 = \epsilon_2 = 1.0$. For $\epsilon_0 = \epsilon_1 = 10^{-4}$, the oscillations are significantly reduced, and the approximation is closer to the target solution.

N^2	PCHIP	MQSI	DBI			PPI		
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$
17	8.07E-3	7.12E-3	1.05E-2	9.79E-3	8.18E-3	1.05E-2	9.77E-3	8.61E-3
33	1.26E-3	2.62E-3	1.67E-3	1.36E-3	1.06E-3	1.64E-3	1.30E-3	8.87E-4
65	1.44E-4	6.35E-4	1.58E-4	8.84E-5	4.89E-5	1.58E-4	8.84E-5	5.01E-5
129	1.63E-5	6.51E-5	1.13E-5	3.07E-6	2.64E-7	1.13E-5	3.07E-6	2.64E-7
257	1.94E-6	6.74E-6	7.29E-7	1.02E-7	5.39E-10	7.29E-7	1.02E-7	5.39E-10

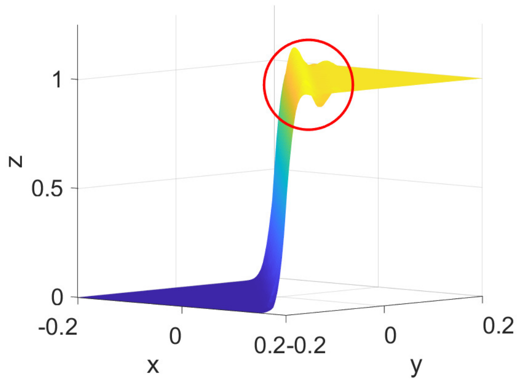
Table 5. L^2-errors when using the PCHIP, MQSI, DBI, and PPI methods to approximate the function $f_5(x, y)$. N represents the number of input points used to build the approximation. The parameters ϵ_0, ϵ_1, and st are set to 0.01, 1, and 3, respectively.



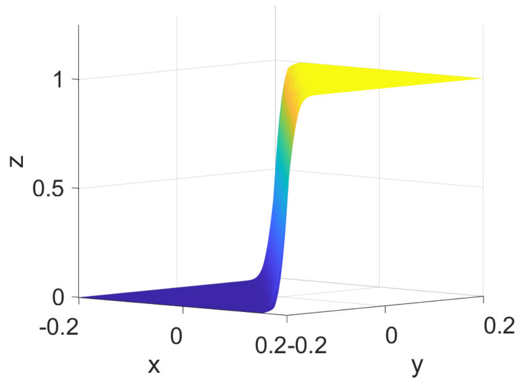
(a) PCHIP



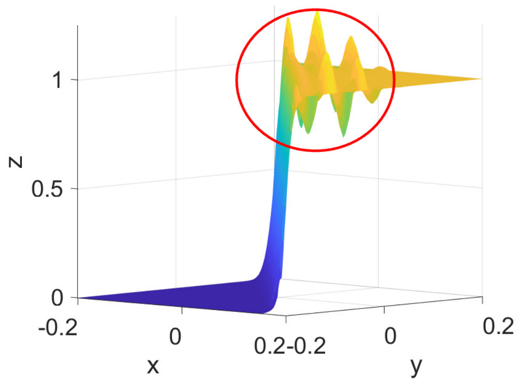
(b) MQSI



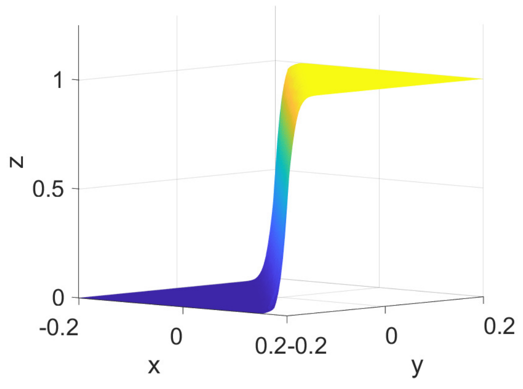
(c) PPI with $\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 1.0$



(d) PPI with $\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 10^{-4}$



(e) PPI with $\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 1.0$



(f) PPI with $\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 10^{-4}$

Fig. 4. Approximation of $f_5(x, y)$ from $N \times N = 17^2$ uniformly spaced points with different interpolation methods. The parameter st is set to 2. The red ellipses highlight examples regions with large oscillations. The surfaces in the top row Figures 4a and 4b are obtained using PCHIP and MQSI. Figures 4c ($\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 1.0$), 4d ($\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 10^{-4}$), 4e ($\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 1.0$), and 4f ($\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 10^{-4}$) are obtained using PPI.

4.6 Example VI: C^0-continuous Surface Function

This example is based on a 2D function used to study positive and monotonic splines [5, 30]. The function is defined as follows:

$$f_6(x,y) = \begin{cases} 2(y-x) & \text{if } 0 \le y-x \le 0.5 \\ 1 & \text{if } y-x \ge 0.5 \\ cos\left(4\pi\sqrt{(x-1.5)^2+(y-0.5)^2}\right) & \text{if } (x-1.5)^2+(y-0.5)^2 \le \frac{1}{16} \\ 0 & otherwise. \end{cases} \tag{28}$$

The function $f_6(x,y)$ is challenging because it is only $\mathbb{C}^0$-continuous at various locations.

Table 6 shows L^2-error norms when using the PCHIP, MQSI, DBI, and PPI methods to approximate the 2D function $f_6(x,y)$ defined in Equation (28). The PCHIP, MQSI, DBI, and PPI methods lead to comparable L^2-error norms. Increasing the target polynomial degree does not significantly improve the approximation for DBI and PPI, as shown in Table 6. The global error is dominated by the local around the $\mathbb{C}^0$. The approximation for both DBI and PPI can be improved by using an underlying mesh that better captures the $\mathbb{C}^0$-continuity.

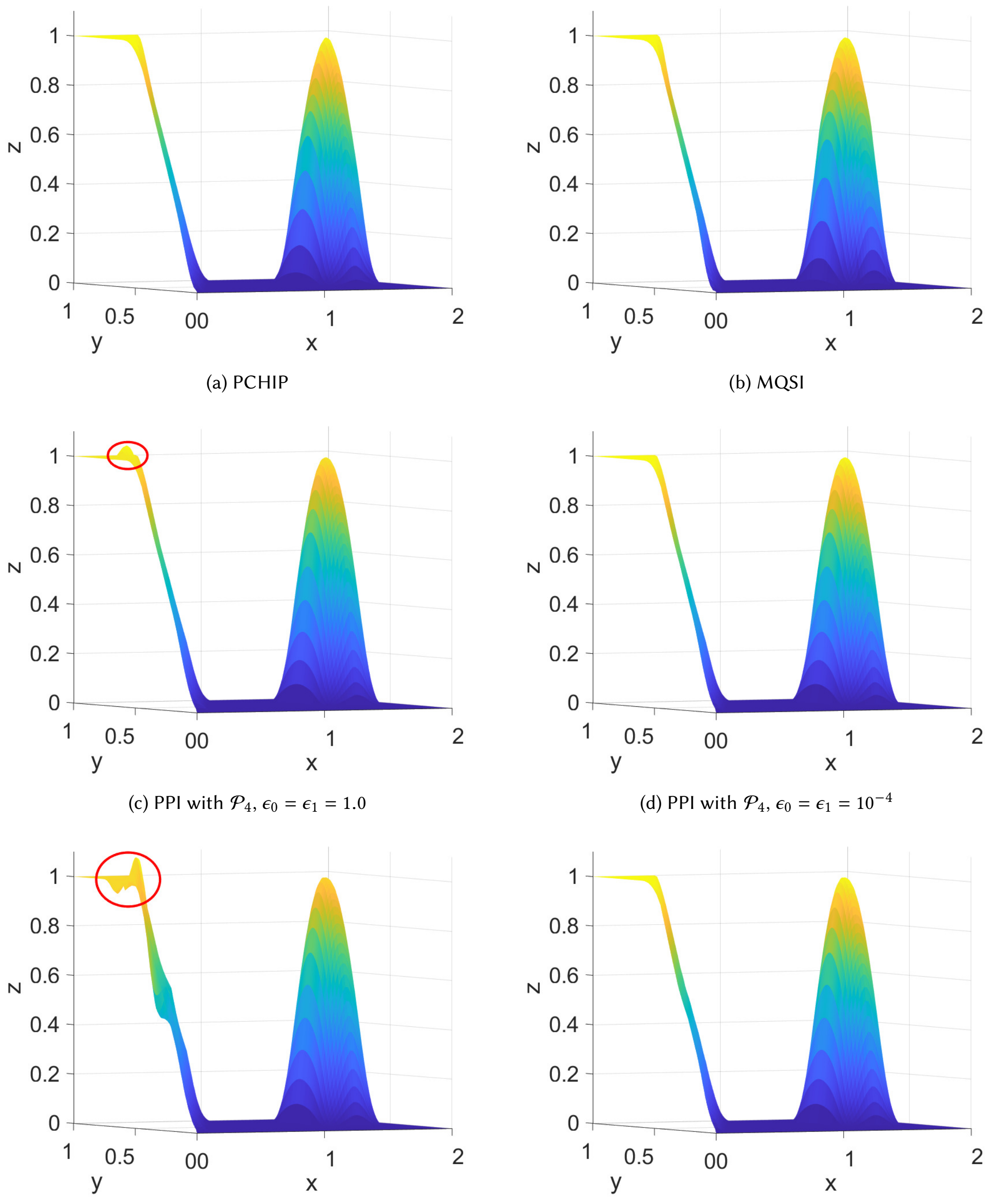
Figure 5 shows approximation plots of $f_6(x,y)$ using $N \times N = 17 \times 17$ uniformly spaced points with PCHIP, MQSI, and PPI. The left and right approximation plots show approximated solutions using the PPI method. For the left plot using PPI, $\epsilon_0 = \epsilon_1 = 1.0$, and for the right plot using PPI, $\epsilon_0 = \epsilon_1 = 10^{-4}$ The target polynomial degree is set to $d = 4$ and $d = 8$ for the second and third rows, respectively. For $\epsilon_0 = \epsilon_1 = 1.0$, the oscillations increase as the target polynomial degree increases from $d = 4$ to $d = 8$. The oscillations observed for $\epsilon_1 = \epsilon_0 = 1$ are removed for small values of ϵ_0 and ϵ_1, shown in Figures 5d and 5f.

N^2	PCHIP	MQSI	DBI			PPI		
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_8$
17	1.91E-2	2.19E-2	1.72E-2	1.69E-2	1.63E-2	1.72E-2	1.68E-2	1.59E-2
33	6.92E-3	7.44E-3	6.16E-3	5.81E-3	5.88E-3	6.16E-3	5.80E-3	5.87E-3
65	2.47E-3	2.45E-3	2.24E-3	2.14E-3	2.11E-3	2.24E-3	2.14E-3	2.11E-3
129	8.99E-4	8.69E-4	8.21E-4	7.77E-4	7.63E-4	8.20E-4	7.77E-4	7.63E-4
257	3.23E-4	3.04E-4	2.97E-4	2.81E-4	2.76E-4	2.96E-4	2.81E-4	2.76E-4

Table 6. L^2-errors when using the PCHIP, MQSI, DBI, and PPI methods to approximate the function $f_6(x,y)$. N represents the number of input points used to build the approximation. The parameters ϵ_0, ϵ_1, and st are set to 0.01, 1, and 3, respectively.

5 MAPPING ERROR ANALYSIS AND EXAMPLES

This section provides an analysis of the mapping error between two different meshes in the context of time-dependent PDEs when DBI and PPI are employed to interpolate data values between the meshes. In addition to the error analysis, the Runge, tropical warm pool international cloud experiment (TWP-ICE), and Barbados oceanographic and meteorological experiment (BOMEX) examples are used to evaluate the use of the PPI and DBI to map data values between two meshes. The Runge and TWP-ICE examples use meshes that emulate the advection and reaction meshes used in NEPTUNE. These meshes are constructed by linearly scaling the NEPTUNE vertical mesh points to the desired interval for the



(a) PCHIP

(b) MQSI

(c) PPI with $\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 1.0$

(d) PPI with $\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 10^{-4}$

(e) PPI with $\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 1.0$

(f) PPI with $\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 10^{-4}$

Fig. 5. Approximation of $f_5(x, y)$ from $N \times N = 17^2$ uniformly spaced points with different interpolation methods. The parameter st is set to 2. The red ellipses highlight examples regions with large oscillations. The surfaces in the top row Figures 5a and 5b are obtained using PCHIP and MQSI. Figures 5c ($\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 1.0$), 5d ($\mathcal{P}_4$, $\epsilon_0 = \epsilon_1 = 10^{-4}$), 5e ($\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 1.0$), and 5f ($\mathcal{P}_8$, $\epsilon_0 = \epsilon_1 = 10^{-4}$) are obtained using PPI.

Runge and TWP-ICE examples. In the BOMEX example, the advection mesh is composed of uniformly spaced points, and the reaction mesh is constructed using the mid-point of each interval from the advection mesh.

5.1 Mapping Error Analysis

In addition to the development and study of the DBI and PPI methods, it is important to provide some insight into the behavior of the mapping error in the context of time-dependent PDEs. An example of a time-dependent problem where a positivity-preserving mapping is required is the US Navy Environmental Prediction System Using the NUMA Core (NEPTUNE) [19]. NEPTUNE is a next-generation global NWP system being developed at the Naval Research Laboratory (NRL) and the Naval Postgraduate School (NPS). In NEPTUNE, the physics and dynamics are calculated using different meshes and require mapping the solution values between both meshes. NEPTUNE uses nonuniform structured meshes that have vertical columns with nonuniformly spaced points inside each column. The mapping must preserve positivity for quantities such as density and cloud water mixing ratio. The cloud water mixing ratio is the amount of cloud water in air. At each time step, the dynamics (advection) solutions, which are calculated on the advection mesh, are mapped to the reaction mesh to be used as input for the physics calculations. The physics results are then mapped back to the dynamics to be used as input for the next time step. Enforcing positivity alone may still lead to large oscillations and approximation errors. Using the DBI method will remove the large oscillations but will truncate any hidden extremum and may be too restrictive for high-order accuracy in some cases. For simulations in which different structured meshes are used and mapping is required, the errors from both the DBI and unconstrained PPI will propagate into other calculations and may even cause the simulation to fail. This section provides an analysis of the mapping error when interpolating from one mesh to another and back to the starting mesh. The mapping error is considered within time-dependent PDEs. For example, when interpolating the data values between the advection and reaction mesh in NEPTUNE, a mapping error is introduced in addition to the physics and time integration errors. The error in approximating a function $u(x)$ with the Newton polynomial $U_n(x)$ over the interval I_i is

$$E_n(x) = u(x) - U_n(x) = \frac{u^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^{n} (x - x_k^e), \quad x \in I_i \tag{29}$$

where $\xi \in [x_n^l, x_n^r]$. Given that ξ and $u^{(n+1)}$ are not known, the local interpolation error in Equation (29) can approximated as follows:

$$\tilde{E}_n = U[x_n^l \cdots x_n^r] \prod_{k=0}^{n} \Delta x_k, \tag{30}$$

where

$$\Delta x_k = max\Big(|x(i) - x_k^e|, |x(i+1) - x_k^e|\Big).$$

The error approximation in Equation (30) is based on the mean value theorem for divided differences, which states that there exist $\xi_0 \in [x_n^l, x_n^r]$ such that

$$U[x_n^l \cdots x_n^r] = \frac{u^{(n+1)}(\xi_0)}{(n+1)!}.$$

Equation (30) approximates the local interpolation error for each interval when mapping from one set of points to another. To consider a mapping error for interpolating from one mesh to another and back to the starting mesh, let $\mathcal{M}_A$ and $\mathcal{M}_R$ be the advection and reaction mesh, respectively. In addition, let I_{AR} and I_{RA} be the interpolation operators that map a given set of data values from $\mathcal{M}_A$ to $\mathcal{M}_R$ and from $\mathcal{M}_R$ to $\mathcal{M}_A$, respectively. We consider an advection-reaction problem where the advection part is calculated on $\mathcal{M}_A$ and the reaction on $\mathcal{M}_R$. A simple forward Euler time integration is used. Let $\bar{u}_\tau$ and $\hat{u}_\tau$ be the approximate and the exact solution at time τ. The dynamics/advection part is written as

$$\bar{u}^1_{\tau+\Delta\tau} = \bar{u}_\tau + \Delta\tau F\big(\bar{u}_\tau\big), \tag{31}$$

and the physics/reaction $\bar{w}_{\tau+\Delta\tau}$ is expressed as

$$\bar{w}_{\tau+\Delta\tau} = H\bar{u}^1_{\tau+\Delta\tau}, \tag{32}$$

where $H\bar{u}^1_{\tau+\Delta\tau} = I_{AR}G(I_{RA}\bar{u}^1_{\tau+\Delta\tau})$. Let $\bar{E}_{\tau+\Delta\tau}$ be the global space and time error accumulated up to $\tau + \Delta\tau$ after the advection in Equation (31) and before mapping the solution values to $\mathcal{M}_R$. $\bar{E}_{\tau+\Delta\tau}$ does not include the mapping errors at $\tau + \Delta\tau$. The final solution after applying the operator H in Equation (32) is

$$\bar{u}_{\tau+\Delta\tau} = \bar{u}^1_{\tau+\Delta\tau} + H\bar{u}^1_{\tau+\Delta\tau}. \tag{33}$$

The true solution $\hat{u}_{\tau+\Delta\tau}$ at the end of time step $\tau + \Delta\tau$ and after the mapping from $\mathcal{M}_A$ to $\mathcal{M}_R$ and back $\mathcal{M}_A$ to can be expressed as

$$\hat{u}_{\tau+\Delta\tau} = \bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau} + \hat{H}\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big), \tag{34}$$

where $\hat{H}$ is assumed to be the corresponding “exact" operator for H. Subtracting Equation (34) from (33) gives an expression for the true error that can be written as

$$E^G_{\tau+\Delta\tau} = \bar{E}_{\tau+\Delta\tau} + \hat{H}\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) - H\bar{u}^1_{\tau+\Delta\tau},$$

where E^G is the global space and time error including the mapping errors at $\tau + \Delta\tau$. Adding and subtracting $H\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big)$ yields

$$E^G_{\tau+\Delta\tau} = \bar{E}_{\tau+\Delta\tau} + \hat{H}\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) - H\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) + H\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) - H\bar{u}^1_{\tau+\Delta\tau}.$$

Using a Taylor expansion of $H\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big)$ about $\bar{u}^1_{\tau+1}$ and dropping the high-order terms, we can approximate the total errors as

$$E^G_{\tau+1} \approx \bar{E}_{\tau+\Delta\tau} + \hat{H}\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) - H\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) + \frac{\partial H}{\partial u}(u^1_{\tau+\Delta\tau})E_{\tau+\Delta\tau}. \tag{35}$$

The results in Equation (35) indicate that the total error is dependent on

- the existing global space and time error $\bar{E}_{\tau+\Delta\tau}$, which does not include the mapping error at $\tau + \Delta\tau$,
- the mapping error $E^M_{\tau+\Delta\tau}$ at $\tau + \Delta\tau$,

$$\begin{aligned} E^M_{\tau+\Delta\tau} &= \hat{H}\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) - H\big(\bar{u}^1_{\tau+\Delta\tau} + \bar{E}_{\tau+\Delta\tau}\big) \\ &= \hat{H}\hat{u}^1_{\tau+\Delta\tau} - H\hat{u}^1_{\tau+\Delta\tau}, \text{ and} \end{aligned}$$

- a multiplier of the existing global space and time error $\bar{E}_{\tau+\Delta\tau}$,

$$E^N_{\tau+\Delta\tau} = \frac{\partial H}{\partial u}(u^1_{\tau+\Delta\tau})\bar{E}_{\tau+\Delta\tau}.$$

Mapping data values from $\mathcal{M}_A$ to $\mathcal{M}_R$ and back to $\mathcal{M}_A$ introduces the interpolation errors that degrade the solution if $E^M_{\tau+\Delta\tau}$ is greater than the existing global space and time error $\bar{E}_{\tau+\Delta\tau}$. This problem is resolved when the mapping error is kept smaller than the existing global space and time error. Similar ideas in the context of time dependent differential equations are explored in [4, 11, 22]. The studies in [11] and [22] develop strategies for balancing the space and time error for better error control and improved performance while [4] show that in mesh adaptivity the spatial interpolation error must be controlled and kept smaller than the temporal error.

5.2 1D Modified Runge Function

This example is based on the modified version of the Runge function defined in Equation (23) and two meshes that emulate the dynamics/advection ($\mathcal{M}_A$) and physics/reaction ($\mathcal{M}_R$) meshes used in NEPTUNE. No actual PDE is solved in this example. Here, we consider the advection time step of the trivial case where the identity operator is used to represent the advection and reaction. The function $f_1(x)$ is evaluated on advection mesh $\mathcal{M}_A$ to create the initial data values. Given that the identity operator is used for both the advection and reaction, these initial data values are mapped to the reaction mesh $\mathcal{M}_R$ and back to the starting mesh M_A. Using the identity operator allows for a study of the mapping error without the influence of the advection and reaction.

Table 7 shows L^2-norms of the mapping errors over the grid points for $f_1(x)$ when using the PCHIP, MQSI, DBI, and PPI methods to map the data values from the advection mesh to the reaction mesh and back to the advection mesh. For $N = 64$ points, increasing the interpolant degree does not significantly improve the approximation. The global error is dominated by the local error in the regions with steep gradients that are to the left and right of the peak at $x = 0$. The mapping errors can be improved by increasing the resolution and adding more points in the regions with steep gradients. The resolution is increased by adding one or three uniformly spaced points in each interval from the initial profile with 64 points. Increasing the resolution leads to better approximations when mapping data values between both meshes, and the error decreases as we increase the polynomial degree from 3 to 7. This example demonstrates that in cases with steep gradients, using the PPI method with high-order interpolants may not significantly improve the approximation unless there is sufficient resolution. In order to benefit from the positivity and the high-order interpolants, it is important to be in the regime where the problem has sufficiently many points to observe convergence as the polynomial degree increases. Overall, the PPI method leads to smaller errors compared to the other methods as the resolution and polynomial degree increase.

N	PCHIP	MQSI		DBI			PPI	
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_7$	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_7$
64	2.92E-03	1.93E-03	4.93E-03	4.78E-04	7.12E-05	3.99E-03	3.13E-04	2.85E-05
127	3.81E-04	5.57E-04	3.58E-03	3.81E-04	6.50E-05	2.85E-03	1.02E-04	3.65E-06
253	6.71E-05	1.48E-04	3.41E-03	3.69E-04	6.49E-05	2.46E-03	3.50E-05	8.62E-07

Table 7. L^2-norm of mapping errors for the modified Runge function $f_4(x)$ when using the PCHIP, MQSI, DBI, and PPI methods to map the data values from the advection mesh to the reaction mesh and back to the advection mesh. The target polynomials are set to $d = 3$, $d = 5$, and $d = 7$. N represents the number of input points used for both meshes. The parameter st is set to 3.

5.3 TWP-ICE Example

This study uses the tropical warm pool international cloud experiment (TWP-ICE) test case from the common community physics package (CCPP) [8]. The input mesh for the simulation is configured to emulate a vertical column in NEPTUNE. The simulation result at time $t = 1440$ sec is extracted and scaled, and used to evaluate different interpolation approaches when mapping solution values between advection and reaction meshes. The domain and range are scaled to $[-1, 1]$ and $[0, 1]$, respectively. This study considers the cloud water mixing ratio profile, which represents the amount of cloud water in air. The extracted profile is then fitted using a radial basis function interpolation to construct an analytical function that can be used as the starting point of the mapping evaluation. The radial basis function is based on multiquadrics:

$$b_i = \sqrt{1 + (\epsilon|x - x_i|)^2}.$$

The parameter ϵ is approximated using cross validation [7]. The initial values are obtained by evaluating the analytical function on the advection mesh. These values are then mapped to the reaction mesh and back to the advection mesh.

Table 8 shows L^2-norms of the mapping errors for the extracted profile when using the PCHIP, DBI, and PPI methods to map the data values from the advection to physics mesh and back to advection mesh. For $N = 64$, the global error is dominated by the local error at a few points located in the regions with steep gradients. Increasing the polynomial degree does not significantly improve the approximation compared to using PCHIP for $N = 64$. More points are required to better approximate the underlying profile in the regions with steep gradients. The resolutions are increased by adding one and three uniformly spaced points in each interval from the initial $N = 64$ mesh points. Table 8 shows that with the increased resolution, the approximation improves as the polynomial degree increases. The number of points used in each region with steep gradients increased as more points were added. This example provides an application example using simulation data from TWP-ICE. In cases of coarse resolution (64 points), the PCHIP and MQSI results are comparable and going to higher degree interpolants doesn't significantly improve the approximation for DBI and PPI. The approximation improves with higher degree interpolants when the resolution is increased, as shown in Table 8. The results from this experiment suggest that increasing the resolution is needed for the mapping between meshes to benefit from the high-order interpolants from the PPI methods.

N	PCHIP	MQSI		DBI			PPI	
	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_7$	$\mathcal{P}_3$	$\mathcal{P}_5$	$\mathcal{P}_7$
64	4.66E-03	3.25E-03	1.17E-02	3.15E-03	6.77E-04	5.11E-03	9.86E-04	1.12E-04
127	1.56E-03	2.05E-03	1.12E-02	3.10E-03	6.49E-04	2.30E-03	3.10E-04	1.83E-05
253	4.89E-04	6.92E-04	1.11E-02	3.11E-03	6.46E-04	1.41E-03	1.45E-04	3.70E-06

Table 8. L^2-norm of mapping errors for the TWP-ICE profile when using the PCHIP, MQSI, DBI, and PPI methods to map the data values from the advection mesh to the reaction mesh and back to the advection mesh. The target polynomials are set to $d = 3$, $d = 5$, and $d = 7$. N represents the number of input points used for both meshes. The parameter st is set to 3.

5.4 BOMEX Example

Here, a maritime shallow convection example based on the 1D Barbados Oceanographic and Meteorological Experiment (BOMEX) [9] is used to study the effects of the different interpolation methods in an application. BOMEX is a single-column shallow convection test case used to measure and study changes in temperature, wind, water vapor mixing ratio, rain water mixing ratio, and cloud water mixing ratio. The mixing ratios represent the amount of water vapor, rain water, and cloud water in air. In this simulation, the dynamics/advection is modeled by

$$\frac{\partial X}{\partial t} = \mathcal{L}X,$$

where X is the state variable that contains the wind, temperature, water vapor mixing ratio, cloud water mixing ratio, and rain water mixing ratio. The dynamics are approximated using fifth-order weighted essentially nonoscillatory (WENO) and third-order Runge-Kutta methods [39]. The physics component of the simulation uses the hybrid eddy-diffusivity mass-flux and free atmospheric turbulence (hybrid EDMF) and Kessler microphysics schemes from [8] to alter the results of the dynamics and incorporate the physics properties. The dynamics and physics results are calculated on different meshes. At each time step, the dynamics are calculated on the advection mesh $\mathcal{M}_A$, and the results are interpolated to the reaction mesh $\mathcal{M}_R$ for the use of the physics routines. The physics terms are calculated using the reaction mesh $\mathcal{M}_R$, and the results are interpolated back to the advection mesh $\mathcal{M}_A$.

As in [32], let q_c be the cloud water mixing ratio profile in the different experiments. Figures 6a - 7f show the cloud mixing ratio profile q_c at $t = 5$ hours that is used as input for the physics routines. The physics calculations require positive input values for q_c. Figure 6a 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 6a. Figures 6b - 7f 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.

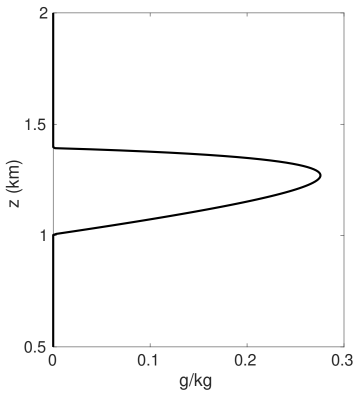
Figure 6b shows the cloud mixing ratio profiles q_c for the target and approximated solution at $t = 5$ hours. In the case of the approximated solution, a fifth-order standard polynomial interpolation is used when mapping between the advection and reaction meshes. For a given interval I_i, the polynomial 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 6b demonstrate that using the standard polynomial interpolation leads to oscillations, negative 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 $max(q_c) = 0.28g/kg$.

The negative values in Figure 6b can be removed via "clipping", which is a procedure that consists of removing the negative values by setting them to zero. Figure 6c 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 6c shows that using "clipping" still allows for oscillations and a positive bias in the prediction of the cloud mixing ratio q_c. The total cloud mixing ratio is 2.09 times greater than the target solution, and the peak $max(q_c) = 0.46g/kg$ is larger than the target peak $max(q_c) = 0.28g/kg$.

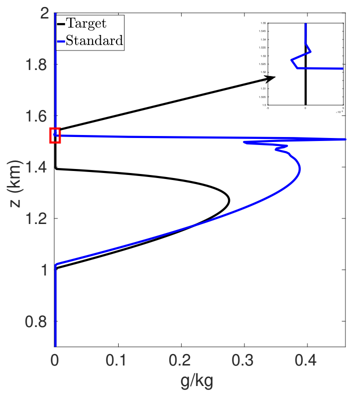
Using PCHIP to map between the advection and reaction meshes eliminates the negative values, removes oscillations, and reduces the positive bias in the cloud mixing ratio prediction compared to the standard interpolation with and without "clipping". Figure 6d shows the target profile q_c and an approximated profile that uses PCHIP for mapping solution values between advection and reaction 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 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 DBI and PPI. Here, the MQSI method is not used because it oversmoothes the state variable X at each time step and leads to no production of cloud mixing ratio.

Figures 7a-7f show 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 oscillations 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 $st = 3$. Figures 7a-7f show that using the DBI and PPI methods with $\epsilon_0 = \epsilon_1 = 10^{-5}$ to map data values between the advection and reaction 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 a 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 $max(q_c) = 0.28g/kg$.

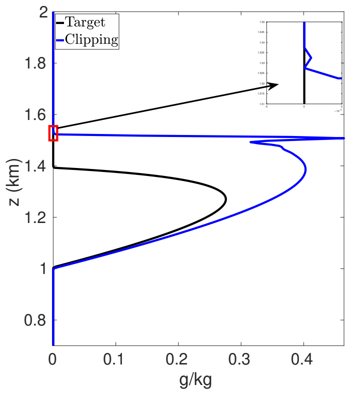
In summary, using DBI and PPI methods to map data values between both the advection and reaction meshes produces better approximation results compared to the standard interpolation and PCHIP methods. Tables 9 and 10 provide a summary of the maximum values and the total amount of cloud mixing ratios for 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 cloud mixing ratios compared to the standard interpolation and PCHIP approaches. The results from Tables 9 and 10 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 a better approximation.



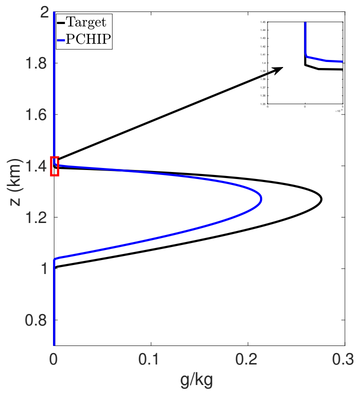
(a) Target (no mapping required).



(b) Standard interpolation.

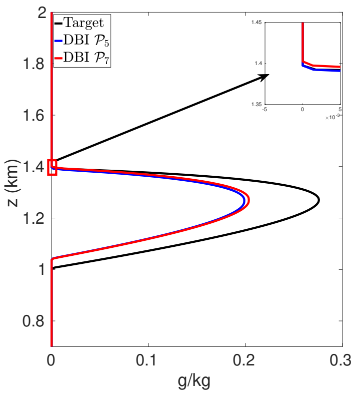


(c) Standard interpolation with “clipping".

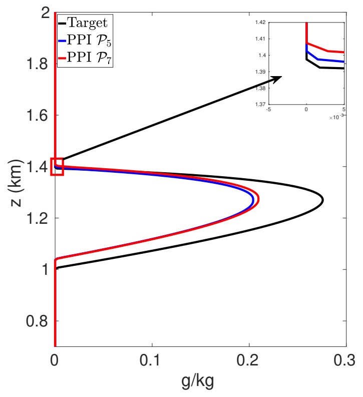


(d) PCHIP.

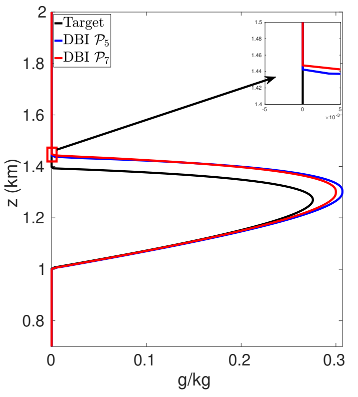
Fig. 6. Cloud mixing ratio q_c profile from the BOMEX test case at $t = 5$ hours with $nz = 600$ points. A fifth-order WENO and third-order Runge-Kutta schemes with $CFL = 0.1$ are used for the dynamics (advection). 6a. The black plot in 6b, 6c, and 6d represents the target profile where the same mesh is used for the dynamics and physics calculations. In 6b, 6c, and 6d, the profiles in blue use different meshes for the dynamics and physics calculations which require mapping the solution values between both meshes. A standard polynomial interpolation, a standard polynomial interpolation with “clipping", and PCHIP methods are used for the mapping in 6b, 6c, and 6d, respectively.



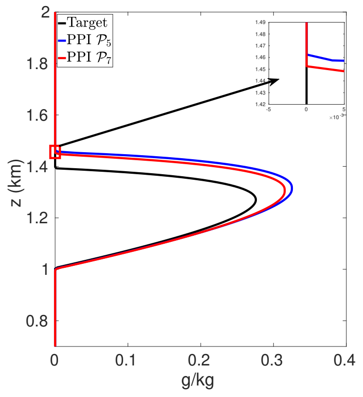
(a)



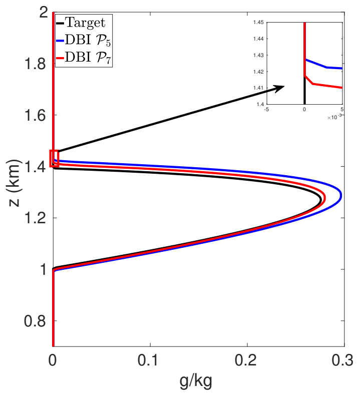
(b)



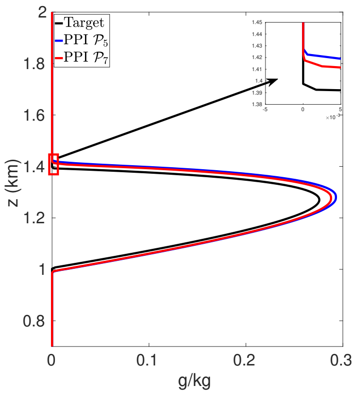
(c)



(d)



(e)



(f)

Fig. 7. Cloud mixing ratio q_c profile from the BOMEX test case at $t = 5$ hours with $nz = 600$ points with $\epsilon_0 = \epsilon_1 = 10^{-5}$. The profile in black is the target solution. The profiles on the left (7a, 7c, 7e) and right (7b, 7d, 7f) are obtained using the DBI and PPI methods, respectively, to map solution values between meshes. The maximum polynomial degrees are set to $d = 5$ and $d = 7$ for the blue and red plots, respectively. The input parameter st is set to 1, 2, and 3 for the advection (7a, 7b), reaction (7c, 7d), and third (7e, 7e, 7f) row, respectively. A fifth-order WENO and third-order Runge-Kutta schemes with $CFL = 0.1$ are used for the dynamics (advection).

	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 9. Maximum values of q_c and the total amount of the cloud mixing ratio at $t = 5$ hours 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 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 10. Maximum values of q_c and the total amount of the cloud mixing ratio at $t = 5$ hours 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 g/kg.

6 DISCUSSION AND CONCLUDING REMARKS

In this work we introduced HiPPIS: a high-order 1D, 2D, and 3D data-bounded and positivity-preserving interpolation software for structured meshes. The software implementation is based on the mathematical framework in Section 2 and the algorithms in Section 3. The software is self-contained and can be incorporated into larger codes that require data-bounded or positivity-preserving interpolation. The interface is designed to be similar to commonly used PCHIP and splines interfaces. The algorithms used in the software extend the DBI and PPI methods introduced in [29] by adding more options for the stencil construction process that can be set by the user with the parameter st. For a given interval I_i, the algorithm starts with the stencil $\mathcal{V}_0 = \{x_i, x_{i+1}\}$ and successively appends points to the left and/or right of $\mathcal{V}_0$ to form the final stencil. The stencil construction is done in accordance with the DBI and PPI conditions outlined in Equations (21a) and (21b). In addition to the different options for the stencil selection process, the software introduces a parameter ϵ_1 that can be used to adjust the bounds of the interpolants in the intervals where extrema are detected.

Various 1D and 2D examples are employed to evaluate the use of the DBI and PPI software in different contexts. The results in Section 4 show that for Examples I, II, IV, and V, which are based on underlying smooth functions, the DBI and PPI methods produce more accurate results compared to PCHIP and MQSI. For Example III, and VI which are obtained from discontinuous and C^0-continuous functions, all the methods have comparable errors. Figures 2-5 show that the parameters ϵ_0 and ϵ_1 can be used to reduce undesired oscillations from the PPI method. We only report the results for $st = 3$ because the differences between $st = 1$, $st = 2$, and $st = 3$ are negligible for the examples in Section 4.

Section 5 provides an analysis and numerical comparison of the mapping error when the DBI and PPI methods are used to map data values between different meshes. The error analysis in Section 5.1 shows that it is important to keep mapping errors smaller than the already existing global errors from other calculations. Removing negative values and spurious oscillations can help reduce the mapping error.

The results in Tables 7 and 8 show that using small values for parameters ϵ_0 and ϵ_1 improves the approximation in cases where the input data are coarse. Small values of ϵ_0 and ϵ_1 further restrict how much the interpolant is allowed to grow beyond the data values. The parameters ϵ_0 and ϵ_1 are used to adjust the lower and upper bounds on each interpolant according to Equations (12) and (13). The study of the modified Runge example in Section 5.2 and TWP-ICE example in Section 5.3 demonstrated that for a profile with steep gradients or fronts, more points are required to better take advantage of the DBI and PPI algorithm. If there are not sufficiently many points in the regions with steep gradients or fronts, increasing the polynomial degree may not improve the accuracy. The results in Tables 7 and 8 show that once the resolution is sufficiently increased, the approximations improve as the polynomial degree increases.

In the BOMEX test case, prioritizing a symmetry ($st = 2$) or locality ($st = 3$) leads to better approximations compared to the ENO stencil ($st = 1$) using the DBI and PPI methods. Using the ENO stencil ($st = 1$) produces significantly less cloud mixing ratio compared to both the prioritizing symmetry and locality. In the BOMEX example with parameters ϵ_0 and ϵ_1 greater than 10^{-5}, the PPI method allows for oscillations that degrade the approximation compared to the DBI and PCHIP approaches. The MQSI method is not used for the BOMEX example because it oversmoothes the different variables at each time step and leads to no production of cloud mixing ratio.

In summary, this work provided (1) a high-order DBI and PPI software for 1D, 2D, and 3D structured meshes; (2) an analysis of the mapping error when using the DBI or PPI to map data values between meshes; and (3) an evaluation of the DBI and PPI methods in the context of function approximation and interpolating data values between different meshes. As this work continues, we plan to investigate different approaches for extending the DBI and PPI methods to unstructured 2D and 3D meshes.

ACKNOWLEDGMENTS

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).

REFERENCES

[1] Balsara, D. S. Self-adjusting, positivity preserving high order schemes for hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics 231*, 22 (2012), 7504–7517.

[2] Berzins, M. Adaptive polynomial interpolation on evenly spaced meshes. *SIAM Review 49*, 4 (2007), 604–627.

[3] Berzins, M. Nonlinear data-bounded polynomial approximations and their applications in eno methods. *Numerical Algorithms 55*, 2 (2010), 171–189.

[4] Berzins, M., Capon, P. J., and Jimack, P. K. On spatial adaptivity and interpolation when using the method of lines. *Applied Numerical Mathematics 26*, 1 (1998), 117–133.

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

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

[7] Fasshauer, G. E., and Zhang, J. G. On choosing "optimal" shape parameters for rbf approximation. *Numerical Algorithms 45*, 1 (2007), 345–368.

[8] Firl, G., Carson, L., Harrold, M., Bernardet, L., and Heinzeller, D. Common community physics package single column model (scm), Sept 2020.

[9] Friedman, H. A., Conrad, G., and McFadden, J. D. Essa research flight facility aircraft participation in the barbados oceanographic and meteorological experiment. *Bulletin of the American Meteorological Society 51*, 9 (1970), 822–834.

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

[11] Goodyer, C. E., and Berzins, M. Adaptive timestepping for elastohydrodynamic lubrication solvers. *SIAM Journal on Scientific Computing 28*, 2 (2006), 626–650.

[12] Green, K. R., Bohn, T. A., and Spiteri, R. J. Direct function evaluation versus lookup tables: When to use which? *SIAM Journal on Scientific Computing 41*, 3 (2019), C194–C218.

[13] Harten, A., Engquist, B., Osher, S., and Chakravarthy, S. R. Uniformly high order accurate essentially non-oscillatory schemes, iii. *Journal of*

Computational Physics 131, 1 (1997), 3 – 47.

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

[15] Hu, X. Y., Adams, N. A., and Shu, C.-W. Positivity-preserving method for high-order conservative schemes solving compressible euler equations. *Journal of Computational Physics 242* (2013), 169–180.

[16] Hussain, M., Hussain, M. Z., and Cripps, R. J. C2 rational quintic function. *Journal of Prime Research in Mathematics 5* (2009), 115–123.

[17] Hussain, M. Z., Hussain, M., and Yameen, Z. A C2-continuous rational quintic interpolation scheme for curve data with shape control. *Journal of The National Science Foundation of Sri Lanka 46* (2018), 341.

[18] Hussain, M. Z., and Sarfraz, M. Positivity-preserving interpolation of positive data by rational cubics. *Journal of Computational and Applied Mathematics 218*, 2 (2008), 446 – 458. The Proceedings of the Twelfth International Congress on Computational and Applied Mathematics.

[19] James D. Doyle and P. A. Reinecke, K. C. Viner, S. Gabersek, M. Martini, D. D. Flagg, J. Michalakes, D. R. Ryglicki, and F. X. Giraldo. Next generation nwp using a spectral element dynamical core, January 2017.

[20] Karim, A., Ariffin, S., Voon Pang, K., and Saaban, A. Positivity preserving interpolation using rational bicubic spline. *Journal of Applied Mathematics 2015* (2015).

[21] Krogh, F. T. Efficient algorithms for polynomial interpolation and numerical differentiation. *Mathematics of Computation 24*, 109 (1970), 185–190.

[22] Lawson, J., Berzins, M., and Dew, P. M. Balancing space and time errors in the method of lines for parabolic equations. *SIAM Journal on Scientific and Statistical Computing 12*, 3 (1991), 573–594.

[23] Light, D., and Durran, D. Preserving nonnegativity in discontinuous galerkin approximations to scalar transport via truncation and mass aware rescaling (tmar). *Monthly Weather Review 144*, 12 (2016), 4771–4786.

[24] Liu, H., Gao, Z., Jiang, C., and Lee, C. Numerical study of combustion effects on the development of supersonic turbulent mixing layer flows with weno schemes. *Computers and Fluids 189* (2019), 82–93.

[25] Lux, T., Watson, L. T., Chang, T., and Thacker, W. Algorithm 1031: Mqsi–monotone quintic spline interpolation. *ACM Trans. Math. Softw. 49*, 1 (mar 2023).

[26] Lux, T. C. H., Watson, L. T., and Chang, T. H. An algorithm for constructing monotone quintic interpolating splines. In *SpringSim '20: Proceedings of the 2020 Spring Simulation Conference, May 2020* (2019), pp. 1–12.

[27] Moler, C. B. *Numerical computing with MATLAB*. SIAM, 2004.

[28] Ouermi, T. A. J., Kirby, R. M., and Berzins, M. Numerical testing of a new positivity-preserving interpolation algorithm, 2020.

[29] Ouermi, T. A. J., Kirby, R. M., and Berzins, M. Eno-based high-order data-bounded and constrained positivity-preserving interpolation. *Numerical Algorithms* (2022).

[30] Piah, A. R. M., Goodman, T. N. T., and Unsworth, K. Positivity-preserving scattered data interpolation. In *Mathematics of Surfaces XI* (Berlin, Heidelberg, 2005), R. Martin, H. Bez, and M. Sabin, Eds., Springer Berlin Heidelberg, pp. 336–349.

[31] Rogerson, A., and Meiburg, E. A numerical study of the convergence properties of eno schemes. *Journal of Scientific Computing 5*, 2 (1990), 151–167.

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

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

[34] Sarfraz, M. A c2 rational cubic spline alternative to the nurbs. *Computers and Graphics 16*, 1 (1992), 69 – 77.

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

[36] Schmidt, J. W., and Hess, W. Positivity of cubic polynomials on intervals and positive spline interpolation. *BIT 28*, 2 (feb 1988), 340–352.

[37] Sekora, M., and Colella, P. Extremum-preserving limiters for muscl and ppm, 2009.

[38] Shu, C.-W. Numerical experiments on the accuracy of eno and modified eno schemes. *Journal of Scientific Computing 5*, 2 (1990), 127–149.

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

[40] Skamarock, W. C., and Weisman, M. L. The impact of positive-definite moisture transport on nwp precipitation forecasts. *Monthly Weather Review 137*, 1 (2009), 488–494.

[41] Subbareddy, P. K., Kartha, A., and Candler, G. V. Scalar conservation and boundedness in simulations of compressible flow. *Journal of Computational Physics 348* (2017), 827–846.

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

[43] Tal-Ezer, H. High degree polynomial interpolation in newton form. *SIAM Journal on Scientific and Statistical Computing 12*, 3 (1991), 648–667.

[44] Ulrich, G., and Watson, L. T. Positivity conditions for quartic polynomials. *SIAM Journal on Scientific Computing 15*, 3 (1994), 528–544.

[45] Zhang, X. On positivity-preserving high order discontinuous galerkin schemes for compressible navier–stokes equations. *Journal of Computational Physics 328* (2017), 301 – 343.

[46] Zhang, X., and Shu, C.-W. Positivity-preserving high order finite difference weno schemes for compressible euler equations. *J. Comput. Phys. 231*, 5 (Mar. 2012), 2245–2258.