


Article

Alternating Polynomial Reconstruction Method for Hyperbolic Conservation Laws

Shijian Lin ¹ , Qi Luo ², Hongze Leng ^{1,*} and Junqiang Song ¹

¹ College of Meteorology and Oceanography, National University of Defense Technology, Changsha 410000, China; linshijian10@nudt.edu.cn (S.L.); junqiang@nudt.edu.cn (J.S.)

² Department of Mathematics, National University of Defense Technology, Changsha 410000, China; luoqi10@nudt.edu.cn

* Correspondence: hzleng@nudt.edu.cn

Abstract: We propose a new multi-moment numerical solver for hyperbolic conservation laws by using the alternating polynomial reconstruction approach. Unlike existing multi-moment schemes, our approach updates model variables by implementing two polynomial reconstructions alternately. First, Hermite interpolation reconstructs the solution within the cell by matching the point-based variables containing both physical values and their spatial derivatives. Then the reconstructed solution is updated by the Euler method. Second, we solve a constrained least-squares problem to correct the updated solution to preserve the conservation laws. Our method enjoys the advantages of a compact numerical stencil and high-order accuracy. Fourier analysis also indicates that our method allows a larger CFL number compared with many other high-order schemes. By adding a proper amount of artificial viscosity, shock waves and other discontinuities can also be computed accurately and sharply without solving an approximated Riemann problem.

Keywords: hyperbolic conservation laws; multi-moment; high-order accuracy; local reconstruction



Citation: Lin, S.; Luo, Q.; Leng, H.; Song, J. Alternating Polynomial Reconstruction Method for Hyperbolic Conservation Laws. *Mathematics* **2021**, *9*, 1885. <https://doi.org/10.3390/math9161885>

Academic Editors: Juan Ramón Torregrosa Sánchez, Alicia Cordero Barbero and Juan Carlos Cortés López

Received: 9 July 2021
Accepted: 6 August 2021
Published: 8 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In this paper, we aim to solve the hyperbolic conservation law

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \quad (1)$$

where u and $f(u)$ can be either scalars or vectors.

The classical higher-order numerical schemes such as the finite difference (FD) and the finite volume (FV) frameworks usually involve only one degree of freedom (DOF) per cell. That means to achieve the k th order discretization of the first-order spatial derivative, a stencil of at least $k + 1$ cells is required. The compactness of a scheme can help simplify the polynomial interpolation in unstructured meshes and reduce communication costs between computational nodes on modern supercomputers. In this paper, the “compact” is used specifically for numerical schemes with a minimal stencil, which excludes so-called Padé schemes [1,2] since its approximation of the derivative involves solving a diagonal system of the full difference stencil.

A natural thought for the design of a compact high-order scheme is involving more than one DOF per cell so as to construct higher-order polynomials under the same stencil width. A series of multi-moment schemes is developed based on this idea. The constrained interpolation profile (CIP) scheme [3,4], which uses the semi-Lagrangian method to advance in time, individually updates the physical variable and its derivatives on each cell by the governing equations in different forms. The interpolated differential operator (IDO) method [5] is similar to CIP but uses an explicit time-advancing approach. For the IDO method, a stencil containing three adjacent cells suffices to approximate the first-order spatial derivative with fifth-order accuracy in one dimension, which makes it possible to

obtain comparably accurate results with spectral methods in direct numerical simulation [6]. For IDO, updating the derivative-value variables induces the problem of computing the time derivative of u_x , which involves f_{uu} . However, for the system of conservation laws, f_{uu} has a fairly complicated form and can be difficult to compute. Goodrich et al. proposed a similar Hermite method [7] on staggered grids.

Neither CIP nor IDO are conservative schemes, which means they are not ideal for shock-capturing problems and long-term simulations. To address the non-conservativeness, so-called CIP-conservative semi-Lagrangian (CIP-CSL) schemes [8,9] and the IDO scheme in conservative form (IDO-CF) [10,11] add the cell-averaged value as an extra independent model variable to evolve. The cell-averaged variable is advanced by the flux-form conservation law and is hence exactly conserved, while other kinds of variables are updated using the differential form of conservation laws. However, these conservative CIP algorithms need to find polynomials that match cell-averaged variables across cells, which can be time-consuming when encountered with non-uniform meshes. A framework termed the multi-moment finite volume method (MM-FVM) has been proposed on the basis of CIP-CSL. Compared with CIP-CSL, MM-FVM can construct the high-order interpolation on a local base and is more flexible when treating unstructured grids, as shown in [12]. Nonetheless, MM-FVM uses the semi-Lagrangian method in the time integration of point-based variables, which is difficult to implement in the multi-dimensional case.

Another class of high-order conservative schemes based on local flux reconstructions also uses a compact stencil and has become increasingly attractive in recent decades. Examples include the well-known discontinuous Galerkin (DG) [13–15], spectral volume [16,17], and spectral difference [18] methods and, more recently, the flux reconstruction/correction via procedure (FR/CPR) method [19,20]. These methods evolve k DOFs in each cell for a k th order spatial accuracy (in one dimension) and compute the numerical flux on cell boundaries to represent the interaction between adjacent cells. Thus, polynomial interpolation across cells is not needed, which makes these methods compact and flexible in handling unstructured meshes. The multi-moment constrained finite volume (MCV) method proposed by Li and Xiao [21] can be viewed as a combination of CIP and the flux reconstruction approach. The MCV method introduces high-order spatial derivatives of numerical flux and reconstructs the local flux function more directly. MCV is generalized as multi-moment constrained flux reconstruction (MMC-FR) [22]. However, the nonlinear aliasing phenomenon [23,24] in treating nonlinear flux functions, which causes instability and accuracy loss, is one typical demerit for these methods based on local flux reconstruction.

This paper presents a novel conservative multi-moment approach termed the AltPoly method, which uses the alternating polynomial reconstruction as the time integration method. AltPoly divides the computational domain into cells and adopts point-based and cell-averaged values as model variables. The point-based variables including the physical variable and its spatial derivatives are updated together, which is different from the conventional multi-moment methods that update different types of model variables according to governing equations in different forms. AltPoly then solves a constrained least-squares problem to reconstruct the solution within the cell and update the model variables, which preserves conservativeness exactly. Solving Riemann problems on cell boundaries is not needed. Furthermore, AltPoly can also handle non-uniform meshes in one dimension efficiently with the aid of the coordinate transform.

The remainder of this paper is organized as follows: Section 2 describes the formulation of AltPoly and constructs artificial viscosity for discontinuous problems. Section 3 studies the accuracy and stability of AltPoly by Fourier analysis. Section 4 validates the numerical convergence rate and capability of solving some widely used benchmark problems involving both the scalar equation and Euler systems in 1D. Finally, a short summary in Section 5 ends this paper.

2. Formulation of AltPoly Method

This section gives the formulation of the AltPoly method for a one-dimensional conservation law. For concreteness and ease of comprehension, we only present the detailed formulation for AltPoly with 2 point-value variables and one extra cell-averaged variable per cell, which is termed AltPoly-3. As for the general AltPoly- r with r model variables within a cell, the formulation is similar to AltPoly-3 and is hence briefly illustrated in Appendix A.

To begin with, we first introduce some notation. The bold font lower-case letter denotes a column vector, e.g., \mathbf{a} , \mathbf{u} . The transpose \mathbf{a} is denoted by \mathbf{a}^T . \mathbf{e}_k denotes the k th standard unit vector in a space in proper dimensions. $[\mathbf{a}_k]_{k=1}^m$ denotes a column vector in the form of $[a_1, \dots, a_m]^T$. Bold font capital letters such as \mathbf{A} denote matrices. The submatrix of \mathbf{A} constructed from rows $\{r_1, r_2, \dots, r_k\}$ and columns $\{c_1, c_2, \dots, c_l\}$ is denoted as $\mathbf{A}[r_1, r_2, \dots, r_k; c_1, c_2, \dots, c_l]$. $\|\mathbf{a}\|$ denotes the 2-norm of the vector \mathbf{a} , while $\|\mathbf{A}\|$ for a matrix \mathbf{A} is defined by $\|\mathbf{A}\| = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}$.

2.1. Spatial Discretization and Model Variables

Suppose the spatial domain $[x_{\min}, x_{\max}]$ is divided into N non-overlapping cells, among which the j -th cell is $I_j := [x_{j-1/2}, x_{j+1/2}]$, $j = 1, \dots, N$. It should be noted that the cells do not necessarily have a uniform length. In the j -th element, we consider computing the time integration of three model variables: u , u_x at the middle point of the element, and additionally the element-integrated value $\bar{u}_j = \frac{1}{\Delta x_j} \int_{x_{j-1/2}}^{x_{j+1/2}} u \, dx$.

2.2. Updating Model Variables

Denote the model variables at time level $t = t_0$ as $\{u_j^{(t_0)}, (u_x)_j^{(t_0)}, \bar{u}_j^{(t_0)}\}$. This part illustrates the procedure to obtain $\{u_j^{(t_1)}, (u_x)_j^{(t_1)}, \bar{u}_j^{(t_1)}\}$ with $t_1 = t_0 + \Delta t$, where Δt is the time step length. In a nutshell, this procedure consists of 3 steps:

1. Hermite interpolation across cells using point-based variables;
2. Computing solution values on selected solution points based on reconstructed polynomials and updating solution values and cell-averaged variables via the Euler forward method;
3. Reconstructing the solution polynomial that matches updated solution values under the constraint of cell-averaged variables to preserve conservativeness.

Details are presented in the following and algorithm illustrations are presented in Figure 1 for convenience of illustration.

2.2.1. Step 1: Hermite Interpolation across Cells

In the first step, we find the interpolation polynomial $h_{j-\frac{1}{2}}(x)$ on the interval $[x_{j-1}, x_j]$ that matches $u_{j-1}^{(t_0)}, (u_x)_{j-1}^{(t_0)}, u_j^{(t_0)}, (u_x)_j^{(t_0)}$ as shown in Figure 1a. This step does not involve cell-averaged variables $\bar{u}_j^{(t_0)}$. This interpolation polynomial is of the third order since there are 4 conditions in total. Before constructing $h_{j-\frac{1}{2}}(x)$, we introduce a local coordinate

$$s(x) := \frac{x - \frac{1}{2}(x_j + x_{j+1})}{x_{j+1} - x_j} \in [-1, 1] \tag{2}$$

for $x \in [x_j, x_{j+1}]$ as is usually done in the finite element method [25]. The local coordinate can unify the interpolation templates so as to reduce the computational cost. The polynomial $h_{j-\frac{1}{2}}(x)$ on the local coordinate s is denoted by $\tilde{h}_{j-\frac{1}{2}}(s) := h_{j-\frac{1}{2}}(x(s))$. Notice that $\frac{\partial \tilde{h}}{\partial s}$ and $\frac{\partial h}{\partial x}$ can be connected by the following rule:

$$\frac{\partial \tilde{h}(s)}{\partial s} = \frac{\partial h(x(s))}{\partial s} = \frac{\partial h}{\partial x} \frac{\partial x}{\partial s} = (x_j - x_{j-1}) \frac{\partial h}{\partial x}. \tag{3}$$

Then the coefficients of $\tilde{h}_{j-\frac{1}{2}}(s) = \sum_{i=0}^3 a_i s^i$ can be determined by the following system:

$$\left\{ \begin{aligned} \tilde{h}_{j-\frac{1}{2}}(-1) &= \sum_{k=0}^3 (-1)^k a_k = u_{j-1}^{(t_0)}, \\ \frac{\partial \tilde{h}_{j-\frac{1}{2}}(-1)}{\partial s} &= \sum_{k=0}^3 (-1)^{k-1} k a_k = (x_j - x_{j-1}) \cdot (u_x)_{j-1}^{(t_0)}, \\ \tilde{h}_{j-\frac{1}{2}}(1) &= \sum_{k=0}^3 a_k = u_j^{(t_0)}, \\ \frac{\partial \tilde{h}_{j-\frac{1}{2}}(1)}{\partial s} &= \sum_{k=0}^3 k a_k = (x_j - x_{j-1}) \cdot (u_x)_j^{(t_0)}. \end{aligned} \right. \tag{4}$$

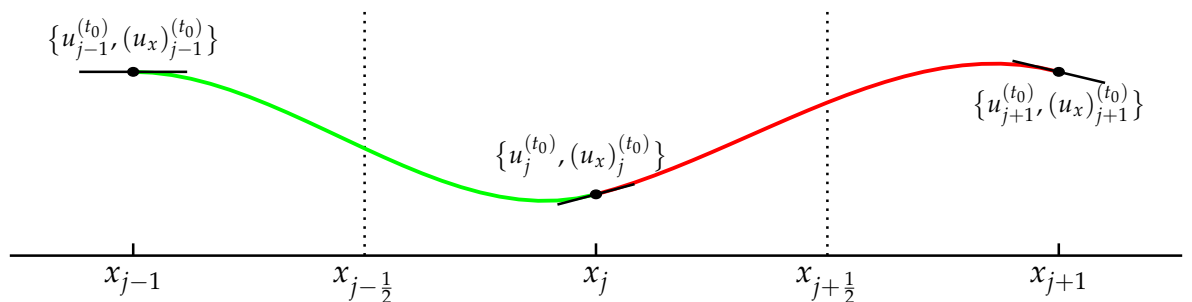
The matrix form is

$$H a = d, \tag{5}$$

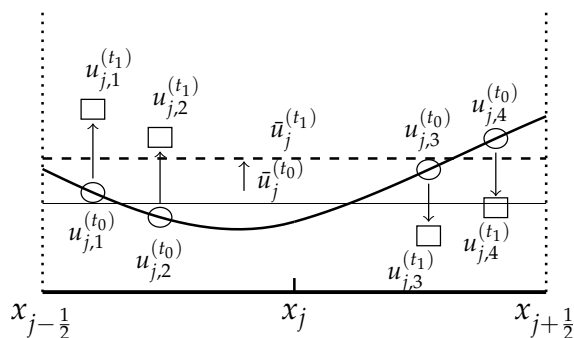
where

$$H = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix}, a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}, \text{ and } d = \begin{bmatrix} u_{j-1}^{(t_0)} \\ (x_j - x_{j-1}) \cdot (u_x)_{j-1}^{(t_0)} \\ u_j^{(t_0)} \\ (x_j - x_{j-1}) \cdot (u_x)_j^{(t_0)} \end{bmatrix}. \tag{6}$$

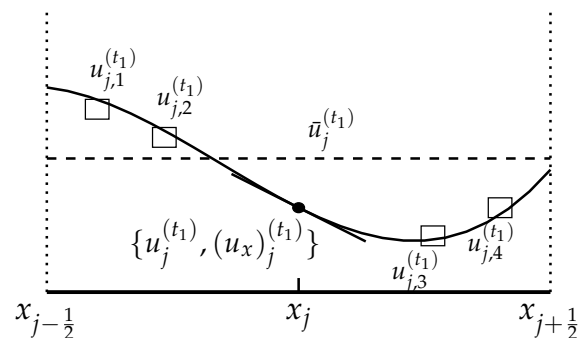
In practice, the inverse of H is computed and stored in advance so as to reduce the computation cost in solving (4).



(a) Step 1



(b) Step 2



(c) Step 3

Figure 1. Algorithm illustration.

2.2.2. Step 2: Updating Solution Values on Solution Points

Based on Step 1, we can now approximate and update the solution in cell I_j using polynomials $h_{j-\frac{1}{2}}(x)$ and $h_{j+\frac{1}{2}}(x)$. For AltPoly-3, four symmetric solution points $x_{j,k} = x_j + \zeta_k \Delta x_j / 2, k = 1, \dots, 4$ on I_j are needed in this step with $\zeta_k \in [-1, 1]$. In this paper, we choose $\zeta_1 = -1, \zeta_2 = -1 + \sigma, \zeta_3 = 1 - \sigma$ and $\zeta_4 = 1$ as solution points with $\sigma = 0.03$. To update the solution values on these solution points, we need to compute

$$u_{j,k} = \begin{cases} h_{j-\frac{1}{2}}(x_{j,k}), & \text{if } x_{j,k} \leq x_j \\ h_{j+\frac{1}{2}}(x_{j,k}), & \text{if } x_{j,k} > x_j, \end{cases} \text{ and } (u_x)_{j,k} = \begin{cases} \frac{\partial}{\partial x} h_{j-\frac{1}{2}}(x_{j,k}), & \text{if } x_{j,k} \leq x_j \\ \frac{\partial}{\partial x} h_{j+\frac{1}{2}}(x_{j,k}), & \text{if } x_{j,k} > x_j, \end{cases} \tag{7}$$

to approximate u and u_x . For convenience, we transfer the solution points ζ_k into local coordinate $s(\zeta_k)$ for $k = 1 \dots, 4$. Combining the conservative law (1) we can then compute $u_{j,k}^{(t_1)}$ by the Euler forward method:

$$u_{j,k}^{(t_1)} = u_{j,k} - f_u(u_{j,k}) \cdot (u_x)_{j,k} \Delta t, \tag{8}$$

where the function f_u stands for $\frac{\partial f(u)}{\partial u}$.

The cell-averaged variable $\bar{u}_j^{(t_1)}$ is updated by the flux form of the conservation law:

$$\bar{u}_j^{(t_1)} = \bar{u}_j^{(t_0)} - \frac{f(u_{j+\frac{1}{2}}) - f(u_{j-\frac{1}{2}})}{\Delta x_j} \Delta t. \tag{9}$$

where $u_{j+\frac{1}{2}}$ is the function value at the cell boundary $x_{j+1/2}$ computed by

$$u_{j+1/2} = h_{j+\frac{1}{2}}(x_{j+\frac{1}{2}}) = \tilde{h}_{j+\frac{1}{2}}(s(x_{j+\frac{1}{2}})). \tag{10}$$

When the computing grid is uniform, (10) is simply $u_{j+1/2} = \tilde{h}_{j+\frac{1}{2}}(0)$.

Figure 1b describes this step.

Remark 1. The Euler time integration used for updating solution values on solution points is simple and easy to implement. However, the semi-Lagrangian time integration scheme [26] might be robust with a larger time step.

2.2.3. Step 3: Updating Model Variables

The cell-averaged variable \bar{u}_j has been updated (9). As for the point-value model variables, the four updated solution values on I_j suffice to reconstruct a Lagrange interpolation polynomial $l_j(x)$, so as to obtain the updated model variables $u_j^{(t_1)}$ and $(u_x)_j^{(t_1)}$. However, such a direct routine may cause the mismatch of u_j and \bar{u}_j .

Similar to Step 1, we introduce a local coordinate $\zeta \in [-1, 1]$:

$$\zeta(x) := \frac{x - x_j}{\Delta x_j}, \text{ for } x \in I_j. \tag{11}$$

Let $\tilde{l}_j(\zeta) = \sum_{i=0}^3 b_i \zeta^i$ be $l_j(x)$ under the local coordinate ζ . Then all conditions for $\tilde{l}_j(\zeta)$ are listed as

$$\tilde{l}_j(\zeta_k) = \sum_{i=0}^3 b_i \zeta_k^i = u_{j,k}^{(t_1)}, \text{ for } k = 1, \dots, 4, \tag{12}$$

$$\frac{1}{2} \int_{-1}^1 \tilde{l}_j(\zeta) d\zeta = b_0 + \frac{1}{3} b_2 = \bar{u}_j^{(t_1)}. \tag{13}$$

This is an over-determined linear system since it contains 5 conditions and 4 unknowns. However, (13) must hold for ensuring the conservation law. In other words, we need to find the optimal coefficients $\{b_i\}_{i=1,\dots,4}$ to fit (12) under the constraint (13). This is a typical constrained least-squares problem. To solve this, we rewrite the constraint (13) as

$$b_2 = 3(\bar{u}_j^{(t_1)} - b_0), \tag{14}$$

and plug it into (12). Then the constrained least-squares problem is cast into the standard least-squares problem as

$$A\mathbf{b} = \mathbf{y}, \tag{15}$$

where

$$A = \begin{bmatrix} 1 - \zeta_1^2 & \zeta_1 & \zeta_1^3 \\ \vdots & \vdots & \vdots \\ 1 - \zeta_4^2 & \zeta_4 & \zeta_4^3 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_3 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} u_{j,1}^{(t_1)} \\ \vdots \\ u_{j,4}^{(t_1)} \end{bmatrix} - 3\bar{u}_j^{(t_1)} \begin{bmatrix} \zeta_1^2 \\ \vdots \\ \zeta_4^2 \end{bmatrix}.$$

Then the solution of (15) is

$$\mathbf{b} = A^\dagger \mathbf{y} = (A^T A)^{-1} A^T \mathbf{y}, \tag{16}$$

where A^\dagger is the generalized inverse or the Moore–Penrose inverse [27] of non-square matrix A .

From the polynomial expression of $l_j(x)$, we can now retrieve the point-value model variables for the next time level:

$$u_j^{(t_1)} = l_j(x_j) = \tilde{l}_j(0) = b_0, \tag{17}$$

$$(u_x)_j^{(t_1)} = \frac{\partial l_j(x_j)}{\partial x} = \frac{\partial \tilde{l}_j(0)}{\partial \zeta} \frac{\partial \zeta}{\partial x} = \frac{b_1}{\Delta x_j}. \tag{18}$$

So far, (9), (17) and (18) together give the updating rule for all model variables.

This polynomial reconstruction of $l_j(x)$ based on solving constrained squares problem (13) can be efficiently implemented if we compute and store A^\dagger in advance.

2.3. Runge–Kutta Time Integration

Up to this point, we have only presented AltPoly using the Euler forward method, which is merely of first-order temporal accuracy. Since the presented AltPoly is not in the semi-discretization form, the Runge–Kutta algorithm cannot be directly applied for AltPoly. Therefore, a slight modification of the Runge–Kutta algorithm is adopted here.

To proceed, we introduce the following notation. Let \mathbf{u} be a compounded list of all model variables, and its j -th entry is a vector $\mathbf{u}_j = \{u_j, (u_x)_j, \bar{u}_j\}$. Denote the result of AltPoly upon \mathbf{u} after one step with length Δt of the temporal forward Euler method by $\mathcal{E}(\mathbf{u}, \Delta t)$ and the corresponding increment part is denoted by

$$\mathcal{D}(\mathbf{u}, \Delta t) = \mathcal{E}(\mathbf{u}, \Delta t) - \mathbf{u}. \tag{19}$$

Then, one time step of the Runge–Kutta procedure for AltPoly is

$$\mathcal{R}(\mathbf{u}^{(t_0)}) = \mathbf{u}^{(t_0)} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{20}$$

where

$$k_1 = \mathcal{D}(u^{(t_0)}, \Delta t), \tag{21a}$$

$$k_2 = \mathcal{D}(u^{(t_0)} + \frac{1}{2}k_1, \frac{1}{2}\Delta t), \tag{21b}$$

$$k_3 = \mathcal{D}(u^{(t_0)} + \frac{1}{2}k_2, \frac{1}{2}\Delta t), \tag{21c}$$

$$k_4 = \mathcal{D}(u^{(t_0)} + k_3, \Delta t). \tag{21d}$$

2.4. Artificial Viscosity

To tackle problems that involve discontinuities such as shocks and contact discontinuities, we add artificial viscosities to the proposed method. The critical step is depicting the smoothness of a cell and determining whether the discontinuity exists. Our shock sensor borrows the idea from the sub-cell shock capturing for DG [28].

Within each cell, we define the following *truncated cell average* based on the moments for AltPoly- r :

$$\bar{u}_j^{\text{trun}} = \sum_{k=0}^{\tilde{k}} \frac{1}{2k+1} \left(\frac{\partial^{2k} u}{\partial x^{2k}} \right)_j (\Delta x_j)^{2k}, \tag{22}$$

where $2\tilde{k}$ is the largest even number smaller than r . It can be seen that \bar{u}_j^{trun} utilizes the even-order partial differential point-value moments and is an approximation of \bar{u}_j with the error of magnitude $\mathcal{O}(\Delta x_j^{2\tilde{k}+2})$. Therefore, we expect that the solution is continuous and then \bar{u}_j^{trun} is close enough to \bar{u}_j , and hence declaim that a discontinuity exists in I_j if $|\bar{u}_j - \bar{u}_j^{\text{trun}}|$ exceeds some threshold. Based on this, the following smoothness indicator is defined:

$$\theta_j = \frac{|\bar{u}_j - \bar{u}_j^{\text{trun}}|}{u_{\text{range}} + \epsilon}, \tag{23}$$

where $u_{\text{range}} = u_{\text{max}} - u_{\text{min}}$ and the parameter $\epsilon = 10^{-10}$ is added to avoid dividing zero. This smoothness indicator has a simple form and can be calculated directly from the model variables. Then the amount of viscosity at $x_{j+\frac{1}{2}}$ is calculated by the following smooth function,

$$v_j = \begin{cases} 0, & \text{if } \vartheta_j \in (-\infty, \vartheta); \\ \frac{\tilde{v}}{2} \left(1 + \sin \frac{\pi(\vartheta_j - \vartheta)}{\kappa - \vartheta} \right) & \text{if } \vartheta_j \in [\vartheta, \kappa]; \\ \tilde{v} & \text{if } \vartheta_j \in (\kappa, +\infty), \end{cases} \tag{24}$$

where $\vartheta_j = \log_{10} \theta_j$, the parameters $\vartheta \sim (\Delta x_j)^{2\tilde{k}}$, $\tilde{v} = \Delta x_j/4$, and κ is large enough to maintain a sharp but smooth shock profile.

So far, we have obtained the viscosity at the middle of each cell. Next we use the linear interpolation to construct the viscosity function between the middle points of two adjacent cells. Then, to solve the problem with this viscosity function, one only needs to replace $f(u)$ with $f_{\text{vis}}(u, u_x) = f(u) - \nu(x)u_x$, and modify (8) and (9), respectively.

3. Fourier (von Neumann) Analysis

This part gives the Fourier analyses of stability and accuracy for AltPoly in solving a linear advection equation. We only give a detailed analysis for AltPoly-3 for concreteness. AltPoly- r with other choices of r can be analyzed following a similar routine.

3.1. Formulation of the Amplification Matrix

Consider the linear scalar equation

$$u_t + u_x = 0. \tag{25}$$

Suppose the computational domain is $[0, L]$ with a uniform mesh spacing Δx . The initial condition is periodic: $u_{\text{init}}(x) = e^{iwx/\Delta x}$ where i denotes the imaginary unit, and

$w = 2\pi k\Delta x/L \in [0, \pi)$ is the scaled wavenumber. The initial values of model variables on the cell $I_j = [x_{j-1/2}, x_{j+1/2}]$ and its neighbor cells are given by

$$u_j = e^{iw x_j/\Delta x}, \quad u_{j\pm 1} = e^{\pm iw} u_j, \tag{26a}$$

$$(u_x)_j = \frac{iw}{\Delta x} e^{iw x_j/\Delta x}, \quad (u_x)_{j\pm 1} = e^{\pm iw} (u_x)_j, \tag{26b}$$

$$\bar{u}_j = \frac{1}{iw} (e^{iw/2} - e^{-iw/2}) e^{iw x_j/\Delta x}, \quad \bar{u}_{j\pm 1} = e^{\pm iw} \bar{u}_j. \tag{26c}$$

The discrete Fourier transform of the series $\{u_j\}$ is

$$\{\hat{u}_{k'} | \hat{u}_{k'} = \sum_j u_j e^{-2\pi i k' x_j/L}\}.$$

$\hat{u}_{k'}$ is not 0 only when $k' = k$ under the given initial condition. Hence, we only need to consider the Fourier coefficient \hat{u}_k and denote it as \hat{u} in the following text for simplicity. \hat{u}_x and $\hat{\bar{u}}$ are defined similarly for series $\{(u_x)_j\}$ and $\{\bar{u}_j\}$.

The key procedure in our analysis is rewriting the AltPoly-3 formulation in Section 2 into the matrix form:

$$\hat{\mathbf{u}}^{(t_1)} = \mathbf{S} \hat{\mathbf{u}}^{(t_0)}, \tag{27}$$

where $\hat{\mathbf{u}}^{(t_0)}$ is defined as $\hat{\mathbf{u}}^{(t_0)} = [\hat{u}^{(t_0)}, \hat{u}_x^{(t_0)} \Delta x/2, \hat{\bar{u}}_j^{(t_0)}]$, and \mathbf{S} denotes the amplification matrix connecting $\hat{\mathbf{u}}^{(t_1)}$ at the next time level $t = t_1 = t_0 + \Delta t$.

The coefficient vector $\mathbf{a}_{j\pm 1/2}$ of the reconstructed polynomial $\tilde{h}(s)$ in Step 1 of AltPoly-3 is

$$\mathbf{a}_{j\pm 1/2} = \mathbf{H}^{-1} \mathbf{d}_{j\pm 1/2} = \mathbf{H}^{-1} \mathbf{P} \hat{\mathbf{u}}^{(t_0)} e^{iw x_j/\Delta x} \tag{28}$$

where $\mathbf{d}_{j-1/2} = [u_{j-1}^{(t_0)}, (u_x)_{j-1}^{(t_0)} \Delta x/2, u_j^{(t_0)}, (u_x)_j^{(t_0)} \Delta x/2]^T$,

$$\mathbf{P} = \begin{bmatrix} e^{-iw} & 0 & 0 \\ 0 & e^{-iw} & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{29}$$

According to (26), we have $\mathbf{d}_{j+1/2} = e^{iw} \mathbf{d}_{j-1/2}$ and $\mathbf{a}_{j+1/2} = e^{iw} \mathbf{a}_{j-1/2}$ according to the periodicity of the spatial profile. According to (10), the function values at the cell boundaries $x_{j\pm 1/2}$ are

$$u_{j-1/2}(0) = \mathbf{e}_1^T \mathbf{a}_{j-1/2}, \quad u_{j+1/2}(0) = e^{iw} u_{j-1/2}, \tag{30}$$

where \mathbf{e}_k denotes the k -th standard unit vector.

Now we turn to Step 2 of AltPoly-3. As the solution points $x_{j,1}$ and $x_{j,2}$ are in the domain of $h_{j-1/2}(x)$, we transfer them into the local coordinate in $[x_{j-1}, x_j]$. Let $s_1 = s(x_{j,1}) = 0$ and $s_2 = s(x_{j,2}) = \alpha$. Then we can derive

$$\begin{bmatrix} u_{j,1} \\ u_{j,2} \end{bmatrix} = \begin{bmatrix} 1 & s_1 & s_1^2 & s_1^3 \\ 1 & s_2 & s_2^2 & s_2^3 \end{bmatrix} \mathbf{a}_{j-\frac{1}{2}}, \quad \begin{bmatrix} (u_x)_{j,1} \\ (u_x)_{j,2} \end{bmatrix} = \frac{2}{\Delta x} \begin{bmatrix} 0 & 1 & 2s_1 & 3s_1^2 \\ 0 & 1 & 2s_2 & 3s_2^2 \end{bmatrix} \mathbf{a}_{j-\frac{1}{2}}. \tag{31}$$

Similarly, the variable value and first spatial derivatives on solution points $x_{j,3}, x_{j,4}$ can be computed by

$$\begin{bmatrix} u_{j,3} \\ u_{j,4} \end{bmatrix} = \begin{bmatrix} 1 & s_3 & s_3^2 & s_3^3 \\ 1 & s_4 & s_4^2 & s_4^3 \end{bmatrix} \mathbf{a}_{j+\frac{1}{2}}, \quad \begin{bmatrix} (u_x)_{j,3} \\ (u_x)_{j,4} \end{bmatrix} = \frac{2}{\Delta x} \begin{bmatrix} 0 & 1 & 2s_3 & 3s_3^2 \\ 0 & 1 & 2s_4 & 3s_4^2 \end{bmatrix} \mathbf{a}_{j+\frac{1}{2}}, \tag{32}$$

where $s_3 = -\alpha$ and $s_4 = 0$.

Combining $a_{j+1/2} = e^{iw} a_{j-1/2}$, the updating of solution values on solution points can be expressed in the following matrix form:

$$[u_{j,m}]_{1 \leq m \leq 4} = \mathbf{W} \mathbf{B} \mathbf{a}_{j-1/2}, \quad [(u_x)_{j,m}]_{1 \leq m \leq 4} = \frac{2}{\Delta x} \mathbf{W} \mathbf{C} \mathbf{a}_{j-1/2} \tag{33}$$

where the items of \mathbf{B} , \mathbf{C} are, respectively:

$$B_{m,n} = s_m^{n-1}, \quad C_{m,n} = (n-1) s_m^{n-2} \text{ for } 1 \leq m, n \leq 4, \tag{34}$$

with s_0 defined as 1, and \mathbf{W} is a diagonal matrix with 1, $1, e^{iw}, e^{iw}$ on the diagonal line.

Since $f_u = 1$, the updated solution values here are computed by

$$[u_{j,m}^{(t_1)}]_{m=1}^4 = (\mathbf{W} \mathbf{B} - 2c \mathbf{W} \mathbf{C}) \mathbf{a}_{j-1/2}, \tag{35}$$

where the CFL number $c = |\partial_u f(u)| \Delta t / \Delta x = \Delta t / \Delta x$ for this linear problem.

As for Step 3, the updated cell average over the j th cell is

$$\bar{u}_j^{(t_1)} = \bar{u}_j - c(u_{j+1/2} - u_{j-1/2}) = \bar{u}_j - c(e^{iw} - 1) \mathbf{e}_1^T \mathbf{a}_{j-1/2}. \tag{36}$$

Solving a constrained least-squares problem we have

$$\mathbf{b} = \mathbf{A}^\dagger \left([u_{j,m}^{(t_1)}]_{1 \leq m \leq 4} - 3[\xi_m^2]_{1 \leq m \leq 4} \bar{u}_j^{(\Delta t)} \right). \tag{37}$$

From \mathbf{b} we can easily obtain point-value variables of the next time level:

$$u_j^{(t_1)} = \mathbf{e}_1^T \mathbf{b}, \quad (u_x)_j^{(t_1)} = \frac{2}{\Delta x} \mathbf{e}_2^T \mathbf{b}. \tag{38}$$

Gathering (28), (30), (35), (37) and (38), we finally obtain the amplification matrix \mathbf{S} in (27):

$$\mathbf{S} = \begin{bmatrix} \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{bmatrix} \mathbf{A}^\dagger \left((\mathbf{W} \mathbf{B} - 2c \mathbf{W} \mathbf{C} + 3c \xi (\mathbf{e}^{iw} - 1) \mathbf{e}_1^T) \mathbf{H}^{-1} \mathbf{P} - 3 \xi \mathbf{e}_3^T \right) \\ \mathbf{e}_3^T - c(\mathbf{e}^{iw} - 1) \mathbf{e}_1^T \mathbf{H}^{-1} \mathbf{P} \end{bmatrix}, \tag{39}$$

which is the basis for the accuracy and the stability analysis as follows.

3.2. Accuracy Analysis

At the beginning, we assumed AltPoly-3 only had third-order spatial accuracy since all reconstruction polynomials are of the fourth-order, which has third-order spatial accuracy for first-order derivatives. Instead, however, we observed fourth-order spatial accuracy in numerical tests. Analyzing only truncation error is not enough to explain this phenomenon. We believe that the cell-averaged moment \bar{u}_j and the constrained polynomial reconstruction help improve the spatial accuracy.

The amplification matrix \mathbf{S} can be divided into two parts,

$$\mathbf{S} = \mathbf{S}_0 + c \mathbf{S}_1, \tag{40}$$

where \mathbf{S}_0 and \mathbf{S}_1 are independent from c . Let $\hat{\mathbf{u}} = [1, iw/2, \frac{1}{iw}(e^{iw/2} - e^{-iw/2})]$. The principal eigenvalue of \mathbf{S}_0 is 1 (see Appendix B for details). We denote the corresponding eigenvector as $\boldsymbol{\psi}$, namely

$$\mathbf{S}_0 \boldsymbol{\psi} = \boldsymbol{\psi}, \tag{41}$$

and we make the last items of $\boldsymbol{\psi}$ and $\hat{\mathbf{u}}$ equal, namely $\psi_3 = \frac{1}{iw}(e^{iw/2} - e^{-iw/2})$.

With the assistance of Wolfram Mathematica[®] 12, we decompose the following terms into Taylor series

$$\boldsymbol{\psi} = \hat{\mathbf{u}} + \varepsilon_1 w^4 \mathbf{e}_1 + \mathcal{O}(w^5) \tag{42}$$

and

$$S_1 \boldsymbol{\psi} = -i w \boldsymbol{\psi} + \varepsilon_2 w^4 \mathbf{e}_2 + \mathcal{O}(w^5), \tag{43}$$

where the scalars $\varepsilon_1, \varepsilon_2$, and the vectors $\mathbf{v}_1, \mathbf{v}_2$ are independent with w .

Suppose we choose a proper c that ensures the stability, namely $\|S^i\| \leq M$ for $i \in \mathbb{N}$ and some positive constant M . Divide the time domain $[0, T]$ uniformly with step length Δt and denote $n = T/\Delta t$. It is obvious that the exact solution at $t = T$ for (53) with the initial condition $u_{\text{init}}(x) = e^{iwx/\Delta x}$ is $u_{\text{exact}} = e^{iw(x-T)/\Delta x} = e^{-incw} u_{\text{init}}(x)$. Hence the error of the numerical solution by AltPoly-3 is

$$\begin{aligned} \|S^n \hat{\mathbf{u}} - e^{-incw} \hat{\mathbf{u}}\| &= \|S^n(\hat{\mathbf{u}} - \boldsymbol{\psi} + \boldsymbol{\psi}) - e^{-incw}(\hat{\mathbf{u}} - \boldsymbol{\psi} + \boldsymbol{\psi})\| \\ &\leq \|S^n \boldsymbol{\psi} - e^{-incw} \boldsymbol{\psi}\| + \|S^n(\hat{\mathbf{u}} - \boldsymbol{\psi})\| + \|e^{-incw}(\hat{\mathbf{u}} - \boldsymbol{\psi})\| \\ &= \|S^n \boldsymbol{\psi} - e^{-incw} \boldsymbol{\psi}\| + \mathcal{O}(w^4). \end{aligned} \tag{44}$$

However,

$$\begin{aligned} S^n \boldsymbol{\psi} &= S^{n-1}((S_0 + cS_1)\boldsymbol{\psi}) = S^{n-1}(S_0 \boldsymbol{\psi} + cS_1 \boldsymbol{\psi}) \\ &= S^{n-1}(\boldsymbol{\psi} - icw \boldsymbol{\psi} + \varepsilon_2 w^4 c \mathbf{e}_2 + c\mathcal{O}(w^5)) \\ &= S^{n-1}(e^{-icw} \boldsymbol{\psi} + \mathcal{O}(cw)^2 + \varepsilon_2 w^4 c \mathbf{e}_2 + c\mathcal{O}(w^5)) \\ &= e^{-icw} S^{n-1} \boldsymbol{\psi} + S^{n-1} \mathcal{O}(c^2 w^2) + c S^{n-1}(\varepsilon_2 w^4 \mathbf{e}_2 + \mathcal{O}(w^5)) \\ &= \dots \\ &= e^{-incw} \boldsymbol{\psi} + \sum_{k=1}^n e^{-i(n-k)cw} S^{k-1} \mathcal{O}(c^2 w^2) + c \sum_{k=1}^n e^{-i(n-k)cw} S^{k-1}(\varepsilon_2 \mathbf{e}_2 w^4 + \mathcal{O}(w^5)) \end{aligned} \tag{45}$$

According to the bound result (A16) in Appendix B, $\|S^k \mathbf{e}_2\| \leq \mu_1^k + (k-1)c\mu_2^k + \mathcal{O}(w)$ with some positive constants $\mu_1, \mu_2 \in (0, 1)$. Then one can deduce that

$$\begin{aligned} \|S^n \boldsymbol{\psi} - e^{-incw} \boldsymbol{\psi}\| &\leq \sum_{k=1}^n \|S^{k-1}\| \mathcal{O}(c^2 w^2) + c \sum_{k=1}^n ((\mu_1^k + (k-1)c\mu_2^k + \mathcal{O}(w))w^4 + \mathcal{O}(w^5)) \\ &\leq n\mathcal{O}(c^2 w^2) + c\left(\frac{\mu_1}{1-\mu_1} + \frac{c\mu_2^2}{(1-\mu_2)^2}\right)\mathcal{O}(w^4) + nc\mathcal{O}(w^5). \end{aligned} \tag{46}$$

Remember that $w \sim \Delta x, c = \Delta t/\Delta x$, and $n = T/\Delta t$, then we have

$$n\mathcal{O}(c^2 w^2) = \frac{T}{\Delta t} \mathcal{O}(\Delta t^2) = \mathcal{O}(\Delta t), \tag{47a}$$

$$nc\mathcal{O}(w^5) = \frac{T}{\Delta t} \frac{\Delta t}{\Delta x} \mathcal{O}(\Delta x^5) = \mathcal{O}(\Delta x^4), \tag{47b}$$

$$c\mathcal{O}(w^4) = \frac{\Delta t}{\Delta x} \mathcal{O}(\Delta x^4) = \mathcal{O}(\Delta t \Delta x^3). \tag{47c}$$

Combining (44)–(47), we can see that the error order of AltPoly-3 is

$$\|S^n \hat{\mathbf{u}} - e^{-incw} \mathbf{u}\| = \mathcal{O}(\Delta t + \Delta t \Delta x^3 + \Delta x^4) \tag{48}$$

To ensure stability, the order of magnitude of Δt is usually not higher than that of Δx , namely $c \sim \mathcal{O}(1)$. Therefore, the truncated error (48) can be simply reduced to $\mathcal{O}(\Delta t + \Delta x^4)$. That means AltPoly-3 using the Euler time scheme has fourth-order spatial accuracy and first-order temporal accuracy.

When using RK4 as time integration strategy (20), the amplification matrix S_{RK4} for this linear equation can be expressed as

$$S_{\text{RK4}} = I + \frac{1}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4), \tag{49}$$

where

$$K_1 = (S - I), \tag{50a}$$

$$K_2 = (S - I)(I + K_1/2), \tag{50b}$$

$$K_3 = (S - I)(I + K_2/2), \tag{50c}$$

$$K_4 = (S - I)(I + K_3). \tag{50d}$$

Expanding S_{RK4} , we have

$$S_{RK4} = \frac{1}{24}(I + 8S + 6S^2 + S^4).$$

One can easily show that S_{RK4} has fifth-order truncated temporal accuracy and then the total error becomes

$$\|S_{RK4}^n \mathbf{u} - e^{-incw} \mathbf{u}\| = O(\Delta t^4 + \Delta t \Delta x^3 + \Delta x^4). \tag{51}$$

3.3. Stability Analysis

To obtain stable time integration, the CFL number c should be carefully chosen such that the magnitude of the principal eigenvalue of the amplification matrix is not greater than 1 for all wavenumbers k . The principal eigenvalue of the amplification matrix S_{RK4} for AltPoly with various moments can be seen in Figure 2. It is remarkable that all schemes have a fairly large range of c except for AltPoly-5. In particular, AltPoly-3, which has fourth-order spatial accuracy, is stable even with $c > 1$, which is better than many other existing explicit high-order schemes using a compact stencil. As more moments are used, the stability condition becomes more stringent. For AltPoly-6, the tolerated CFL number is approximately $c \in [0.2, 0.4]$. However, the maximum principal eigenvalue is around $1 + 10^{-2}$ when $c \leq 0.2$ for AltPoly-6, which means short-term time integration can still maintain stability.

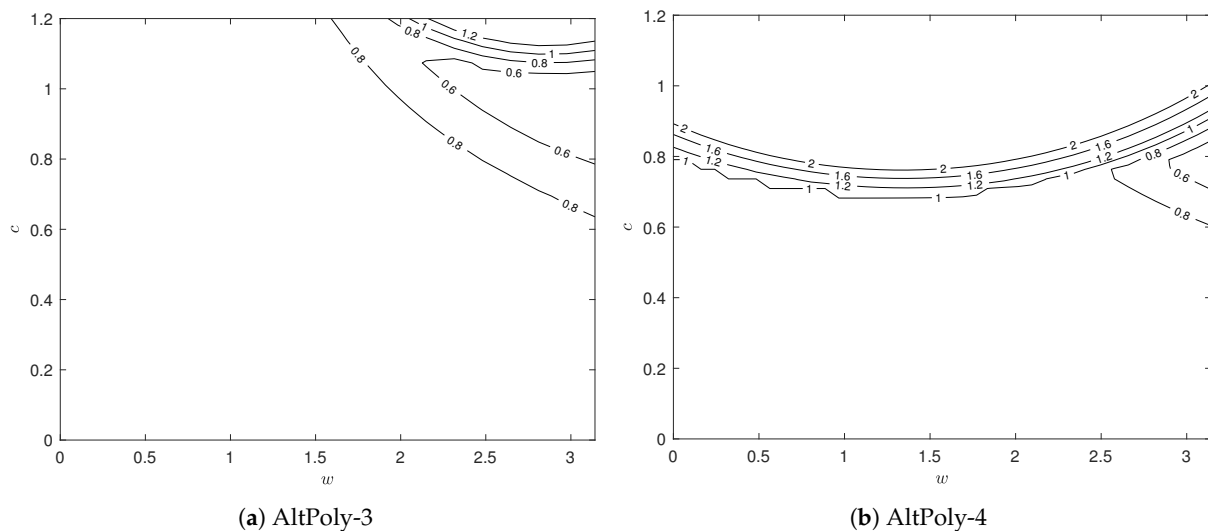


Figure 2. Cont.

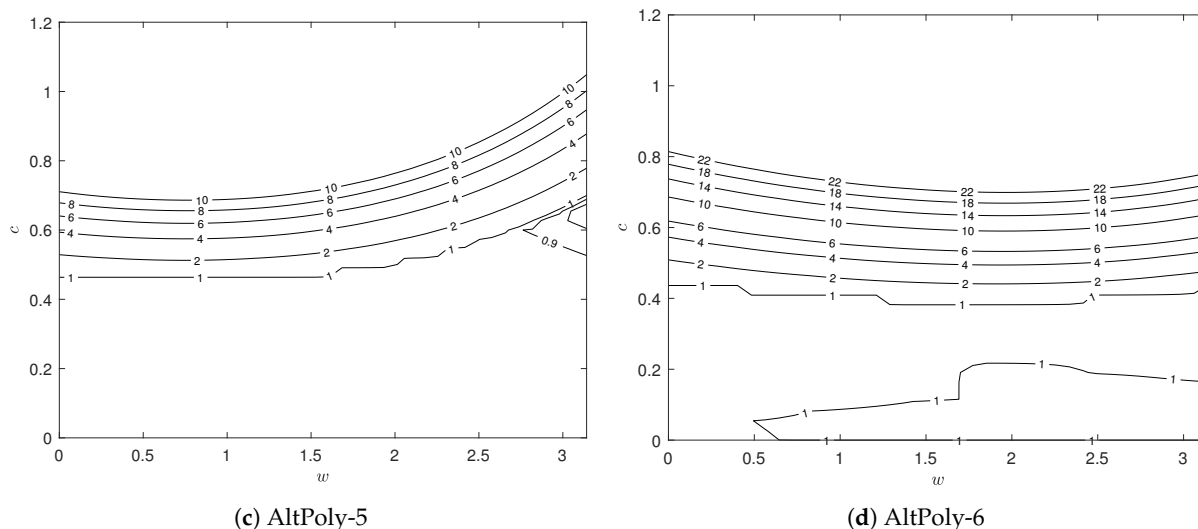


Figure 2. Contour plots of the principal eigenvalues of the amplification matrices of AltPoly schemes using fourth-order RK time integration for the linear advection equation.

4. Numerical Results

In this part, we solve several types of one-dimensional hyperbolic law to illustrate the performance of the proposed method. We choose the four-stage Runge–Kutta method (20) for the time discretization.

For all numerical experiments, AltPoly- r with the moment number r varying from 3 to 6 are tested on uniform meshes. In addition, we check the accuracy of the proposed method on non-uniform meshes. Non-uniform grids are generated by adding a 40% random perturbation upon the element boundary of the uniform mesh. Specifically, suppose $x_{j+1/2}$ and Δx are separately the cell boundary and cell length for the uniform grid, then the element boundary generated for the non-uniform grid is

$$\tilde{x}_{j+\frac{1}{2}} = x_{j+\frac{1}{2}} + \beta \zeta_j \Delta x, \quad \zeta_j \sim \text{Unif}(-1, 1), \tag{52}$$

where ζ_j are independently identically distributed random variables, $\text{Unif}(-1, 1)$ denotes the uniform distribution over $[-1, 1]$, and β is the amount of perturbation, which is set as 0.2 in our experiments.

Special attention might be paid to specify the initial values of model variables. This is trivial if the initial profile is analytically given. Otherwise, these unknowns can be specified by implementing high-order interpolations and numerical integration.

Since the temporal accuracy is only fourth-order, we take $\Delta t \sim (\Delta x)^{(2r-2)/4}$ for AltPoly- r in the accuracy tests.

4.1. Linear Scalar Equation

In this subsection, we test the performance of AltPoly schemes on the following scalar equation [29]:

$$u_t + u_x = 0. \tag{53}$$

We first check the accuracy of the proposed method through grid refinement tests. The initial condition is given as $u(x, 0) = \sin(\pi x)$ with a periodical boundary condition on $x \in [0, 2\pi)$. The exact solution is $u(x, t) = \sin(x - t)$. We refine the grid and record numerical errors after one period ($t = 2$).

Two typical norms for errors are adopted here, i.e., $E_1 = \frac{1}{N} \sum_{j=1}^N |u_j - u_j^{\text{true}}|$, and $E_\infty = \max_{1 \leq j \leq N} |u_j - u_j^{\text{true}}|$. The errors and the convergence rates are presented in Table 1. We can see that the spatial accuracy order is $2r - 2$ for the AltPoly- r scheme, which matches

our theoretical analysis in Section 3. Additionally, the order of accuracy is also maintained well for non-uniform meshes.

Table 1. Accuracy test for linear hyperbolic law with initial condition $u(x, 0) = \sin(x)$ at $t = 2\pi$.

r	N	L_1 Error	Order	L_∞ Error	Order	L_1 Error	Order	L_∞ Error	Order
Uniform Grid					Non-Uniform Grid				
3	10	1.79×10^{-3}	–	2.76×10^{-3}	–	2.10×10^{-3}	–	3.33×10^{-3}	–
	20	1.15×10^{-4}	3.95	1.82×10^{-4}	3.93	1.32×10^{-4}	4.00	2.09×10^{-4}	3.99
	30	2.31×10^{-5}	3.96	3.63×10^{-5}	3.97	2.68×10^{-5}	3.93	4.25×10^{-5}	3.93
	40	7.35×10^{-6}	3.98	1.16×10^{-5}	3.97	8.46×10^{-6}	4.00	1.35×10^{-5}	3.99
	60	1.46×10^{-6}	3.99	2.29×10^{-6}	3.99	1.68×10^{-6}	3.98	2.67×10^{-6}	3.99
4	10	1.32×10^{-4}	–	2.05×10^{-4}	–	1.33×10^{-4}	–	2.06×10^{-4}	–
	20	2.06×10^{-6}	6.01	3.22×10^{-6}	5.99	2.06×10^{-6}	6.01	3.23×10^{-6}	5.99
	30	1.81×10^{-7}	5.99	2.84×10^{-7}	5.98	1.82×10^{-7}	5.99	2.86×10^{-7}	5.98
	40	3.23×10^{-8}	6.00	5.06×10^{-8}	6.00	3.24×10^{-8}	6.00	5.08×10^{-8}	6.00
	60	2.84×10^{-9}	6.00	4.45×10^{-9}	5.99	2.85×10^{-9}	5.99	4.48×10^{-9}	5.99
5	10	1.92×10^{-4}	–	2.97×10^{-4}	–	1.92×10^{-4}	–	2.98×10^{-4}	–
	20	7.60×10^{-7}	7.98	1.18×10^{-6}	7.97	7.60×10^{-7}	7.98	1.18×10^{-6}	7.98
	30	2.95×10^{-8}	8.01	4.63×10^{-8}	8.00	2.95×10^{-8}	8.01	4.63×10^{-8}	7.99
	40	2.97×10^{-9}	7.99	4.64×10^{-9}	7.99	2.96×10^{-9}	7.99	4.65×10^{-9}	7.99
	60	1.16×10^{-10}	8.00	1.81×10^{-10}	8.00	1.16×10^{-10}	8.00	1.82×10^{-10}	8.00
6	20	1.57×10^{-6}	–	2.46×10^{-6}	–	1.57×10^{-6}	–	2.46×10^{-6}	–
	30	2.73×10^{-8}	10.00	4.28×10^{-8}	9.99	2.73×10^{-8}	9.99	4.28×10^{-8}	9.99
	40	1.54×10^{-9}	9.99	2.41×10^{-9}	9.99	1.54×10^{-9}	9.99	2.41×10^{-9}	9.99
	60	2.49×10^{-11}	10.17	3.96×10^{-11}	10.14	2.28×10^{-11}	10.39	3.82×10^{-11}	10.22

Next, we use a complicated initial condition from [30] to evaluate the ability of the proposed method in handling both continuous and discontinuous profiles. This initial condition contains four piece-wise functions:

$$u(x, 0) = \begin{cases} \frac{1}{6}(G(x, \beta, z - \delta) + G(x, \beta, z + \delta) + 4G(x, \beta, z)) & \text{for } x \in [-0.8, -0.6], \\ 1 & \text{for } x \in [-0.4, -0.2], \\ 1 - |10(x - 0.1)|, & \text{for } x \in [0, 0.2], \\ \frac{1}{6}(F(x, \alpha, a - \delta) + F(x, \alpha, a + \delta) + 4F(x, \alpha, a)) & \text{for } x \in [0.4, 0.6], \\ 0, & \text{otherwise,} \end{cases} \quad (54)$$

where the involved functions are

$$G(x, \beta, z) = \exp(-\beta(x - z)^2)$$

and

$$F(x, \alpha, a) = \sqrt{\max(1 - \alpha^2(x - a)^2, 0)},$$

with the constants set as $\alpha = 10, a = 0.5, z = -0.7, \delta = 0.005$, and $\beta = \log 2 / (36\delta^2)$. As for the initial profile, we suggest using the routine used in the discontinuous Galerkin method.

A smaller time step can help reduce the oscillations around discontinuities. We take $\Delta t = 0.3\Delta x, 0.1\Delta x, 0.075\Delta x$, and $0.01\Delta x$, respectively, for AltPoly-3, AltPoly-4, AltPoly-5, and AltPoly-6.

The computational domain is divided into 200 elements. We first compute this problem without adding artificial viscosity, for which higher-order methods can produce better results in both continuous and discontinuous regions. The results at $t = 2$ are illustrated

in Figures 3 and 4 separately. Specifically, AltPoly-3 and AltPoly-4 generate significant oscillations around $x = -0.4$ and $x = -0.2$. These oscillations are reasonable as our method is based on the smoothness assumption and the polynomial does have its limitations on presenting discontinuities. However, it is interesting that spurious oscillations are significantly reduced and the transitions around strong discontinuities are quite sharp for AltPoly-5 and AltPoly-6. After adding a proper amount of artificial viscosity, the proposed method is able to smooth the artificial oscillations for AltPoly method of various orders while not losing high accuracy in the smooth regions.

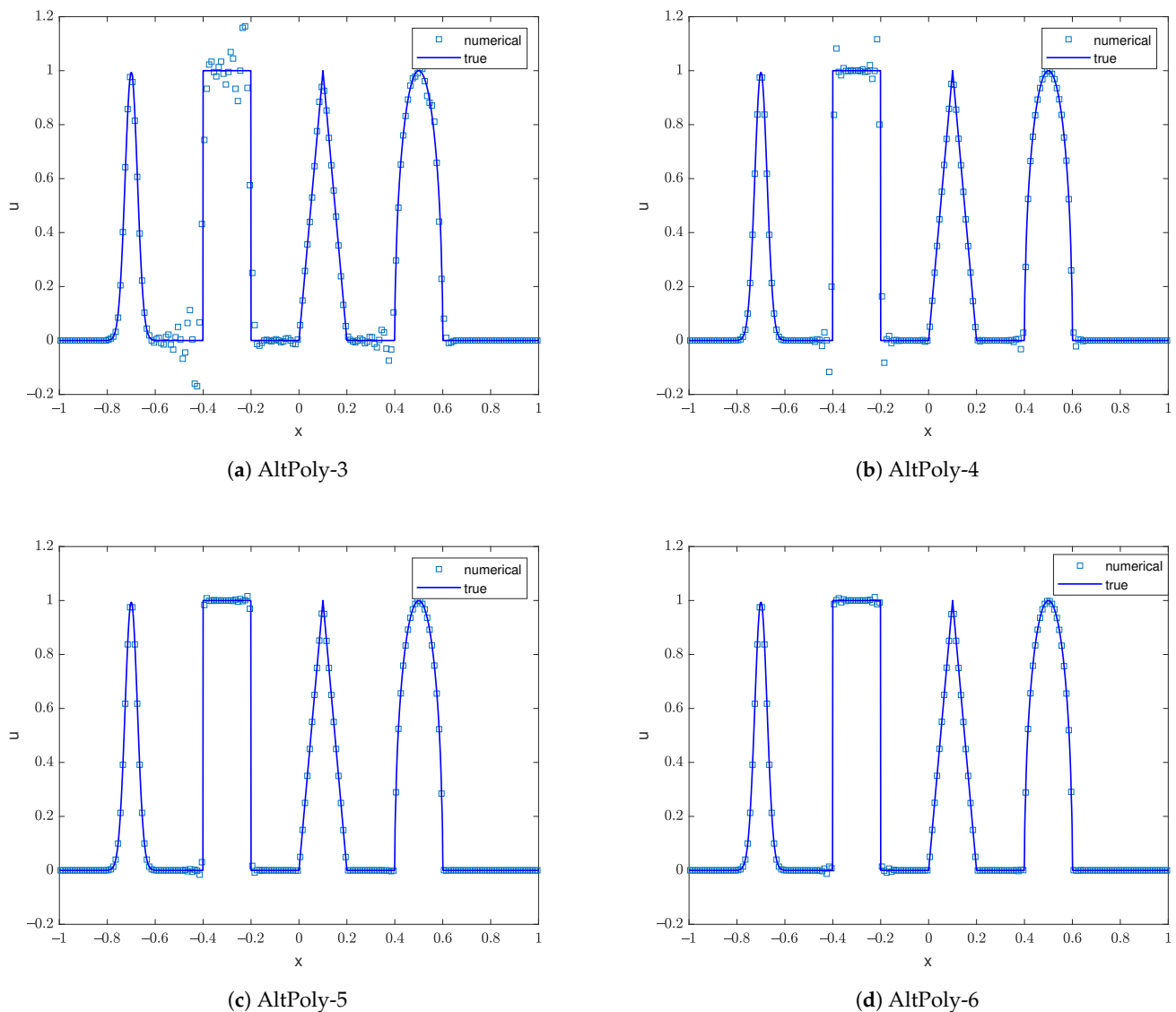


Figure 3. Numerical results of the advection of a complicated profile at $t = 2$ via AltPoly without artificial viscosity using $N = 200$ cells.

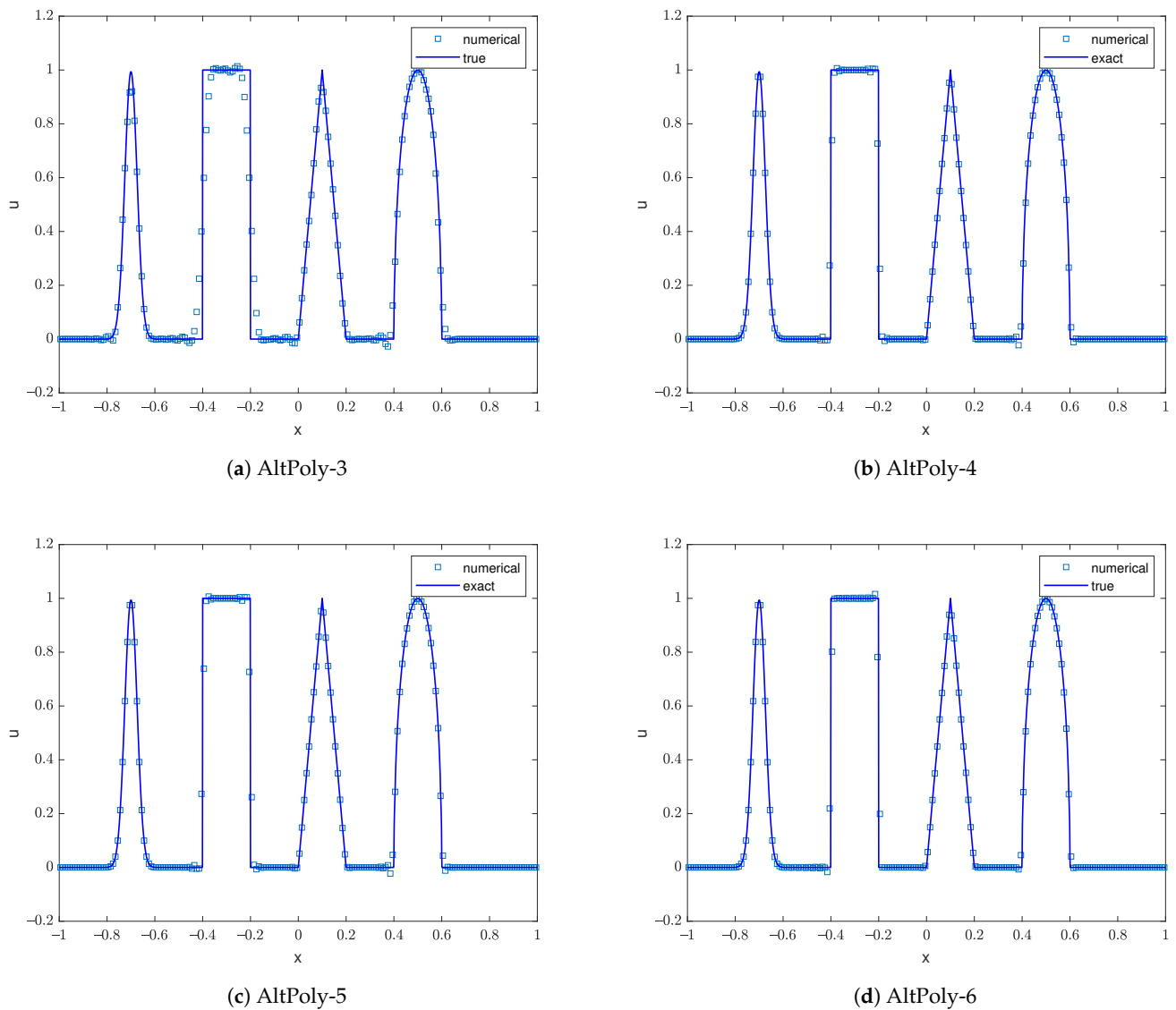


Figure 4. Numerical results of the advection of a complicated profile at $t = 2$ via AltPoly with artificial viscosity using $N = 200$ cells.

4.2. Nonlinear Burgers Equation

This subsection considers the 1D inviscid Burgers equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \tag{55}$$

The initial condition is $u(x,0) = 0.5 + \sin(x)$ for $x \in [0, 2\pi)$ with the periodical boundary condition.

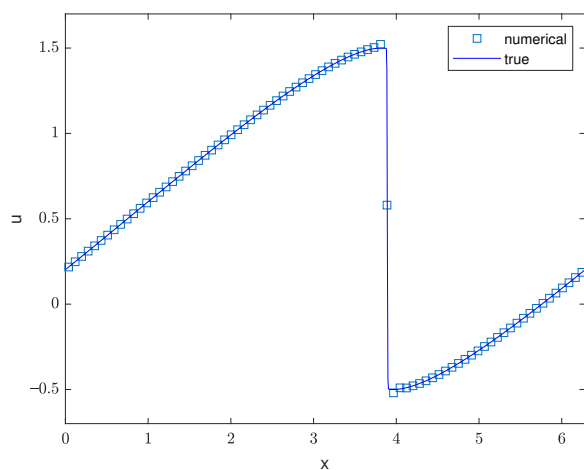
Although the initial profile is smooth, shock waves exist when t is large enough since the flux function is nonlinear. To evaluate the order of accuracy, we calculate the numerical solution before the shock wave occurs and compute the errors in respect of grid refinement. Table 2 lists the errors at $t = 0.5$ when the solution is still smooth. The numerical convergence rate results are consistent with our theoretical analysis.

The numerical results for the Burgers equation at $t = 1.5$ are shown in Figure 5 when a shock has already emerged. To handle the shock discontinuity, we add a proper amount of artificial viscosity according to Section 2.4. We can see that our schemes sup-

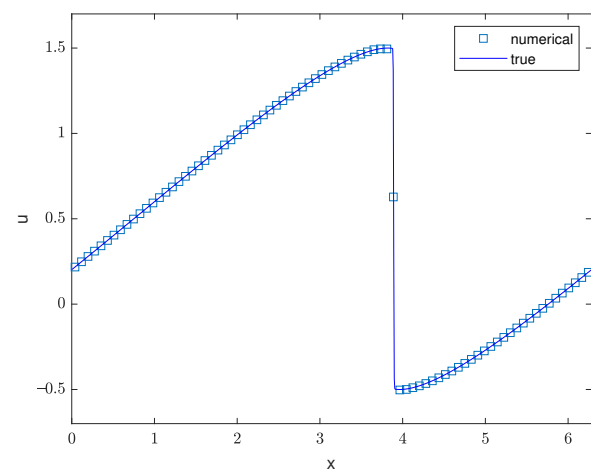
press the oscillations well around the discontinuity while still maintaining a sharp and accurate profile.

Table 2. Accuracy test for Burgers equation with $u(x, 0) = 0.5 + \sin(x)$ at $t = 0.5$.

r	N	L_1 Error	Order	L_∞ Error	Order	L_1 Error	Order	L_∞ Error	Order
Uniform Grid					Non-Uniform Grid				
3	20	8.41×10^{-6}	—	1.32×10^{-5}	—	2.01×10^{-5}	—	5.46×10^{-5}	—
	30	1.59×10^{-6}	4.10	2.50×10^{-6}	4.11	2.99×10^{-6}	4.70	9.14×10^{-6}	4.41
	40	4.93×10^{-7}	4.08	7.71×10^{-7}	4.08	8.52×10^{-7}	4.37	2.54×10^{-6}	4.45
	60	9.52×10^{-8}	4.05	1.50×10^{-7}	4.04	1.90×10^{-7}	3.70	5.71×10^{-7}	3.68
	80	2.98×10^{-8}	4.03	4.69×10^{-8}	4.03	5.39×10^{-8}	4.38	1.74×10^{-7}	4.12
4	20	1.98×10^{-5}	—	1.21×10^{-4}	—	2.11×10^{-5}	—	1.29×10^{-4}	—
	30	2.18×10^{-6}	5.43	1.83×10^{-5}	4.66	2.13×10^{-6}	5.65	1.73×10^{-5}	4.96
	40	4.24×10^{-7}	5.70	3.60×10^{-6}	5.65	4.16×10^{-7}	5.68	3.53×10^{-6}	5.52
	60	3.58×10^{-8}	6.09	2.94×10^{-7}	6.17	3.65×10^{-8}	6.00	2.99×10^{-7}	6.09
	80	6.38×10^{-9}	5.99	5.41×10^{-8}	5.89	6.54×10^{-9}	5.97	5.28×10^{-8}	6.03
5	20	8.48×10^{-5}	—	5.80×10^{-4}	—	9.04×10^{-5}	—	5.55×10^{-4}	—
	30	4.07×10^{-6}	7.49	3.04×10^{-5}	7.28	3.90×10^{-6}	7.75	3.01×10^{-5}	7.19
	40	4.41×10^{-7}	7.73	3.40×10^{-6}	7.61	4.39×10^{-7}	7.60	3.38×10^{-6}	7.60
	60	1.83×10^{-8}	7.84	1.42×10^{-7}	7.83	1.84×10^{-8}	7.82	1.43×10^{-7}	7.80
	80	1.81×10^{-9}	8.04	1.49×10^{-8}	7.85	1.81×10^{-9}	8.06	1.48×10^{-8}	7.87
6	20	4.54×10^{-4}	—	2.74×10^{-3}	—	4.68×10^{-4}	—	2.73×10^{-3}	—
	30	1.30×10^{-5}	8.77	1.00×10^{-4}	8.15	1.29×10^{-5}	8.86	1.00×10^{-4}	8.15
	40	7.54×10^{-7}	9.89	5.87×10^{-6}	9.87	7.62×10^{-7}	9.83	5.85×10^{-6}	9.87
	60	1.32×10^{-8}	9.98	1.02×10^{-7}	9.99	1.32×10^{-8}	10.00	1.03×10^{-7}	9.97
	80	7.38×10^{-10}	10.03	6.09×10^{-9}	9.81	7.67×10^{-10}	9.90	6.09×10^{-9}	9.82



(a) AltPoly-3



(b) AltPoly-4

Figure 5. Cont.

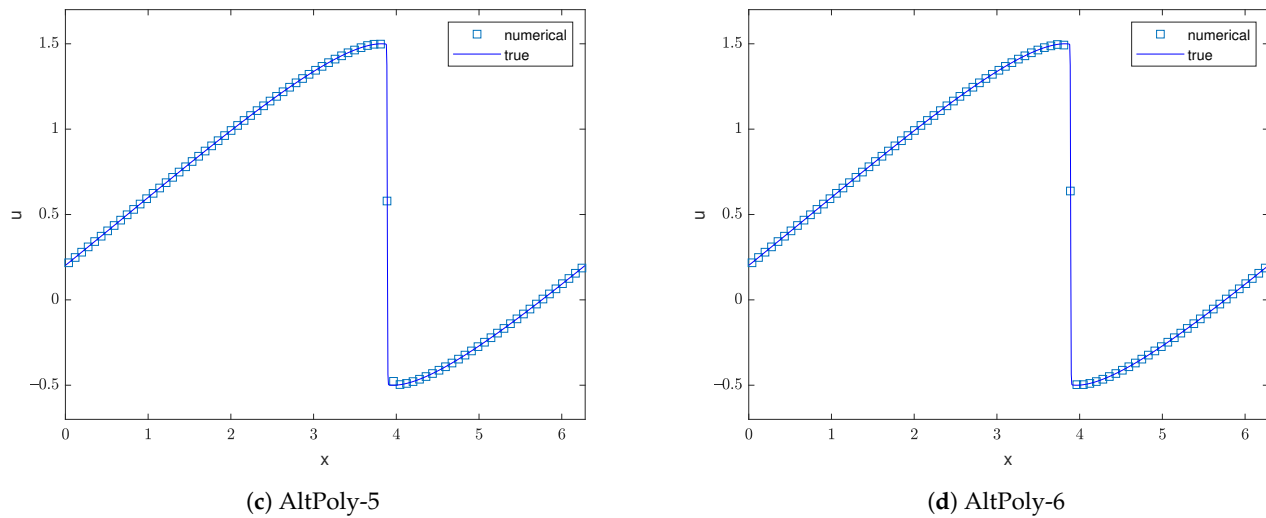


Figure 5. Numerical results of the Burgers equation at $t = 1.5$ with $N = 80$ cells and $\Delta t = 0.15\Delta x$.

4.3. Euler Equations

We then use our scheme to solve the one-dimensional Euler system:

$$\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ m \\ e \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} m \\ \rho u^2 + p \\ eu + pu \end{pmatrix} = \mathbf{0}, \tag{56}$$

where ρ refers to the density, u the velocity, $m = \rho u$ the momentum, e the total energy, and p the pressure:

$$p = (\gamma - 1) \left(e - \frac{1}{2} \rho u^2 \right) \tag{57}$$

with $\gamma = 1.4$ for a perfect dry gas. Just like in the scalar case, we introduce point-valued and cell-averaged moments for each variable. The cell-averaged moments are updated by the flux-form Equation (56). In Step 2 of AltPoly, solution values on auxiliary points are updated using the following non-conservative form equations:

$$\rho_t = -m_x, \tag{58}$$

$$m_t = -(m_x u + m u_x + p_x), \tag{59}$$

$$e_t = -(e_x u + e u_x + p_x u + p u_x), \tag{60}$$

where

$$u_x = \frac{\partial}{\partial x} \left(\frac{m}{\rho} \right) = \frac{m_x}{\rho} - \frac{m \rho_x}{\rho^2},$$

$$p_x = (\gamma - 1) \left(e_x - \frac{1}{2} (m_x u + m u_x) \right).$$

Neither characteristic decomposition nor Riemann solver are needed here.

We first check the order of accuracy for this nonlinear system by simulating advection of the density perturbation. The initial conditions are $\rho(x_0) = 1 + 0.2 \sin x$, $v(x, 0) = 1$, and $p(x, 0) = 1$ for $x \in [0, 2\pi)$ with the periodic boundary condition. The exact solution is a traveling wave: $\rho(x, t) = 1 + 0.2 \sin(x - t)$, $v(x, t) = 1$ and $p(x, t) = 1$. The error results of the density are listed in Table 3 and a satisfactory convergence rate is obtained.

Table 3. Accuracy test for the one-dimensional Euler equation with $u(x, 0) = 0.5 + \sin(x)$ at $t = 2\pi$.

r	N	L_1 Error	Order	L_∞ Error	Order	L_1 Error	Order	L_∞ Error	Order
Uniform Grid					Non-Uniform Grid				
3	10	3.26×10^{-4}	—	5.03×10^{-4}	—	3.88×10^{-4}	—	6.22×10^{-4}	—
	15	6.09×10^{-5}	4.13	9.55×10^{-5}	4.10	7.22×10^{-5}	4.15	1.16×10^{-4}	4.15
	20	1.84×10^{-5}	4.16	2.85×10^{-5}	4.21	2.21×10^{-5}	4.12	3.50×10^{-5}	4.16
	30	3.45×10^{-6}	4.13	5.41×10^{-6}	4.09	4.32×10^{-6}	4.03	6.93×10^{-6}	3.99
	40	1.07×10^{-6}	4.09	1.67×10^{-6}	4.09	1.30×10^{-6}	4.18	2.09×10^{-6}	4.17
4	10	1.12×10^{-5}	—	1.72×10^{-5}	—	1.20×10^{-5}	—	2.40×10^{-5}	—
	15	9.81×10^{-7}	6.00	1.54×10^{-6}	5.96	9.91×10^{-7}	6.16	1.55×10^{-6}	6.75
	20	1.76×10^{-7}	5.97	2.76×10^{-7}	5.98	1.78×10^{-7}	5.97	2.79×10^{-7}	5.97
	30	1.55×10^{-8}	5.99	2.43×10^{-8}	5.99	1.57×10^{-8}	5.99	2.47×10^{-8}	5.98
	40	2.77×10^{-9}	5.99	4.35×10^{-9}	5.98	2.81×10^{-9}	5.98	4.42×10^{-9}	5.98
5	10	2.55×10^{-7}	—	3.94×10^{-7}	—	2.55×10^{-7}	—	3.96×10^{-7}	—
	15	1.02×10^{-8}	7.95	1.59×10^{-8}	7.91	1.02×10^{-8}	7.95	1.59×10^{-8}	7.92
	20	1.02×10^{-9}	7.99	1.58×10^{-9}	8.03	1.02×10^{-9}	7.98	1.58×10^{-9}	8.03
	30	3.96×10^{-11}	8.01	6.22×10^{-11}	7.98	3.97×10^{-11}	8.01	6.24×10^{-11}	7.98
	40	3.98×10^{-12}	7.98	6.49×10^{-12}	7.85	3.99×10^{-12}	7.99	6.47×10^{-12}	7.88
6	10	5.08×10^{-7}	—	7.85×10^{-7}	—	5.08×10^{-7}	—	7.85×10^{-7}	—
	12	8.35×10^{-8}	9.90	1.27×10^{-7}	9.98	8.35×10^{-8}	9.90	1.27×10^{-7}	9.98
	15	8.90×10^{-9}	10.03	1.40×10^{-8}	9.91	8.91×10^{-9}	10.03	1.40×10^{-8}	9.90
	20	5.03×10^{-10}	9.99	7.90×10^{-10}	9.98	1.44×10^{-9}	10.01	2.25×10^{-9}	10.02
	24	8.34×10^{-11}	9.86	1.33×10^{-10}	9.77	5.02×10^{-10}	9.97	7.92×10^{-10}	9.90

We then consider the classical shock tube problem, which is described by Euler equations with the following Riemann-type initial distribution:

$$(\rho, v, p) = \begin{cases} (\rho_L, v_L, p_L), & 0 \leq x \leq 0.5; \\ (\rho_R, v_R, p_R), & 0.5 < x \leq 1, \end{cases} \quad x \in [0, 1]. \tag{61}$$

Two initial conditions are tested here. The first one is known as Sod’s problem and the initial condition is defined as

$$(\rho_L, v_L, p_L) = (1, 0, 1), \quad (\rho_R, v_R, p_R) = (0.125, 0, 0.1). \tag{62}$$

The second one is Lax’s problem, which starts from the following initial condition:

$$(\rho_L, v_L, p_L) = (0.445, 0.698, 3.258), \quad (\rho_R, v_R, p_R) = (0.5, 0, 0.571). \tag{63}$$

We compute these shock tube problems with 100 cells using the AltPoly schemes equipped with the artificial viscosity. Generally, a certain degree of spurious oscillation still exists but is in a tolerant range. As shown in Figure 6 and 7, all AltPoly schemes can depict shocks sharply using only one or two cells. The higher-order schemes (AltPoly-5 and AltPoly-6) capture the contact discontinuities with only one or two cells, while more cells are needed for the lower-order AltPoly schemes.

To show the accuracy of our smoothness indicator (23), Figure 8 illustrates the cells where the artificial viscosity takes effect on the x - t plane. Generally, the number of cells recognized as discontinuous for lower-order schemes is significantly smaller than that for higher-order schemes. This might be because lower-order schemes have a larger dissipation and the smoothness indicator is less accurate than that of high-order schemes. It is remarkable that the artificial viscosity vanishes in most areas and is in effect only

around the shock wave. The contact discontinuity does not trigger the smoothness indicator as it can be smoothed by the artificial viscosity in the first few time steps.

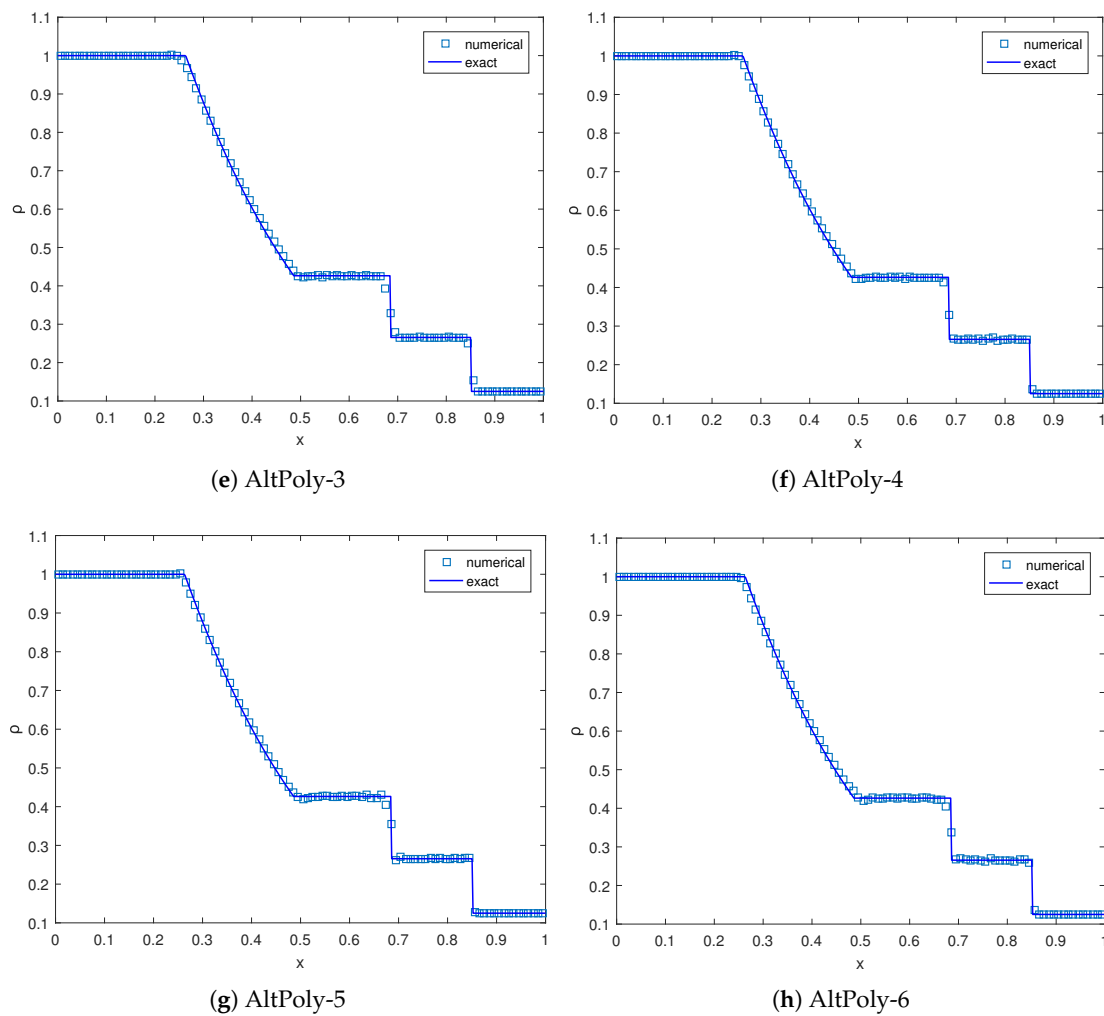


Figure 6. Numerical results of Sod's problem at $t = 0.13$ with $N = 100$ cells. $\Delta t = 0.2\Delta x, 0.15\Delta x, 0.1\Delta x,$ and $0.05\Delta x,$ respectively.

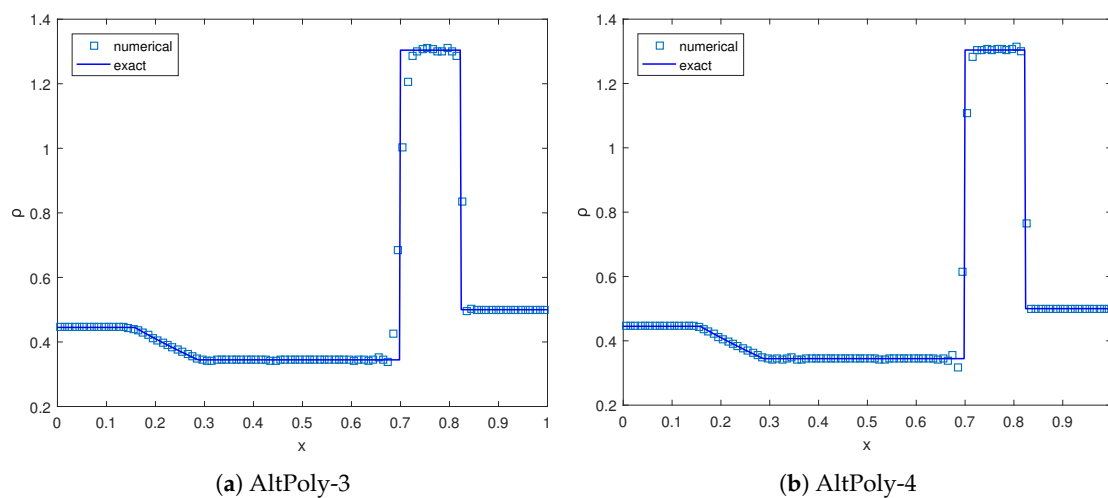


Figure 7. Cont.

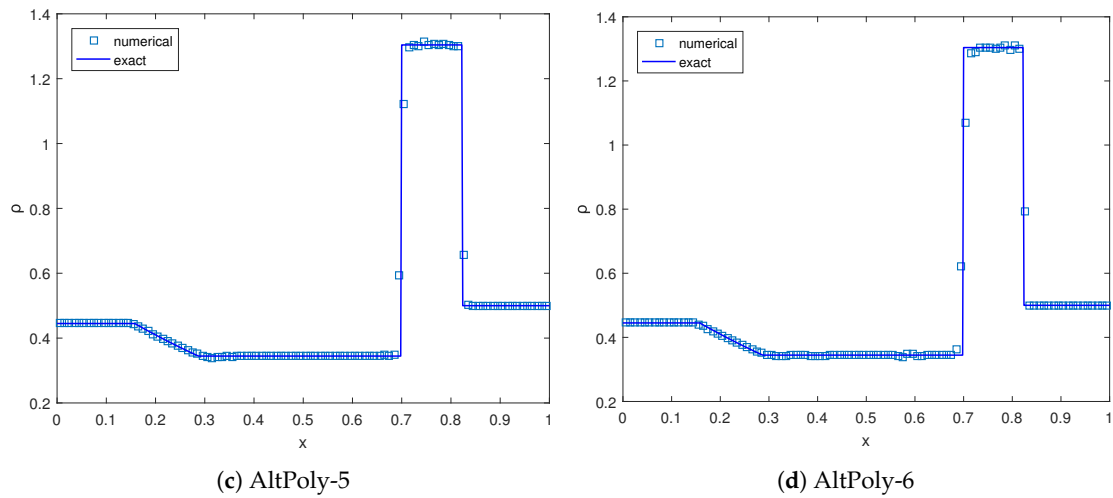


Figure 7. Numerical results of Lax’s problem at $t = 0.13$ with $N = 100$ cells. $\Delta t = 0.1\Delta x, 0.1\Delta x, 0.08\Delta x,$ and $0.05\Delta x,$ respectively.

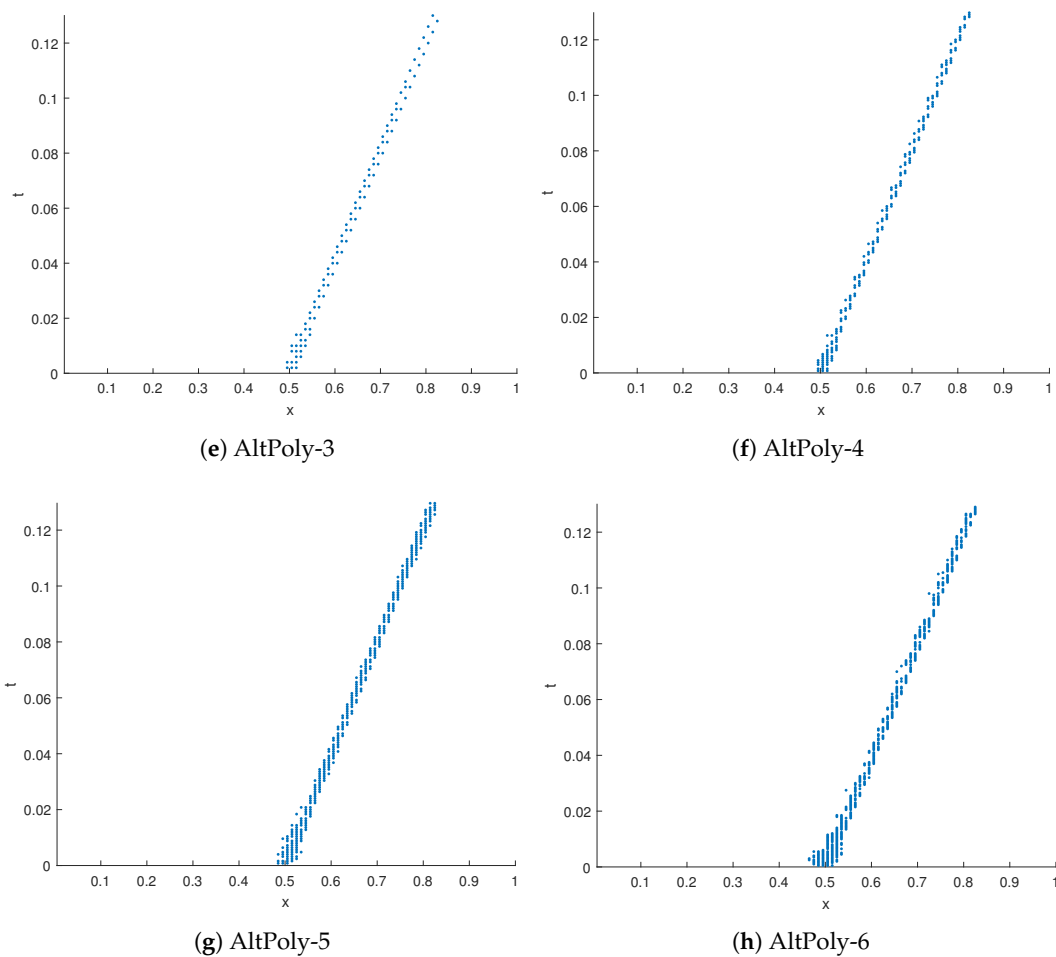


Figure 8. The cells where artificial viscosity takes effect in Lax’s problem.

5. Conclusions

We have presented a high-order conservative solver termed AltPoly for 1D hyperbolic conservation laws. This method can be categorized as a multi-moment scheme since it adopts both point-values at the middle of the cell and cell-averaged values as the model

variables. AltPoly updates the model variables via alternately implementing two polynomial reconstructions. The point-based variables are used to reconstruct a high-order accurate approximation of the solution via Hermite interpolation. The cell-averaged variables updated by the flux-form equations serve as the constraint to guarantee numerical conservation. The solution within a cell is corrected by polynomial reconstruction via solving a constrained least-squares problem. In other words, our method directly reconstructs the solution to maintain the conservation, which is different from existing schemes based on the local flux reconstruction. Fourier analysis shows the accuracy and the admissible range of the CFL number for the linear scalar equation. Further, our scheme can be extended to other kinds of PDEs such as parabolic equations, advection–diffusion equations, etc., which will be our future work.

Author Contributions: S.L., conceptualization, methodology, software, validation, writing—original draft preparation; Q.L., methodology, formal analysis, validation, writing—review and editing; H.L., supervision, methodology, writing—review and editing; J.S., project administration, resources. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundation of China No. 41605070.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Outline of AltPoly- r

Steps 1 and 2 of AltPoly- r are almost the same as those of AltPoly-3 and are hence omitted here. In this part, we focus on introducing the constrained polynomial reconstruction in Step 3 of AltPoly- r for $r \geq 3$.

Following the notations in Section 2.2.3, the core of Step 3 is reconstructing the polynomial $\tilde{l}_j(\xi) = \sum_{i=0}^{2r-3} b_i \xi^i$ from the following conditions:

$$\tilde{l}_j(\xi_k) = \sum_{i=0}^{2r-3} b_i \xi_k^i = u_{j,k}^{(t_1)}, \quad \text{for } k = 1, \dots, 2r - 2, \tag{A1}$$

$$\frac{1}{2} \int_{-1}^1 \tilde{l}_j(\xi) d\xi = b_0 + \frac{1}{3} b_2 + \dots + \frac{1}{2r-3} b_{2r-4} = \bar{u}_j^{(t_1)}, \tag{A2}$$

where $u_{j,k}^{(t_1)}$ and $\bar{u}_j^{(t_1)}$ are obtained in Step 2. Since the hyperbolic conservation law must be obeyed, we rewrite the constraint condition (A2) into the expression of b_{2r-4} ,

$$b_{2r-4} = (2r - 3)(\bar{u}_j^{(t_1)} - b_0 - \frac{1}{3} b_2 - \dots - \frac{1}{2r-5} b_{2r-6}), \tag{A3}$$

and plug it into (A1) to arrive at

$$b_0 + b_1 \xi_k + \dots + b_{2r-5} \xi_k^{2r-5} + (2r - 3)(\bar{u}_j^{(t_1)} - b_0 - \frac{1}{3} b_2 - \dots - \frac{1}{2r-5} b_{2r-6}) \xi_k^{2r-4} + b_{2r-3} \xi_k^{2r-3} = u_{j,k}^{(t_1)}. \tag{A4}$$

We rewrite it in matrix form:

$$Ab = y, \tag{A5}$$

where

$$A = A_1 - A_2,$$

$$\begin{aligned}
 A_1 &= \begin{bmatrix} 1 & \zeta_1 & \zeta_1^2 & \cdots & \zeta_1^{2r-4} & \zeta_1^{2r-3} \\ 1 & \zeta_2 & \zeta_2^2 & \cdots & \zeta_2^{2r-4} & \zeta_2^{2r-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \zeta_{2r-2} & \zeta_{2r-2}^2 & \cdots & \zeta_{2r-2}^{2r-4} & \zeta_{2r-2}^{2r-3} \end{bmatrix}, \\
 A_2 &= (2r-3) \begin{bmatrix} \zeta_1^{2r-4} & 0 & \frac{1}{3}\zeta_1^{2r-4} & 0 & \frac{1}{5}\zeta_1^{2r-4} & \cdots & 0 & \frac{1}{2r-5}\zeta_1^{2r-4} & 0 & 0 \\ \zeta_2^{2r-4} & 0 & \frac{1}{3}\zeta_2^{2r-4} & 0 & \frac{1}{5}\zeta_2^{2r-4} & \cdots & 0 & \frac{1}{2r-5}\zeta_2^{2r-4} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \zeta_{2r-2}^{2r-4} & 0 & \frac{1}{3}\zeta_{2r-2}^{2r-4} & 0 & \frac{1}{5}\zeta_{2r-2}^{2r-4} & \cdots & 0 & \frac{1}{2r-5}\zeta_{2r-2}^{2r-4} & 0 & 0 \end{bmatrix}, \\
 \mathbf{b} &= \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{2r-5} \\ b_{2r-3} \end{bmatrix}, \mathbf{y} = \begin{bmatrix} u_{j,1}^{(t_1)} \\ u_{j,2}^{(t_1)} \\ \vdots \\ u_{j,2r-2}^{(t_1)} \end{bmatrix} - (2r-3) \begin{bmatrix} \zeta_1^{2r-4} \\ \zeta_2^{2r-4} \\ \vdots \\ \zeta_{2r-2}^{2r-4} \end{bmatrix}.
 \end{aligned}$$

This linear system is overdetermined since it has $2r - 3$ unknowns and $2r - 2$ equations. The least-squares solution is

$$\mathbf{b} = A^\dagger \mathbf{y} = (A^T A)^{-1} A^T \mathbf{y}, \tag{A6}$$

where A^\dagger denotes the generalized inverse or Moore–Penrose inverse [27] of A . From \mathbf{b} we can obtain the updated point-based model variables.

In practice, we find that A^\dagger has a large condition number when $r \geq 5$ and we calculate it via symbolic computation using mathematical software to reduce the error. Throughout this paper, we used the symmetrical solution points $\{-1, -1 + \sigma_1, -1 + \sigma_2, \dots, -1 + \sigma_{r-2}, 1 - \sigma_{r-2}, 1 - \sigma_{r-1}, \dots, 1 - \sigma_1, 1\}$, for AltPoly- r with the choices of $\{\sigma_i\}$ shown in Table A1. Through numerical experiments, we found that smaller $\{\sigma_i\}$ can generally induce a more stable numerical scheme, which allows for a larger CFL number. However, when $\{\sigma_i\}$ approaches 0, the condition number of A tends to infinity, which will cause a loss of accuracy. Based on numerical trials, the choices of $\{\sigma_i\}$ are determined as a compromise between stability and accuracy.

Table A1. Choice of $\{\sigma_i\}$ for solution points.

r	Solution Points
3	$\sigma_1 = 0.03$
4	$\sigma_1 = 0.05, \sigma_2 = 0.1$
5	$\sigma_1 = \frac{1}{34}, \sigma_2 = \frac{2}{11}, \sigma_3 = \frac{1}{4}$
6	$\sigma_1 = 0.02, \sigma_2 = 0.05, \sigma_3 = 0.1, \sigma_4 = 0.12$

Appendix B. Some Technical Results Used in Section 3

Via symbolic computation by Mathematica, we have the following Taylor expansions in decimal form:

$$\begin{aligned}
 S_0 e_1 &= -(0.267 + 0.256 \cos w) e_1 + 0.379 \sin w e_2 \\
 &= -0.523 e_1 + w v_1 + \mathcal{O}(w^2)
 \end{aligned} \tag{A7}$$

$$\begin{aligned} S_0 e_2 &= 0.129 \sin w e_1 + (0.508 - 0.246 \cos w) e_2 + \\ &= 0.129 w e_1 + 0.262 e_2 + \mathcal{O}(w^2) \end{aligned} \tag{A8}$$

$$\begin{aligned} S_1 e_1 &= -0.739i \sin w e_1 + 4.523 \sin^2\left(\frac{w}{2}\right) e_1 - i \sin w e_3 \\ &= w v_2 + \mathcal{O}(w^2) \end{aligned} \tag{A9}$$

$$S_1 e_2 = -0.521 e_1 + 1.261i w e_2 + \mathcal{O}(w^2) \tag{A10}$$

where the vectors v_3 and v_4 are independent with w .

We can see that the magnitudes of all entries in $S_0[1 : 2; 1 : 2]$ are smaller than 1 when w is small enough. In addition, from (39) we know that

$$S_0 e_3 = e_3. \tag{A11}$$

Therefore the principal eigenvalue of S_0 is 1 as long as w is small enough.

Now we try to bound $\|S^n e_1\|$ and $\|S^n e_2\|$. We first assume that $\|S^i\| \leq M$ for any $i \geq 1$, which naturally holds as long as the scheme is stable. Then we have

$$\begin{aligned} \|S^i e_1\| &= \|S^{i-1}((S_0 + cS_1)e_1)\| \\ &= \|S^{i-1}(-0.523e_1 + wv + \mathcal{O}(w^2))\| \\ &= \dots \\ &= \|(-0.523)^n e_1 + \sum_{k=1}^i (-0.523)^{n-i} S^{k-1}(wv_1 + \mathcal{O}(w^2))\| \\ &\leq 0.523^i + M(w\|v_1\| + \mathcal{O}(w^2)) = 0.523^i + \mathcal{O}(w). \end{aligned} \tag{A12}$$

Hence,

$$\begin{aligned} S^i e_2 &= S^{i-1}(0.262e_2 + \mathcal{O}(w) + c((-0.521)e_1 + \mathcal{O}(w))) \\ &= S^{i-2}(0.262^2 e_2 + (0.262 + S)(\mathcal{O}(w) + c(-0.521e_1 + \mathcal{O}(w)))) \\ &= \dots \\ &= 0.262^i e_2 + \underbrace{\sum_{k=1}^n (0.262^{k-1} S^{i-k})(\mathcal{O}(w) + c\mathcal{O}(w))}_{A_1} - 0.521c \underbrace{\sum_{k=1}^i 0.262^{k-1} S^{i-k} e_1}_{A_2} \end{aligned} \tag{A13}$$

Since the magnitude of $c = \Delta t / \Delta x$ is not greater than $\mathcal{O}(1)$ and $\|S^i\| \leq M$, then $\|A_1\|$ can be estimated by

$$\|A_1\| \leq \sum_{k=0}^{i-1} 0.262^{k-1} M \mathcal{O}(w) \leq \frac{M \mathcal{O}(w)}{1 - 0.262} = \mathcal{O}(w). \tag{A14}$$

As for $\|A_2\|$, the following bound can be deduced based on (A12),

$$\begin{aligned} \|A_2\| &\leq \sum_{k=1}^{i-1} 0.262^{k-1} \|S^{i-k} e_1\| \leq \sum_{k=1}^{i-1} 0.262^{k-1} (0.523^{i-k} + \mathcal{O}(w)) \\ &\leq (i - 1)0.523^{i-1} + \mathcal{O}(w). \end{aligned} \tag{A15}$$

Combining (A14) and (A15), $\|S^i e_2\|$ can be bounded by

$$\|S^i e_2\| \leq 0.262^i + (i - 1)0.523^i c + \mathcal{O}(w). \tag{A16}$$

References

1. Lele, S.K. Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.* **1992**, *103*, 16–42. [[CrossRef](#)]
2. Mohebbi, A.; Dehghan, M. High order compact solution of the one-space-dimensional linear hyperbolic equation. *Numer. Methods Part. Differ. Equ. Int. J.* **2008**, *24*, 1222–1235. [[CrossRef](#)]
3. Yabe, T.; Ishikawa, T.; Wang, P.; Aoki, T.; Kadota, Y.k.; Ikeda, F. A universal solver for hyperbolic equations by cubic-polynomial interpolation II. Two-and three-dimensional solvers. *Comput. Phys. Commun.* **1991**, *66*, 233–242. [[CrossRef](#)]
4. Yabe, T.; Xiao, F.; Utsumi, T. The constrained interpolation profile method for multiphase analysis. *J. Comput. Phys.* **2001**, *169*, 556–593. [[CrossRef](#)]
5. Aoki, T. Interpolated differential operator (IDO) scheme for solving partial differential equations. *Comput. Phys. Commun.* **1997**, *102*, 132–146. [[CrossRef](#)]
6. Imai, Y.; Aoki, T. Stable coupling between vector and scalar variables for the IDO scheme on collocated grids. *J. Comput. Phys.* **2006**, *215*, 81–97. [[CrossRef](#)]
7. Goodrich, J.; Hagstrom, T.; Lorenz, J. Hermite methods for hyperbolic initial-boundary value problems. *Math. Comput.* **2006**, *75*, 595–630. [[CrossRef](#)]
8. Tanaka, R.; Nakamura, T.; Yabe, T. Constructing exactly conservative scheme in a non-conservative form. *Comput. Phys. Commun.* **2000**, *126*, 232–243. [[CrossRef](#)]
9. Yabe, T.; Tanaka, R.; Nakamura, T.; Xiao, F. An exactly conservative semi-Lagrangian scheme (CIP-CSL) in one dimension. *Mon. Weather Rev.* **2001**, *129*, 332–344. [[CrossRef](#)]
10. Imai, Y.; Aoki, T.; Takizawa, K. Conservative form of interpolated differential operator scheme for compressible and incompressible fluid dynamics. *J. Comput. Phys.* **2008**, *227*, 2263–2285. [[CrossRef](#)]
11. Onodera, N.; Aoki, T.; Kobayashi, H. Large-eddy simulation of turbulent channel flows with conservative IDO scheme. *J. Comput. Phys.* **2011**, *230*, 5787–5805. [[CrossRef](#)]
12. Li, S.; Shimuta, M.; Xiao, F. A 4th-order and single-cell-based advection scheme on unstructured grids using multi-moments. *Comput. Phys. Commun.* **2005**, *173*, 17–33. [[CrossRef](#)]
13. Cockburn, B.; Shu, C.W. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Math. Comput.* **1989**, *52*, 411–435.
14. Cockburn, B.; Lin, S.Y.; Shu, C.W. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems. *J. Comput. Phys.* **1989**, *84*, 90–113. [[CrossRef](#)]
15. Qiu, J.; Shu, C.W. Runge–Kutta discontinuous Galerkin method using WENO limiters. *SIAM J. Sci. Comput.* **2005**, *26*, 907–929. [[CrossRef](#)]
16. Wang, Z.J. Spectral (finite) volume method for conservation laws on unstructured grids. basic formulation: Basic formulation. *J. Comput. Phys.* **2002**, *178*, 210–251. [[CrossRef](#)]
17. Wang, Z.J.; Zhang, L.; Liu, Y. Spectral (finite) volume method for conservation laws on unstructured grids IV: Extension to two-dimensional systems. *J. Comput. Phys.* **2004**, *194*, 716–741. [[CrossRef](#)]
18. Liu, Y.; Vinokur, M.; Wang, Z.J. Spectral difference method for unstructured grids I: Basic formulation. *J. Comput. Phys.* **2006**, *216*, 780–801. [[CrossRef](#)]
19. Huynh, H.T. A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods. In Proceedings of the 18th AIAA Computational Fluid Dynamics Conference, Miami, FL, USA, 25–28 June 2007; p. 4079.
20. Wang, Z.J.; Gao, H. A unifying lifting collocation penalty formulation including the discontinuous Galerkin, spectral volume/difference methods for conservation laws on mixed grids. *J. Comput. Phys.* **2009**, *228*, 8161–8186. [[CrossRef](#)]
21. Li, S.; Xiao, F. High order multi-moment constrained finite volume method. Part I: Basic formulation. *J. Comput. Phys.* **2009**, *228*, 3669–3707. [[CrossRef](#)]
22. Xiao, F.; Li, S.; Chen, C.; Li, X. A note on the general multi-moment constrained flux reconstruction formulation for high order schemes. *Appl. Math. Model.* **2013**, *37*, 5092–5108. [[CrossRef](#)]
23. Kirby, R.M.; Karniadakis, G.E. De-aliasing on non-uniform grids: Algorithms and applications. *J. Comput. Phys.* **2003**, *191*, 249–264. [[CrossRef](#)]
24. Kanevsky, A.; Carpenter, M.H.; Hesthaven, J.S. Idempotent filtering in spectral and spectral element methods. *J. Comput. Phys.* **2006**, *220*, 41–58. [[CrossRef](#)]
25. Zienkiewicz, O.C.; Taylor, R.L.; Zhu, J.Z. *The Finite Element Method: Its Basis and Fundamentals*; Elsevier: Amsterdam, The Netherlands, 2005.
26. Staniforth, A.; Côté, J. Semi-Lagrangian integration schemes for atmospheric models—A review. *Mon. Weather Rev.* **1991**, *119*, 2206–2223. [[CrossRef](#)]
27. Ben-Israel, A.; Greville, T.N. *Generalized Inverses: Theory and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2003; Volume 15.
28. Persson, P.O.; Peraire, J. Sub-cell shock capturing for discontinuous Galerkin methods. In Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 9–12 January 2006; p. 112.
29. Dehghan, M.; Shokri, A. A meshless method for numerical solution of a linear hyperbolic equation with variable coefficients in two space dimensions. *Numer. Methods Part. Differ. Equ. Int. J.* **2009**, *25*, 494–506. [[CrossRef](#)]
30. Jiang, G.S.; Shu, C.W. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.* **1996**, *126*, 202–228. [[CrossRef](#)]