



Article

Reconstructed Interpolating Differential Operator Method with Arbitrary Order of Accuracy for the Hyperbolic Equation

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 family of multi-moment methods with arbitrary orders of accuracy for the hyperbolic equation via the reconstructed interpolating differential operator (RDO) approach. Reconstruction up to arbitrary order can be achieved on a single cell from properly allocated model variables including spatial derivatives of varying orders. Then we calculate the temporal derivatives of coefficients of the reconstructed polynomial and transform them into the temporal derivatives of the model variables. Unlike the conventional multi-moment methods which evolve different types of moments by deriving different equations, RDO can update all derivatives uniformly via a simple linear transform more efficiently. Based on difference in introducing interaction from adjacent cells, the central RDO and the upwind RDO are proposed. Both schemes enjoy high-order accuracy which is verified by Fourier analysis and numerical experiments.

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



Citation: Lin, S.; Luo, Q.; Leng, H.; Song, J. Reconstructed Interpolating Differential Operator Method with Arbitrary Order of Accuracy for the Hyperbolic Equation. *Axioms* **2021**, *10*, 295. <https://doi.org/10.3390/axioms10040295>

Academic Editor: Maya Briani

Received: 15 September 2021

Accepted: 2 November 2021

Published: 6 November 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 the last few decades, high-order numerical methods are becoming increasingly popular in the research community due to the advantages over the low-order methods in achieving higher resolution more efficiently with lower computational cost [1]. The conventional high-order approaches based on high-order finite difference (FD) and the finite volume (FV) methods only use one degree of freedom (DOF) on each node or cell. For these methods, a high-order approximation requires a wide stencil which introduces huge communication costs between computational nodes on supercomputers.

To narrow the stencil, a natural and feasible approach is to include two or more DOFs per cell so as to construct a higher-order polynomial. The constrained interpolation profile (CIP) scheme [2,3] evolves two moments, i.e., the physical variable and the spatial derivative simultaneously and independently according to the governing equations in different forms. Specifically, CIP describes the spatial profile using the third-order Hermite interpolation and updates the profile according to the local analytic solution in via the semi-Lagrangian approach. Following this pioneering work, the so-called CIP-conservative semi-Lagrangian (CIP-CSL) schemes [4,5] add the cell-averaged value as an extra moment to ensure the conservativeness for the scalar conservative advection transport. Successive studies have resulted in a more general framework, the so-called CIP/multi-moment finite volume method (CIP/MM FVM) which has been applied to various fluid dynamic simulations [6–8]. The interpolated differential operator (IDO) method [9] is similar to CIP but IDO updates the model variables using the Eulerian method instead of the semi-Lagrangian method. A conservative variant of IDO scheme also predicts the cell-averaged moment separately [10]. Both CIP and IDO require an additional equation for the spatial derivative moment which can be difficult to obtain when treating nonlinear and high-dimensional problems.

A compact CIP/MM-FVM formulation that uses high-order derivative moments can reach arbitrary accuracy order has been presented in [11]. This approach predicts the high-order derivative moments in the Eulerian representation via solving a linearized high-order derivative Riemann derivative problem [12]. As a more flexible variant of CIP/MM-FVM, the multi-moment constrained finite volume (MCV) [13] treats the cell-averaged moments as constraints instead of explicit model variables. Following this principle, the MCV method has been generalized to the unstructured meshes [14,15] recently.

High-order schemes based on local flux reconstruction are another representative class of high-order schemes which also use compact stencil for spatial discretization and are increasingly attractive. Examples includes the discontinuous Galerkin(DG) [16], the spectral volume [17] and the flux reconstruction [18] method.

This article presents a new efficient compact multi-moment method termed reconstructed interpolating differential operator (RDO) method which does not involve the cross-cell reconstruction for the hyperbolic equations. This work can be viewed as an generalization of the original IDO scheme [9] to arbitrary orders, non-uniform meshes and higher dimensions. Moreover, this work proposes a novel updating rules for model variables via a direct linear transform, which does not require introducing additional equations for different types of moments. The core idea is simple and direct. In each cell, the solution is approximated by a polynomial to interpolate the moments including the point values and high-order derivatives at the cell boundaries. The next step computes the temporal derivatives of the polynomial coefficients with the aid of solution points. Then the temporal derivatives are retrieved from the temporal derivatives of polynomial coefficient via a linear transform. To deal with the non-uniqueness of temporal derivatives computed from different cells at cell boundaries, the central and the upwind RDO are proposed. Fourier analysis and numerical tests indicate that our scheme can achieve arbitrary orders of accuracy.

The remaining part of this paper is organized as follows. Section 2 gives the basic formulation of RDO in one dimension. Section 3 describes the generalization of RDO to higher dimensions. Then Section 4 investigates the accuracy order and stability by classical Fourier analysis. Numerical tests in Section 5 validate the Fourier analysis results and compare the computational efficiency and numerical performance of RDO of various orders. Section 6 ends this paper with a few conclusions and discussions.

2. Basic Formulation for the One-Dimensional Problem

Consider the 1D scalar hyperbolic equation

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

where x is the spatial coordinate, t denotes time, u is the conserved variable and $f(u)$ is the flux.

To begin with, we introduce some notations here. The bold lower-case letter such as \mathbf{a}, \mathbf{u} stands for a column vector. The bold capital letter like \mathbf{A} denotes a matrix. The column vector $[a_1, \dots, a_m]^T$ is denoted by $[a_k]_{k=1}^m$ for simplicity. \mathbf{A}^T denotes the transpose of \mathbf{A} . The ℓ_p norm of the vector \mathbf{a} is denoted by $\|\mathbf{a}\|_p$ and $\|\mathbf{a}\|$ is a simplified form of $\|\mathbf{a}\|_2$ throughout this paper. The matrix norm of \mathbf{H} is defined by $\|\mathbf{H}\| = \max_{\|z\|=1} \|\mathbf{H}z\| / \|z\|$.

2.1. Spatial Discretization

We divide the computational domain $[x_{\min}, x_{\max}]$ into N non-overlapping cells or elements, among which the j -th cell is $I_j := [x_{j-1/2}, x_{j+1/2}]$. For convenience, we denote the middle point of I_j by $x_j = (x_{j-1/2} + x_{j+1/2})/2$ and the length of I_j by $\Delta x_j = x_{j+1/2} - x_{j-1/2}$.

Model variables including point-based value and derivatives are independently evolved in our scheme. For simplicity, the RDO scheme that have M model variables in each cell is denoted by RDO- M . The allocation of model variables has a slight difference between $M = 2K$ and $M = 2K + 1$. For RDO- $2K$, $2K$ model variables in each cell are all located at

the cell boundaries. These model variables are $\{u_{j\pm 1/2}, (u_x)_{j\pm 1/2}, \dots, (u_{x^{K-1}})_{j\pm 1/2}\}$, where $(u_{x^k})_{j-1/2}$ is the approximation of $u_{x^k}(x_{j-1/2})$ at $x = x_{j-1/2}$.

For RDO- $(2K + 1)$, we add the function value at the middle of the cell, i.e., $u_j \approx u(x_j)$ as an extra model variable and use a $2K$ -th order polynomial to represent the solution polynomial. Specifically, the $(2K + 1)$ model variables used in RDO- $(2K + 1)$ are $\{u_{j\pm 1/2}, (u_x)_{j\pm 1/2}, \dots, (u_{x^{K-1}})_{j\pm 1/2}\} \cup \{u_j\}$.

2.2. Updating Model Variables

This section only presents the detailed formulation of RDO- $2K$ ($K \geq 2$). The procedure for RDO- $(2K + 1)$ is almost the same and hence omitted here for simplicity. The procedure to compute time derivatives of model variables can be divided into three stages as follows.

2.2.1. First Stage: Reconstructing Polynomial from Model Variables

This stage finds a solution polynomial $h_j(x)$ on $I_j = [x_{j-1/2}, x_{j+1/2}]$ to interpolate model variables defined at cell boundaries. The interpolation polynomial is of $(2K - 1)$ -th order since there are totally $2K$ model variables. We first introduce a local coordinate to map I_j to the standard cell $[-1, 1]$,

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

The usage of $s(x)$ unifies the interpolation templates and hence reduces the computational cost in polynomial reconstruction. The solution polynomial $h_j(x)$ under the local coordinate s is then $\tilde{h}_j(s) = h_j(x(s))$. $\frac{d\tilde{h}_j}{ds}$ and $\frac{dh_j}{dx}$ are connected by the chain rule

$$\frac{d\tilde{h}_j(s)}{ds} = \frac{dx}{ds} \frac{dh_j}{dx} = \frac{\Delta x_j}{2} \frac{dh_j}{dx}. \tag{3}$$

Suppose $\tilde{h}_j(s) = \sum_{i=0}^{2K-1} a_i s^i$. Then the polynomial coefficients $\{a_i\}_{i=0}^{2K-1}$ can be determined by the following linear system:

$$\frac{d^k}{ds^k} \tilde{h}_j(s) \Big|_{s=\pm 1} = \sum_{i=k}^{2K-1} \frac{i!}{(i-k)!} a_i (\pm 1)^i = \left(\frac{\Delta x_j}{2}\right)^k \cdot (u_{x^k})_{j\pm 1/2}, \tag{4}$$

for $k = 0, \dots, K$. This is a linear system about $2K$ unknowns $\{a_0, \dots, a_{2K-1}\}$ and we can write it into the matrix form

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

where \mathbf{H} is the coefficient matrix and \mathbf{d}_j denotes the rescaled model variables

$$\mathbf{d}_j = \left[u_{j-1/2}, \frac{\Delta x_j}{2} (u_x)_{j-1/2}, \dots, \left(\frac{\Delta x_j}{2}\right)^{K-1} (u_{x^{K-1}})_{j-1/2}, \right. \\ \left. u_{j+1/2}, \frac{\Delta x_j}{2} (u_x)_{j+1/2}, \dots, \left(\frac{\Delta x_j}{2}\right)^{K-1} (u_{x^{K-1}})_{j+1/2} \right]^T. \tag{6}$$

The coefficients of the solution polynomial is then

$$\mathbf{a} = \mathbf{H}^{-1} \mathbf{d}_j. \tag{7}$$

Through the high-order Hermite interpolation (4), the solution polynomial is $(K - 1)$ -th smooth globally. Introducing the local coordinate makes the inverse of the coefficient matrix \mathbf{H} the same for all cells so as to reduce the computational costs in solving the linear system in each cell. When the condition number of this linear system is too high as K increases, one can use the symbolic tool in Maple or Matlab to obtain a sufficiently accurate inverse of the matrix to avoid the loss of accuracy.

2.2.2. Second Stage: Computing Temporal Derivatives on Solution Points

This part computes the temporal derivative of the obtained solution polynomial $h_j(x)$. For this, we choose $2K$ solution points $x_{j,k} = x_j + \zeta_k \Delta x_j$, $k = 1, \dots, 2K$ on I_j with $\zeta_k \in [-1, 1]$. $\{\zeta_k\}_{k=1}^{2K}$ are the same for each element. Through numerical and theoretical analysis later, we find that a simple choice of uniformly distributed solution points is sufficient for good numerical performance, i.e.,

$$\zeta_k = -1 + \frac{2(k-1)}{2K-1}, \quad k = 1, \dots, 2K. \tag{8}$$

We can now easily update the solution polynomials via updating solution values located on selected solution points. Current solution values and approximated spatial derivatives are, respectively,

$$u_{j,k} = \tilde{h}_j(s(x_{j,k})), \quad (u_x)_{j,k} = \left(\frac{\Delta x_j}{2}\right)^{-1} \frac{d\tilde{h}_j(s)}{ds} \Big|_{s=s(x_{j,k})}, \quad k = 1, \dots, 2K. \tag{9}$$

where $\frac{d\tilde{h}_j}{ds}$ can be obtained by

$$\left[\frac{d\tilde{h}_j}{ds} \Big|_{s=s(x_{j,k})} \right]_{k=1}^{2K} = D \mathbf{a}, \quad D = \begin{bmatrix} 0 & \zeta_1 & \dots & (2K-1)\zeta_1^{2K-2} \\ 0 & \zeta_2 & \dots & (2K-1)\zeta_2^{2K-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \zeta_M & \dots & (2K-1)\zeta_M^{2K-2} \end{bmatrix}. \tag{10}$$

According to (1), the temporal derivatives of solution values become

$$\frac{du_{j,k}}{dt} = -f_u(u_{j,k}) \cdot (u_x)_{j,k}, \quad k = 1, \dots, 2K. \tag{11}$$

Then we use $\frac{du_{j,k}}{dt}$ to retrieval the temporal derivatives of the coefficients of solution polynomials $h_j(s)$. This can be done by solving the following linear systems

$$\frac{d\tilde{h}_j(s(x_{j,k}))}{dt} = \sum_{i=0}^{2K-1} \frac{da_i}{dt} s(x_{j,k})^i = \frac{du_{j,k}}{dt}, \quad k = 1, \dots, 2K. \tag{12}$$

Note that $s(x_{j,k}) = \zeta_k$. This is a $2K \times 2K$ linear system about $\left\{ \frac{da_i}{dt} \right\}$ and the coefficient matrix is the Vandermonde matrix

$$\mathbf{V} = \begin{bmatrix} 1 & \zeta_1 & \dots & \zeta_1^{2K-1} \\ 1 & \zeta_2 & \dots & \zeta_2^{2K-1} \\ \dots & \dots & \dots & \dots \\ 1 & \zeta_{2K} & \dots & \zeta_{2K}^{2K-1} \end{bmatrix}. \tag{13}$$

This coefficient matrix remains the same for all cells. Therefore, we also can store the inverse in advance to reduce computational costs.

2.2.3. Third Stage: Retrieving the Temporal Derivatives

Using $\left\{ \frac{da_i}{dt} \right\}$, the temporal derivatives of model variables can be directly retrieved by the following linear transform,

$$\left(\frac{d}{dt} (u_{x^k})_{j \pm 1/2} \right)_j = \left(\frac{\Delta x_j}{2} \right)^{-k} \sum_{i=k}^{2K-1} \frac{i!}{(i-k)!} \frac{da_i}{dt} (\pm 1)^i, \quad k = 0, \dots, K-1. \tag{14}$$

The subscript j in the left-hand side of (14) indicates that this temporal derivative is computed from the cell I_j alone. At the cell boundary $x = x_{j+1/2}$, we obtain the temporal derivatives from the I_j and I_{j-1} at the same time, which can take different values. This leaves us the problem about how to define the proper temporal derivatives of $\left\{ (u_{x^k})_{j+1/2} \right\}_{k=0}^K$ properly.

A natural thought is choosing the one from the upwind direction so as to obey the physical property of the hyperbolic equation. We denote $\frac{d\phi_{j+1/2}}{dt}$ computed in the cell I_j by $\left(\frac{d\phi_{j+1/2}}{dt} \right)_j$ for the model variable ϕ . Then the upwind temporal derivative of ϕ is

$$\frac{d}{dt}\phi_{j+1/2} = \left(\frac{d\phi_{j+1/2}}{dt} \right)_{j_{\text{up}}}, \tag{15}$$

where j_{up} denotes the index in the upwind direction which is determined by the sign of $f_u(u_{j+1/2})$. To be more specific,

$$j_{\text{up}} = \begin{cases} j, & \text{if } f_u(u_{j+1/2}) \geq 0; \\ j + 1, & \text{otherwise.} \end{cases} \tag{16}$$

This scheme is termed upwind RDO scheme due to the upwind property.

Another idea to determine $\frac{d}{dt}\phi_{j-1/2}$ is simply computing the algebraic average of the temporal derivatives from two cells just like the classical central difference scheme. Then the central temporal derivative of some model variables ϕ is

$$\frac{d}{dt}\phi_{j-1/2} = \frac{1}{2} \left(\left(\frac{d\phi_{j+1/2}}{dt} \right)_j + \left(\frac{d\phi_{j+1/2}}{dt} \right)_{j+1} \right) \tag{17}$$

One may consider a sum weighted by cell length in (17) when encountering the non-uniform mesh like the spectral element method [19]. However, (17) works sufficiently well, as shown in the numerical tests on non-uniform meshes.

2.3. Time Integration

We have obtained the spatial discretization and temporal derivatives of model variables, which can be described in the semi-discrete form

$$\frac{d\Phi}{dt} = \mathcal{L}(\Phi), \tag{18}$$

where Φ denotes the collection of all model variables. Suppose the value of Φ at n -th time step is $\Phi^{(n)}$, then $\Phi^{(n+1)}$ at the next time step is computed by the fourth-order Runge–Kutta method [20] to ensure the numerical accuracy in time. We have

$$\Phi^{(n+1)} = \Phi^{(n)} + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{19}$$

where Δt is the time step length and

$$\begin{cases} k_1 = \mathcal{L}(\Phi^{(n)}), \\ k_2 = \mathcal{L}(\Phi^{(n)} + \frac{1}{2}k_1\Delta t), \\ k_3 = \mathcal{L}(\Phi^{(n)} + \frac{1}{2}k_2\Delta t), \\ k_4 = \mathcal{L}(\Phi^{(n)} + k_3\Delta t). \end{cases} \tag{20}$$

2.4. Discussion

The involved polynomial interpolation procedures (4) and (14) are implemented within the cell. In other words, no cross-cell polynomial interpolation is needed in the proposed schemes, which means the scheme is compact and makes it possible to solve the problems on the non-uniform mesh more flexibly than the conventional IDO methods.

We can also construct RDO-2 and RDO-3 by assigning merely function values at cell boundaries without derivative moments. Then central and the upwind RDO-2 are, respectively, the classical central difference and the upwind difference method. Besides, the spectral element method [21] with three element base function in one dimension is also the special case of central RDO-3.

3. Generalization to Higher Dimensions

Now we extend the algorithm to rectangular meshes to solve the two dimensional conservative equations and briefly explain how to extend this problem into three dimension. The core issue is to define proper model variables on cell boundaries and find a benign polynomial space to interpolate them. Furthermore, the reconstructed piecewise polynomials should also be at least differentiable at the cell boundaries. Therefore we also use Hermite interpolation to represent the solution polynomials.

3.1. Spatial Discretization

Suppose the computational domain is a rectangle $[a, b] \times [c, d]$ and is divided into $N_1 \times N_2$ non-overlapping cells. We denote the cell at the i -th row and j -th column by $I_{i,j} = [x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}]$. Denote $\Delta x_i = x_{i+1/2} - x_{i-1/2}$ and $\Delta y_j = y_{j+1/2} - y_{j-1/2}$.

To explain the intuition of spatial discretization, we first consider the bicubic Hermite interpolation on the cell $I_{i,j}$. In this case, 16 DOFs are needed as 4 DOFs are assigned on each vertex. These 4 DOFs are u, u_x, u_y and the second-order mixed derivative u_{xy} . A bit different from the 1D situation, u_{xy} are required here to guarantee that the reconstructed piecewise Hermite polynomials possesses continuous normal derivatives along cell edges. A similar allocation of model variables can be seen in the IDO method for solving the two-dimensional Poission Equations [22].

For general RDO-2K in 2D, the needed model variables on each vertex are $\left\{ \frac{\partial^{i+j} u}{\partial x^i \partial y^j} \right\}_{0 \leq i, j \leq K-1}$ which means that each element has totally $4K^2$ model variables. However, since the adjacent cells share the same model variables on common vertices, the number of effective DOFs on each element is only K^2 . We use the following polynomial spaces for representing the solution polynomial is the following full polynomial spaces [23],

$$\mathcal{Q}_{2K} = \{q(x, y) | q(x, y) = \sum_{0 \leq l, m \leq 2K-1} a_{l,m} x^l y^m\}. \tag{21}$$

Its space dimension is $\dim(\mathcal{Q}_{2K}) = (2K)^2$. \mathcal{Q}_{2K} is the tensor product of the 1D Hermite polynomials.

For RDO- $(2K + 1)$, the reconstructed polynomial space becomes \mathcal{Q}_{2K+1} which has $(2K + 1)^2$ dimensions. Hence, each element needs additionally $(2K + 1)^2 - (2K)^2 = 4K + 1$ model variables, which include $u_{i,j}$ that denotes the function value on the barycenter, and K model variables, $u, u_n, u_{n^2}, \dots, u_{n^{K-1}}$ on four edge centers. Here u_{n^k} denotes the k th-order normal derivative on the edge center.

Figure 1a,b illustrate the choices of model variables for RDO-4 and RDO-5, which are representative examples of RDO- $(2K)$ and RDO- $(2K + 1)$, respectively. Dots represent the point-valued variables. Single arrows represent the first derivatives. Paired arrows denote the mixed derivatives u_{xy} .

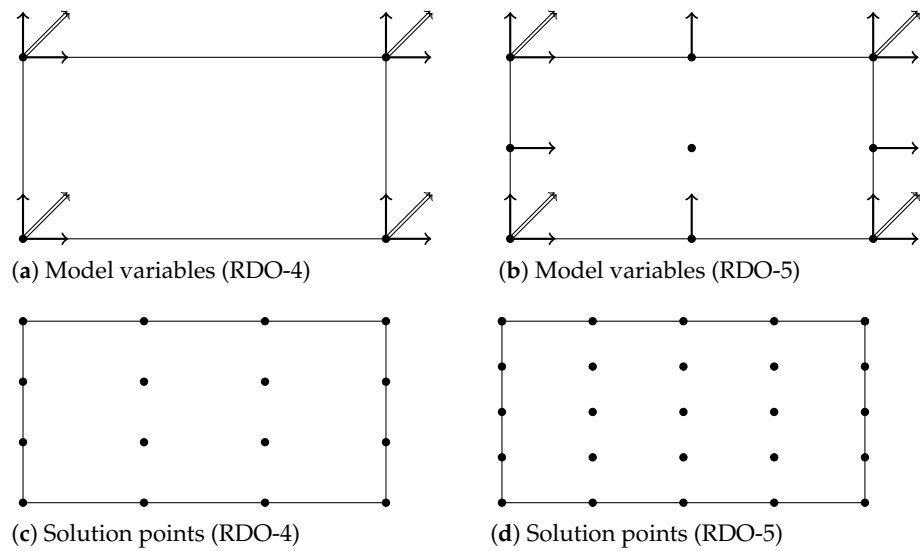


Figure 1. The illustration of model variables and solution points used in RDO-4 and RDO-5 in the two-dimensional space.

3.2. First Stage: Reconstructing Solution Polynomial

The first step interpolates the solution polynomials $h_{i,j}(x, y)$ on $I_{i,j}$. To begin with, we also introduce a local coordinate to map the cell $I_{i,j}$ to the standard cell $[-1, 1] \times [-1, 1]$,

$$\begin{cases} s(x, y) = 2(x - x_i) / \Delta x_i \in [-1, 1]; \\ r(x, y) = 2(y - y_j) / \Delta y_j \in [-1, 1]. \end{cases} \tag{22}$$

Similar to the one-dimensional case, $h_{i,j}(x, y)$ is also transformed into $\tilde{h}_{i,j}(r, s)$ under the local coordinate. Correspondingly, $4K^2$ model variables serves as constraints. The solution polynomial

$$\tilde{h}_{i,j}(r, s) = \sum_{\alpha=0}^{2K-1} \sum_{\beta=0}^{2K-1} a_{\alpha,\beta} r^\alpha s^\beta$$

is then determined by

$$\begin{aligned} \frac{\partial^{l+m}}{\partial r^l \partial s^m} \tilde{h}(r, s) \Big|_{r=\pm 1, s=\pm 1} &= \sum_{\alpha=l}^{2K-1} \sum_{\beta=m}^{2K-1} a_{\alpha,\beta} \frac{l!}{(l-\alpha)!} \frac{m!}{(m-\beta)!} (\pm 1)^\alpha (\pm 1)^\beta \\ &= \left(\frac{\Delta x_i}{2}\right)^l \left(\frac{\Delta y_j}{2}\right)^m (u_{x^l y^m})_{i\pm 1/2, j\pm 1/2}, \quad 0 \leq l, m \leq 2K - 1. \end{aligned} \tag{23}$$

The inverse of the coefficient matrix of the linear system (23) about $\{a_{\alpha,\beta}\}$ can also be solved in advance so as to effectively save computational costs in solving (23).

3.3. Second Stage: Computing Temporal Derivatives on Solution Points

This stage represents the solution polynomial on solution points and update it. The chosen solution points in 2D are tensor products of 1D points. Specifically, these $4K^2$ points in the local coordinate are in the form of

$$(s_k, r_l) = (\xi_k, \zeta_l), \quad k, l \in \{1, \dots, 2K\}. \tag{24}$$

Figure 1c,d illustrate the distribution of solution points for RDO-4 and RDO-5. The solution values and spatial derivatives can be computed as

$$u_{i,j;k,l} = \tilde{h}_{i,j}(s,r), \tag{25}$$

$$(u_x)_{i,j;k,l} = \left(\frac{\Delta x_j}{2}\right)^{-1} \frac{\partial}{\partial s} \tilde{h}_j(s,r), \tag{26}$$

$$(u_y)_{i,j;k,l} = \left(\frac{\Delta y_j}{2}\right)^{-1} \frac{\partial}{\partial r} \tilde{h}_j(s,r). \tag{27}$$

Then the temporal derivatives of solution values become

$$\frac{du_{i,j;k,l}}{dt} = -f_u(u_{i,j;k,l})(u_x)_{i,j;k,l} - g_u(u_{i,j;k,l})(u_y)_{i,j;k,l}. \tag{28}$$

Then the temporal derivatives of coefficients of the solution polynomial, $\frac{da_{\alpha,\beta}}{dt}$ can be obtained by solving the following linear system,

$$\frac{d\tilde{h}_j(s(x_{i,j;k,l}))}{dt} = \sum_{l=0}^{2K-1} \sum_{m=0}^{2K-1} \frac{da_{\alpha,\beta}}{dt} \frac{l!}{(l-\alpha)!} \frac{m!}{(m-\beta)!} (\pm 1)^\alpha (\pm 1)^\beta. \tag{29}$$

3.4. Third Stage: Retrieving the Model Variables

From $\left\{ \frac{da_{i,j}}{dt} \right\}$, we can obtain the temporal derivatives of model variables by the following relationships for RDO-2K

$$\left(\frac{d}{dt} (u^{x^l y^m})_{i\pm 1/2, j\pm 1/2} \right)_{(i,j)} = \left(\frac{\Delta x_i}{2}\right)^l \left(\frac{\Delta y_i}{2}\right)^m \sum_{\alpha=l}^{2K-1} \sum_{\beta=m}^{2K-1} \frac{l!}{(l-\alpha)!} \frac{m!}{(m-\beta)!} \frac{da_{i,j}}{dt} (-1)^{\alpha+\beta} \tag{30}$$

for $l, m \in \{0, \dots, 2K - 1\}$. The subscript (i, j) in RHS of (30) specifies that the temporal derivatives are computed from $I_{i,j}$ and do not involve any external information. To properly introduce the external interaction, both central and upwind schemes can be reconstructed similar to the one dimensional case.

3.5. Three Dimensional Case

In three dimensions, RDO-2K on the cuboid mesh can also be constructed similarly via the polynomial space

$$C_{2K} = \left\{ p(x,y,z) \mid \sum_{0 \leq i,j,k \leq K-1} a_{i,j,k} x^i y^j z^k \right\},$$

and chooses $\left\{ \frac{\partial^{i+j+k} u}{\partial x^i \partial y^j \partial z^k} \right\}_{0 \leq i,j,z \leq K-1}$ on each vertex as model variables. There are totally $(2K)^3$ model variables in each cell. The remaining procedure is exactly the same as in lower dimensional space and we do not repeat the formulation here.

4. Fourier Analysis

This part analyses the accuracy and the stability of presenting scheme in solving the linear advection equation. For concreteness, we only give a detailed analysis for RDO-4 since the generalized RDO-M can be analyzed following the same routine.

4.1. Formulation of the Amplification Matrix

Consider the linear scalar advection equation

$$u_t + u_x = 0. \tag{31}$$

Suppose the spatial domain is $[0, L]$ and is discretized by a uniform mesh spacing Δx . The initial condition is periodic, $u_{ini} = e^{i\omega x / \Delta x}$ where i denotes $\sqrt{-1}$ and the scaled

wavenumber is $w = 2\pi k\Delta x/L \in [0, \pi)$. Then the initial model variables for the upwind RDO-4 on $I_j = [x_{j-1/2}, x_{j+1/2}]$ are

$$u_{j\pm 1/2} = e^{iw(x_{j\pm 1/2} - x_j)/\Delta x}, \tag{32a}$$

$$(u_x)_{j\pm 1/2} = \frac{iw}{\Delta x} e^{iw(x_{j\pm 1/2} - x_j)/\Delta x}. \tag{32b}$$

The core in Fourier analysis is reformulating the algorithm into the matrix form

$$\frac{d}{dt} \mathbf{d} = \frac{1}{\Delta x} \mathbf{S} \mathbf{d} \tag{33}$$

using the periodicity of u_{ini} and \mathbf{S} denotes the amplification matrix. We do this step by step as follows.

Stage 1 described in Section 2 reconstructs the Hermite polynomial coefficients \mathbf{a}_j from the rescaled model variables \mathbf{d}_j in the cell I_j ,

$$\mathbf{a}_j = \mathbf{H}^{-1} \mathbf{d}_j, \tag{34}$$

where \mathbf{H} is defined by (5) and \mathbf{d}_j is the composition of the model variables after the coordinate transform,

$$\begin{aligned} \mathbf{d}_j &= [u_{j-1/2}, (u_x)_{j-1/2}\Delta x/2, u_{j+1/2}, (u_x)_{j+1/2}\Delta x/2]^T \\ &= [1, \frac{iw}{2}, 1, \frac{iw}{2}]^T e^{iwx_j/\Delta x}. \end{aligned} \tag{35}$$

In Stage 2, the solution values and derivatives on solution points are, respectively,

$$\left[u_{j,k} \right]_{k=1}^4 = \mathbf{V} \mathbf{H}^{-1} \mathbf{d}_j, \tag{36}$$

and

$$\left[(u_x)_{j,k} \right]_{k=1}^4 = \frac{2}{\Delta x} \mathbf{D} \mathbf{H}^{-1} \mathbf{d}_j, \tag{37}$$

where \mathbf{V} and \mathbf{D} can be found in (13) and (10). According to (31), the temporal derivatives on the solution points are

$$\left[\frac{d}{dt} u_{j,k} \right]_{k=1}^4 = -\frac{2}{\Delta x} \mathbf{D} \mathbf{H}^{-1} \mathbf{d}_j. \tag{38}$$

Then the temporal derivatives on solution points are transformed back to the temporal derivatives of model variables, which is the inverse problem of (37):

$$\left(\frac{d}{dt} \mathbf{d}_j \right)_j = \mathbf{H} \mathbf{V}^{-1} \left[\frac{d}{dt} u_{j,k} \right]_{k=1}^4 = -\frac{2}{\Delta x} \mathbf{H} \mathbf{V}^{-1} \mathbf{D} \mathbf{H}^{-1} \mathbf{d}_j = \frac{1}{\Delta x} \tilde{\mathbf{S}} \mathbf{d}_j, \tag{39}$$

where the subscript j in the left-hand side indicates that the temporal derivatives are computed locally from j -th cell, and $\tilde{\mathbf{S}} = -2\mathbf{H}\mathbf{V}^{-1}\mathbf{D}\mathbf{H}^{-1}$.

In the linear case, $\mathbf{V}^{-1}\mathbf{D}$ is an invariant about the choice of solution points. Furthermore, the matrix $\mathbf{V}^{-1}\mathbf{D}$ has a simple structure

$$\mathbf{V}^{-1}\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{40}$$

Therefore, it can be easily concluded that RDO schemes using different sets of solution points are identical in solving the linear advection equation as the following theorem tells.

Theorem 1. For RDO- M schemes ($M \geq 4$), the numerical result is independent of the solution points chosen in solving the linear advection equation on uniform meshes.

Proof. This can be proved by showing that $V^{-1}D$ is invariant about the choice of solution points $\{\xi_1, \dots, \xi_M\}$. In fact, one can easily check that

$$\begin{aligned}
 V^{-1}D &= \begin{bmatrix} 1 & \xi_1 & \dots & \xi_1^{M-1} \\ 1 & \xi_2 & \dots & \xi_2^{M-1} \\ \dots & \dots & \dots & \dots \\ 1 & \xi_M & \dots & \xi_M^{M-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 & \xi_1 & \dots & (M-1)\xi_1^{M-2} \\ 0 & \xi_2 & \dots & (M-1)\xi_2^{M-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \xi_M & \dots & (M-1)\xi_M^{M-2} \end{bmatrix} \\
 &= \begin{bmatrix} 0 & 1 & & & & \\ & 0 & 2 & & & \\ & & 0 & 3 & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & M-1 \\ & & & & & 0 \end{bmatrix}. \tag{41}
 \end{aligned}$$

Moreover, it is noticed that the in-cell temporal derivatives (39) only relies on $H, V^{-1}D$ and d_j . H and d_j are independent with the choice of solution points. Therefore, we can conclude that the semi-discrete scheme is independent of solution points chosen. \square

Equation (39) gives the in-cell numerical approximation of derivatives of model derivatives. Denote the temporal derivatives $p_j = \left(\frac{d}{dt}d_j\right)_j$. From the periodicity of the initial profile, we know that

$$d_{j\pm 1} = e^{\pm iw}d_j, \tag{42}$$

$$p_{j\pm 1} = e^{\pm iw}p_j \tag{43}$$

Then for the upwind discretization, the periodicity of the initial profile (42) tells that

$$\begin{aligned}
 \frac{d}{dt}d_j &= [p_{j-1}(3), p_{j-1}(4), p_j(3), p_j(4)]^T \\
 &= [e^{-iw}p_j(3), e^{-iw}p_j(4), p_j(3), p_j(4)]^T. \tag{44}
 \end{aligned}$$

Hence, we can obtain the amplification matrix S_{up} for the upwind scheme as follows,

$$\frac{d}{dt}d_j = \frac{1}{\Delta x}T_{up}\tilde{S}d_j = \frac{1}{\Delta x}S_{up}d_j, \quad S_{up} := T_{up}\tilde{S} \tag{45}$$

with

$$T_{up} = \begin{bmatrix} 0 & 0 & e^{-iw} & 0 \\ 0 & 0 & 0 & e^{-iw} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{46}$$

Similarly, the temporal derivatives of d_j for the central RDO-4 are

$$\begin{aligned}
 \frac{d}{dt}d_j &= \frac{1}{2}[p_{j-1}(3) + p_j(1), p_{j-1}(4) + p_j(2), p_j(3) + p_{j+1}(1), p_j(4) + p_{j+1}(2)]^T \\
 &= \frac{1}{2}[e^{-iw}p_j(3) + p_j(1), e^{-iw}p_j(4) + p_j(2), e^{iw}p_j(1) + p_j(3), e^{iw}p_j(2) + p_j(4)]^T. \tag{47}
 \end{aligned}$$

Combining (39) and (47) we have

$$\frac{d}{dt} \mathbf{d}_j = \frac{1}{\Delta x} \mathbf{T}_{\text{central}} \tilde{\mathbf{S}} \mathbf{d}_j = \frac{1}{\Delta x} \mathbf{S}_{\text{central}} \mathbf{d}_j, \quad \mathbf{S}_{\text{central}} := \mathbf{T}_{\text{central}} \tilde{\mathbf{S}} \tag{48}$$

where

$$\mathbf{T}_{\text{central}} = \begin{bmatrix} 1/2 & 0 & e^{-iw}/2 & 0 \\ 0 & 1/2 & 0 & e^{-iw}/2 \\ e^{iw}/2 & 0 & 1/2 & 0 \\ 0 & e^{iw}/2 & 0 & 1/2 \end{bmatrix}. \tag{49}$$

4.2. Accuracy Analysis

We can see that both the central and the upwind RDO-4 have fourth spatial local discretization errors. However, this does not suffice to ensure the third order accuracy in numerical experiments. In fact, the order accuracy is determined by the principal eigenvalues $\sigma(w)$ according to [18], with $\sigma(w)$ being the principle eigenvalue of the amplification matrix \mathbf{S} . Other eigenvalues of \mathbf{S} are spurious ones. Specifically, if $\sigma(w)$ can be expanded as a Taylor series in w ,

$$\sigma(w) = -iw + \mathcal{O}(w^{m+1}), \tag{50}$$

then the scheme is m -th order accurate. Since it is difficult to calculate an expression manually in practice, we use the symbolic computation tools in Matlab®.

For the upwind RDO-4, we can obtain the principal eigenvalue of the amplification matrix $\mathbf{S}_{\text{upwind}}$ is

$$\sigma(w) = -iw - \frac{1}{72}w^4 + \mathcal{O}(w^5). \tag{51}$$

This indicates that the upwind RDO-4 is third order accurate in space. Besides, the term $-\frac{1}{72}w^4$ introduces a small amount of dissipation which tends to smear the solution, which is certificated in the numerical experiments.

The principal eigenvalue of the amplification matrix $\mathbf{S}_{\text{central}}$ for central RDO-4 is

$$\sigma(w) = -iw - \frac{i}{16}w^3 + \mathcal{O}(w^5), \tag{52}$$

and we can see that only second order spatial accuracy is maintained for the central scheme. It is also worthy noting that $-\frac{i}{16}w^3$ introduces slight dispersion, which means that phase speed varies as w changes. Therefore, during the long-term simulation of a complicated initial profile that contains waves of different frequency, the profile will be distorted finally. This is also verified in numerical experiments in Section 5.

The accuracy analysis for higher-order RDO schemes is exactly the same as RDO-4. However, a concise Taylor expansion of principal eigenvalue like (51) and (52) is very difficult (if not impossible) to obtain even using the mathematical software. Here we try to calculate the truncation order of $\sigma(w)$ of a scheme by the following approximation. First, choose a proper w , say $w = \pi/4$. Then the error is

$$\delta(w) = \sigma(w) + iw. \tag{53}$$

Next, we halve the wave number w and obtain the error corresponding to $w/2$

$$\delta(w/2) = \sigma(w/2) + iw/2. \tag{54}$$

Noticing that

$$\mathcal{O}(w/2) \approx \frac{1}{2^{m+1}} \mathcal{O}(w^{m+1}), \tag{55}$$

for the m -th order accurate scheme, one has

$$\delta(w/2) \approx \frac{1}{2^{m+1}} \delta(w). \tag{56}$$

Hence the order of accuracy can be approximated by a simple logarithmic operation

$$m \approx \frac{\log(|\delta(w)|/|\delta(w/2)|)}{\log(2)} - 1. \tag{57}$$

The modulus operation is required here since the error result can be complex valued. Errors and estimated accuracy orders for the upwind and the central schemes can be seen in Tables 1 and 2. The errors for the upwind schemes agree with the local discretization error well. The upwind scheme using M model variables per cell achieves $(M - 1)$ -th accuracy order strictly. However, the accuracy orders for the central schemes varies quite irregularly. Specifically, when $M = 5$ and 9 , the accuracy orders for central schemes remain consistent with that of upwind schemes. When $M = 6$ and 7 , central schemes achieve higher accuracy orders than upwind schemes. Nevertheless, there also some choices of M like $M = 4$ and 8 such that the accuracy order of the central scheme is lower than the corresponding upwind scheme. It is remarkable that when $M = 7$, the central scheme enjoys two orders higher accuracy than that of the local discretization error.

Table 1. Errors and accuracy orders for central RDO schemes via Fourier analysis.

RDO- M	Coarse Mesh Error ($w = \pi/4$)	Fine Mesh Error ($w = \pi/8$)	Order
$M = 5$	$-7.88 \times 10^{-7} + 3.16 \times 10^{-6}i$	$-1.24 \times 10^{-8} + 1.00 \times 10^{-7}i$	4.02
$M = 6$	$-3.14 \times 10^{-5} - 4.27 \times 10^{-6}i$	$-5.04 \times 10^{-7} - 3.40 \times 10^{-8}i$	4.97
$M = 7$	$-1.36 \times 10^{-7} + 2.85 \times 10^{-7}i$	$-5.42 \times 10^{-10} + 2.34 \times 10^{-9}i$	6.04
$M = 8$	$-1.00 \times 10^{-7} - 1.00 \times 10^{-8}i$	$-3.98 \times 10^{-10} - 1.98 \times 10^{-11}i$	6.98
$M = 9$	$-2.42 \times 10^{-10} + 5.31 \times 10^{-10}i$	$-2.34 \times 10^{-13} + 1.07 \times 10^{-12}i$	8.05

Table 2. Errors and accuracy orders for upwind RDO schemes via Fourier analysis.

RDO- M	Coarse Mesh Error ($w = \pi/4$)	Fine Mesh Error ($w = \pi/8$)	Order
$M = 5$	$3.36 \times 10^{-6}i$	$1.02 \times 10^{-7}i$	4.16
$M = 6$	$1.17 \times 10^{-5}i$	$8.71 \times 10^{-8}i$	6.08
$M = 7$	$8.97 \times 10^{-9}i$	$1.74 \times 10^{-11}i$	8.00
$M = 8$	$-2.17 \times 10^{-7}i$	$-1.76 \times 10^{-9}i$	5.95
$M = 9$	$6.46 \times 10^{-10}i$	$1.13 \times 10^{-12}i$	8.15

Tables 1 and 2 and also contain abundant information about the dissipation and dispersion of the schemes. It should be noted that a positive real part of $\delta(w)$ will cause the scheme eventually blow up if no additional dissipation is added in time stepping. In Table 2, the real parts of $\delta(w)$ are negative for all upwind schemes which introduces a certain amount of spurious dissipation so as to stabilize the scheme. From Table 1 we can see that $\delta(w)$ are pure imaginary numbers with all choices of M , which means that the dispersion is dominant in the numerical error for the central schemes. However, it can be easily observed that the dispersion and dissipation error decays rapidly as M increases for the proposed schemes.

In (39), one can see that the numerical error in computing H^{-1} dominates the error of temporal derivatives, since H and $V^{-1}D$ have explicit expression. However, the error of computing H^{-1} is inevitably increasing as M increases, which indicates that there is a limit of accuracy for the presented method. Denote the numerical inverse of H by \hat{H} , which is obtained by Matlab. It is well known that the error $\|H^{-1} - \hat{H}\|$ hinges on the

condition number of H . Moreover, a more accurate estimate $\|H^{-1} - \hat{H}\|$ can be bounded by the following inequality,

$$\begin{aligned} \|H^{-1} - \hat{H}\| &= \frac{1}{\|H\|} \|H\| \|H^{-1} - \hat{H}\| \\ &\geq \frac{1}{\|H\|} \|I - H\hat{H}\| := E_{\text{inv}}. \end{aligned} \tag{58}$$

Then E_{inv} determines the smallest error that RDO- M can reach through the grid refinement. The results of $\text{cond}(H)$ and E_{inv} with M varying from 4 to 11 are shown in Table 3. We can see that $\text{cond}(H)$ and E_{inv} grow quickly as M increases. Hence, there is a tradeoff between the higher convergence rate and E_{inv} . In this work, we suggest that $M \leq 9$ is ideal for the practical numerical experiments.

Table 3. $\text{cond}(H)$ and the E_{recon} as the function of M .

M	4	5	6	7	8	9	10	11
$\text{cond}(H)$	9.92	1.40×10^2	4.13×10^2	1.49×10^3	5.25×10^4	9.24×10^5	3.71×10^7	5.12×10^8
E_{inv}	1.60×10^{-16}	4.55×10^{-16}	1.21×10^{-15}	1.39×10^{-15}	3.24×10^{-14}	3.17×10^{-13}	4.56×10^{-12}	1.24×10^{-11}

4.3. Stability Analysis

Now we investigate the maximum tolerant time step length when using the Runge–Kutta time integration scheme. Let the time step length be $\Delta t = c\Delta x$ and c is the Courant number for (31). The amplification matrix for the proposed schemes using the fourth order Runge Kutta method for time integration (19) can be simplified as

$$R = I + cS + \frac{c^2 S^2}{2} + \frac{c^3 S^3}{6} + \frac{c^4 S^4}{24}. \tag{59}$$

To ensure the stability, the Courant number c should be chosen such that the spectral radius of the amplification matrix R , i.e., the largest absolute value of its eigenvalues, is not greater than 1 for all wavenumbers w . The spectral radius of R for the upwind schemes and the central schemes are presented in Figures 2 and 3, respectively, which can also tell the Courant number limitation related to the stability condition.

Both types of schemes have a fairly large permissible range of c . As more moments are used, the stability condition becomes more strict for both types of RDO schemes. Generally, the Courant number limitation of the central scheme is larger than that of the upwind schemes. For example, the Courant number limit for upwind RDO-5 is about 0.25 and this number is approximately 0.39 for the central RDO-5, although both schemes are fourth-order accurate. The Courant number limits for RDO schemes are presented in Table 4.

Table 4. The Courant number limit for RDO schemes.

M		4	5	6	7
c_{max}	Upwind	0.45	0.25	0.24	0.12
	Central	0.69	0.39	0.33	0.24

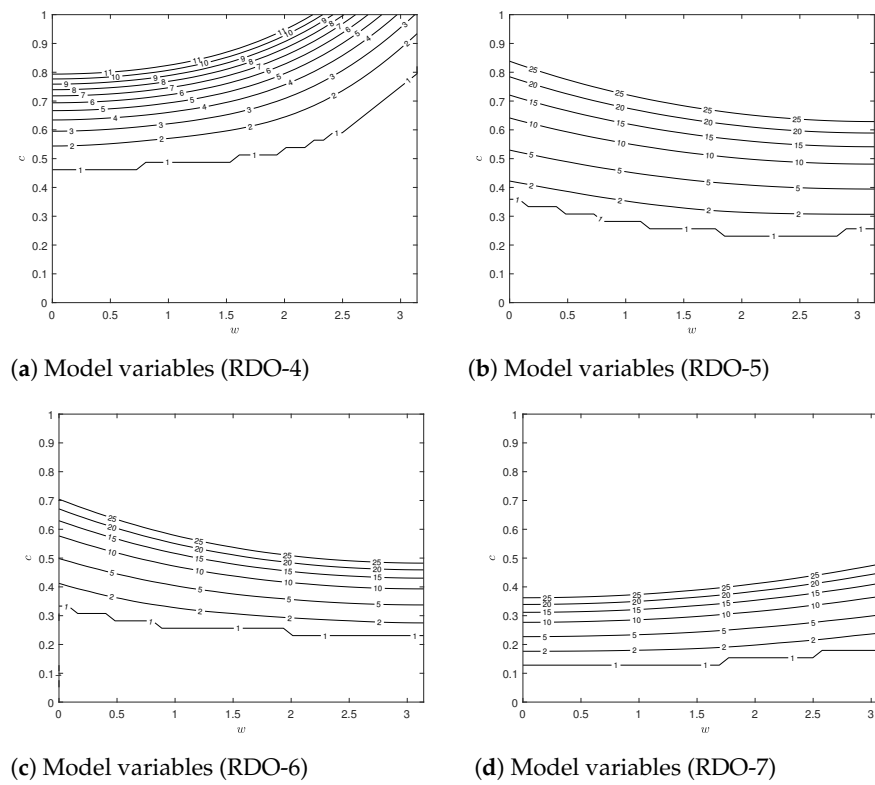


Figure 2. Contour plots of spectral radii of amplification matrices of the upwind schemes.

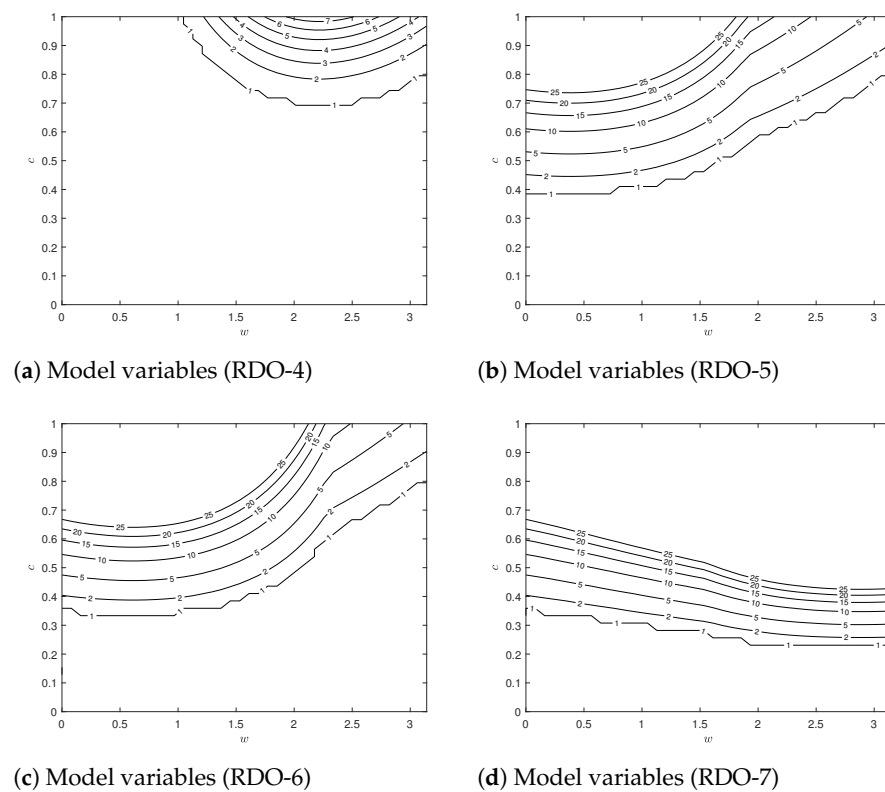


Figure 3. Contour plots of spectral radii of amplification matrices of the central schemes.

5. Numerical Results

This section provides several numerical tests to verify the formal accuracy and other numerical properties like diffusion and dispersion of the proposed schemes.

5.1. Accuracy Test

5.1.1. One-Dimensional Linear Case

We first test the accuracy when solving the scalar hyperbolic Equation (31) analysed in Section 4. The initial condition is $u(x, 0) = \sin(x)$ on $[0, 2\pi)$ with the periodical boundary condition. The exact solution is $u(x, t) = \sin(x - t)$.

Both the central and the upwind RDO- M with the number of moments M varying from 4 to 9 are tested on uniform meshes. To measure the accuracy, two typical norms for errors are used here,

$$E_1 = \sum_{j=1}^N |u_j - u_j^{\text{true}}|, \tag{60}$$

$$E_\infty = \max_{1 \leq j \leq N} |u_j - u_j^{\text{true}}|. \tag{61}$$

We study the convergence rate by recording the errors at $t = 2\pi$ and gradually refining the grids. The time step length is set as a very small number such that the temporal error does not influence the accuracy results. The error results are presented in Table 5. In general, the orders of accuracy computed from the grid refinement agree with the Fourier analysis results in Tables 1 and 2 well, with same trends in both orders of accuracy and error magnitudes. It can be observed that the upwind RDO- M achieves $(M - 1)$ -th order of accuracy for all M , which agrees with the local discretization error. On the other hand, the loss of accuracy order when $M = 4, 8$ and the supravergence phenomenon when $M = 6, 7$ indicated by Fourier analysis are also observed for the central RDO schemes in this test.

Besides the uniform meshes, we also consider the non-uniform meshes that are randomly generated by setting cell boundaries as

$$x_{j-\frac{1}{2}} = x_{j-\frac{1}{2}} + \eta \zeta_j \Delta x, \quad \zeta_j \sim \text{Unif}(-1, 1), \quad j = 1, \dots, N, \tag{62}$$

where N denotes the number of cells, $\Delta x = L/N$, $\{\zeta_j\}_{j=1}^N$ are independently random variables, η represents the magnitude of perturbation and $\text{Unif}(-1, 1)$ denotes the uniform distribution over $[-1, 1]$. We set $\eta = 0.1$ in the present work. In this setting, the maximal and the minimal possible values of Δx_j are, respectively, $0.8\Delta x$ and $1.2\Delta x$.

The accuracy results of RDO schemes over non-uniform meshes are listed in Table 6. It can be observed that the lack of uniformity in the mesh causes slightly larger errors and does not influence the convergence rate essentially. Therefore, the proposed schemes works consistently well on both uniform and non-conform meshes.

5.1.2. Two-Dimensional Linear Case

Consider the following two-dimensional scalar hyperbolic equation

$$u_t + u_x + u_y = 0. \tag{63}$$

The initial condition is $u(x, y; 0) = \sin(x + y)$ on $[0, 2\pi] \times [0, 2\pi]$ with the periodical boundary condition. The exact solution is $u(x, y; t) = \sin(x + y - 2t)$. The errors and the orders of accuracy for the central and the upwind RDO- M with M changing from 4 to 8 are shown in Table 7.

The errors and accuracy orders of the two-dimensional schemes are similar to that of the one-dimensional schemes, which illustrates that both the upwind and the central RDO schemes work well in two dimensions. Specifically, central RDO schemes achieves

significantly lower errors compared with upwind RDO schemes when $M = 6, 7$. However, for any other presented values of M , upwind RDO performs better.

5.1.3. Nonlinear Case

This case considers solving the one-dimensional Burger’s equation

$$u_t + uu_x = 0. \tag{64}$$

The initial condition is $u(x, 0) = 0.5 + \sin x$ with the periodical boundary condition for $x \in [0, 2\pi]$. We compute the numerical solution at $t = 0.2\pi$ when the profile is still smooth and record the errors under grid refinement. The results can be seen in Table 8. Both schemes lose accuracy due to nonlinearity compared with results of the linear cases shown in Table 5. The accuracy orders of central RDO schemes coincide the accuracy orders of local polynomial reconstruction better when $M \geq 5$. However, upwind schemes generally enjoy smaller errors in this test.

Table 5. Accuracy results from grid convergence on uniform meshes for the linear advection equation.

RDO- M	N	E_1	Order	E_∞	Order	E_1	Order	E_∞	Order
Central Schemes					Upwind Schemes				
$M = 4$	10	3.40×10^{-2}	-	5.25×10^{-2}	-	1.31×10^{-2}	-	2.02×10^{-2}	-
	20	8.21×10^{-3}	2.05	1.30×10^{-2}	2.02	1.70×10^{-3}	2.95	2.65×10^{-3}	2.93
	30	3.67×10^{-3}	1.98	5.75×10^{-3}	2.01	5.06×10^{-4}	2.98	7.93×10^{-4}	2.98
$M = 5$	10	2.40×10^{-4}	-	3.71×10^{-4}	-	2.28×10^{-4}	-	3.49×10^{-4}	-
	20	1.37×10^{-5}	4.13	2.17×10^{-5}	4.10	1.42×10^{-5}	3.95	2.12×10^{-5}	3.97
	30	2.70×10^{-6}	4.00	4.24×10^{-6}	4.02	2.95×10^{-6}	4.00	4.12×10^{-6}	3.97
$M = 6$	10	1.20×10^{-5}	-	1.20×10^{-5}	-	5.26×10^{-5}	-	8.13×10^{-5}	-
	20	1.79×10^{-7}	4.07	1.79×10^{-7}	6.07	1.68×10^{-6}	4.97	2.64×10^{-6}	4.94
	30	1.58×10^{-8}	5.99	1.58×10^{-8}	5.99	2.23×10^{-7}	4.99	3.49×10^{-7}	4.99
$M = 7$	10	1.57×10^{-7}	-	2.42×10^{-7}	-	3.88×10^{-7}	-	5.99×10^{-7}	-
	20	6.21×10^{-10}	7.98	9.72×10^{-10}	7.96	6.24×10^{-9}	5.96	9.68×10^{-9}	5.95
	30	2.80×10^{-11}	7.64	6.07×10^{-11}	6.84	5.48×10^{-10}	6.00	8.59×10^{-10}	5.97
$M = 8$	10	2.96×10^{-7}	-	4.58×10^{-7}	-	1.06×10^{-7}	-	1.64×10^{-7}	-
	20	4.80×10^{-9}	5.95	7.59×10^{-9}	5.91	8.48×10^{-10}	6.97	1.33×10^{-9}	6.94
	30	4.22×10^{-10}	6.00	6.62×10^{-10}	6.02	4.98×10^{-11}	6.99	7.82×10^{-11}	7.00
$M = 9$	5	1.89×10^{-7}	-	2.93×10^{-7}	-	1.81×10^{-7}	-	2.71×10^{-7}	-
	10	5.38×10^{-10}	8.46	8.31×10^{-10}	8.46	5.85×10^{-10}	8.28	9.24×10^{-10}	8.20
	15	1.96×10^{-11}	8.17	3.07×10^{-11}	8.13	2.24×10^{-11}	8.05	3.49×10^{-11}	8.08

Table 6. Accuracy results from grid convergence on non-uniform meshes generated by (62) for the linear advection equation.

RDO- M	N	E_1	Order	E_∞	Order	E_1	Order	E_∞	Order
Central Schemes					Upwind Schemes				
$M = 4$	10	3.99×10^{-2}	-	5.80×10^{-2}	-	1.39×10^{-2}	-	2.19×10^{-2}	-
	20	8.97×10^{-3}	2.15	1.20×10^{-2}	2.27	1.84×10^{-3}	2.92	2.87×10^{-3}	2.93
	30	4.05×10^{-3}	1.96	5.50×10^{-3}	1.92	5.49×10^{-4}	2.98	8.57×10^{-4}	2.98
$M = 5$	10	2.45×10^{-4}	-	3.81×10^{-4}	-	2.64×10^{-4}	-	4.09×10^{-4}	-
	20	1.43×10^{-5}	4.10	2.29×10^{-5}	4.06	1.43×10^{-5}	4.21	2.34×10^{-5}	4.13
	30	2.79×10^{-6}	4.02	4.51×10^{-6}	4.01	3.06×10^{-6}	3.80	4.67×10^{-6}	3.98

Table 6. Cont.

RDO-M	N	E_1	Order	E_∞	Order	E_1	Order	E_∞	Order
Central Schemes					Upwind Schemes				
M = 6	10	1.36×10^{-5}	-	2.45×10^{-5}	-	6.04×10^{-5}	-	9.24×10^{-5}	-
	20	4.11×10^{-7}	5.05	9.03×10^{-7}	4.76	1.81×10^{-6}	5.06	2.90×10^{-6}	4.99
	30	3.57×10^{-8}	6.03	7.54×10^{-8}	6.12	2.37×10^{-7}	5.01	3.68×10^{-7}	5.09
M = 7	10	2.05×10^{-7}	-	4.64×10^{-7}	-	4.17×10^{-7}	-	6.14×10^{-7}	-
	20	2.30×10^{-9}	6.48	4.96×10^{-9}	6.55	7.04×10^{-9}	5.89	1.02×10^{-8}	5.91
	30	1.71×10^{-10}	6.40	3.58×10^{-10}	6.48	5.88×10^{-10}	6.12	9.65×10^{-10}	5.82
M = 8	10	3.23×10^{-7}	-	5.94×10^{-7}	-	1.41×10^{-7}	-	2.19×10^{-7}	-
	20	5.34×10^{-9}	5.92	9.25×10^{-9}	6.00	1.02×10^{-9}	7.11	1.62×10^{-9}	7.08
	30	4.55×10^{-10}	6.07	7.58×10^{-10}	6.17	6.13×10^{-11}	6.94	9.62×10^{-11}	6.96
M = 9	5	2.07×10^{-7}	-	3.33×10^{-7}	-	2.03×10^{-7}	-	3.27×10^{-7}	-
	10	5.83×10^{-10}	8.47	9.50×10^{-10}	8.45	7.11×10^{-10}	8.15	1.12×10^{-9}	8.19
	15	2.16×10^{-11}	8.12	3.40×10^{-11}	8.21	2.50×10^{-11}	8.26	3.71×10^{-11}	8.41

Table 7. Accuracy results on uniform meshes for the 2D linear advection equation $u_t + u_x + u_y = 0$ at $t = 2\pi$.

RDO-M	$N_x \times N_y$	E_1	Order	E_∞	Order	E_1	Order	E_∞	Order
Central Schemes					Upwind Schemes				
M = 4	10×10	6.76×10^{-2}	-	1.04×10^{-1}	-	2.64×10^{-2}	-	4.08×10^{-2}	-
	20×20	1.64×10^{-2}	2.04	2.59×10^{-2}	2.01	3.43×10^{-3}	2.95	5.36×10^{-3}	2.93
	30×30	7.34×10^{-3}	1.99	1.15×10^{-2}	2.01	1.01×10^{-3}	3.00	1.59×10^{-3}	2.99
M = 5	10×10	7.48×10^{-4}	-	1.16×10^{-3}	-	4.37×10^{-4}	-	6.75×10^{-4}	-
	20×20	4.42×10^{-5}	4.08	6.95×10^{-5}	4.06	2.79×10^{-5}	3.97	4.33×10^{-5}	3.96
	30×30	8.69×10^{-6}	4.01	1.36×10^{-5}	4.02	5.46×10^{-6}	4.02	8.56×10^{-6}	4.00
M = 6	10×10	3.78×10^{-5}	-	5.84×10^{-5}	-	1.04×10^{-4}	-	1.60×10^{-4}	-
	20×20	5.60×10^{-7}	6.08	8.86×10^{-7}	6.04	3.35×10^{-6}	4.95	5.29×10^{-6}	4.92
	30×30	4.96×10^{-8}	5.98	7.78×10^{-8}	6.00	4.44×10^{-7}	4.98	6.97×10^{-7}	5.00
M = 7	10×10	2.56×10^{-6}	-	3.95×10^{-6}	-	4.91×10^{-6}	-	7.59×10^{-6}	-
	20×20	1.11×10^{-8}	7.84	1.75×10^{-8}	7.82	4.45×10^{-8}	6.79	6.97×10^{-8}	6.77
	30×30	4.60×10^{-10}	7.86	7.41×10^{-10}	7.80	2.88×10^{-9}	6.75	4.67×10^{-9}	6.67
M = 8	5×5	3.74×10^{-4}	-	5.78×10^{-4}	-	4.13×10^{-5}	-	6.38×10^{-5}	-
	10×10	6.05×10^{-6}	5.95	9.34×10^{-6}	5.95	3.31×10^{-7}	6.96	5.12×10^{-7}	6.96
	15×15	5.21×10^{-7}	6.04	8.16×10^{-7}	6.01	1.93×10^{-8}	7.01	3.03×10^{-8}	6.97

Table 8. Accuracy results for the Burger’s equation with $u(x, 0) = 0.5 + \sin x$ at $t = 0.2\pi$.

RDO-M	N	E_1	Order	E_∞	Order	E_1	Order	E_∞	Order
Central Schemes					Upwind Schemes				
M = 4	10	2.33×10^{-3}	-	1.04×10^{-1}	-	8.38×10^{-4}	-	4.64×10^{-3}	-
	20	7.17×10^{-4}	1.70	2.59×10^{-2}	2.01	1.30×10^{-4}	2.69	1.07×10^{-3}	2.12
	30	3.23×10^{-4}	1.97	1.15×10^{-2}	2.01	4.22×10^{-5}	2.77	3.98×10^{-4}	2.44
M = 5	10	3.69×10^{-4}	-	1.71×10^{-3}	-	2.63×10^{-5}	-	9.20×10^{-5}	-
	20	1.95×10^{-5}	4.24	1.53×10^{-4}	3.48	3.41×10^{-6}	2.95	1.35×10^{-5}	2.77
	30	2.17×10^{-6}	5.42	2.47×10^{-5}	4.49	7.21×10^{-7}	3.83	4.70×10^{-6}	2.61

Table 8. Cont.

RDO-M	N	E_1	Order	E_∞	Order	E_1	Order	E_∞	Order
Central Schemes					Upwind Schemes				
M = 6	10	1.68×10^{-4}	-	5.82×10^{-4}	-	1.78×10^{-5}	-	1.05×10^{-4}	-
	20	5.09×10^{-6}	5.04	3.08×10^{-5}	4.24	1.41×10^{-6}	3.66	1.67×10^{-5}	2.65
	30	5.46×10^{-7}	5.51	5.27×10^{-6}	4.36	2.48×10^{-7}	4.28	3.64×10^{-6}	3.76
M = 7	10	2.03×10^{-5}	-	1.65×10^{-4}	-	9.18×10^{-6}	-	3.19×10^{-5}	-
	20	3.63×10^{-7}	5.80	3.18×10^{-6}	5.69	1.42×10^{-7}	6.01	1.08×10^{-6}	4.89
	30	3.14×10^{-8}	6.04	2.03×10^{-7}	6.79	1.47×10^{-8}	5.60	9.37×10^{-8}	6.02
M = 8	10	2.72×10^{-5}	-	1.33×10^{-4}	-	1.62×10^{-7}	-	9.62×10^{-7}	-
	20	7.96×10^{-8}	8.42	1.03×10^{-6}	7.02	1.42×10^{-8}	3.51	1.60×10^{-7}	2.59
	30	7.20×10^{-9}	5.93	9.95×10^{-8}	5.76	2.17×10^{-9}	4.63	2.60×10^{-8}	4.49
M = 9	10	1.72×10^{-6}	-	1.20×10^{-5}	-	5.30×10^{-8}	-	4.72×10^{-7}	-
	20	3.20×10^{-8}	5.74	1.76×10^{-7}	6.08	9.41×10^{-10}	5.81	1.26×10^{-8}	5.23
	30	5.73×10^{-10}	9.92	2.68×10^{-9}	10.32	6.01×10^{-11}	6.78	7.08×10^{-10}	7.10

5.2. Advection of the Gaussian Profile

To investigate the diffusion and the dispersion of the proposed schemes and verify the long-term stability, a Gaussian profile used in [18]

$$u(x, 0) = e^{-10x^2} \tag{65}$$

is set as the initial condition for the advection Equation (31) over the $[-1, 1]$ with the periodic boundary condition. The number of cells is $N = 20$ and the time step length is $\Delta t = 0.005$ for all schemes. The numerical solution by the central and the upwind RDO-M with various M after 4 periods and 40 periods through the domain can be seen in Figures 4 and 5, respectively. Overall, the maximum value and the shape can be retained better as M increases, due to the improvement of computational accuracy. The profiles advanced by the upwind the central RDO-8 coincide the exact solution very well.

The diffusion and the dispersion agree with the Fourier analysis results in Tables 1 and 2 well. Table 1 indicates that the error of central RDO-M schemes are dominated by dispersion for all M . By contrast, Table 2 reveals that the upwind RDO-M suffers from diffusion mainly for all M with the only exception being $M = 5$. Correspondingly, the numerical solution by central RDO-4 and RDO-5 are remarkably distorted by the dispersion, especially in the long-term simulation. On the other hand, the upwind RDO schemes maintains a better and meanwhile a more flat profile than the central schemes do for all choices of M due to diffusion.

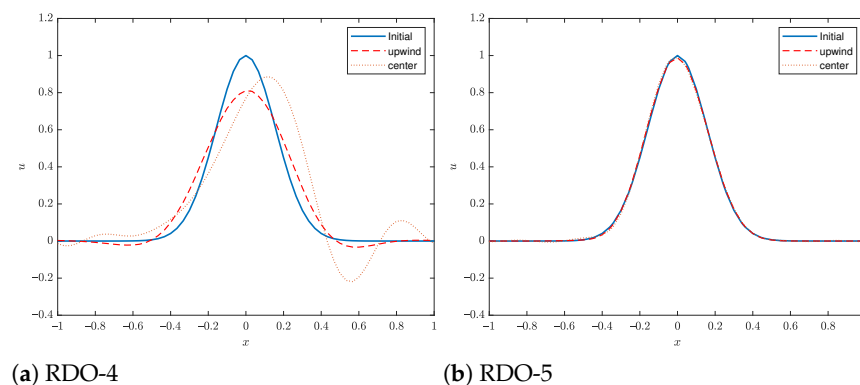


Figure 4. Cont.

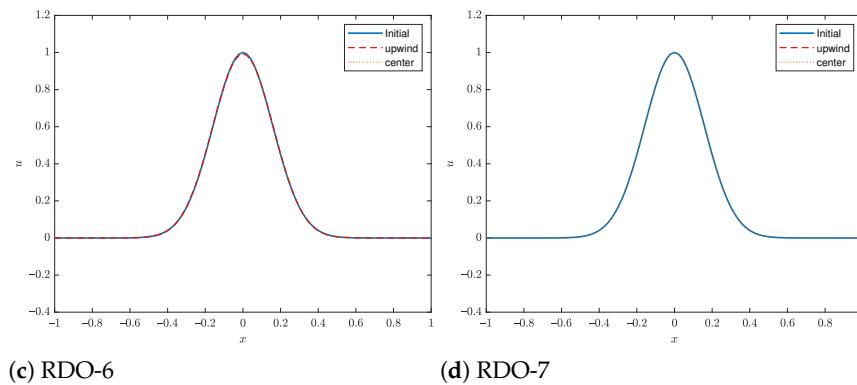


Figure 4. Numerical results at $t = 8$ (4 periods) for the advection equation with the bell function as the initial profile.

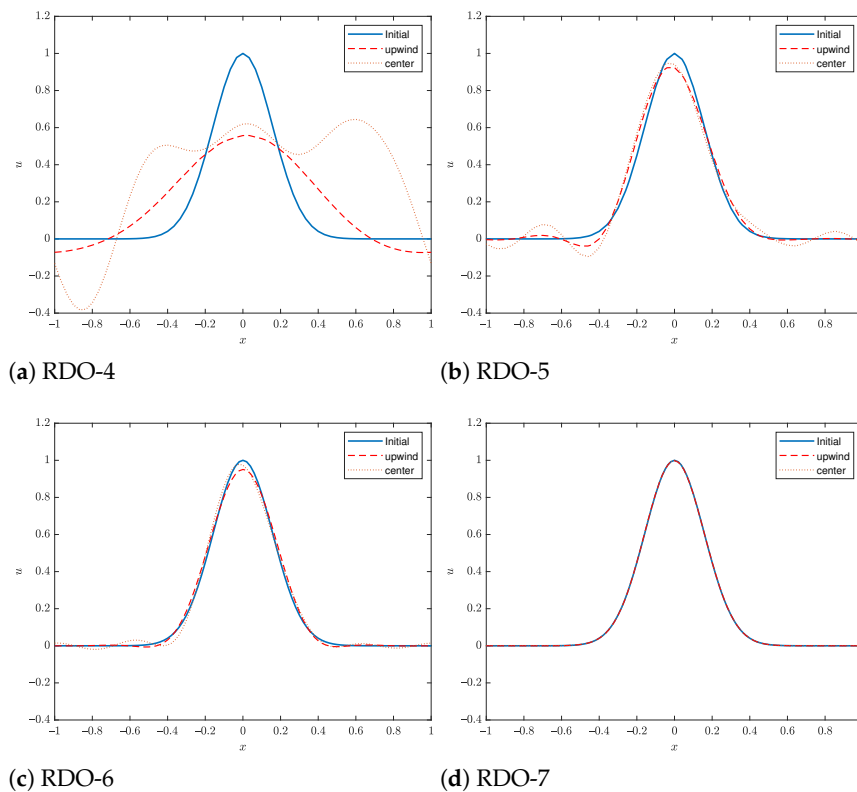


Figure 5. Numerical results at $t = 80$ (40 periods) for the advection equation with the bell function as the initial profile.

5.3. Comparison of Computational Costs

From the stability analysis, we can know that the higher-order schemes enjoy better accuracy at the expense of decreasing the CFL number and hence the efficiency. Therefore, it is necessary to compare the practical computational efficiency for RDO- M to achieve the same accuracy.

Then we consider simulating the advection of the Gaussian profile in the last test. All schemes use the fourth-order Runge–Kutta method with the temporal step length $\Delta t = 0.95c_{\max}\Delta x$ for time integration, where c_{\max} denotes the largest tolerable Courant number. Furthermore, we take $\Delta t = 0.7c_{\max}$ for the non-uniform mesh since the stability condition is more stringent in the presence of mesh non-uniformness. The target accuracy is set as $E_1 = 10^{-5}$. The number of cells N is increased gradually until the norm-1 error reaches the target accuracy. The minimal required N and the corresponding CPU time for

RDO schemes are shown in Table 9. It should be remarked that the total DOFs are not MN since the model variables at each cell boundary are shared by two cells.

For both uniform and non-uniform meshes, the significant reduction can be easily observed in the number of cells, the computational storage (i.e., total DOFs) and CPU time as M and the accuracy orders increases. RDO schemes on non-uniform meshes generally require slightly more CPU time to reach the target accuracy as shown in Table 9. Hence, high-order schemes are more attractive in terms of computational efficiency.

Table 9. The minimal required number of cells to reach the accuracy $E_1 \leq 10^{-5}$ and the corresponding resumed CPU time for RDO- M .

	RDO- M	$M = 4$	$M = 5$	$M = 6$	$M = 7$	$M = 4$	$M = 5$	$M = 6$	$M = 7$
Mesh Type		Central Schemes				Upwind Schemes			
Uniform	Minimal N	1129	67	48	27	271	62	44	22
	Total DOFs	2258	201	144	108	542	186	132	88
	CPU time (s)	55.8	0.46	0.25	0.14	6.31	0.48	0.37	0.14
Non-uniform	Minimal N	1187.2	73.4	51.6	30.0	292.4	71.2	49.8	27.2
	Total DOFs	2374.4	220.2	154.8	120.0	584.8	213.6	149.4	108.8
	CPU time (s)	70.46	0.65	0.37	0.19	7.93	0.62	0.47	0.18

6. Conclusions

A family of compact multi-moment schemes that use high-order derivative moments have been proposed, analyzed and tested in one dimension and two dimensions. Using the model variables at cell boundaries, the proposed schemes reconstruct the solution polynomial within a single cell and then retrieve the temporal derivatives of model variables efficiently. The central scheme also achieves excellent performance in several tests, although this scheme does not possess clear physical meaning as the upwind scheme does. This indicates that one may extend the scheme to the more complicated scenarios without considering finding the upwind direction.

The future work will focus on extending this formulation to the more generalized unstructured triangular mesh using the Argyris polynomials [24] which also have continuous gradient globally. Furthermore, designing the conservative counterpart of the current version is also on our schedule.

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

Funding: This work is 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.

References

1. Wang, Z.J.; Fidkowski, K.; Abgrall, R.; Bassi, F.; Caraeni, D.; Cary, A.; Deconinck, H.; Hartmann, R.; Hillewaert, K.; Huynh, H.T.; et al. High-order CFD methods: Current status and perspective. *Int. J. Numer. Methods Fluids* **2013**, *72*, 811–845. [CrossRef]
2. 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]
3. Yabe, T.; Xiao, F.; Utsumi, T. The constrained interpolation profile method for multiphase analysis. *J. Comput. Phys.* **2001**, *169*, 556–593. [CrossRef]

4. Tanaka, R.; Nakamura, T.; Yabe, T. Constructing exactly conservative scheme in a non-conservative form. *Comput. Phys. Commun.* **2000**, *126*, 232–243. [[CrossRef](#)]
5. 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](#)]
6. Xiao, F.; Akoh, R.; Li, S. Unified formulation for compressible and incompressible flows by using multi-integrated moments II: Multi-dimensional version for compressible and incompressible flows. *J. Comput. Phys.* **2006**, *213*, 31–56. [[CrossRef](#)]
7. 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](#)]
8. Lin, S.; Luo, Q.; Leng, H.; Song, J. Alternating Polynomial Reconstruction Method for Hyperbolic Conservation Laws. *Mathematics* **2021**, *9*, 1885. [[CrossRef](#)]
9. Aoki, T. Interpolated differential operator (IDO) scheme for solving partial differential equations. *Comput. Phys. Commun.* **1997**, *102*, 132–146. [[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. Xiao, F.; Li, S. CIP/Multi-moment finite volume method with arbitrary order of accuracy. *arXiv* **2012**, arXiv:1207.6822.
12. Toro, E.F.; Titarev, V.A. Derivative Riemann solvers for systems of conservation laws and ADER methods. *J. Comput. Phys.* **2006**, *212*, 150–165. [[CrossRef](#)]
13. Li, S.; Xiao, F. High order multi-moment constrained finite volume method. Part I: Basic formulation. *J. Comput. Phys.* **2009**, *228*, 3669–3707. [[CrossRef](#)]
14. Deng, X.; Xie, B.; Xiao, F. Multimoment finite volume solver for euler equations on unstructured grids. *AIAA J.* **2017**, *55*, 2617–2629. [[CrossRef](#)]
15. Cheng, L.; Deng, X.; Xie, B.; Jiang, Y.; Xiao, F. Low-dissipation BVD schemes for single and multi-phase compressible flows on unstructured grids. *J. Comput. Phys.* **2021**, *428*, 110088. [[CrossRef](#)]
16. 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.
17. 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](#)]
18. 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.
19. Taylor, M.; Tribbia, J.; Iskandarani, M. The spectral element method for the shallow water equations on the sphere. *J. Comput. Phys.* **1997**, *130*, 92–108. [[CrossRef](#)]
20. Süli, E.; Mayers, D.F. *An Introduction to Numerical Analysis*; Cambridge University Press: Cambridge, UK, 2003.
21. Patera, A.T. A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *J. Comput. Phys.* **1984**, *54*, 468–488. [[CrossRef](#)]
22. Sakurai, K.; Aoki, T.; Lee, W.H.; Kato, K. Poisson equation solver with fourth-order accuracy by using interpolated differential operator scheme. *Comput. Math. Appl.* **2002**, *43*, 621–630. [[CrossRef](#)]
23. Zhang, S. On the full C1-Qk finite element spaces on rectangles and cuboids. *Adv. Appl. Math. Mech* **2010**, *2*, 701–721. [[CrossRef](#)]
24. Ciarlet, P.G. *The Finite Element Method for Elliptic Problems*; SIAM: Philadelphia, PA, USA, 2002.