Geosci. Model Dev., 18, 8175–8201, 2025
https://doi.org/10.5194/gmd-18-8175-2025

Geoscientific Model Development
Open Access

Model description paper

HOPE: an arbitrary-order non-oscillatory finite-volume shallow water dynamical core with automatic differentiation

Lilong Zhou[1,2,3], **Wei Xue**[1], **and Xueshun Shen**[2,3]

[1]Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
[2]Department of Model Technology, CMA Earth System Modeling and Prediction Centre (CEMC), Beijing, 100081, China
[3]State Key Laboratory of Severe Weather Meteorological Science and Technology (LaSW), Beijing, 100081, China

Correspondence: Wei Xue (xuewei@tsinghua.edu.cn) and Xueshun Shen (shenxs@cma.gov.cn)

Received: 23 April 2025 – Discussion started: 27 May 2025
Revised: 17 September 2025 – Accepted: 22 September 2025 – Published: 5 November 2025

Abstract. This study presents the High Order Prediction Environment (HOPE), an automatically differentiable, non-oscillatory finite-volume dynamical core for shallow water equations on the cubed-sphere grid. HOPE integrates five key features: (1) arbitrary high-order accuracy through genuine two-dimensional reconstruction schemes; (2) essential non-oscillation via adaptive polynomial order reduction in discontinuous regions; (3) exact mass conservation inherited from finite-volume discretization; (4) automatically differentiable and (5) GPU-native scalability through PyTorch-based implementation. Another innovation is the development of a two-way coupled ghost cell interpolation method. This approach incorporates information from adjacent panels on both sides of the boundary, thereby overcoming the integration instability inherent in one-sided ghost cell interpolation approaches when using high-order reconstruction. Leveraging the linear operator nature of this interpolation scheme, we optimized its computation: information exchange across the panel boundary is achieved through a single matrix-vector multiplication instead of iterative coupling, without accuracy loss. Numerical experiments demonstrate the capabilities of HOPE: The 11th-order scheme reduces errors to near double-precision round-off levels in steady-state geostrophic flow tests on coarse grids. Maintenance of Rossby–Haurwitz waves over 100 simulation days without crashing. A cylindrical dam-break test case confirms the genuinely two-dimensional WENO scheme exhibits significantly better isotropy compared to dimension-by-dimension approaches. Moreover, normalized conservation errors in total energy, total potential enstrophy, and total zonal angular momentum significantly reduce with increasing order of the reconstruction scheme. Two implementations are developed: a Fortran version for convergence analysis and a PyTorch version leveraging automatic differentiation and GPU acceleration. The PyTorch implementation maps reconstruction and quadrature operation to 2D convolution and Einstein summation respectively, achieving about 2× speedup on single NVIDIA RTX3090 GPU versus Dual Intel E5-2699v4 CPUs execution. This design enables seamless coupling with neural network parameterizations, positioning HOPE as a foundational tool for next-generation differentiable atmosphere models.

1 Introduction

Recent years have witnessed a surge in research integrating numerical weather prediction (NWP) with artificial intelligence (AI) techniques. A prominent advancement in this domain is the hybrid modeling paradigm, which synergizes the complementary strengths of both approaches. This framework implements numerical dynamical cores within AI software platforms such as TensorFlow or PyTorch, thereby enabling seamless integration of AI models into the numerical solution process for atmospheric dynamical partial differential equations (PDEs). Unlike the fully surrogated methods, such as Pangu-Weather (Bi et al., 2022), FengWu (Chen et al., 2023), GraphCast (Lam et al., 2023), NowcastNet (Zhang et al., 2023), hybrid model integrates traditional PDE-based dynamical cores with neural network (NN)-based physical parameterizations. The auto-differentiable nature of the dynamical core enables training losses to propagate through the entire model during backpropagation, allowing the NN-based parameterization module to access more comprehensive residual information. NeuralGCM (Kochkov et al., 2024) exemplifies this hybrid approach by combining a spectral numerical dynamical core with NN-based physical parameterizations. The governing equation-based dynamical core imposes rigorous physical constraints within the framework, effectively mitigating the blurriness characteristic of purely data-driven models. Furthermore, NeuralGCM demonstrates superior power spectra performance compared to conventional data-driven meteorological models. While the implementation of a spectral dynamical core in NeuralGCM theoretically enables infinite-order accuracy, the global nature of spectral expansion restricts the method's scalability. Furthermore, in contrast to finite-volume algorithms which inherently ensure strict mass conservation, achieving strict mass conservation with NeuralGCM's spectral dynamical core requires supplementary modifications.

To address these shortcomings, we present the High Order Prediction Environment (HOPE) dynamical core with following contributions:

1. A new-generation shallow-water model architecture integrating:
   i. Arbitrary high-order accuracy (up to 13th-order verified) via tensor product polynomial (TPP).
   ii. A finite-volume scheme requiring only information from a local stencil surrounding each cell to perform state updates, enabling massively parallel scalability.
   iii. Inherent mass conservation from finite-volume discretization.
   iv. A WENO (Weighted Essentially Non-Oscillatory) based, adaptive polynomial order reduction for essential non-oscillation.
2. A novel two-way coupled ghost cell interpolation scheme achieving:
   i. Arbitrary odd-order convergence through central stencil interpolation.
   ii. Single sparse matrix-vector operation replacing iterative procedures (Eq. A.12).
   iii. Overcome numerical instability beyond 7th-order accuracy.
3. PyTorch-based high performance differentiable implementation featuring:
   i. GPU acceleration through convolution/einsum operator in PyTorch, $2\times$ speedup on single RTX3090 GPU vs. Dual Intel Xeon 2699v4 CPUs.
   ii. Automatic ghost cell interpolation matrix generation via native auto-differentiation.
   iii. Seamless integration with NN modules for hybrid modeling.

In the following part of the introduction, we introduce the relevant work on constructing the HOPE model, and from this, we elaborate on the challenges and motivations for establishing the algorithm of the dynamical core. High-order accuracy is an extremely appealing trait for the design of a dynamical core, particularly in high-resolution atmospheric simulations. A dynamical core model with high-order accuracy produces significantly less simulation error in smooth regions compared to a low-order model. Furthermore, even when the resolution is equivalent or coarser, a high-order model is capable of resolving finer details than a low-order one.

A high-order finite volume model was developed on cubed sphere, named MCORE (Ullrich et al., 2010; Ullrich and Jablonowski, 2012). High-order reconstruction requires information from cells external to panel boundaries (commonly termed ghost cells). Due to coordinate discontinuities across the six panels of the cubed-sphere grid, MCORE implements an interpolation scheme for ghost cells based on one-side information. This approach employs a two-dimensional reconstruction stencil to interpolate prognostic variables onto Gaussian quadrature points within each cell, followed by integration to obtain cell-averaged values. The authors assert that MCORE's convergence rate can theoretically be of arbitrary order. However, during the design of the ghost cell interpolation for HOPE, we initially attempted to use a one-sided reconstruction stencil similar to MCORE. While stable integration was achieved with the 3rd-, 5th-, and 7th-order schemes, the model became unstable when schemes of 9th-order or higher were used. In other words, for HOPE, overcoming the 7th-order accuracy limitation necessitated the development of a new ghost cell interpolation scheme.

Therefore, we designed a bilateral interpolation algorithm. This algorithm employs an iterative procedure that incorporates information from both adjacent panels of the cubed-sphere grid simultaneously. This enabled stable model integration even with higher-order schemes. Though not detailed in the paper, our testing confirmed stable integration even at 13th-order accuracy.

In this article, we devise the reconstruction based on tensor product polynomial (TPP). When the stencil width is k, our method achieves kth order accuracy, surpassing MCORE by one order of accuracy with the same stencil width. In addition, we have developed a new class of ghost interpolation schemes that abandon the use of one-sided stencils and instead adopt central stencils. This new approach enables the scheme to overcome the non-physical oscillations arising from interpolation at panel boundaries. Our method allows for arbitrary order of accuracy while the field is smooth enough, and we have verified this by testing up to the 11th order.

Nevertheless, higher-order reconstruction does not invariably yield superior simulation outcomes, as elucidated by analyzing the properties of the Taylor series remainder term. The accuracy of approximating a function via a Taylor series requires two essential conditions: (1) the existence of higher-order derivatives of the function at the expansion point, and (2) the convergence of the series within the relevant domain. When the field exhibits poor continuity – where higher-order derivatives either do not exist or lead to increasing residuals with series order – employing higher-order approximations can introduce significant errors. Therefore, for reconstruction schemes based on polynomial functions, high-order accuracy should only be adopted when the field is sufficiently smooth. Conversely, for discontinuous or poorly continuous fields, reducing the reconstruction order is necessary to maintain numerical stability and effectiveness.

The Weighted Essentially Non-Oscillatory (WENO) scheme is an adaptive limiter widely employed in computational fluid dynamics (CFD) to address this challenge. Originally developed for one-dimensional problems (Liu et al., 1994), WENO was later extended to two dimensions by Shi et al. (2002) using two distinct approaches: a genuinely two-dimensional (WENO2D) scheme and a dimension-by-dimension reconstruction. In this work, we implement WENO2D scheme to enforce the non-oscillatory property. This approach effectively suppresses non-physical oscillations near sharp discontinuities while preserving high-order accuracy in smooth regions.

The remainder of this paper is organized as follows: Sect. 2 details the governing equations on the cubed-sphere grid. Section 3 presents the numerical methods, including reconstruction schemes, panel boundary treatment method, and temporal marching scheme. Section 4 focuses on HOPE's high-performance implementation leveraging PyTorch's built-in operators for GPU acceleration. The adoption of PyTorch simultaneously enables automatic differentiation capabilities through its computational graph construction. Section 5 validates model performance through standard test cases, followed by conclusions and future directions in Sect. 6.

2 Governing equation on cubed sphere

The cubed-sphere grid partitions the spherical domain into six panels, each with a structured and rectangular computational space. This configuration facilitates high-order spatial reconstruction and efficient massive-thread parallelism (see Fig. 1). Early work on solving the primitive equations on the cubed-sphere grid dates back to Sadourny (1972). In recent decades, the cubed-sphere grid has been widely adopted in high-order-accuracy atmospheric models. For instance, Chen and Xiao (2008) developed a shallow water model using the multi-moment constrained finite volume method on the cubed sphere, achieving 3rd–4th order accuracy. Ullrich et al. (2010) designed a high-order finite volume dynamical core based on this grid, Nair et al. (2005a, b) implemented a discontinuous Galerkin model on the cubed sphere.

In this study, we also employ the equiangular cubed-sphere grid. Although the mesh is non-orthogonal, the computational space can still be treated as a rectangular grid by adopting a generalized curvilinear coordinate system. In this section, we present the shallow water equations in generalized curvilinear coordinates and discuss specialized treatments for topography.

Shallow water equation set on gnomonic equiangular cubed sphere grid is written as

$$\begin{cases} \frac{\partial\sqrt{G}\phi}{\partial t}+\frac{\partial\sqrt{G}\phi u}{\partial x}+\frac{\partial\sqrt{G}\phi v}{\partial y}=0 \\ \frac{\partial\sqrt{G}\phi u}{\partial t}+\frac{\partial\sqrt{G}\left(\phi uu+\frac{1}{2}G^{11}\phi^2\right)}{\partial x}+\frac{\partial\sqrt{G}\left(\phi uv+\frac{1}{2}G^{12}\phi^2\right)}{\partial y}=\psi_M^1+\psi_C^1+\psi_B^1 \\ \frac{\partial\sqrt{G}\phi v}{\partial t}+\frac{\partial\sqrt{G}\left(\phi uv+\frac{1}{2}G^{21}\phi^2\right)}{\partial x}+\frac{\partial\sqrt{G}\left(\phi vv+\frac{1}{2}G^{22}\phi^2\right)}{\partial y}=\psi_M^2+\psi_C^2+\psi_B^2 \end{cases}. \quad (1)$$

The gnomonic equiangular coordinates are represented by (x, y, p), where $(x, y) \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ are local equiangular coordinate of each panel and $p = 1, 2, 3, \ldots, n_p$ is panel index as shown in Fig. 1b; $n_p = 6$ is the number of panels. $\phi = gh$ is geopotential, h is fluid thickness, u and v are the contravariant wind components in the x and y directions, respectively, g is gravity acceleration. ψ_M, ψ_C, ψ_B are the metric term, Coriolis term and bottom topography influence term

$$\psi_M=\begin{pmatrix}\psi_M^1\\ \psi_M^2\end{pmatrix}=\frac{2\sqrt{G}}{\delta^2}\begin{pmatrix}-XY^2\phi uu+Y\left(1+Y^2\right)\phi uv\\ X\left(1+X^2\right)\phi uv-X^2Y\phi vv\end{pmatrix} \quad (2)$$

$$\psi_C=-\sqrt{G}\sqrt{G}f\mathbf{k}\times\phi\mathbf{u}=\sqrt{G}f\begin{pmatrix}-G^{12} & G^{11}\\ -G^{22} & G^{12}\end{pmatrix}\begin{pmatrix}\sqrt{G}\phi u\\ \sqrt{G}\phi v\end{pmatrix} \quad (3)$$

$$\psi_B=-\sqrt{G}\phi G^{ij}\frac{\partial\phi_s}{\partial x^j}=-\sqrt{G}\phi\begin{pmatrix}G^{11}\frac{\partial\phi_s}{\partial x}+G^{12}\frac{\partial\phi_s}{\partial y}\\ G^{21}\frac{\partial\phi_s}{\partial x}+G^{22}\frac{\partial\phi_s}{\partial y}\end{pmatrix} \quad (4)$$

where $X = \tan x, Y = \tan y, \delta = \sqrt{1 + X^2 + Y^2}$, $f = 2\Omega \sin\theta$ is Coriolis parameter, $\phi_s = gh_s$ is surface geopotential, and h_s is surface height.

$$\sin\theta=\begin{cases} Y/\delta, & p\in\{1,2,3,4\}\\ 1/\delta, & p=5\\ -1/\delta, & p=6\end{cases} \quad (5)$$

The contravariant metric on cubed-sphere is

$$G^{ij}=\frac{\delta^2}{r^2\left(1+X^2\right)\left(1+X^2\right)}\begin{pmatrix}1+Y^2 & XY\\ XY & 1+X^2\end{pmatrix}. \quad (6)$$

The covariant metric

$$G_{ij}=\frac{r^2\left(1+X^2\right)\left(1+Y^2\right)}{\delta^4}\begin{pmatrix}1+X^2 & -XY\\ -XY & 1+Y^2\end{pmatrix} \quad (7)$$

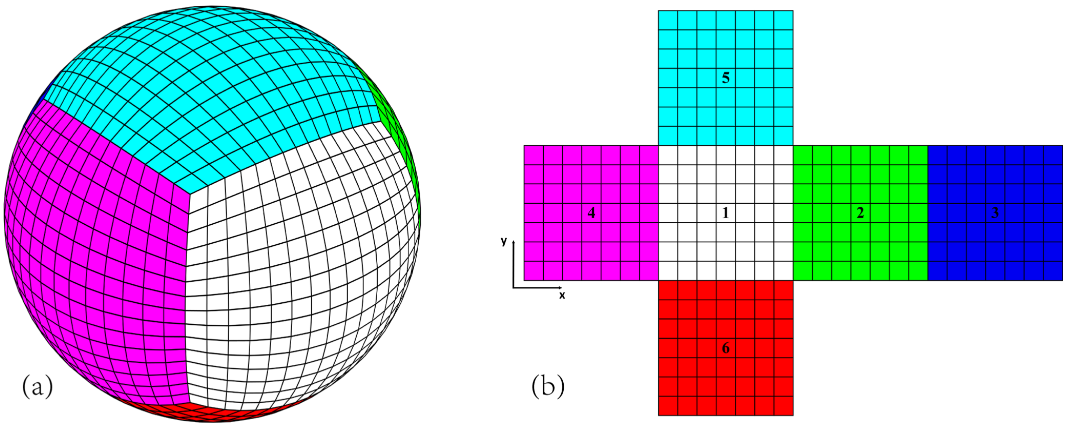

Figure 1. Cubed sphere grid. **(a)** Physical space; **(b)** computational space. Six panels are identified by indices from 1 to 6.

and the metric determinant is given by

$$\sqrt{G} = \sqrt{\det\left(G_{ij}\right)} = \frac{r^2\left(1+X^2\right)\left(1+Y^2\right)}{\delta^3}. \tag{8}$$

r is radius of earth.

The contravariant wind vector $\boldsymbol{V} = (u, v)$ can be convert to wind vector on spherical LAT/LON coordinate $\boldsymbol{V}_\mathrm{s} = (u_\mathrm{s}, v_\mathrm{s})$ by the following formula

$$\begin{pmatrix} u_\mathrm{s} \\ v_\mathrm{s} \end{pmatrix} = \mathbf{J}\begin{pmatrix} u \\ v \end{pmatrix} \tag{9}$$

where $\mathbf{J}$ is a 2×2 conversion matrix, the expressions are different in each panel

$$\mathbf{J} = r\begin{pmatrix} \cos\theta\frac{\partial\lambda}{\partial x} & \cos\theta\frac{\partial\lambda}{\partial y} \\ \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial y} \end{pmatrix} =$$
$$\begin{cases} r\begin{pmatrix} \cos\theta & 0 \\ -\sin\theta\cos\theta\tan\lambda_p & \cos\lambda_p\cos^2\theta + \frac{\sin^2\theta}{\cos\lambda_p} \end{pmatrix}, & p \in \{1,2,3,4\} \\ r\begin{pmatrix} \cos\lambda\sin\theta\Gamma_1 & \sin\lambda\sin\theta\Gamma_2 \\ -\sin\lambda\sin^2\theta\Gamma_1 & \cos\lambda\sin^2\theta\Gamma_2 \end{pmatrix}, & p = 5 \\ r\begin{pmatrix} -\cos\lambda\sin\theta\Gamma_1 & \sin\lambda\sin\theta\Gamma_2 \\ \sin\lambda\sin^2\theta\Gamma_1 & \cos\lambda\sin^2\theta\Gamma_2 \end{pmatrix}, & p = 6 \end{cases} \tag{10}$$

$$\lambda_p = \lambda - \frac{\pi}{2}(p-1),\ \Gamma_1 = 1 + \frac{\sin^2\lambda}{\tan^2\theta},\ \Gamma_2 = 1 + \frac{\cos^2\lambda}{\tan^2\theta} \tag{11}$$

where λ, θ are longitude and latitude. The relation between $\mathbf{J}$ and G_{ij} is

$$G_{ij} = \mathbf{J}^T\mathbf{J}. \tag{12}$$

To discretize and solve the equation system, we first perform reconstruction on the prognostic variables to obtain their values at the cell interfaces. These reconstructed values are then used within a Riemann solver to compute the numerical fluxes. During the numerical experiments, we observed that reconstructing $\sqrt{G}\phi$ directly leads to non-physical oscillations. This occurs because topography may induce discontinuities in the variable ϕ, while high-order reconstruction fundamentally requires smoothness of the field.

To address this, inspired by the approach mentioned by Ii and Xiao (2010), we instead reconstruct $\sqrt{G}\phi_t$, where $\phi_t = \phi + \phi_\mathrm{s}$ is total geopotential. The detailed formulation of this reconstruction method is presented in Sect. 3. Crucially, $\sqrt{G}\phi_t$ is used exclusively during the reconstruction step. The prognostic variable remains $\sqrt{G}\phi$ to ensure exact mass conservation.

The momentum equations need to be modified as follow

$$\begin{cases} \frac{\partial\sqrt{G}\phi}{\partial t} + \frac{\partial\sqrt{G}\phi u}{\partial x} + \frac{\partial\sqrt{G}\phi v}{\partial y} = 0 \\ \frac{\partial\sqrt{G}\phi u}{\partial t} + \frac{\partial\sqrt{G}\left(\phi uu + \frac{1}{2}G^{11}\phi_t^2\right)}{\partial x} + \frac{\partial\sqrt{G}\left(\phi uv + \frac{1}{2}G^{12}\phi_t^2\right)}{\partial y} = \psi_\mathrm{M}^1 + \psi_\mathrm{C}^1 + \psi_\mathrm{B}^1 \\ \frac{\partial\sqrt{G}\phi v}{\partial t} + \frac{\partial\sqrt{G}\left(\phi uv + \frac{1}{2}G^{21}\phi_t^2\right)}{\partial x} + \frac{\partial\sqrt{G}\left(\phi vv + \frac{1}{2}G^{22}\phi_t^2\right)}{\partial y} = \psi_\mathrm{M}^2 + \psi_\mathrm{C}^2 + \psi_\mathrm{B}^2 \end{cases} \tag{13}$$

and the bottom topography influence term is now expressed as

$$\psi_\mathrm{B} = \sqrt{G}\phi_\mathrm{s}G^{ij}\frac{\partial\phi_t}{\partial x^j} = \sqrt{G}\phi_\mathrm{s}\begin{pmatrix} G^{11}\frac{\partial\phi_t}{\partial x} + G^{12}\frac{\partial\phi_t}{\partial y} \\ G^{21}\frac{\partial\phi_t}{\partial x} + G^{22}\frac{\partial\phi_t}{\partial y} \end{pmatrix}. \tag{14}$$

The reconstruction variables are $(\sqrt{G}\phi_t, \sqrt{G}\phi u, \sqrt{G}\phi v)$.

We write the governing equation set to vector form

$$\frac{\partial\boldsymbol{q}}{\partial t} + \frac{\partial\boldsymbol{F}(\boldsymbol{q})}{\partial x} + \frac{\partial\boldsymbol{G}(\boldsymbol{q})}{\partial y} = \boldsymbol{S}(\boldsymbol{q}) \tag{15}$$

$$\boldsymbol{q} = \begin{bmatrix} \sqrt{G}\phi \\ \sqrt{G}\phi u \\ \sqrt{G}\phi v \end{bmatrix},\ \boldsymbol{F} = \begin{bmatrix} \sqrt{G}\phi u \\ \sqrt{G}\left(\phi uu + \frac{1}{2}G^{11}\phi_t^2\right) \\ \sqrt{G}\left(\phi uv + \frac{1}{2}G^{21}\phi_t^2\right) \end{bmatrix},$$

$$\boldsymbol{G} = \begin{bmatrix} \sqrt{G}\phi v \\ \sqrt{G}\left(\phi uv + \frac{1}{2}G^{12}\phi_t^2\right) \\ \sqrt{G}\left(\phi vv + \frac{1}{2}G^{22}\phi_t^2\right) \end{bmatrix}$$

$$\boldsymbol{S} = \begin{bmatrix} 0 \\ \psi_\mathrm{M}^1 + \psi_\mathrm{C}^1 + \psi_\mathrm{B}^1 \\ \psi_\mathrm{M}^2 + \psi_\mathrm{C}^2 + \psi_\mathrm{B}^2 \end{bmatrix}. \tag{16}$$

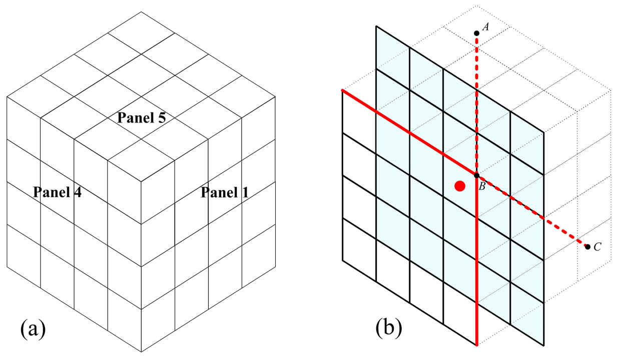

Figure 2. **(a)** Adjacent area of panels 1, 4 and 5. **(b)** 5×5 reconstruction stencil nearby panel corner is represented by shade. The cell contains red dot is the target cell on panel 4; the magenta points are overlapped GQPs shared by panel 1 and panel 5; red solid lines are boundary of panel 4, red dash lines are extension line of panel 4 boundary line. A and C are points on dash line, B is the upper right corner point of panel 4.

3 Numerical discretization

The finite volume method computes the temporal tendency of cell-averaged quantities by evaluating the net flux across cell interfaces. The interfacial flux is obtained through Gaussian quadrature, where the field values at quadrature points are reconstructed spatially and then processed by a Riemann solver to determine the flux magnitude.

In this section, we present two distinct spatial reconstruction approaches: (1) a two-dimensional tensor product polynomial (TPP) method, and (2) a two-dimensional weighted essentially non-oscillatory (WENO2D) scheme based on tensor product polynomials. Each reconstruction yields two potential values at every Gaussian quadrature point (GQP). These values are then resolved into a single flux value using the Low Mach number Approximate Riemann Solver (LMARS) (Chen et al., 2013) or AUSM+-up (Liou, 2006; Ullrich et al., 2010). Even with an approximate Riemann solver like LMARS, the scheme preserves high-order because it combines high-order reconstructions from both sides of the cell interface to determine the flux. Finally, the total flux across each cell edge is computed by applying linear Gaussian quadrature integration along the interface.

According to the finite volume scheme, average Eq. (15) on cell i, j, we have

$$\frac{\partial \overline{\boldsymbol{q}}_{i,j}}{\partial t} + \frac{\overline{\boldsymbol{F}}_{i+\frac{1}{2},j} - \overline{\boldsymbol{F}}_{i-\frac{1}{2},j}}{\Delta x} + \frac{\overline{\boldsymbol{G}}_{i,j+\frac{1}{2}} - \overline{\boldsymbol{G}}_{i,j-\frac{1}{2}}}{\Delta y} = \overline{\boldsymbol{S}}_{i,j} \quad (17)$$

$$\frac{\partial \overline{\boldsymbol{q}}_{i,j}}{\partial t} = \frac{1}{\Delta x \Delta y}\frac{\partial}{\partial t}\iint_{\Omega_{i,j}} \boldsymbol{q}\,\mathrm{d}x\mathrm{d}y, \overline{\boldsymbol{S}}_{i,j} = \frac{1}{\Delta x \Delta y}\iint_{\Omega_{i,j}} S\mathrm{d}x\mathrm{d}y \quad (18)$$

$$\overline{\boldsymbol{F}}_{i-\frac{1}{2},j} = \frac{1}{\Delta y}\int_{e_{i-\frac{1}{2}}} \boldsymbol{F}\mathrm{d}y,\ \overline{\boldsymbol{F}}_{i+\frac{1}{2},j} = \frac{1}{\Delta y}\int_{e_{i+\frac{1}{2}}} \boldsymbol{F}\boldsymbol{d}y \quad (19)$$

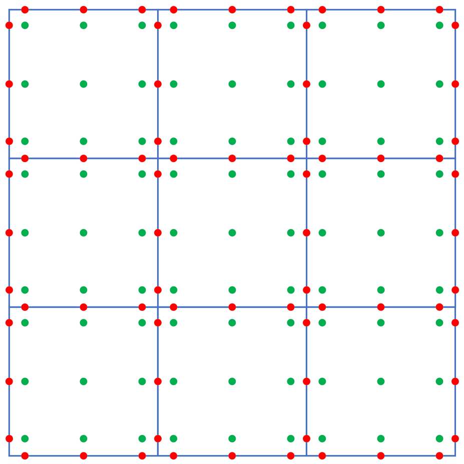

Figure 3. Function points on cell. Red points are edge quadrature points (EQP) or called flux points, green points are inner cell quadrature points (CQP).

$$\overline{\boldsymbol{G}}_{i,j-\frac{1}{2}} = \frac{1}{\Delta x}\int_{e_{j-\frac{1}{2}}} \boldsymbol{G}\mathrm{d}x,\ \overline{\boldsymbol{G}}_{i,j+\frac{1}{2}} = \frac{1}{\Delta x}\int_{e_{j+\frac{1}{2}}} \boldsymbol{G}\mathrm{d}x \quad (20)$$

where $\Omega_{i,j}$ represents the region overlapped by cell (i, j), $e_{i-\frac{1}{2}}$, $e_{i+\frac{1}{2}}$, $e_{j-\frac{1}{2}}$, $e_{j+\frac{1}{2}}$ are left, right, bottom, top edges of cell (i, j).

The physical interpretation of equation Eq. (17) is that the average tendency of prognostic field $\boldsymbol{q}$ within cell (i, j) is governed by the average net flux and average source. In this study, we calculate these averages using Gaussian quadrature, the function points within each cell are illustrated in Fig. 3, the EQPs are share by adjacent cells, and CQPs are exclusive for each cell.

Average on edge by 1D scheme:

$$\overline{\boldsymbol{F}}_{i+\frac{1}{2},j} \approx \sum_{\xi=1}^{m_\mathrm{e}} w_\xi \boldsymbol{F}_\xi = \boldsymbol{w}\overrightarrow{\boldsymbol{F}} \tag{21}$$

where m_e is the number of quadrature points on each edge, $\boldsymbol{w} = (w_1, w_2, \ldots, w_{m_\mathrm{e}})$ is the 1D Gaussian quadrature coefficient vector. $\overrightarrow{\boldsymbol{F}} = (\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_{m_\mathrm{e}})^T$ is the vector of flux, the elements of $\overrightarrow{\boldsymbol{F}}$ represent the flux on EQPs.

Average in cell by 2D scheme:

$$\overline{\boldsymbol{S}}_{i,j} \approx \sum_{\xi=1}^{m_\mathrm{c}} W_\xi \boldsymbol{S}_\xi = \boldsymbol{W}\overrightarrow{\boldsymbol{S}} \tag{22}$$

where m_c is the number of quadrature points on each cell, $\boldsymbol{W} = (W_1, W_2, \ldots, W_{m_\mathrm{c}})$ is the 2D Gaussian quadrature coefficient matrix, $\overrightarrow{\boldsymbol{S}} = (\mathbf{S}_1, \boldsymbol{S}_2, \ldots, \mathbf{S}_{m_\mathrm{c}})^T$ is the vector of source term, the elements of $\boldsymbol{S}$ represent the source value on GQPs, superscript T stands for transpose matrix.

HOPE employs an equiangular cubed-sphere grid, where each panel undergoes uniform angular discretization into $n_\mathrm{c} \times n_\mathrm{c}$ cells. In the computational space (equiangular coordinates), each cell spans an angular interval of $\frac{\pi}{2n_\mathrm{c}}$, therefore

$$\Delta x = \Delta y = \frac{\pi}{2n_\mathrm{c}}. \tag{23}$$

This uniformity ensures that all cells are geometrically identical in the computational space, thereby avoiding the need for cell-specific treatment during reconstruction studies. In the following part of this section, we set a new computational space for reconstruction process. The local coordinate system $(\hat{x}, \hat{y})$ is established such that within each reconstruction stencil, the origin (0, 0) is located at the stencil center, the central cell spans [−0.5, 0.5] in both $\hat{x}$ and $\hat{y}$ directions, as shown in Fig. 4a. All of the cells have the same size in $\hat{x}$, $\hat{y}$ directions:

$$\Delta\hat{x} = \Delta\hat{y} = 1. \tag{24}$$

On the cubed-sphere grid, a fixed reconstruction scheme yields consistent stencils across all cells. This structural homogeneity renders the reconstruction operation computationally equivalent to two-dimensional convolution, thereby enabling efficient GPU acceleration through PyTorch's built-in conv2d function.

3.1 Tensor product polynomial (TPP) reconstruction

HOPE employs genuinely two-dimensional reconstruction, simultaneously incorporating information in both spatial dimensions to minimize dimensional splitting errors. For computational efficiency, reconstruction algorithms using square stencils are computationally equivalent to convolution operations. This equivalence allows efficient implementation via PyTorch's *conv2d* function for acceleration.

To construct genuinely 2D reconstructions, the functional form of the reconstruction basis must be selected. A bivariate polynomial of degree d contains $\frac{(d+1)(d+2)}{2}$ terms. As illustrated in Fig. 4b, the 6 terms of a bivariate quadratic polynomial ($d = 2$) are insufficient to cover a square stencil. To address this, we adopt Tensor Product Polynomials (TPP) as basis functions. We denote a TPP function containing $n \times n$ terms as TPPn. Determining the coefficients of TPPn requires information from a $n \times n$ block of cells. When using a TPP reconstruction stencil of size $n \times n$, HOPE achieves fifth-order accuracy when simulating smooth flow fields. We therefore designate a TPP reconstruction stencil of size $n \times n$ as an nth order TPP stencil, the 3rd and 5th order TPP stencils are shown in Fig. 4c and d.

A TPPn polynomial is expressed as

$$p\left(\hat{x}, \hat{y}\right) = \sum_{\hat{j}=1}^{n}\sum_{\hat{i}=1}^{n} a_k \hat{x}^{\hat{i}-1}\hat{y}^{\hat{j}-1} = \sum_{k=1}^{N} a_k c_k\left(\hat{x}, \hat{y}\right) \tag{25}$$

where n is width of stencil. a_k is the coefficient of each term, the term index $k = \hat{i} + n(\hat{j} - 1)$, and $c_k(\hat{x}, \hat{y}) = \hat{x}^\alpha \hat{y}^\beta$, $\alpha = k - \mathrm{int}\left(\frac{k-1}{n}\right)n - 1$, $\beta = \mathrm{int}\left(\frac{k-1}{n}\right)$, "int" is equivalent to Fortran's intrinsic function int( ) that truncates to integer values. $N = n^2$ is the cell number in stencil and also the term number of the TPP. We define column vectors $\boldsymbol{c}(\hat{x}, \hat{y}) = \{c_k(\hat{x}, \hat{y}) | k = 1, 2, 3, \ldots, N\}$ and $\boldsymbol{a} = \{a_k | k = 1, 2, 3, \ldots, N\}$, the point value on $(\hat{x}, \hat{y})$ can be written as

$$p\left(\hat{x}, \hat{y}\right) = \boldsymbol{c}\left(\hat{x}, \hat{y}\right) \cdot \boldsymbol{a}. \tag{26}$$

The volume integration average (VIA) of prognostic field q on cell $\Omega_{i,j}$ is represented by

$$\overline{q}_{i,j} = \frac{1}{\Delta\hat{x}_{i,j}\Delta\hat{y}_{i,j}} \int\int_{\Omega_{i,j}} p\left(\hat{x}, \hat{y}\right) \mathrm{d}\hat{x}\mathrm{d}\hat{y}. \tag{27}$$

$\Delta\hat{x}_{i,j}$, $\Delta\hat{y}_{i,j}$ are length of edges $\hat{x}$, $\hat{y}$ of cell $\Omega_{i,j}$ in computational space. The VIA value $\overline{\boldsymbol{q}}_i$ on each cell is predicted by time integration, we wish to determine the coefficient vector $\boldsymbol{a}$ by these VIA values. HOPE employs an equiangular cubed-sphere grid, wherein each cell in computational space can be considered a perfectly identical square, according to Eq. (24), we may assume without loss of generality that $\Delta\hat{x}_{i,j} = \Delta\hat{y}_{i,j} = 1$, and Eq. (27) becomes

$$\overline{q}_{i,j} = \int\int_{\Omega_{i,j}} p\left(\hat{x}, \hat{y}\right) \mathrm{d}\hat{x}\mathrm{d}\hat{y} = \int\int_{\Omega_{i,j}} \boldsymbol{c} \cdot \boldsymbol{a} \mathrm{d}\hat{x}\mathrm{d}\hat{y} = \boldsymbol{\psi}_{i,j} \cdot \boldsymbol{a} \tag{28}$$

where

$$\boldsymbol{\psi}_{i,j} = \int\int_{\Omega_{i,j}} \boldsymbol{c}\mathrm{d}\hat{x}\mathrm{d}\hat{y} = \begin{pmatrix} \int\int_{\Omega_{i,j}} c_1 \mathrm{d}\hat{x}\mathrm{d}\hat{y} \\ \int\int_{\Omega_{i,j}} c_2 \mathrm{d}\hat{x}\mathrm{d}\hat{y} \\ \vdots \\ \int\int_{\Omega_{i,j}} c_N \mathrm{d}\hat{x}\mathrm{d}\hat{y} \end{pmatrix},$$

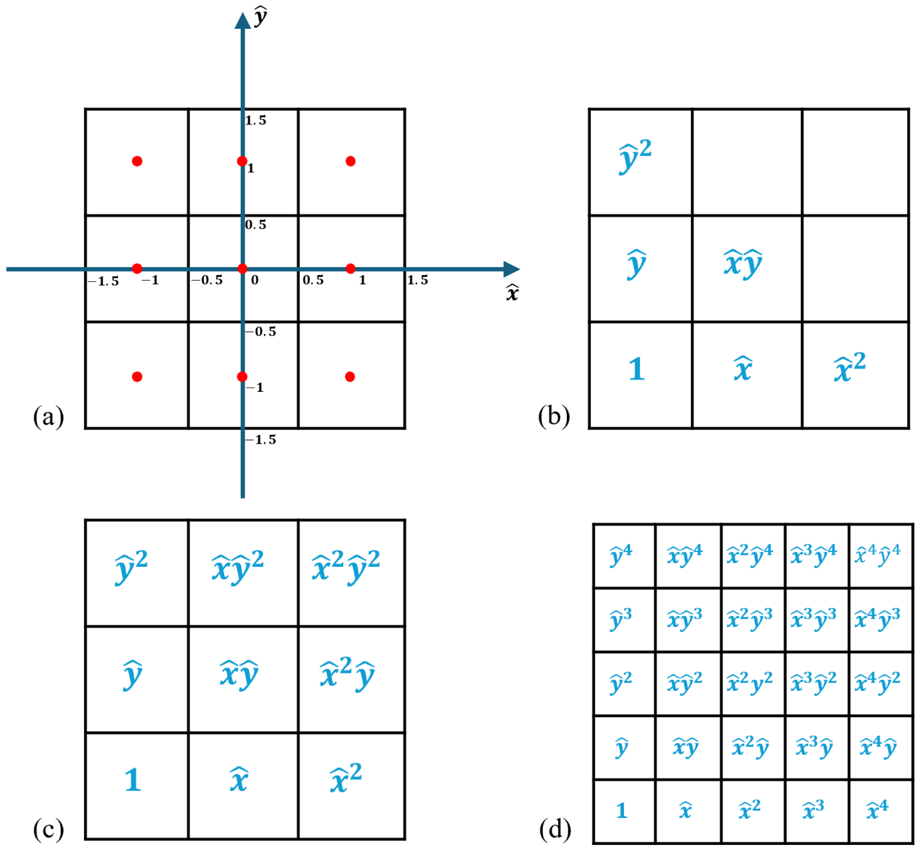
$\hat{y}$
$\hat{x}$
1.5
1
0.5
0
−0.5
−1
−1.5
(a)
(b)
(c)
(d)
$\hat{y}^2$
$\hat{y}$
$\hat{x}\hat{y}$
1
$\hat{x}$
$\hat{x}^2$
$\hat{x}\hat{y}^2$
$\hat{x}^2\hat{y}^2$
$\hat{x}^2\hat{y}$
$\hat{y}^4$ $\hat{x}\hat{y}^4$ $\hat{x}^2\hat{y}^4$ $\hat{x}^3\hat{y}^4$ $\hat{x}^4\hat{y}^4$
$\hat{y}^3$ $\hat{x}\hat{y}^3$ $\hat{x}^2\hat{y}^3$ $\hat{x}^3\hat{y}^3$ $\hat{x}^4\hat{y}^3$
$\hat{y}^2$ $\hat{x}\hat{y}^2$ $\hat{x}^2y^2$ $\hat{x}^3\hat{y}^2$ $\hat{x}^4\hat{y}^2$
$\hat{y}$ $\hat{x}\hat{y}$ $\hat{x}^2\hat{y}$ $\hat{x}^3\hat{y}$ $\hat{x}^4\hat{y}$
1 $\hat{x}$ $\hat{x}^2$ $\hat{x}^3$ $\hat{x}^4$

Figure 4. Reconstruction coordinate and polynomial terms on stencils. **(a)** Local reconstruction coordinate (the red points denote cell centers); **(b)** 2nd degree polynomial stencil; **(c)** TPP3 stencil; **(d)** TPP5 stencil.

combining N cells. We have following linear system

$$\mathbf{A}\boldsymbol{a} = \overline{\boldsymbol{q}} \tag{29}$$

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{\psi}_1^T \\ \boldsymbol{\psi}_2^T \\ \vdots \\ \boldsymbol{\psi}_N^T \end{pmatrix}, \ \overline{\boldsymbol{q}} = \begin{pmatrix} \overline{\boldsymbol{q}}_1 \\ \overline{\boldsymbol{q}}_2 \\ \vdots \\ \overline{\boldsymbol{q}}_N \end{pmatrix} \tag{30}$$

and polynomial coefficient $\boldsymbol{a}$ can be obtain by solving Eq. (29).

$$\boldsymbol{a} = \mathbf{A}^{-1}\overline{\boldsymbol{q}} \tag{31}$$

The reconstruction values on M points can be obtained by following formula

$$\boldsymbol{p} = \mathbf{C}\boldsymbol{a} = \mathbf{C}\mathbf{A}^{-1}\overline{\boldsymbol{q}} = \mathbf{R}\overline{\boldsymbol{q}} \tag{32}$$

where $\boldsymbol{p} = \begin{pmatrix} p\left(\hat{x}_1, \hat{y}_1\right) \\ p\left(\hat{x}_2, \hat{y}_2\right) \\ \vdots \\ p\left(\hat{x}_M, \hat{y}_M\right) \end{pmatrix}$, $\mathbf{C} = \begin{pmatrix} \boldsymbol{c}_1^T \\ \boldsymbol{c}_2^T \\ \vdots \\ \boldsymbol{c}_M^T \end{pmatrix}$, $\boldsymbol{c}_m^T = \boldsymbol{c}^T\left(\hat{x}_m, \hat{y}_m\right)$, $m = 1, 2, \ldots, M$, superscript T stands for transpose matrix, $(\hat{x}_m, \hat{y}_m)$ represents the mth function point on target cell. The reconstruction matrix

$$\mathbf{R} = \mathbf{C}\mathbf{A}^{-1}. \tag{33}$$

In practical implementation, the reconstruction matrix $\mathbf{R}$ needs to be computed only once during model initialization and stored in memory. Crucially, a fundamental advantage of our cubed-sphere grid dynamical core implementation lies in employing a globally shared reconstruction matrix $\mathbf{R}$. This unification signifies that a single instance of $\mathbf{R}$ applies identically to all grid cells, thereby significantly reducing memory/VRAM requirements, and enabling straightforward utilization of PyTorch's conv2d for accelerated reconstruction. For example, the TPP reconstruction procedure can be directly formulated as a two-dimensional convolutional operation using $\mathbf{R}$ as the convolution kernel.

3.2 Genuine two-dimensional WENO

Weighted Essentially Non-Oscillatory (WENO) represents an adaptive algorithm that dynamically preserves high-order approximation accuracy in smooth flow regions while automatically degenerating to robust low-order reconstruction near discontinuities for effective shock capturing. Shi

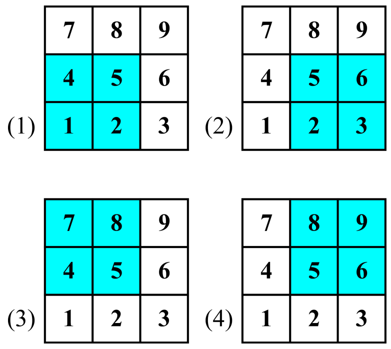

Figure 5. Stencils of 3rd order WENO 2D. The high order stencil contains cells No. 1–9, blue ones represent the cells in sub-stencils (1)–(4).

et al. (2002) mentioned two different approaches for constructing a fifth-order finite volume WENO scheme: the "Genuine 2D" method and the "Dimension by Dimension" method.

For HOPE, within the Genuine 2D framework, nth order accuracy WENO scheme employs a $n \times n$ master stencil partitioned into $\frac{(n+1)^2}{4}$ distinct $\frac{(n+1)}{2} \times \frac{(n+1)}{2}$ sub-stencils, for example:

a. WENO3: third-order reconstruction utilizes a 3×3 cell stencil that decomposes into four 2×2 sub-stencils.

b. WENO5: fifth-order accuracy employs a 5×5 master stencil partitioned into nine distinct 3×3 sub-stencils (complete schematic representations of these decomposition strategies are provided in Figs. 5 and 6).

The scheme's theoretical order of accuracy fundamentally depends on the proper determination of optimal linear weights for the multidimensional stencil combination. These weights, when correctly derived, enable the weighted superposition of sub-stencils to recover full high-order accuracy in smooth solution regions. While Shi et al. (2002) indicated the theoretical possibility of computing these weights through Lagrange interpolation basis analysis, they omitted specific implementation details. In this section, we present the methods for constructing genuine two-dimensional WENO (WENO 2D) schemes using least squares method.

We construct WENO 2D based on TPP and square stencil. As mentioned in previous section, a nth order stencil contains $N = n^2$ cells, and the full stencil (also called high-order stencil) width is $h = n$. Decomposing the high-order stencil into $s = \left(\frac{n+1}{2}\right)^2$ sub-stencils, there are $s_\mathrm{c} = s$ cells in each sub-stencil (also called low-order stencil), and the sub-stencil width is $l = \frac{n+1}{2}$. We define p_H as the high- order reconstruction polynomial, and p_i represents ith sub-stencil reconstruction polynomial, they share the same expression as Eq. (25) with different stencil width and coefficient a. For the reconstruction point $(\hat{x}, \hat{y})$, suppose

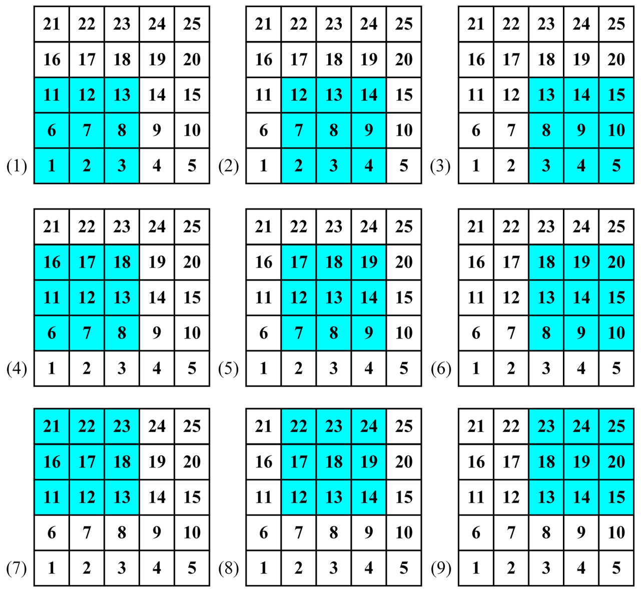

Figure 6. Stencils of 5th order WENO 2D. The high order stencil contains cells No. 1–25, blue ones represent the cells in sub-stencils (1)–(9).

$p_\mathrm{H}(\hat{x}, \hat{y})$ is the reconstruction value of high-order stencil, the reconstruction values of sub-stencils are stored in vector $\boldsymbol{p} = (p_1(\hat{x}, \hat{y}), p_2(\hat{x}, \hat{y}), \cdots, p_\mathrm{s}(\hat{x}, \hat{y}))^T$. The intention of constructing the optimal linear weights is to determine the unique weights $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \cdots, \gamma_\mathrm{s})$, such that

$$p_\mathrm{H} = \mathbf{R}_\mathrm{H}\overline{\boldsymbol{q}} = \boldsymbol{\gamma} \cdot \boldsymbol{p} \tag{34}$$

where the elements of vector $\overline{\boldsymbol{q}} = (q_1, q_2, \cdots, q_N)^T$ represent VIA of each cell in high-order stencil. $\mathbf{R}_\mathrm{H} = (r_{H_j})$, $j = 1, 2, \ldots, N$ is the reconstruction matrix of high-order stencil.

It should be noted that Eq. (34) appears overdetermined at first glance. However, subsequent analysis demonstrates that the solution obtained via the least squares method satisfies Eq. (34) exactly. Specifically, in the case of square stencils, the rank of the system defined by Eq. (34) becomes s, resulting in a unique solution for the linear system. This finding aligns with observations presented in Hu and Shu (1999) regarding their research on Triangular Meshes.

The computation of $\boldsymbol{\gamma}$ requires the integration of both high-order and low-order reconstruction matrices into a unified linear system. For each sub-stencil i we define the reconstruction matrix $\mathbf{R}_i = (r_{ik})$, $k = 1, 2, \ldots, s_\mathrm{c}$ (computed via Eq. 33). and $\mathbf{R}_{\mathrm{L}_i} = (r_{\mathrm{L}_{ij}})$, $j = 1, 2, \ldots, N$ is the extension matrix of $\mathbf{R}_i$. The matrix relationship is expressed as

$$(\mathbf{R}_i)_{1\times s_\mathrm{c}}(\mathbf{E})_{s_\mathrm{c}\times N} = \left(\mathbf{R}_{\mathrm{L}_i}\right)_{1\times N} \tag{35}$$

where the subscripts denote matrix dimensions. The correspondence matrix $\mathbf{E} = (e_{ij})$, $i = 1, 2, \ldots, s_\mathrm{c}$; $j = 1, 2, \ldots, N$ encodes the cell relationships between stencils:

when the ith cell in low-order stencil is the same as the jth cell in high order stencil, $e_{ij} = 1$, otherwise, $e_{ij} = 0$.

Substitute Eq. (32) into Eq. (34), yield

$$\mathbf{R}_{\mathrm{H}}\overline{q} = \sum_{i=1}^{s} \mathbf{R}_{\mathrm{L}_i} \gamma_i \overline{q}. \tag{36}$$

We set $\mathbf{R}_{\mathrm{L}} = (\mathbf{R}_{\mathrm{L}_1}, \mathbf{R}_{\mathrm{L}_2}, \ldots, \mathbf{R}_{\mathrm{L}_s})^T$, Eq. (36) becomes

$$\mathbf{R}_{\mathrm{L}}\boldsymbol{\gamma} = \mathbf{R}_{\mathrm{H}}. \tag{37}$$

The unknown optimal weights vector $\boldsymbol{\gamma}$ can be determined by following least square procedure

$$\boldsymbol{\gamma} = \left(\mathbf{R}_{\mathrm{L}}^T \mathbf{R}_{\mathrm{L}}\right)^{-1} \mathbf{R}_{\mathrm{L}}^T \mathbf{R}_{\mathrm{H}}. \tag{38}$$

However, the elements of $\boldsymbol{\gamma}$ could be negative, which would cause unstable. A split technique mentioned by Shi et al. (2002) was adopted to solve this problem. The optimal weights can be split into two parts:

$$\tilde{\gamma}^+ = \frac{\theta|\boldsymbol{\gamma}| + \boldsymbol{\gamma}}{2}, \ \tilde{\gamma}^- = \boldsymbol{\gamma}^+ - \boldsymbol{\gamma} \tag{39}$$

where the constant $\theta = 3$. For keeping the sum of weights to 1, $\tilde{\gamma}^\pm$ and new value of $\boldsymbol{\gamma}^\pm$ can be rescaled as:

$$\sigma^\pm = \sum_{i=1}^{s} \tilde{\gamma}_i^\pm \tag{40}$$

and

$$\gamma_i^\pm = \frac{\tilde{\gamma}_i^\pm}{\sigma^\pm} \ i = 1, 2, \ldots, s \tag{41}$$

where $\tilde{\gamma}_i^\pm$ is the ith element of $\tilde{\gamma}^\pm$, $\gamma_i^\pm$ is the ith element of $\boldsymbol{\gamma}^\pm$.

The WENO scheme adaptively assigns nonlinear weights ω_i, $(i = 1, 2, \ldots, s)$ to each candidate stencil to suppress unphysical oscillations during high-order reconstruction. Essentially, it gives greater weight to stencils identified as smooth over the local cell, while suppressing the influence of those containing discontinuities by assigning them smaller weights. Several nonlinear weighting schemes have been developed to meet these criteria, including WENO-JS (Jiang and Shu, 1996), WENO-Z (Borges et al., 2008), WENO-Z+ (Acker et al., 2016), WENO-Z+M (Luo and Wu, 2021), among others.

In this work, we employ the WENO-Z formulation as our baseline scheme. While most existing WENO schemes were originally developed for one-dimensional problems, we propose a two-dimensional extension of WENO-Z through modification of τ, a crucial coefficient that governs the scheme's higher-order accuracy properties.

For stencil i the nonlinear weight is given as

$$\omega_i^\pm = \frac{\alpha_i^\pm}{\sum_{i=1}^{s} \alpha_i^\pm} \tag{42}$$

$$\alpha_i^\pm = \gamma_i^\pm \left[1 + \left(\frac{\tau}{\beta_i + \varepsilon}\right)^2\right] \tag{43}$$

$$\tau = \frac{2}{(s-1)s} \sum_{\eta=1}^{s-1} \sum_{\psi=\eta+1}^{s} \left|\beta_\psi - \beta_\eta\right| \tag{44}$$

where $\varepsilon = 10^{-14}$ is introduced to prevent division by zero. The smooth indicators β_i quantify the smoothness of reconstruction functions within the target cell. We employ a formulation analogous to that described in Zhu and Shu (2019).

As mentioned in Eq. (24), all cells participating in reconstruction within HOPE's computational space can be treated as identical unit squares with $\Delta\hat{x} = \Delta\hat{y} = 1$. Thus, the smooth indicator for sub-stencil i is expressed as:

$$\beta_i = \sum_{\zeta=1}^{l} \int\int_{\Omega} \left[\frac{\partial^\zeta}{\partial \hat{x}^{\zeta_1} \partial \hat{y}^{\zeta_2}} p_i\left(\hat{x}, \hat{y}\right)\right]^2 \mathrm{d}\hat{x}\mathrm{d}\hat{y} \tag{45}$$

where $\zeta_1 + \zeta_2 = \zeta$ and $\zeta > 0$, $\zeta_1, \zeta_2 \in [0, n]$, and l is the sub-stencil width.

The reconstruction value on point $(\hat{x}, \hat{y})$ is expressed by:

$$q\left(\hat{x}, \hat{y}\right) = \sum_{i=1}^{s} \left(\sigma^+ \omega_i^+ - \sigma^- \omega_i^-\right) p_i\left(\hat{x}, \hat{y}\right). \tag{46}$$

3.3 Treatment of the panel boundaries

The cubed sphere grid comprises 12 panel boundaries, and the flux across the interface between any two panels must be computed at the quadrature points situated on the edges of the boundary cells, as depicted in Fig. 7a. However, a challenge arises because the coordinates across these panel boundaries are discontinuous. Given that the TPP reconstruction necessitates a square stencil, the values of the cells outside the domain (referred to as ghost cells) must be computed through interpolation within the adjacent panel, as illustrated in Fig. 7b. While Ullrich et al. (2010) proposed a one-sided interpolation scheme, our testing with the HOPE model revealed that using a similar one-sided ghost cell interpolation approach around panel boundaries resulted in instability when scheme exceeded 7th order of accuracy. To address this limitation, we redesigned the ghost cell interpolation scheme to incorporate information from both panels adjacent to the boundary. This modified approach ensures stable integration even for very high-order schemes, as validated in tests up to 13th-order accuracy.

3.3.1 Ghost cell interpolation

To achieve arbitrary high-order accuracy, we propose a ghost cell interpolation scheme that incorporates information from both sides of the panel boundary. Since the ghost cell values are inherently unknown prior to interpolation, our approach involves an initial estimation through an iterative process.

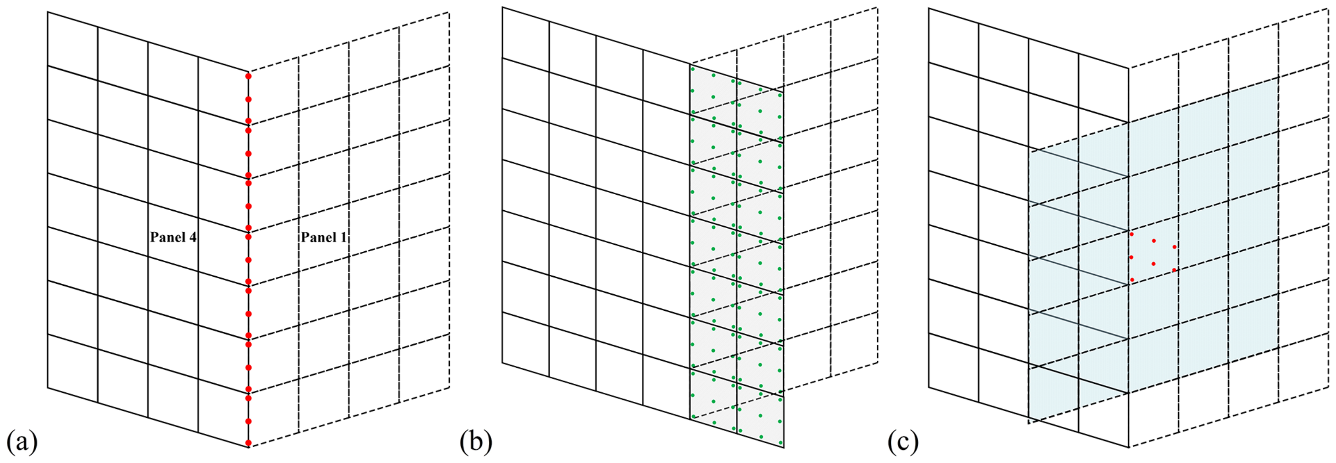

Figure 7. Points and cells close to panel boundary. **(a)** Flux points (red points) on the interface between Panel 1 and Panel 4, the flux across each panel at these points are determined by Riemann solver, which merges the reconstruction outcomes from both panels into a single flux value; **(b)** ghost cells (shaded cells) out of Panel 4 boundary, with green points representing the GQPs in these cells; **(c)** cells requirement for 5th order ghost cell interpolation stencil, red points represent the GQPs located in the ghost cell on Panel 4, the blue shaded region represents the TPP reconstruction stencil (on Panel 1) to interpolate these red GQPs.

Specifically, the method iteratively performs ghost cell interpolation until the increments of the cell values converge to within a specified tolerance.

Through mathematical analysis (detailed in the Appendix), we demonstrate that this iterative process can be expressed as a linear mapping, thereby eliminating the need for actual iterations. However, direct computation of the mapping matrix requires inversion of a large matrix, which poses significant computational and memory challenges. To address this, we implement the iterative interpolation process using PyTorch and leverage its automatic differentiation capability to efficiently obtain the interpolation matrix.

The complete methodology, as derived in the Appendix, proceeds as follows:

1. Initialization: all ghost cell values are initialized to zero (denoted as $\boldsymbol{g}^{(0)} = 0$, where the superscript indicates the iteration number).

2. Interpolation: the Gaussian quadrature points (GQPs) in the ghost cells are interpolated using the Tensor Product Polynomial (TPP) stencil. For instance, considering two adjacent panels (Fig. 7a), any out-domain cell in Panel 4 (shaded cell in Fig. 7b) contains GQPs that physically reside in Panel 1. These GQPs are interpolated using the TPP stencil shown in Fig. 7c, which incorporates relevant ghost cells from Panel 1.

3. Update and convergence check: after interpolating all GQPs, the ghost cell values are updated via Gaussian quadrature (Eq. 22), yielding $\boldsymbol{g}^{(1)}$. The L_2-norm residual $r^{(k)} = \left\|\boldsymbol{g}^{(k+1)} - \boldsymbol{g}^{(k)}\right\|_2$ is then computed. Steps 2–3 repeat until $r^{(k)} < \epsilon$, where $\epsilon = 1 \times 10^{-14}$ for double precision and $\epsilon = 1 \times 10^{-5}$ for single precision. In practice, convergence typically occurs within 10 iterations, so we fix the iteration count at 10 for consistency.

This process establishes a linear mapping $\mathcal{G} : \boldsymbol{q} \rightarrow \boldsymbol{g}$ from known cell values to ghost cell values. As proven in Eq (A12), the mapping's linearity implies that $\mathcal{G} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}}$ forms a matrix, which we efficiently compute using PyTorch's autograd functionality. This approach avoids explicit matrix inversion while maintaining numerical precision.

It is important to note that overlapping GQPs occur at the corner positions of the cubed-sphere grid, as illustrated by the magenta points in Fig. 2b. These points lie on the interface shared by adjacent panels (e.g., Panel 1 and Panel 5). Consequently, during ghost value interpolation, two distinct interpolated values are obtained at these overlapping points – one from each adjoining panel. The final interpolated value is computed as the average of these two values. Since the interpolation performed on each individual panel is high-order, the approximation order is preserved when taking this average.

$\mathcal{G}$ is a sparse matrix containing many zero entries. To avoid unnecessary memory costs, we adopt the Compressed Sparse Row (CSR) format for storing $\mathcal{G}$. Furthermore, the size of $\mathcal{G}$ is extremely large-making direct application of "torch.autograd.functional.jacobian" to generate $\mathcal{G}$ computationally infeasible. Our implementation for generating ghost cell interpolation matrix achieves significant acceleration and substantially reduces VRAM demand compared to PyTorch's native "torch.autograd.functional.jacobian" function. The key optimizations are:

1. Parallel Multi-Row Computation: utilizing "torch.vmap" to encapsulate "torch.func.vjp", en-

abling simultaneous computation of multiple matrix rows.

2. CSR Compression and Incremental Disk Storage:
   a. Employing Compressed Sparse Row (CSR) format for matrix representation.
   b. Implementing incremental disk storage, where computed data batches are immediately written to disk after processing, avoiding prolonged VRAM retention.
3. Tunable Batch Processing: adjusting the number of rows processed per iteration maximizes GPU utilization while respecting VRAM constraints (e.g., 24 GB on NVIDIA RTX 3090).

It should be note that the model grid does not change during simulation, the ghost interpolation matrix $\mathcal{G}$ needs to be calculated only once in initialization progress.

3.3.2 Fields conversion between panels

Due to the differing coordinate systems across panels, field variables must be appropriately transformed when transferring information between adjacent panels. To illustrate this process, we consider the interface between Panel 1 and Panel 4, as depicted in Figs. 2a and 7a. Although flux points are shared between the two panels, their coordinate representations are discontinuous across the interface.

To ensure consistency, two key transformations are required:

1. Metric reset for mass variables: the mass-related prognostic quantities must be recomputed in the target panel's coordinate system to maintain metric consistency.
2. Wind vector transformation: velocity components (or other vector quantities) must be converted from the source panel's local coordinate frame to that of the target panel.

This coordinate conversion ensures proper continuity and physical consistency when interpolating or exchanging data across panel boundaries.

Suppose $\boldsymbol{q}_1 = [(\sqrt{G}\phi)_1, (\sqrt{G}\phi u)_1, (\sqrt{G}\phi v)_1]^T$ and $\boldsymbol{q}_4 = [(\sqrt{G}\phi)_4, (\sqrt{G}\phi u)_4, (\sqrt{G}\phi v)_4]^T$ represent the fields on panel 1 and 4. The mass field conversion from panel 4 to panel 1 is expressed by

$$(\sqrt{G}\phi)_4^1 = \frac{\sqrt{G}_4}{\sqrt{G}_1}(\sqrt{G}\phi)_1 \tag{47}$$

the subscript represents the target panel and the superscript stands for source panel.

The transformation of momentum vectors between panels is performed in two sequential steps to maintain proper tensor consistency. The contravariant momentum components in Panel 1 are first projected onto the global spherical coordinate system using the transformation matrix $\mathbf{J}$, as defined in Eq. (10). The resulting spherical momentum components are then transformed into the contravariant representation specific to Panel 4, ensuring compatibility with the target panel's local coordinate system.

$$\begin{bmatrix} \left(\sqrt{G}\phi u_s\right)_1 \\ \left(\sqrt{G}\phi v_s\right)_1 \end{bmatrix} = \mathbf{J}_1 \begin{bmatrix} (\sqrt{G}\phi u)_1 \\ (\sqrt{G}\phi v)_1 \end{bmatrix} \tag{48}$$

$$\begin{bmatrix} (\sqrt{G}\phi u)_4 \\ (\sqrt{G}\phi v)_4 \end{bmatrix} = \mathbf{J}_4^{-1}\frac{\sqrt{G}_4}{\sqrt{G}_1} \begin{bmatrix} \left(\sqrt{G}\phi u_s\right)_1 \\ \left(\sqrt{G}\phi v_s\right)_1 \end{bmatrix} \tag{49}$$

where $\mathbf{J}_1$ is the $\mathbf{J}$ matrix on panel 1, $\mathbf{J}_4^{-1}$ is the inverse matrix of $\mathbf{J}$ on panel 4. (u_s, v_s) are zonal wind and meridional wind, (u, v) are contravariant wind components. Since the vector conversion is linear process, Eqs. (48) and (49) can be merged into following equation

$$\begin{bmatrix} (\sqrt{G}\phi u)_4 \\ (\sqrt{G}\phi v)_4 \end{bmatrix} = \mathbf{C}_{1,4} \begin{bmatrix} (\sqrt{G}\phi u)_1 \\ (\sqrt{G}\phi v)_1 \end{bmatrix} \tag{50}$$

where matrix $\mathbf{C}_{1,4} = \frac{\sqrt{G}_4}{\sqrt{G}_1}\mathbf{J}_4^{-1}\mathbf{J}_1$, the subscript stands for converting vector form panel 1 to panel 4.

The mass and vector are also need to be converted on GQPs in the same manner.

3.4 Riemann solver

Following spatial reconstruction, discontinuous solutions arise on either side of each flux point location. Since the majority of atmospheric flow speeds correspond to small Mach numbers, we adopt the Low Mach number Approximate Riemann Solver (Chen et al., 2013) and AUSM$^+$-up (Liou, 2006; Ullrich et al., 2010) as Riemann solvers to determine the flux at the edge quadrature points (EQPs).

3.4.1 Low Mach number approximate Riemann solver (LMARS)

Spatial reconstruction gives the left and right state vector,

$$\boldsymbol{q}_L = \begin{bmatrix} (\sqrt{G}\phi)_L \\ (\sqrt{G}\phi u)_L \\ (\sqrt{G}\phi v)_L \end{bmatrix}, \boldsymbol{q}_R = \begin{bmatrix} (\sqrt{G}\phi)_R \\ (\sqrt{G}\phi u)_R \\ (\sqrt{G}\phi v)_R \end{bmatrix}. \tag{51}$$

First of all, we convert the contravariant wind u to physical speed $u^\perp$ that is perpendicular to the cell edge.

$$u^\perp = \frac{u}{\sqrt{G^{ii}}}, \ i = \begin{cases} 1, & x \text{ direction} \\ 2, & y \text{ direction} \end{cases} \tag{52}$$

For example, in x direction, $u^\perp = \frac{u}{\sqrt{G^{11}}}$, and there is no summation over i in Eq. (52).

The wind speed m^* and geopotential ϕ are calculated by

$$m^* = \frac{1}{2}\left(u_{\mathrm{L}}^{\perp} + u_{\mathrm{R}}^{\perp} - \frac{\phi_{\mathrm{R}} - \phi_{\mathrm{L}}}{c}\right) \tag{53}$$

$$\phi = \frac{1}{2}\left[\phi_{\mathrm{L}} + \phi_{\mathrm{R}} - c\left(u_{\mathrm{R}}^{\perp} - u_{\mathrm{L}}^{\perp}\right)\right] \tag{54}$$

$$c = \frac{c_{\mathrm{L}} + c_{\mathrm{R}}}{2} \tag{55}$$

$$c_{\mathrm{L}} = \sqrt{\phi_{\mathrm{L}}},\ c_{\mathrm{R}} = \sqrt{\phi_{\mathrm{R}}}. \tag{56}$$

c is the phase speed of external gravity wave, the subscript "L", "R" represent the left and right side of cell edge.

Once m^* is determined, we convert it back to contravariant speed by

$$m = m^*\sqrt{G^{ii}}. \tag{57}$$

We define the pressure-driven flux as

$$P = \frac{1}{2}\sqrt{G}\phi_t^2. \tag{58}$$

The flux across the cell edge is then given by

$$\boldsymbol{F} = \frac{1}{2}m\left[\left(\boldsymbol{q}_{\mathrm{L}} + \boldsymbol{q}_{\mathrm{R}}\right) - \operatorname{sign}(m)\left(\boldsymbol{q}_{\mathrm{R}} - \boldsymbol{q}_{\mathrm{L}}\right)\right] + \boldsymbol{P} \tag{59}$$

$$\boldsymbol{P} = \begin{pmatrix} 0 \\ G^{1i}P \\ G^{2i}P \end{pmatrix},\ i = \begin{cases} 1, & x \text{ direction} \\ 2, & y \text{ direction} \end{cases}. \tag{60}$$

For calculation of $\boldsymbol{G}$ (the flux in y direction) the method is similar.

3.4.2 Advection upstream splitting method for all speeds (AUSM$^+$-up)

The differences between AUSM$^+$-up and LMARS lie in the method of determining the wind speed m^* and pressure-driven flux P. In AUSM$^+$-up

$$m^* = cM \tag{61}$$

where c denotes the gravity phase speed defined in Eq. (55). Mach number M is expressed as

$$M = M_{(4)}^{+}(M_{\mathrm{L}}) + M_{(4)}^{-}(M_{\mathrm{R}}) - K_p \max\left(1 - \sigma\overline{M}^2, 0\right) \frac{P_{\mathrm{R}} - P_{\mathrm{L}}}{c^2\phi} \tag{62}$$

where $M_{\mathrm{L}} = \frac{u_L^{\perp}}{c}$, $M_{\mathrm{R}} = \frac{u_R^{\perp}}{c}$, $\overline{M}^2 = \frac{(u_{\mathrm{L}}^{\perp})^2 + (u_{\mathrm{R}}^{\perp})^2}{2c^2}$, and

$$M_{(4)}^{\pm}(M) = \begin{cases} \frac{1}{2}(M \pm |M|), & |M| \geq 1 \\ M_{(2)}^{\pm}(M)\left[1 \mp 16\beta M_{(2)}^{\mp}(M)\right], & |M| < 1 \end{cases} \tag{63}$$

$$M_{(2)}^{\pm}(M) = \pm\frac{1}{4}(M \pm 1)^2. \tag{64}$$

The pressure-driven flux is expressed as

$$P = P_{(5)}^{+}(M_{\mathrm{L}})P_{\mathrm{L}} + P_{(5)}^{-}(M_{\mathrm{R}})P_{\mathrm{R}} + -2K_u P_{(5)}^{+}(M_{\mathrm{L}}) P_{(5)}^{-}(M_{\mathrm{R}})\phi c\left(u_{\mathrm{R}}^{\perp} - u_{\mathrm{L}}^{\perp}\right) \tag{65}$$

where $P_{\mathrm{L}} = \frac{1}{2}\phi_{\mathrm{L}}^2$, $P_{\mathrm{R}} = \frac{1}{2}\phi_{\mathrm{R}}^2$, and

$$P_{(5)}^{\pm} = \begin{cases} \frac{1}{2}(1 \pm \operatorname{sign}(M)), & |M| \geq 1 \\ M_{(2)}^{\pm}(M)\left[(\pm 2 - M) \mp 16\alpha M M_{(2)}^{\mp}(M)\right], & |M| < 1 \end{cases} \tag{66}$$

The mathematical meaning of sign(M) (returning -1, 0, or 1 based on the sign of M) is standard. The coefficients take the values: $\sigma = 1$, $\alpha = \frac{3}{16}$, $\beta = \frac{1}{8}$, $K_p = \frac{1}{4}$, $K_u = \frac{3}{4}$.

Once m^* and P are computed, the flux across the cell edge can be calculated using Eqs. (57) to (60).

3.5 Temporal integration

We use the explicit Runge–Kutta (RK) as time marching scheme, Wicker and Skamarock (2002) described a 3rd order RK with three stages (achieves third-order accuracy exclusively when applied to linear problems). For the prognostic fields $\boldsymbol{q}$, the integration step from time slot n to $n+1$:

$$\boldsymbol{q}^* = \boldsymbol{q}^n + \frac{\Delta t}{3}\left(\frac{\partial \boldsymbol{q}^n}{\partial t}\right) \tag{67}$$

$$\boldsymbol{q}^{**} = \boldsymbol{q}^* + \frac{\Delta t}{2}\left(\frac{\partial \boldsymbol{q}^*}{\partial t}\right) \tag{68}$$

$$\boldsymbol{q}^{n+1} = \boldsymbol{q}^n + \Delta t\left(\frac{\partial \boldsymbol{q}^{**}}{\partial t}\right) \tag{69}$$

where Δt is the time step, and temporal tendency terms $\frac{\partial \mathbf{q}}{\partial t}$ can be obtain by Eqs. (15) and (16). In our numerical experiments, the choice of different time marching schemes influenced only the integration stability; it had no significant impact on the simulation norm errors, non-oscillatory property, or conservation property.

4 High performance implementation and automatic differentiation

The spatial operator and temporal integration of HOPE can be easily implemented using PyTorch. Specifically, the spatial reconstruction given by Eq. (32) is implemented as a convolution operation, while the Gaussian quadrature can be efficiently expressed as a matrix-vector multiplication. Leveraging PyTorch's highly optimized built-in functions for both convolution and quadrature operations ensures superior performance on GPUs.

Furthermore, PyTorch's role as a versatile AI development platform provides automatic differentiation capabilities across the entire computation graph. This streamlines implementation and enables efficient gradient computation for all components.

For the reconstruction implementation. Suppose the cubed sphere grid comprises n_c cells in x-direction within each panel, including ghost cells. The panel number is n_p, and the shallow water equation involves n_v prognostic variables per cell, we write the cell state tensor $\boldsymbol{q}$ with the shape (n_v, n_p, 1, n_c, n_c). The TPP reconstruction weight tensor $\mathbf{R}$ has shape (n_poc, 1, s_w, s_w), where n_poc is the number of points required to be interpolated within each cell (including EQP and CQP), s_w denotes the stencil width (same as the stencil width represented by n in Sec. 3.1). The reconstruction can be executed by a simple command (pseudo-code):

$$\begin{aligned}\boldsymbol{q}_\mathrm{rec} =&\text{torch.nn.functional.conv2d}\left(\boldsymbol{q}\cdot\text{view}\left(n_\mathrm{v}n_\mathrm{p},1,n_\mathrm{c},n_\mathrm{c}\right),\right.\\ &\left.\mathbf{R}\right)\text{view}\left(n_\mathrm{v},n_\mathrm{p},n_\mathrm{poc},n_\mathrm{c},n_\mathrm{c}\right)\end{aligned} \tag{70}$$

where the shape of $\boldsymbol{q}_\mathrm{rec}$ is (n_v, n_p, n_poc, n_c, n_c).

We exclusively demonstrate the flux computation procedure at cell edges as an illustrative example, where Gaussian quadrature is employed to obtain edge-averaged fluxes. The analogous procedure applies to source term integration at CQPs. The edge state tensor $\boldsymbol{q}_\mathrm{e}$, corresponding to the EQPs along each cell edge, is subsequently expressed as:

$$\boldsymbol{q}_\mathrm{e}=\boldsymbol{q}_\mathrm{rec}(\ldots,\,\text{pes}:\text{pee},\,:,\,:) \tag{71}$$

where subscript "e" represents edges on cell including L(left), R(right), B(bottom), T(top). "pes, pee" are start and end point indices on edge e. The shape of $\boldsymbol{q}_\mathrm{e}$ (including $\boldsymbol{q}_\mathrm{L}$, $\boldsymbol{q}_\mathrm{R}$, $\boldsymbol{q}_\mathrm{B}$, $\boldsymbol{q}_\mathrm{T}$) is ($n_\mathrm{v}$, n_p, n_poe, n_c, n_c). n_poe denotes the number of edge quadrature points (EQPs). This value is computed as $n_\mathrm{poe}=\text{pee}-\text{pes}$ in PyTorch implementations, whereas in Fortran it is calculated as $n_\mathrm{poe}=\text{pee}-\text{pes}+1$, reflecting the difference in array indexing conventions between the two languages.

After spatial reconstruction, the resulting data is utilized in the Riemann solver for EQPs and for source term computation on CQPs. Subsequently, integration is performed on both EQPs and CQPs to calculate the net flux and the cell-averaged source term tendency. The cell-edge flux tensor $\boldsymbol{F}$ with dimensions (n_v, n_p, n_poe, n_c, n_c) is obtained after the Riemann solver.

There is a dimensionality mismatch between the flux tensor and weight tensor during using matrix multiplication. For the Gaussian quadrature implementation, consider an edge Gaussian quadrature weight tensor $\mathbf{g}_\mathrm{e}$ with shape (n_poe), if an edge flux tensor $\tilde{\boldsymbol{F}}$ has shape (n_v, n_p, n_c, n_c, n_poe), the Gaussian quadrature can be expressed by:

$$\boldsymbol{F}_g=\text{torch.matmul}\left(\tilde{\boldsymbol{F}},\mathbf{g}_\mathrm{e}\right) \tag{72}$$

where the shape of $\boldsymbol{F}_g(n_\mathrm{v},n_\mathrm{p},n_\mathrm{c},n_\mathrm{c})$ is the average flux on edge. In this operation, n_poe must occupy the last dimension of $\tilde{\boldsymbol{F}}$, to permit "torch.matmul" execution. We note, however, that in the flux tensor $\boldsymbol{F}$ obtained from the Riemann solver, n_poe corresponds to the third dimension, direct matrix multiplication is therefore not feasible.

To address this dimensionality issue, two methods are available. The first method involves rearranging the n_poc dimension to the last position using the "torch.tensor.permute" operation in PyTorch, this allows Gaussian integrations to be naturally implemented through the "torch.matmul" operation. The second method, which avoids the need for the "permute" operation, maintains the original dimension sequence. Instead, Gaussian integrations are performed using the "torch.einsum" function. This function sums the product of the elements of the input arrays along dimensions specified using a notation based on the Einstein summation convention.

$$\boldsymbol{F}_g=\text{torch.einsum}\left('vnpij,p\rightarrow vnij',\boldsymbol{F},\mathbf{g}_\mathrm{e}\right) \tag{73}$$

We have conducted performance tests comparing the two methods, and the results indicate that the second method is approximately 5% faster than the first. Specifically, the first method took 649 seconds, while the second method took 616 s. The test was set as a one-day steady state geostrophic flow (with details provided in Sect. 5.2) simulation at a resolution of C900 ($\Delta x=\Delta y=0.1°$), using 3rd order accuracy reconstruction stencil. The time step was 30 s, and the default data type used was "torch.float32" (single precision).

The Riemann solver implementation on flux points is way easier, only needs to call "torch.sign" for Eq. (59), while all other operations can be executed using basic arithmetic: addition, subtraction, multiplication, and division. During a Runge–Kutta sub-step, there are no dependencies, and neither "for" loops nor "if" statements are required in the HOPE kernel code. This algorithm fully leverages the capabilities of the GPU.

5 Numerical experiments

The HOPE dynamical core is evaluated using the standard test cases (Test 1, 2, 5, and 6) for the spherical shallow water model as described in Williamson et al. (1992), along with the perturbed jet flow case proposed by Galewsky et al. (2004). Additionally, a dam-break experiment is designed to demonstrate the HOPE model's capability in capturing shock waves.

In our experiments, the grid resolutions are denoted by the count of cells along one dimension on each panel of the cubed sphere; for instance, C90 signifies that each panel is subdivided into a 90×90 grid, corresponding to a grid interval of $\Delta x=\Delta y=1°$.

We measure the conservation errors by defining the normalized error ϵ_r of the variable η as $\epsilon_r=\frac{I_g(\eta^n)-I_g(\eta^0)}{I_g(\eta^0)}$, where η^0 and η^n stand for η value at initial time and time slot n, respectively. The global integral is defined as:

$$I(\eta) = \sum_{p=1}^{n_p} \sum_{j=1}^{n_c} \sum_{i=1}^{n_c} \sqrt{G}_{i,j,p} \eta_{i,j,p} \tag{74}$$

where $\eta_{i,j,p}$ represents the average value of η in cell (i, j, p).

We use three kinds of norm errors to measure the simulation errors,

$$L_1 = \frac{I\left[\phi(i,j,p) - \phi_{\text{ref}}(i,j,p)\right]}{I\left[\phi_{\text{ref}}(i,j,p)\right]} \tag{75}$$

$$L_2 = \sqrt{\frac{I\left[(\phi(i,j,p) - \phi_{\text{ref}}(i,j,p))^2\right]}{I\left[\phi_{\text{ref}}^2(x,y,p)\right]}} \tag{76}$$

$$L_\infty = \frac{\max|\phi(i,j,p) - \phi_{\text{ref}}(i,j,p)|}{\max|\phi_{\text{ref}}(i,j,p)|} \tag{77}$$

the subscript "ref" represents reference state.

5.1 Cosine bell advection

The Solid Body Rotation Cosine Bell (Case 1 from Williamson et al., 1992) is commonly employed to assess noise generated by panel boundaries, as noted by Chen and Xiao (2008) and Ullrich et al. (2010). The wind field is given by

$$u_s = u_0(\cos\theta\cos\alpha + \cos\lambda\sin\theta\sin\alpha) \tag{78}$$

$$v_s = -u_0 \sin\lambda\sin\alpha \tag{79}$$

where u_s, v_s are zonal wind and meridional wind, earth radius is $a = 6\,371\,220$ m, basic flow speed $u_0 = \frac{2\pi a}{12 \cdot 86\,400}$ m s^{-1}. The initial height is defined as

$$h(\lambda,\theta) = \begin{cases} h_0\left(1 + \cos\frac{\pi d_s}{R}\right), & r < R \\ 0, & r \geq R \end{cases} \tag{80}$$

where λ, θ are longitude and latitude. The basic height $h_0 = 1000$ m. The great circle distance between (λ, θ) and the initial center point of cosine bell (λ_c, θ_c) = $(3\pi/2, 0)$ is expressed by $d_s = a \cdot \arccos[\sin\theta_c \sin\theta + \cos\theta_c \cos\theta \cos(\lambda - \lambda_c)]$. The radius $R = a/3$.

Figure 8 presents the norm errors for a 12 d simulation at $\alpha = 0$; results for $\alpha = \pi/2$ are identical. The temporal evolution of L_1 and L_2 norm errors does not exhibit a pronounced signature attributable to panel boundaries. In contrast, the L_∞ norm error evolution shows significant sensitivity to panel boundaries, varying considerably with grid resolution and reconstruction order. When using low resolution and low reconstruction order (TPP3 with C30 grid), oscillations induced by panel boundaries are relatively weak. However, as the model resolution or reconstruction order increases, the influence of panel boundaries on the L_∞ norm error manifests as a distinct four-peak pattern, corresponding to the four longitudinally aligned panel boundaries of the cubed-sphere grid.

Figure 9 shows the 12 d simulation norm errors for $\alpha = \pi/4$. In this test configuration, the cosine bell initially moves alone the interface between Panel 1 and Panel 5, and subsequently moves along the interface between Panel 3 and Panel 6. The temporal evolution of L_1 and L_2 norm errors display two gentle peaks, corresponding to the errors generated as the cosine bell crosses these panel interfaces. Similar to Fig. 8, the L_∞ norm error progressively exceeds the L_1 and L_2 norm errors as grid resolution and reconstruction order increase.

Because the Cosine Bell field lacks infinite continuity, the convergence rate of the norm errors cannot exceed second order in our tests, regardless of the reconstruction order employed. This observation aligns with the key point emphasized in our paper: high-order numerical methods achieve their design accuracy only when the flow field is sufficiently smooth. Discontinuities in the flow field violate the fundamental premise of polynomial reconstruction (as discontinuities impair the continuity of higher derivatives, leading to non-convergence of the Taylor series). This inherent sensitivity to smoothness is precisely the factor causing norm errors to be influenced by cubed-sphere panel boundaries. When using low-order reconstruction schemes at low resolutions, the Tensor Product Polynomial (TPP) reconstruction employs lower-degree polynomials and is consequently less sensitive to the smoothness of the flow field. Conversely, high-order TPP reconstruction requires the flow field to possess higher-order continuity to maintain accuracy; it is thus more sensitive to discontinuities. Insufficiently smooth flow fields can introduce numerical oscillations with high-order schemes. Therefore, while TPP5 and TPP7 yield lower L_∞ norm error magnitudes than TPP3, they exhibit more pronounced oscillations caused by the cubed-sphere panel boundaries.

5.2 Steady state geostrophic flow

Steady state geostrophic flow is the 2nd case in Williamson et al. (1992), it provided an analytical solution for spherical shallow water equations, it was widely used in accuracy test for shallow water models. The analytical solution is a steady state, which means the initial filed is the exact solution. The initial wind field replicates the formulation given in Eqs. (78) and (79), while the initial geopotential is expressed as

$$\phi = \phi_0 - \left(a\Omega u_0 + \frac{u_0^2}{2}\right)(-\cos\lambda\cos\theta\sin\alpha + \sin\theta\cos\alpha)^2 \tag{81}$$

where $\Omega = 7.292 \times 10^{-5}$ s^{-1} is the earth rotation angular velocity, basic geopotential $\phi_0 = 29\,400$ m^2 s^{-2}, $\alpha = 0$ denotes the rotation angle transcribed between the physical north pole and the center of the northern panel on the cubed-sphere grid, and gravity acceleration $g = 9.80616$ m s^{-2}. The conversion

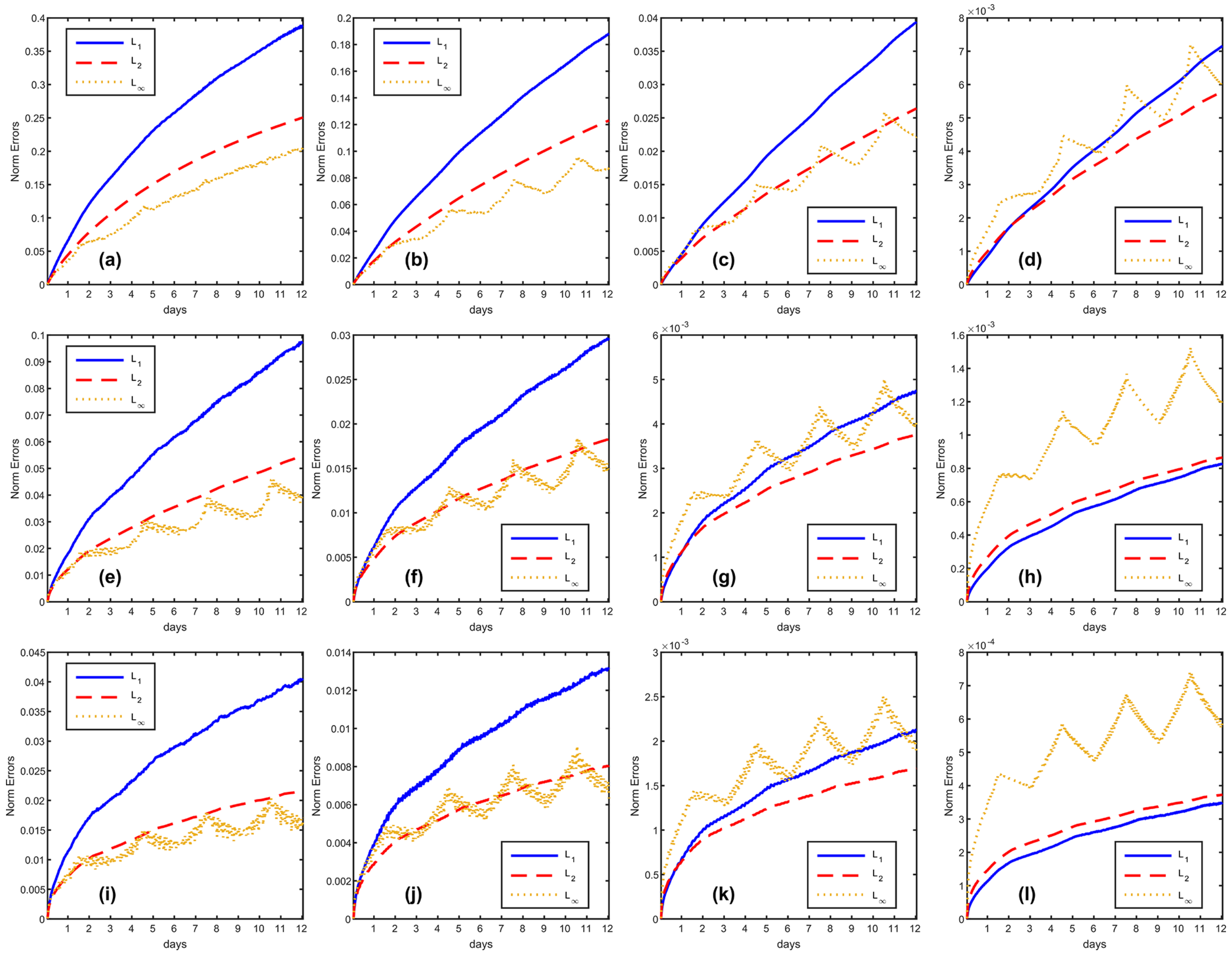

Figure 8. The variation of norm errors during simulation days for the cosine bell advection test case, with direction parameter $\alpha = 0$. The rows represent reconstruction schemes TPP3, TPP5 and TPP7, the columns stand for grid C30, C45, C90 and C180.

between the spherical wind (u_s, v_s) and contravariant wind is given by Eq. (9).

We simulated the steady state geostrophic flow over one period (12 d) to test the norm errors and corresponds convergence rate. Since the norm error becomes too small to express by double precision number, all of the experiments were based on the quadruple precision version of HOPE. Time steps were set to $\Delta t = 600, 400, 200, 100, 50$ s for C30, C45, C90, C180 and C360, respectively.

As shown in Fig. 10, errors near the panel boundaries of the cubed-sphere grid are significantly higher than those in the central regions, confirming the presence of grid imprinting. Furthermore, we implemented the AUSM-up+ Riemann solver (consistent with the scheme described in Ullrich et al., 2010) as an alternative to LMARS. While computationally more complex, AUSM$^+$-up substantially reduces simulation errors. Comparative analysis of Fig. 10a and b demonstrates that the maximum absolute error decreases from 8.792×10^5 (LMARS) to 2.413×10^5 (AUSM$^+$-up), while convergence rates remain unchanged.

Performance benchmarks using HOPE's Fortran implementation on a C90 grid show that simulating 12 d with a 200 s integration time step requires 49.4 s for LMARS versus 57.34 s for AUSM$^+$-up. This demonstrates that Riemann solver selection critically impacts simulation outcomes, consistent with the discussions in Ullrich et al. (2010).

In Table 1, we present the geopotential simulation errors and convergence rate of different order accuracy schemes at various resolutions. It is evident that HOPE is capable of achieving the designed accuracies in all tests. When the resolution exceeds C180, the errors obtained from the TPP7, TPP9 and TPP11 schemes have surpassed the limits expressible by double-precision numbers. This demonstrates HOPE's excellent error convergence for simulating smooth flow fields. It should be noted that high-order accuracy schemes do consume more computational resources.

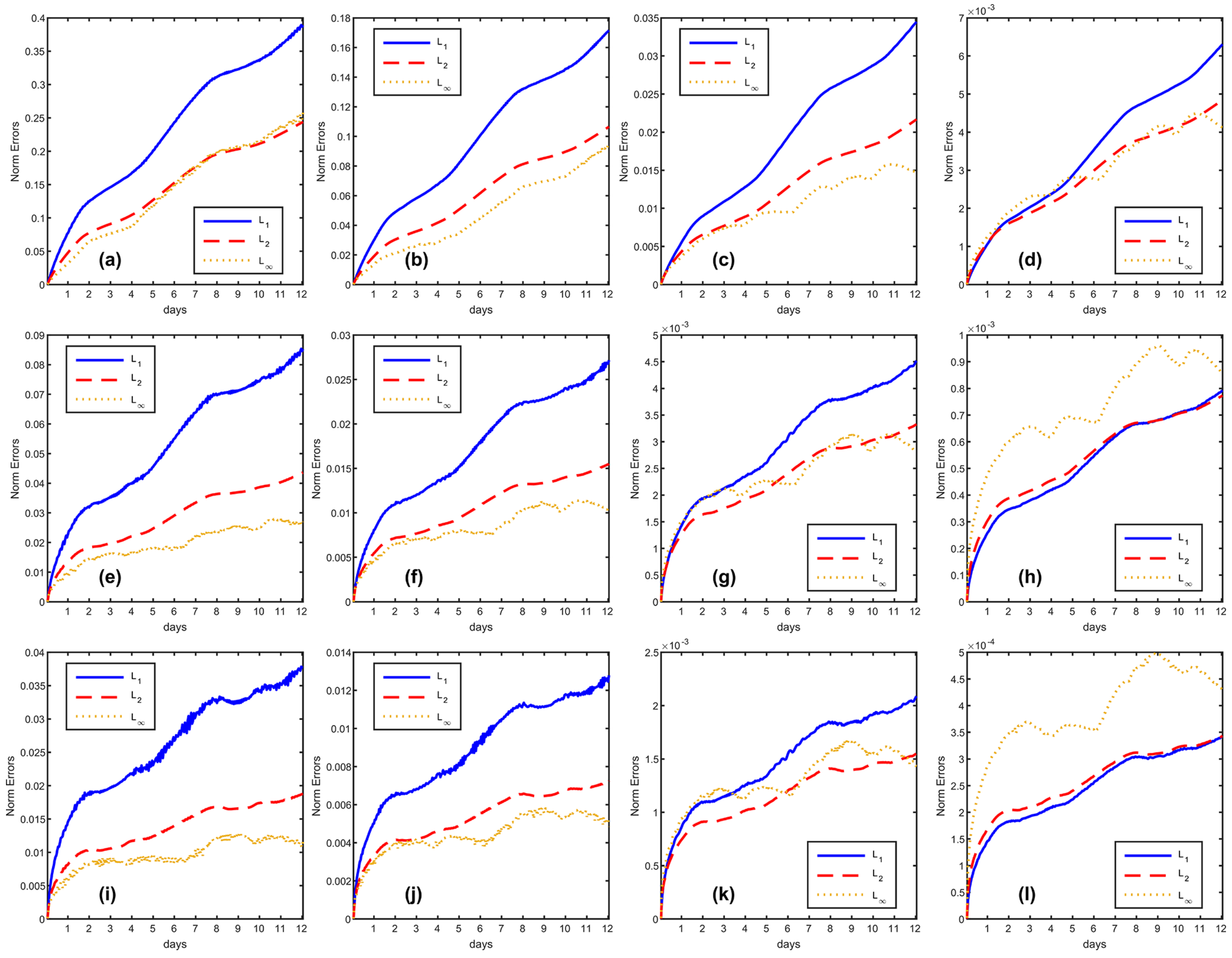

Figure 9. The variation of norm errors during simulation days for the cosine bell advection test case, with direction parameter $\alpha = \pi/4$. The rows represent reconstruction schemes TPP3, TPP5 and TPP7, the columns stand for grid C30, C45, C90 and C180.

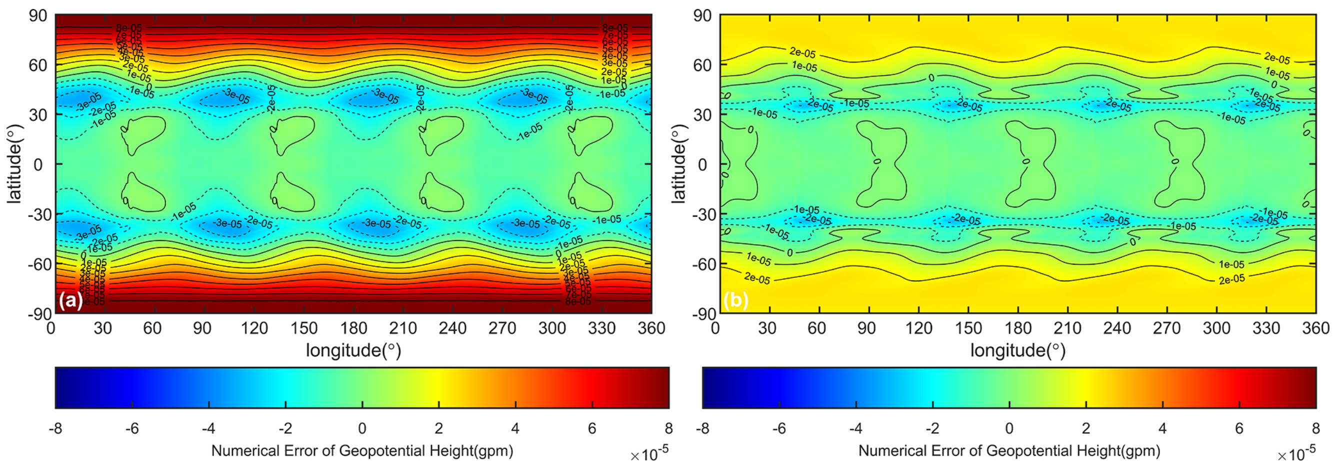

Figure 10. Numerical errors (simulation result minus exact solution) of geopotential for steady state flow with Riemann solvers **(a)** LMARS and **(b)** AUSM$^+$-up. The reconstruction scheme is TPP5, and the model resolution is C90.

HOPE has proven the feasibility of ultra-high-order accuracy finite volume methods on cubed sphere grids. However, in simulating the real atmosphere, a balance between computational efficiency and error must be considered. We believe that 3rd or 5th order accuracy schemes will be more practical for subsequent developments in baroclinic atmosphere model.

At lower resolutions, the simulation error of WENO3 is significantly higher than that of TPP3. However, as the resolution increases, the error of WENO3 progressively approaches that of TPP3. Comparing WENO5 and TPP5 results reveals a marginal increase in norm errors for WENO5, while maintaining 5th-order convergence rates. This confirms WENO5's capability to preserve high accuracy when simulating smooth flows.

It should be noted that HOPE achieves extremely small errors in simulating smooth flow fields even on very coarse resolutions. These errors can be so minute that they fall below the 16 significant digits representable in double precision. Under these conditions, conducting precision tests using double precision alone fails to accurately capture the true convergence rate. To obtain correct error measurements and convergence rate, we must employ FP128 (real (16) in Fortran). However, PyTorch's underlying architecture is built on NVIDIA CUDA, which currently supports only up to FP64 (double precision). Consequently, the PyTorch implementation cannot provide correct simulation errors when utilizing ultra-high-order schemes.

5.3 Zonal flow over an isolated mountain

Zonal flow over an isolated mountain is the 5th case mentioned in Williamson et al. (1992), this case was usually be implemented to test the topography influence in shallow water models. The initial condition is defined by Eqs. (79)–(81), but $h_0 = 5960\,\mathrm{m}$, $\phi_0 = h_0 g$, $u_0 = 20\,\mathrm{m\,s^{-1}}$. The mountain height is expressed as

$$h_\mathrm{s} = h_{\mathrm{s}_0}\left(1 - \frac{d_\mathrm{s}}{R}\right) \tag{82}$$

where $h_{\mathrm{s}_0} = 2000\,\mathrm{m}$; $R = \frac{\pi}{9}$; $d_\mathrm{s} = \sqrt{\min[R^2, (\lambda-\lambda_\mathrm{c})^2 + (\theta-\theta_\mathrm{c})^2]}$. $\lambda_\mathrm{c} = \frac{3\pi}{2}\theta_\mathrm{c} = \frac{\pi}{6}$ are the center longitude and latitude of the mountain, respectively.

HOPE is able to deal with the bottom topography correctly, as shown in Fig. 11, all of the simulation result is consistent with prior researches such as (Nair et al., 2005a; Ullrich et al., 2010; Chen and Xiao, 2008) and so on. Furthermore, as discussed in Bao et al. (2014), some high order Discontinuous Galerkin (DG) method exhibit non-physical oscillation during simulating the over mountain flow, the additional viscosity operators are necessary to alleviate this issue. However, HOPE does not require any explicit viscosity operator to suppress vorticity oscillations, the vorticity fields are smooth all the time as illustrated in Fig. 11j–l. We have tested other schemes as well, including TPP3, TPP7, WENO3, and WENO5, all of the schemes are able to achieve similar simulation results.

In the 15 d simulation of zonal flow over an isolated mountain the total energy exhibited a gradual increase over the integration time, while the total potential enstrophy showed gradual dissipation as the simulation progressed. The AUSM$^+$-up scheme demonstrated stronger energy dissipation compared to the LMARS scheme, as illustrated in Fig. 12.

5.4 Rossby–Haurwitz wave with 4 waves

Rossby–Haurwitz (RH) wave is the 6th test case introduced by Williamson et al. (1992), the RH waves are analytic solution of the spherical nonlinear barotropic vorticity equation, the reference solution is the zonal advection of RH wave without pattern changing, the angular phase speed is given by

$$c = \frac{R(R+3)\omega - 2\Omega}{(R+1)(R+2)} \tag{83}$$

where $R = 4$ is the zonal wavenumber, $\omega = 7.848 \times 10^{-6}\,\mathrm{s^{-1}}$; the earth rotation angular speed $\Omega = 7.292 \times 10^{-5}\,\mathrm{s^{-1}}$. Therefore, we have the period $T \approx 29.52\,\mathrm{d}$. The initial condition expressed as

$$\phi = \phi_0 + a^2\left[A(\theta) + B(\theta)\cos R\lambda + C(\theta)\cos 2R\lambda\right] \tag{84}$$

$$u = a\omega\cos\theta + aK\cos^{R-1}\theta\left(R\sin^2\theta - \cos^2\theta\right)\cos R\lambda \tag{85}$$

$$v = -aKR\cos^{R-1}\theta\sin\theta\sin R\lambda \tag{86}$$

$$\begin{aligned} A(\theta) =& \frac{\omega}{2}(2\Omega+\omega)\cos^2\theta + \frac{1}{4}K^2\cos^{2R}\theta \\ & \left[(R+1)\cos^2\theta + 2R^2 - R - 2 - 2R^2\cos^{-2}\theta\right] \end{aligned} \tag{87}$$

$$\begin{aligned} B(\theta) =& \frac{2(\Omega+\omega)K}{(R+1)(R+2)}\cos^R\theta[R^2+2R+2 \\ & -(R+1)^2\cos^2\theta] \end{aligned} \tag{88}$$

$$C(\theta) = \frac{1}{4}K^2\cos^{2R}\theta\left[(R+1)\cos^2\theta - R - 2\right] \tag{89}$$

where λ, θ are longitude and latitude, $K = \omega$, $\phi_0 = gh_0$, $h_0 = 8000\,\mathrm{m}$, and $a = 6\,371\,220\,\mathrm{m}$ is the earth radius.

According to the study by Thuburn and Li (2000), the Rossby–Haurwitz (RH) wave with wavenumber 4 is inherently dynamically unstable and prone to collapse. This instability can be triggered by minute perturbations, such as those arising from grid structure (breaking initial symmetry), initial condition imperfections, or numerical errors (e.g., truncation or roundoff). Similar conclusions have been verified in subsequent research. In tests conducted by Zhou et al. (2020), the TRiSK framework based on the SCVT grid could only sustain the RH wave pattern for 25 d without collapse. In contrast, Li et al. (2020) successfully maintained the RH wave pattern for 89 d using a similar algorithm on a latitude-longitude grid. Ullrich et al. (2010) developed the

Table 1. Norm errors and convergence rates of steady state geostrophic flow at day 12, with LMARS as Riemann Solver.

TPP3	C30	C45	C90	C180	C360
L_1 error	1.8853×10^{-3}	5.6474×10^{-4}	7.0960×10^{-5}	8.8777×10^{-6}	1.1099×10^{-6}
L_1 rate		2.9731	2.9925	2.9988	2.9998
L_2 error	2.1484×10^{-3}	6.4171×10^{-4}	8.0500×10^{-5}	1.0069×10^{-5}	1.2588×10^{-6}
L_2 rate		2.9802	2.9949	2.9991	2.9998
L_∞ error	4.3242×10^{-3}	1.2932×10^{-3}	1.6201×10^{-4}	2.0275×10^{-5}	2.5350×10^{-6}
L_∞ rate		2.9770	2.9968	2.9983	2.9997
TPP5					
L_1 error	3.6122×10^{-6}	4.7493×10^{-7}	1.4827×10^{-8}	4.6322×10^{-10}	1.4474×10^{-11}
L_1 rate		5.0039	5.0014	5.0004	5.0002
L_2 error	5.2427×10^{-6}	6.9169×10^{-7}	2.1627×10^{-8}	6.7584×10^{-10}	2.1119×10^{-11}
L_2 rate		4.9954	4.9992	5.0000	5.0001
L_∞ error	1.6810×10^{-5}	2.2451×10^{-6}	7.0534×10^{-8}	2.2070×10^{-9}	6.8985×10^{-11}
L_∞ rate		4.9652	4.9923	4.9982	4.9996
TPP7					
L_1 error	8.1697×10^{-8}	4.7967×10^{-9}	3.7678×10^{-11}	2.9547×10^{-13}	2.3125×10^{-15}
L_1 rate		6.9922	6.9922	6.9946	6.9974
L_2 error	8.7991×10^{-8}	5.1644×10^{-9}	4.0507×10^{-11}	3.1728×10^{-13}	2.4823×10^{-15}
L_2 rate		6.9931	6.9943	6.9963	6.9979
L_∞ error	1.4741×10^{-7}	8.6376×10^{-9}	6.7814×10^{-11}	5.3387×10^{-13}	4.1901×10^{-15}
L_∞ rate		6.9971	6.9929	6.9889	6.9934
TPP9					
L_1 error	7.8909×10^{-10}	2.1780×10^{-11}	4.3925×10^{-14}	8.6359×10^{-17}	
L_1 rate		8.8537	8.9538	8.9905	
L_2 error	9.5638×10^{-10}	2.6409×10^{-11}	5.3341×10^{-14}	1.0494×10^{-16}	
L_2 rate		8.8526	8.9516	8.9896	
L_∞ error	2.3946×10^{-9}	6.6773×10^{-11}	1.3547×10^{-13}	2.6644×10^{-16}	
L_∞ rate		8.8285	8.9452	8.9899	
TPP11					
L_1 error	1.1908×10^{-10}	1.3799×10^{-12}	6.7696×10^{-16}	3.3197×10^{-19}	
L_1 rate		10.9943	10.9932	10.9938	
L_2 error	1.3084×10^{-10}	1.5186×10^{-12}	7.4489×10^{-16}	3.6500×10^{-19}	
L_2 rate		10.9904	10.9934	10.9949	
L_∞ error	2.4204×10^{-10}	2.8579×10^{-12}	1.4147×10^{-15}	6.9567×10^{-19}	
L_∞ rate		10.9479	10.9803	10.9898	
WENO3					
L_1 error	2.6438×10^{-3}	7.2239×10^{-4}	7.7012×10^{-5}	8.9622×10^{-6}	
L_1 rate		3.1998	3.2296	3.1032	
L_2 error	4.0817×10^{-3}	9.7196×10^{-4}	9.5476×10^{-5}	1.0553×10^{-5}	
L_2 rate		3.5390	3.3477	3.1775	
L_∞ error	2.5439×10^{-2}	7.7486×10^{-3}	9.6110×10^{-4}	1.0723×10^{-4}	
L_∞ rate		2.9319	3.0112	3.1640	
WENO5					
L_1 error	3.6191×10^{-6}	4.7551×10^{-7}	1.4829×10^{-8}	4.6322×10^{-10}	
L_1 rate		5.0056	5.0030	5.0006	
L_2 error	5.2659×10^{-6}	6.9252×10^{-7}	2.1630×10^{-8}	6.7585×10^{-10}	
L_2 rate		5.0033	5.0008	5.0002	
L_∞ error	1.6873×10^{-5}	2.2466×10^{-6}	7.0539×10^{-8}	2.2070×10^{-9}	
L_∞ rate		4.9727	4.9932	4.9983	

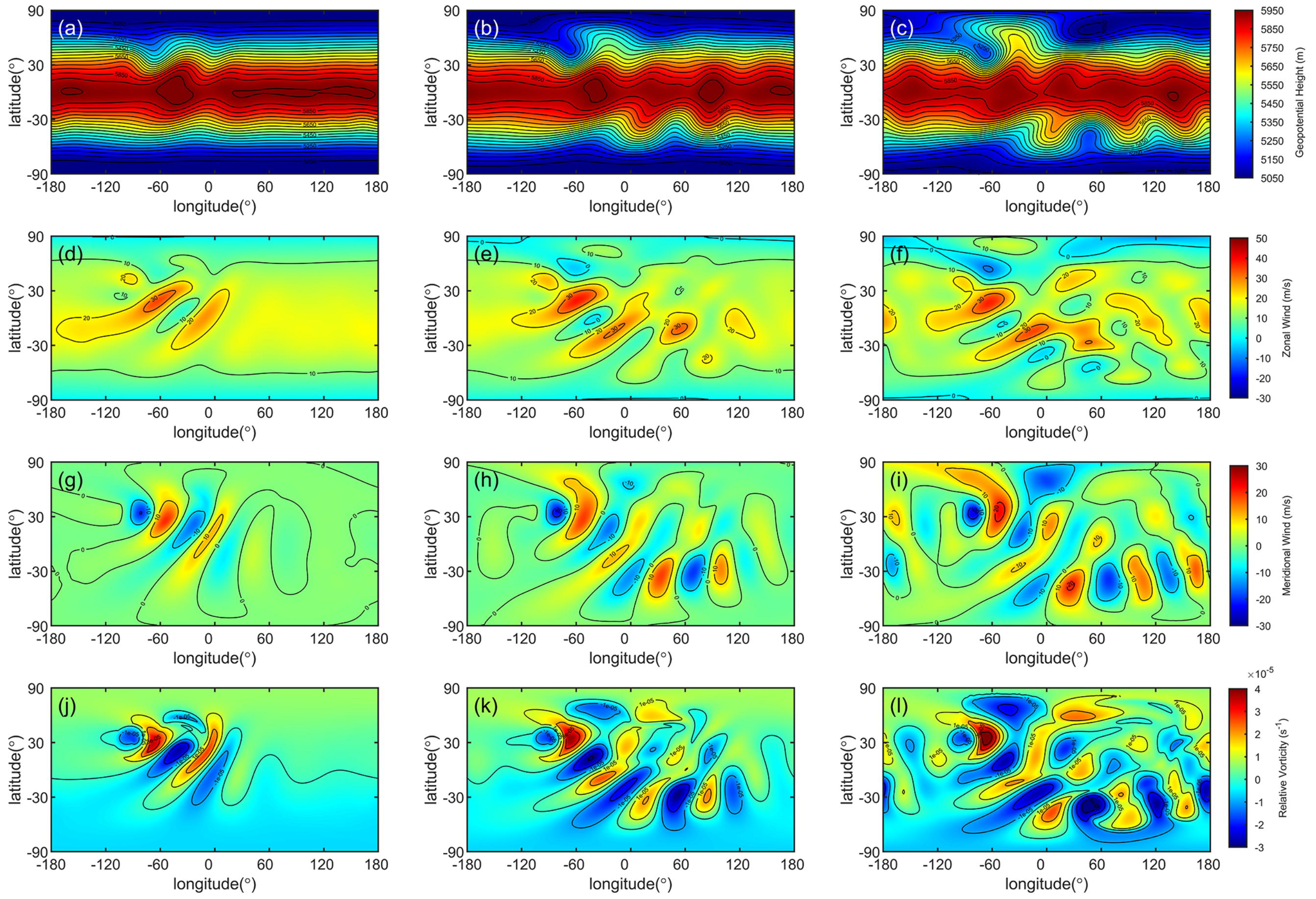

Figure 11. TPP5 (with LMARS) simulation result of the isolated mountain wave on C90 grid. The rows stand for variables: geopotential, zonal wind, meridional wind and relative vorticity, respectively. The columns represent simulation day 5, 10, 15. Geopotential contour from 5050 to 5950 m with interval 50 m. Zonal wind contour from −30 to 50 m s^{-1} with interval 10 m s−1. Meridional wind contour from −30 to 30 m s^{-1} with interval 10 m s^{-1}. Relative vorticity contour from -3×10^{-5} to 4×10^{-5} s^{-1} with interval 1×10^{-5} s^{-1}.

high-order accuracy finite volume model based on a cubed-sphere grid, which was able to sustain the RH wave for up to 90 d. In the most of our experiments, the ability of HOPE to maintain the Rossby–Haurwitz (RH) wave significantly improved with increased order of accuracy and grid resolution. All of the simulation results are based on LMARS in this section.

In the TPP3 simulation, we found that the duration for which the RH wave is maintained increases with higher grid resolution, as exhibit in Fig. 13. When the grid resolution is low (C45, C90), an obvious dissipation phenomenon can be observed. When the resolution reaches C180, the dissipation is significantly reduced, but the waveform has completely collapsed by day 90. When the resolution reaches C360, the simulation results are further improved, with dissipation further reduced, and the RH wave waveform can still barely be maintained on day 90.

A 100 d simulation of the Rossby–Haurwitz wave was conducted using a C90 grid (1° resolution). The total energy simulated with the TPP3, TPP5, TPP7, and TPP9 schemes underwent dissipation to varying degrees. By day 100, the normalized total energy errors reached -1.49×10^{-3}, -1.33×10^{-5}, -1.71×10^{-6}, -4.20×10^{-7}, respectively, indicating significantly stronger dissipation for the TPP3 scheme compared to the other higher-order schemes (Fig. 14a). Figure 14b presents a scaled view of the energy evolution for TPP5, TPP7, and TPP9, clearly demonstrating that increasing the reconstruction order progressively reduces energy dissipation. Furthermore, following the RH wave collapse, a significant drop in total energy was observed for the TPP5 scheme (after approximately 90 d) and the TPP7 scheme (after approximately 95 d).

Analysis of the normalized total potential enstrophy error (Fig. 14c) and the normalized zonal angular momentum error (Fig. 14d) over time yields conclusions consistent with those for total energy. Specifically, the TPP3 scheme exhibited substantially higher dissipation than the higher-order schemes, confirming that employing higher-order reconstruction schemes effectively minimizes dissipation. No-

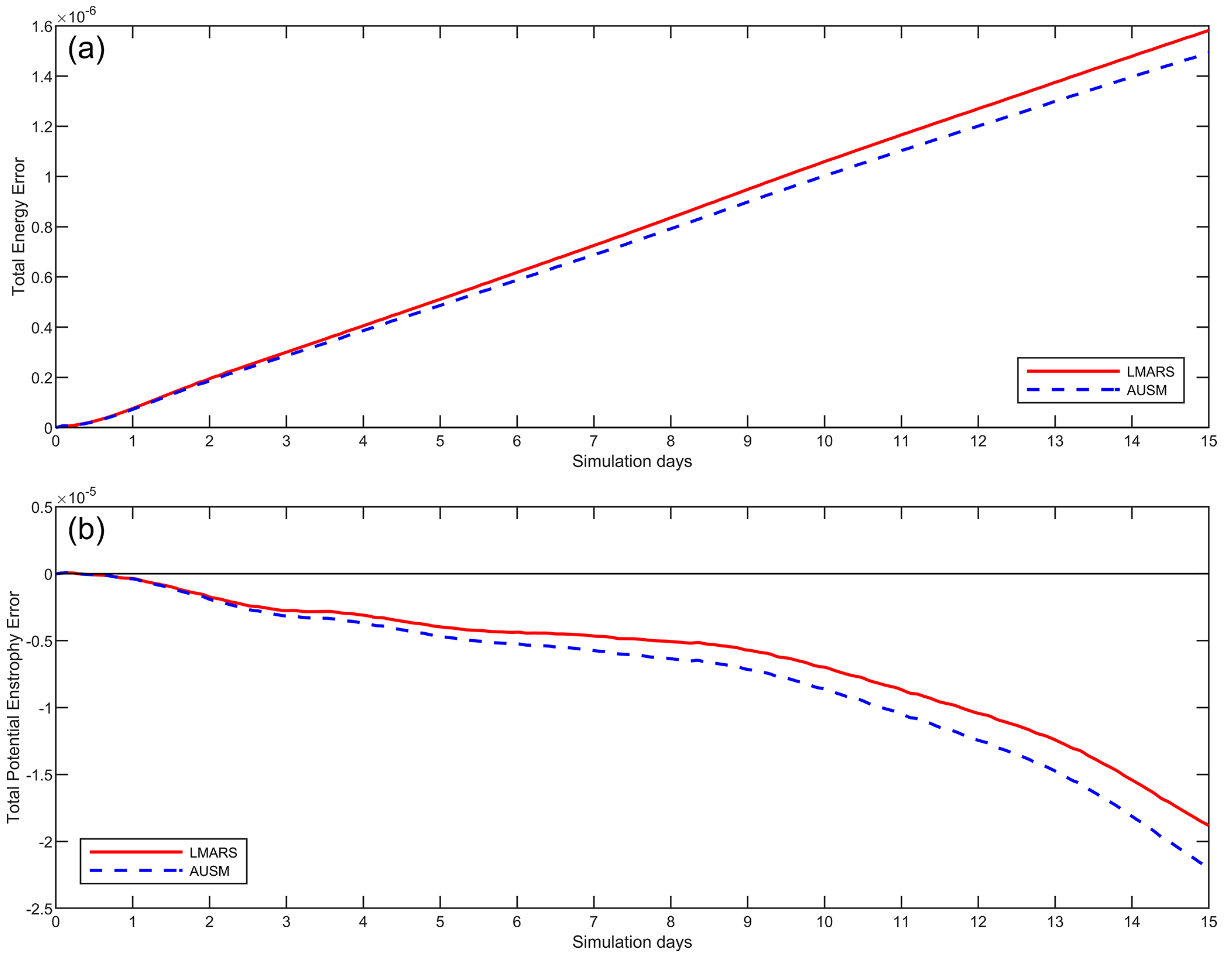

Figure 12. Time series of normalized conservation errors for the zonal flow over isolated mountain simulation on the C90 grid over days 0 to 15. **(a)** Normalized total energy error. **(b)** Normalized total potential enstrophy error. **(c)** Normalized total zonal angular momentum error.

tably, significant dissipation surges occurred in these quantities following the RH wave collapse.

In Fig. 15, we compare the impact of order-of-accuracy on the simulation capability of RH waves by fixing the resolution. By comparing row by row, it can be observed that when the accuracy reaches 5th order or higher, the dissipation is significantly reduced. Both the TPP5 and TPP7 simulations show signs of waveform distortion on day 90, and the waveform completely collapses by day 100. However, when using TPP9 for the simulation, the waveform is well maintained even until day 100.

Figure 16 presents the simulation results on the 80th day for different resolutions and reconstruction schemes. The dissipation decreases as the resolution and reconstruction order improve. At the C45 resolution, both the TPP3 and TPP5 simulations exhibit significant dissipation. Although the TPP7 simulation shows a notable improvement in dissipation, the waveform is severely distorted. The TPP9 scheme produces the best simulation results. As the resolution increases, the simulation performance also improves significantly. When using the C360 resolution, all TPP schemes yield good simulation results.

Significant differences were observed between the 2D WENO scheme and the TPP schemes in this test. Regardless of the specific WENO order employed (3, 5, 7, or 9), all WENO variants maintained the Rossby–Haurwitz (RH) wave pattern for a shorter duration compared to their TPP counterparts of equivalent order. We infer that the nonlinear processes inherent within the WENO scheme introduce asymmetries that disrupt the computational stencil symmetry, leading to a premature collapse of the RH wave.

5.5 Perturbed jet flow

The perturbed jet flow was introduced by Galewsky et al. (2004), this experiment was desired to test the model ability of simulating the fast and slow motion. the initial field is defined as

$$u(\theta) = \begin{cases} \frac{u_{\max}}{e_n} e^{\frac{1}{(\theta-\theta_0)(\theta-\theta_1)}}, & \theta \in (\theta_0, \theta_1) \\ 0, & \text{otherwise} \end{cases} \tag{90}$$

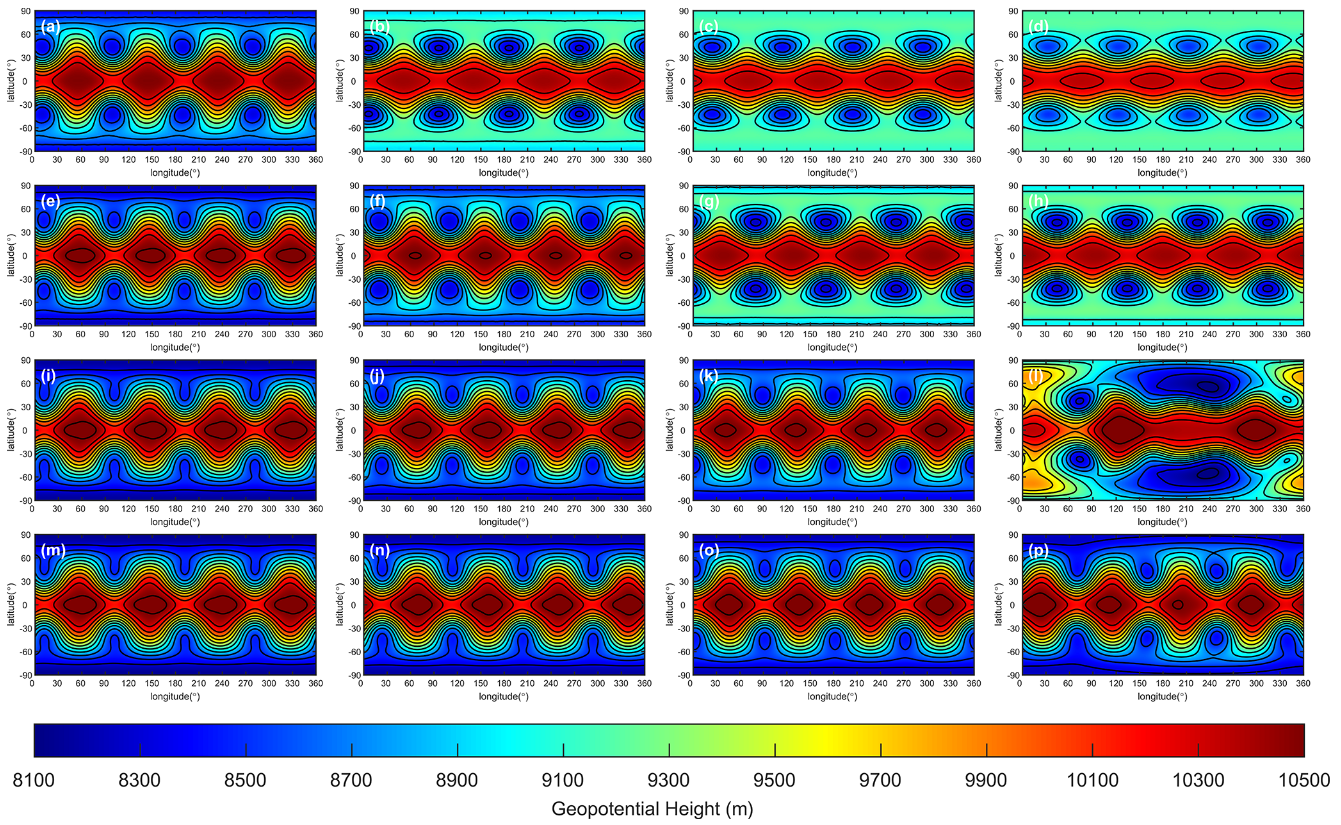

Figure 13. Geopotential of Rossby–Haurwitz wave simulated by TPP3 scheme. The rows represent grid C45, C90, C180 and C360, the columns stand for simulation day 14, 30, 60, 90. Contours from 8100 to 10 500 m with interval 200 m.

$$\phi(\lambda,\theta) = \phi_0 + \phi'(\lambda,\theta) - \int_{-\frac{\pi}{2}}^{\theta} au(\theta')\left[f + \frac{\tan\theta'}{a}u(\theta')\right]\mathrm{d}\theta' \quad (91)$$

$$\phi'(\lambda,\theta) = g\hat{h}\cos\theta e^{-\left(\frac{\lambda}{\alpha}\right)^2 - \left(\frac{\theta_2-\theta}{\beta}\right)^2},\ \lambda \in (-\pi,\pi) \quad (92)$$

where λ, θ represents longitude and latitude, $a = 6\,371\,220$ m is radius of earth, $u_{\max} = 80\,\mathrm{m\,s^{-1}}$, $\theta_0 = \frac{\pi}{7}$, $\theta_1 = \frac{5\pi}{14}$, $\theta_2 = \frac{\pi}{4}$, $e_n = e^{\frac{-4}{(\theta_1-\theta_0)^2}}$, $\alpha = \frac{1}{3}$, $\beta = \frac{1}{15}$, and $\hat{h} = 120$ m. We adopt LMARS as Riemann solver in all of the simulation in this section.

As mentioned in Chen and Xiao (2008), the perturbed jet flow experiment poses a particular challenge for the cubed-sphere grid model. Firstly, the jet stream is located at 45° N, which is very close to the boundaries of panel 5 of the cubed-sphere grid, resulting in a large geopotential gradient in the ghost interpolation region, which leads to larger interpolation error. Furthermore, the location of the geopotential perturbation ϕ' coincides with the boundary between panel 1 and panel 5, which also leads to greater numerical computation errors.

Figure 17 displays the HOPE simulation outcomes at day 6 for varying levels of reconstruction order and resolutions. The four rows correspond to the TPP5, TPP7, TPP9 and TPP11 schemes in terms of reconstruction order. The three columns, meanwhile, represent the resolutions of C45, C90, and C180, respectively. Upon comparing the different columns, it is evident that the perturbed jet flow test case converges as the resolution increases. Figure 17a, d, g, and j illustrate that, with an increase in reconstruction order, the vorticity field patterns become increasingly similar to the high-resolution results shown in the second and third columns of Fig. 17. Notably, HOPE enhances the simulation results by utilizing both higher reconstruction order and higher resolution.

5.6 Dam-break shock wave

In this section we introduce a dam-break case for testing the capability of HOPE to capture the shock wave and comparing the difference between 1D and 2D WENO schemes. The initial condition is configured as a cylinder with a geopotential of 30 000 $\mathrm{m^2\,s^{-2}}$, as shown in Fig. 18a. The geopotential is given by

$$\phi\left(d_s(\lambda,\theta)\right) = \begin{cases} 2\phi_0, & r < r_c \\ \phi_0, & \text{otherwise} \end{cases} \quad (93)$$

where $d_s = \sqrt{(\lambda-\lambda_c)^2 + (\theta-\theta_c)^2}\lambda_c = \pi$, $\theta_c = 0$, $r_c = \frac{\pi}{9}$, $\phi_0 = 30\,000\,\mathrm{m^2\,s^{-2}}$, and the earth rotation angular speed

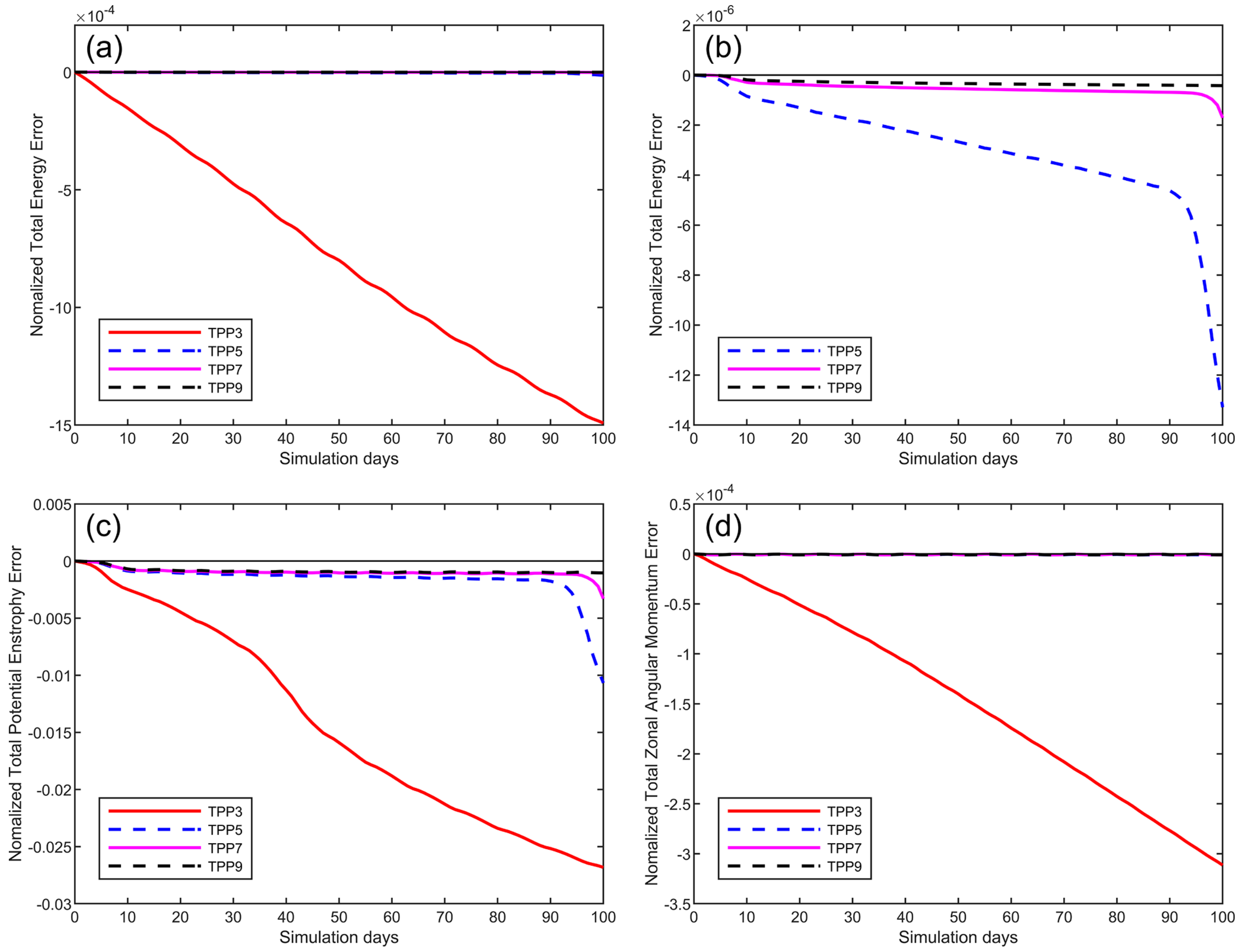

Figure 14. Time series of normalized conservation errors for the Rossby–Haurwitz wave simulation on the C90 grid over days 0 to 100, with LMARS scheme as Riemann solver. **(a)** Normalized total energy error for TPP3, TPP5, TPP7 and TPP9. **(b)** The total energy normalized error for TPP5, TPP7 and TPP9. **(c)** Normalized potential enstrophy error for TPP3, TPP5, TPP7 and TPP9. **(d)** Normalized total zonal angular momentum error for TPP3, TPP5, TPP7 and TPP9.

$\Omega = 0$. We adopt LMARS as Riemann solver in all of the simulation in this section.

In this experiment, we compare WENO5 (WENO scheme with reconstruction width 5) on both 1D and 2D schemes, the WENO-Z (Borges et al., 2008) is adopted as WENO 1D scheme, and WENO 2D scheme is consist with Sect. 3.2. Due to the initial condition being a cylinder, the resulting shock wave should maintain a circular feature. In the simulation results of WENO 1D, numerous radial textures appear (Fig. 18b). The simulation results using the WENO 2D scheme exhibit a smoother circular shape, Fig. 18c. This outcome arises because the 1D reconstruction scheme suffers from dimension split error, whereas the fitting function in the 2D reconstruction scheme incorporates cross terms. Therefore, when simulating fluid fields characterized by isotropic features, the 1D scheme lacks the capability to accurately represent diagonal directional features. Conversely, the 2D scheme correctly captures the inherent isotropic characteristics.

6 Conclusions

This paper presents HOPE, an innovative finite-volume model capable of achieving arbitrary odd-order convergence rate. Through comprehensive numerical experiments, we demonstrate that HOPE exhibits excellent convergence properties when applied to smooth flow fields, with simulation errors decreasing rapidly as the order of accuracy increases.

The model's performance has been rigorously evaluated across several benchmark cases:

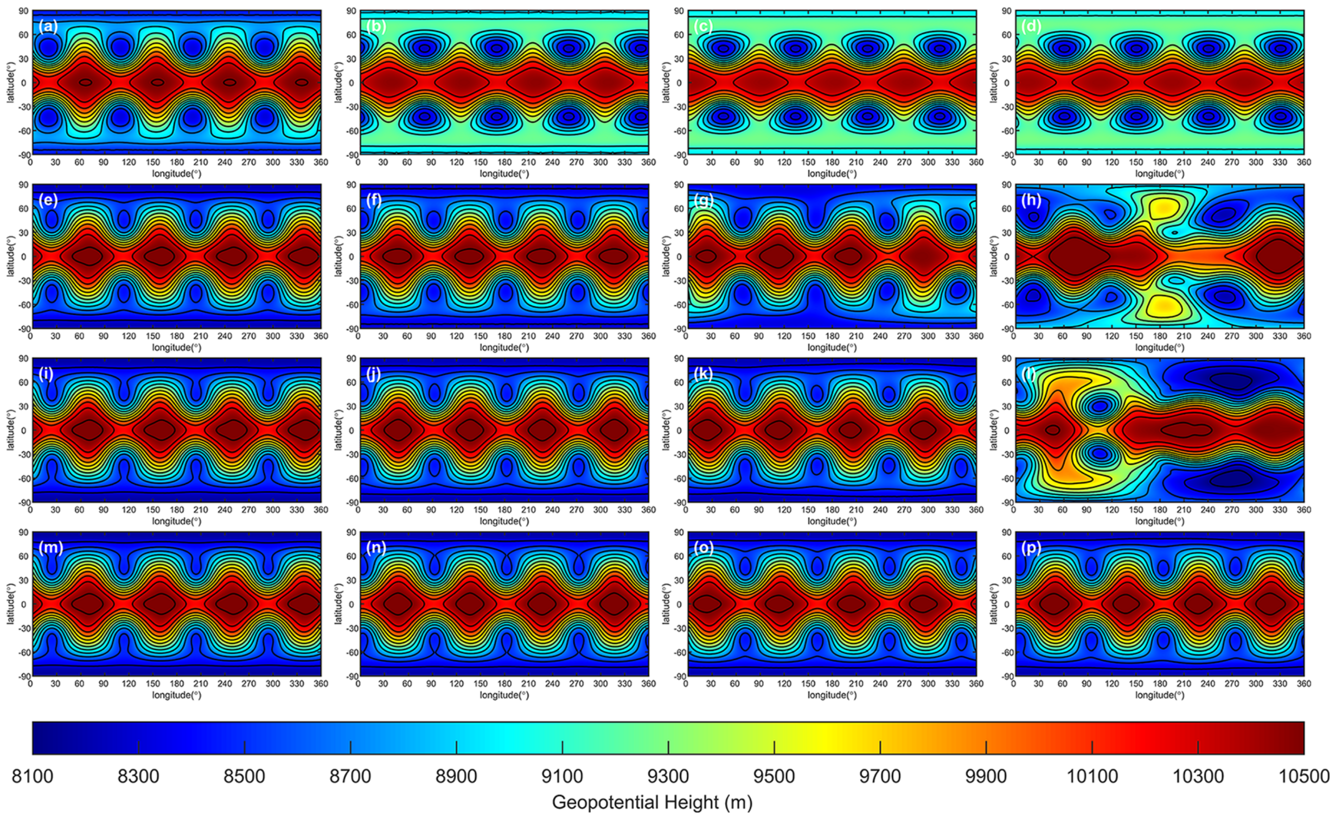

Figure 15. Geopotential of Rossby–Haurwitz wave on C90 grid, the rows represent the spatial reconstruction scheme with TPP3, TPP5, TPP7 and TPP9 the columns stand for simulation day 30, 60, 90 and 100. Contours from 8100 to 10 500 m with interval 200 m.

1. In Rossby–Haurwitz wave simulations, HOPE demonstrates superior waveform preservation capabilities that scale with both spatial resolution and accuracy order.

2. For perturbed jet flow scenarios, the model successfully resolves both fast and slow dynamical features, with significant improvements in solution quality observed at higher orders and finer resolutions.

3. Mountain wave simulations confirm HOPE's ability to accurately represent orographically-forced gravity waves.

4. In the dam break test case featuring cylindrical shock fronts, the two-dimensional WENO reconstruction scheme proves more effective than dimension-split approaches in maintaining circular symmetry.

In the case of steady geostrophic flow, Both WENO3 and WENO5 achieve the expected 3rd-order and 5th-order convergence rates, respectively. However, the computed norm errors for WENO schemes are marginally larger than those obtained with the TPP3 and TPP5 schemes. This observation confirms that the 2D WENO scheme preserves the designed convergence rate in smooth flow regions. Concurrently, in the Dam-Break Shock Wave case, the 2D WENO scheme demonstrates its robust capability for handling discontinuous flow fields. These combined results align perfectly with the primary motivation for introducing the WENO scheme: its adaptive oscillation suppression capability. Specifically, the scheme preserves the high convergence rate in sufficiently smooth regions while automatically reducing the reconstruction order near discontinuities to effectively suppress the development and propagation of non-physical oscillations.

A key innovation of HOPE lies in its computational architecture. The algorithm is specifically designed to harness GPU acceleration through (1) implementation of spatial reconstructions as convolutional operations, and (2) formulation of integration steps as matrix-vector products. These design choices leverage computational patterns widely adopted in machine learning frameworks. By developing HOPE within PyTorch, we inherit automatic differentiation capabilities, enabling straightforward coupling with neural network systems.

This integration facilitates the development of hybrid prediction models that combine a high-order, high-performance dynamical core, and Neural network-based physical parameterizations. Current research efforts have successfully extended this algorithmic framework to a two-dimensional baroclinic model (X–Z dimensions).

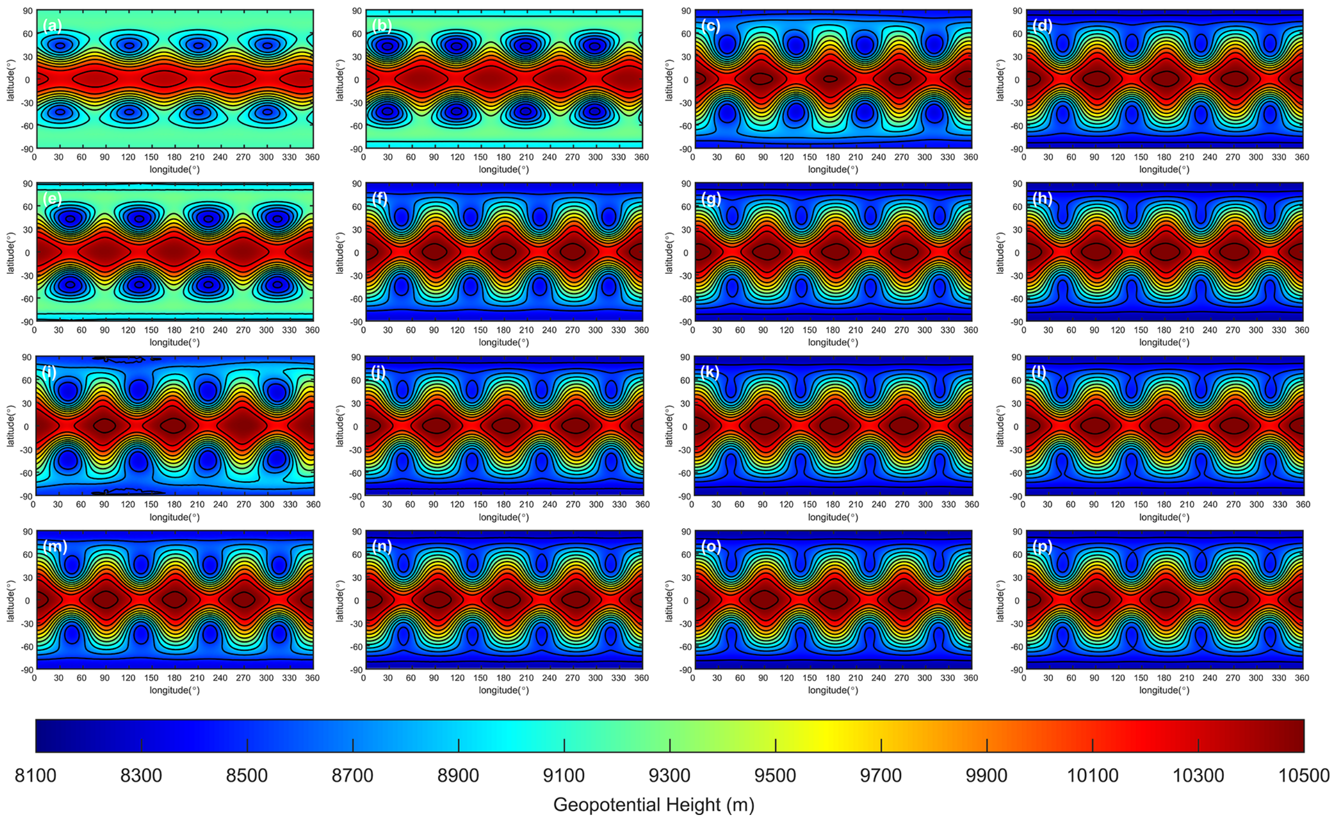
Figure 16. Geopotential of Rossby–Haurwitz wave at simulation day 80. The rows represent spatial reconstruction with TPP3, TPP5, TPP7 and TPP9. The columns stand for grid C45, C90, C180 and C360. Contours from 8100 to 10 500 m with interval 200 m.

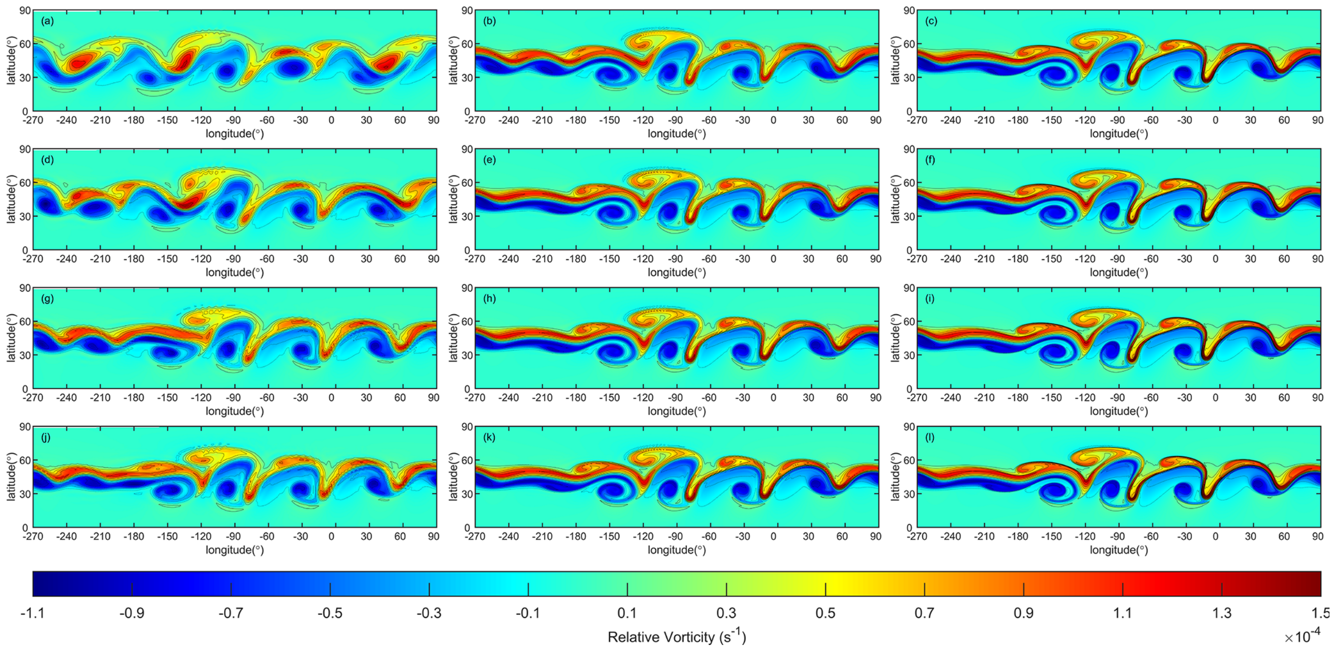
Figure 17. Relative vorticity of perturbed jet flow. **(a)**–**(c)** represent the results of TPP5 scheme with resolutions C45, C90, C180. **(d)**–**(f)** represent the results of TPP7 scheme with resolutions C45, C90, C180. **(g)**–**(i)** represent the results of TPP9 scheme with resolutions C45, C90, C180. **(j)**–**(l)** represent the results of TPP11 scheme with resolutions C45, C90, C180.

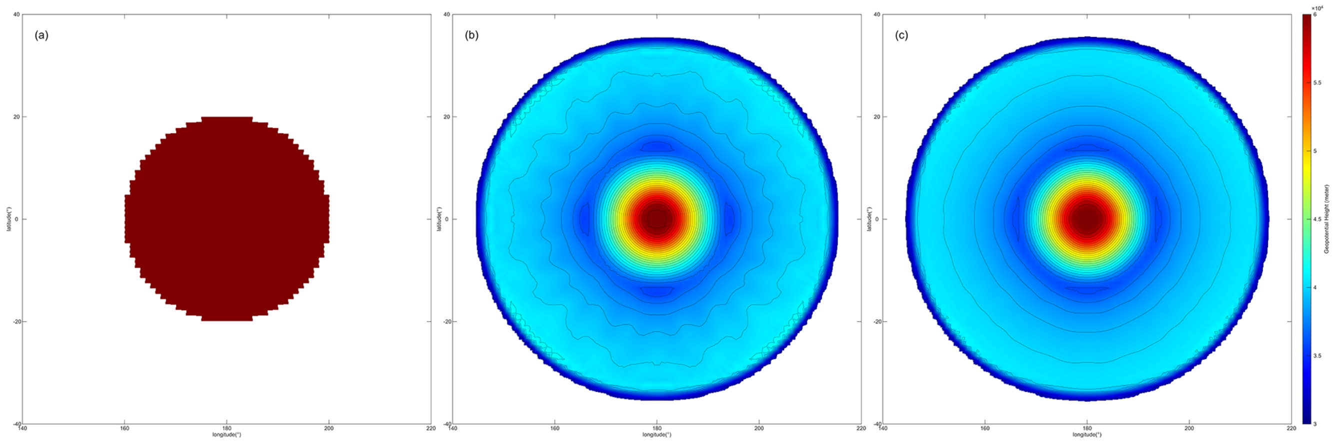

Figure 18. Geopotential of dam-break test case on C90 grid at 2nd hour. **(a)** Initial condition, **(b)** WENO 1D, **(c)** WENO 2D. The horizontal resolution for both schemes is C90. Shaded and contour from 3.2×10^4 to 6×10^4 m, with contour interval 10^3 m.

Future work will focus on developing a global, fully compressible baroclinic model using the HOPE algorithm, further demonstrating its versatility and advantages for modeling complex atmospheric dynamics. The model's unique combination of physical conservation, computational efficiency, and machine learning compatibility positions it as a powerful tool for next-generation atmospheric modeling.

Appendix A

In this appendix, we introduce a novel boundary ghost cell interpolation scheme for cubed sphere, which is able to support HOPE to reach the accuracy over 11th order or even higher.

There are two types of cells, in-domain and out-domain (also named ghost cell, as show in Fig. 7b), we define the set of in-domain cell values $\boldsymbol{q}_{d\times 1} = (q_1, q_2, \ldots, q_d)^T$, the set of out-domain cell values $\boldsymbol{g}_{h\times 1} = (g_1, g_2, \ldots, g_d)^T$, and the set of Gaussian quadrature point values (green points in Fig. 3) in out-domain cells is define as $\boldsymbol{v}_{p\times 1} = (v_1, v_2, \ldots, v_p)$. To identify the shape of the arrays, we denote the array shape using subscripts (this convention will be followed throughout the subsequent text). The purpose of ghost cell interpolation is using the known cell value $\boldsymbol{q}$ to interpolate the unknown $\boldsymbol{g}$.

Define a new set includes the values of domain cell values and ghost cell values

$$\tilde{\boldsymbol{q}}_{(d+h)\times 1} = \boldsymbol{q} \bigcup \boldsymbol{g} = (q_1, q_2, \ldots, q_d, g_1, g_2, \ldots, g_h)^T \tag{A1}$$

Similar to the describe in section 0, we can use a TPP to reconstruct the ghost quadrature points

$$\boldsymbol{v}_{p\times 1} = \mathbf{A}_{p\times(d+h)} \tilde{\boldsymbol{q}}_{(d+h)\times 1} \tag{A2}$$

where $\mathbf{A}_{p\times(d+h)}$ is the interpolation matrix that can be obtain by the similar method to Eq. (29). The ghost cell values are calculated by Gaussian quadrature

$$\boldsymbol{g}_{h\times 1} = \mathbf{B}_{h\times p} \boldsymbol{v}_{p\times 1} \tag{A3}$$

where $\mathbf{B}_{h\times p}$ is the Gaussian quadrature matrix.

$\tilde{\boldsymbol{q}}_{(d+h)\times 1}$ can be decomposed as the linear $\boldsymbol{q}_{d\times 1}$ and $\boldsymbol{v}_{p\times 1}$

$$\begin{aligned}\tilde{\boldsymbol{q}}_{(d+h)\times 1} &= \begin{pmatrix} \mathbf{I}_{d\times d} & 0 \\ 0 & \mathbf{B}_{h\times p} \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_{d\times 1} \\ \boldsymbol{v}_{p\times 1} \end{pmatrix} \\ &= \tilde{\mathbf{B}}_{(d+h)\times(d+p)} \overline{\boldsymbol{q}}_{(d+p)\times 1}\end{aligned} \tag{A4}$$

where $\mathbf{I}_{d\times d}$ is an identity matrix, and

$$\tilde{\mathbf{B}}_{(d+h)\times(d+p)} = \begin{pmatrix} \mathbf{I}_{d\times d} & 0 \\ 0 & \mathbf{B}_{h\times p} \end{pmatrix} \tag{A5}$$

$$\overline{\boldsymbol{q}}_{(d+p)\times 1} = \begin{pmatrix} \boldsymbol{q}_{d\times 1} \\ \boldsymbol{v}_{p\times 1} \end{pmatrix}. \tag{A6}$$

Substitute Eq. (30) into Eq. (26), we have

$$\begin{aligned}\boldsymbol{v}_{p\times 1} &= \mathbf{A}_{p\times(d+h)} \tilde{\mathbf{B}}_{(d+h)\times(d+p)} \overline{\boldsymbol{q}}_{(d+p)\times 1} \\ &= \tilde{\mathbf{A}}_{p\times(d+p)} \overline{\boldsymbol{q}}_{(d+p)\times 1} = \tilde{\mathbf{A}}_{p\times(d+p)} \begin{pmatrix} \boldsymbol{q}_{d\times 1} \\ \boldsymbol{v}_{p\times 1} \end{pmatrix}.\end{aligned} \tag{A7}$$

We found that matrix $\tilde{\mathbf{A}}_{p\times(d+p)}$ can be decomposed into two parts

$$\tilde{\mathbf{A}}_{p\times(d+p)} = \begin{pmatrix} \overline{\mathbf{A}}_{p\times d} & \mathbf{C}_{p\times p} \end{pmatrix}. \tag{A8}$$

Such that

$$\boldsymbol{v}_{p\times 1} = \overline{\mathbf{A}}_{p\times d} \boldsymbol{q}_{d\times 1} + \mathbf{C}_{p\times p} \boldsymbol{v}_{p\times 1}. \tag{A9}$$

Therefore

$$\left(\mathbf{I}_{p\times p} - \mathbf{C}_{p\times p}\right) \boldsymbol{v}_{p\times 1} = \overline{\mathbf{A}}_{p\times d} \boldsymbol{q}_{d\times 1}. \tag{A10}$$

We set $\mathbf{D}_{p\times p} = \mathbf{I}_{p\times p} - \mathbf{C}_{p\times p}$, then $\boldsymbol{v}_{p\times 1}$ can be determined by

$$\boldsymbol{v}_{p\times 1} = \mathbf{D}_{p\times p}^{-1} \overline{\mathbf{A}}_{p\times d} \boldsymbol{q}_{d\times 1}. \tag{A11}$$

Substitute Eq. (A11) into Eq. (A3), we establish the relationship between ghost cell values and in-domain cell values

$$\mathbf{g}_{h\times 1} = \mathbf{B}_{h\times p}\boldsymbol{v}_{p\times 1} = \mathbf{B}_{h\times p}\mathbf{D}^{-1}_{p\times p}\overline{\mathbf{A}}_{p\times d}\boldsymbol{q}_{d\times 1} = \mathcal{G}_{h\times d}\boldsymbol{q}_{d\times 1} \tag{A12}$$

where $\mathcal{G}_{h\times d} = \mathbf{B}_{h\times p}\mathbf{D}^{-1}_{p\times p}\overline{\mathbf{A}}_{p\times d}$. It is clear that Eq. (A.12) is linear, and only rely on the mesh and Gaussian quadrature scheme. Therefore, we need to compute the projection matrix $\mathcal{G}_{h\times d}$ only once for a given mesh and accuracy, this matrix can be computed by a preprocessing system and save it to the hard disk.

Code and data availability. The core data underpinning our findings are outputs produced by the HOPE code (including ghost interpolation weights and simulation results). With this code, the data can be accessed and regenerated. The digital object identifier (DOI) for the HOPE shallow water model is https://doi.org/10.5281/zenodo.16635583 (Zhou, 2025).

Author contributions. LZ developed the algorithm, implemented the code, conducted numerical experiments, and wrote the manuscript. WX provided expert guidance on software architecture design and high-performance computing implementations. XS provided expert guidance on manuscript writing.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Acknowledgements. The authors wish to express their profound respect for the rigorous scientific approach and dedication to pure science demonstrated by the anonymous reviewers. Their numerous valuable suggestions and critiques have significantly enhanced the manuscript's linguistic clarity, argumentative rigor, and experimental completeness.

Financial support. This work is partially supported by the National Natural Science Foundation of China (grant no. U2242210).

Review statement. This paper was edited by Yongze Song and reviewed by two anonymous referees.

References

Acker, F., de R. Borges, R. B., and Costa, B.: An improved WENO-Z scheme, J. Comput. Phys., 313, 726–753, https://doi.org/10.1016/j.jcp.2016.01.038, 2016.

Bao, L., Nair, R. D., and Tufo, H. M.: A Mass and Momentum Flux-Form High-Order Discontinuous Galerkin Shallow Water Model on the Cubed-Sphere, J. Comput. Phys., 271, 224–243, https://doi.org/10.1016/j.jcp.2013.11.033, 2014.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q.: Pangu-Weather: A 3D High-Resolution System for Fast and Accurate Global Weather Forecast, arXiv [preprint], arXiv:2211.02556v1 [physics.ao-ph], https://doi.org/10.48550/arXiv.2211.02556, 2022.

Borges, R., Carmona, M., Costa, B., and Don, W. S.: An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws, J. Comput. Phys., 227, 3191–3211, https://doi.org/10.1016/j.jcp.2007.11.038, 2008.

Chen, C. and Xiao, F.: Shallow Water Model on Cubed-Sphere by Multi-Moment Finite Volume Method, J. Comput. Phys., 227, 5019–5044, https://doi.org/10.1016/j.jcp.2008.01.033, 2008.

Chen, K., Han, T., Chen, X., Ma, L., Gong, J., Zhang, T., Yang, X., Bai, L., Ling, F., Su, R., Ci, Y., and Ouyang, W.: FengWu: Pushing the Skillful Global Medium-range Weather Forecast beyond 10 Days Lead, arXiv [prepint, arXiv:2304.02948v1 [cs.AI], https://doi.org/10.48550/arXiv.2304.02948, 2023.

Chen, X., Andronova, N., Van Leer, B., Penner, J. E., Boyd, J. P., Jablonowski, C., and Lin, S.-J.: A Control-Volume Model of the Compressible Euler Equations with a Vertical Lagrangian Coordinate, Mon. Weather Rev., 141, 2526–2544, https://doi.org/10.1175/mwr-d-12-00129.1, 2013.

Galewsky, J., Scott, R. K., and Polvani, L. M.: An Initial-Value Problem for Testing Numerical Models of the Global Shallow-Water Equations, Tellus A, 56, 429–440, 2004.

Hu, C. and Shu, C.-W.: Weighted Essentially Non-oscillatory Schemes on Triangular Meshes, J. Comput. Phys., 150, 97–127, 1999.

Ii, S. and Xiao, F.: A Global Shallow Water Model Using High Order Multi-Moment Constrained Finite Volume Method and Icosahedral Grid, J. Comput. Phys., 229, 1774–1796, https://doi.org/10.1016/j.jcp.2009.11.008, 2010.

Jiang, G.-S. and Shu, C.-W.: Efficient Implementation of Weighted ENO Schemes, J. Comput. Phys., 126, 202–228, https://doi.org/10.1006/jcph.1996.0130, 1996.

Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., Lottes, J., Rasp, S., Duben, P., Klower, M., Hatfield, S., Battaglia, P., Sanchez-Gonzalez, A., Willson, M., Brenner, M. P., and Hoyer, S.: Neural General Circulation Models for Weather and Climate, Nature, 632, 1060–1066, 2024.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., and Battaglia, P.: GraphCast: Learning Skillful Medium-Range Global Weather Forecasting, Science, 382, 1416–1421, 2023.

Li, J., Wang, B., and Dong, L.: Analysis of and Solution to the Polar Numerical Noise Within the Shallow-Water Model on the Latitude–Longitude Grid, J. Adv. Model. Earth Syst., 12, https://doi.org/10.1029/2020ms002047, 2020.

Liou, M.-S.: A Sequel to AUSM, Part II: AUSM+-up for All Speeds, J. Comput. Phys., 214, 137–170, https://doi.org/10.1016/j.jcp.2005.09.020, 2006.

Liu, X.-D., Osher, S., and Chan, T.: Weighted Essentially Non-oscillatory Schemes, J. Comput. Phys., 115, 200–212, 1994.

Luo, X. and Wu, S.-P.: An improved WENO-Z+ scheme for solving hyperbolic conservation laws, J. Computat. Phys., 445, https://doi.org/10.1016/j.jcp.2021.110608, 2021.

Nair, R. D., Thomas, S. J., and Loft, R. D.: A Discontinuous Galerkin Global Shallow Water Model, Mon. Weather Rev., 133, 876–888, 2005a.

Nair, R. D., Thomas, S. J., and Loft, R. D.: A Discontinuous Galerkin Transport Scheme on the Cubed Sphere, Mon. Weather Rev., 133, 814–828, 2005b.

Sadourny, R.: Conservative Finite-Difference Approximations of the Primitive Equations on Quasi-Uniform Spherical Grids, Mon. Weather Rev., 100, 136–144, 1972.

Shi, J., Hu, C., and Shu, C.-W.: A Technique of Treating Negative Weights in WENO Schemes, J. Comput. Phys., 175, 108–127, https://doi.org/10.1006/jcph.2001.6892, 2002.

Thuburn, J. and Li, Y.: Numerical Simulations of Rossby–Haurwitz Waves, Tellus A, 52, 181–189, https://doi.org/10.1034/j.1600-0870.2000.00107.x, 2000.

Ullrich, P. A. and Jablonowski, C.: MCore: A Non-hydrostatic Atmospheric Dynamical Core Utilizing High-order Finite-Volume Methods, Journal of Computational Physics, 231, 5078-5108, 10.1016/j.jcp.2012.04.024, 2012.

Ullrich, P. A., Jablonowski, C., and van Leer, B.: High-Order Finite-Volume Methods for the Shallow-Water Equations on the Sphere, J. Comput. Phys., 229, 6104–6134, https://doi.org/10.1016/j.jcp.2010.04.044, 2010.

Wicker, L. J. and Skamarock, W. C.: Time-Splitting Methods for Elastic Models Using Forward Time Schemes, Mon. Weather Rev., 130, 2088–2097, 2002.

Williamson, D. L., Drake, J. B., Hack, J. J., Jakob, R., and Swarztrauber, P. N.: A Standard Test Set for Numerical Approximations to the Shallow Water Equations in Spherical Geometry, J. Comput. Phys., 102, 211–224, 1992.

Zhang, Y., Long, M., Chen, K., Xing, L., Jin, R., Jordan, M. I., and Wang, J.: Skilful Nowcasting of Extreme Precipitation with NowcastNet, Nature, 619, 526–532, https://doi.org/10.1038/s41586-023-06184-4, 2023.

Zhou, L.: High Order Prediction Environment, Zenodo [code], https://doi.org/10.5281/zenodo.16635583, 2025.

Zhou, L., Feng, J., Hua, L., and Zhong, L.: Extending Square Conservation to Arbitrarily Structured C-grids with Shallow Water Equations, Geosci. Model Dev., 13, 581–595, https://doi.org/10.5194/gmd-13-581-2020, 2020.

Zhu, J. and Shu, C.-W.: A New Type of Multi-resolution WENO Schemes with Increasingly Higher Order of Accuracy on Triangular Meshes, J. Comput. Phys., 392, 19–33, https://doi.org/10.1016/j.jcp.2019.04.027, 2019.