High-Order Spectral Element Transport Scheme

LI,CHIN-KAI

Department of Applied Mathematics National Chung Hsing University (NCHU)

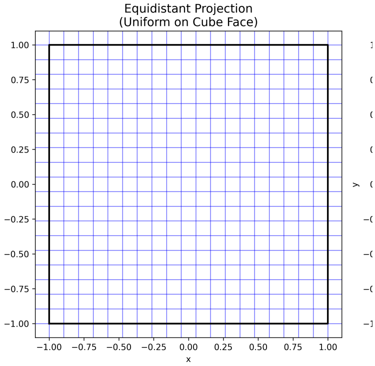
December 16, 2025

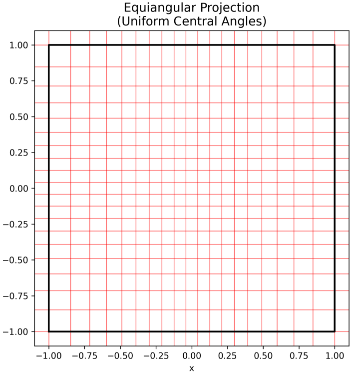
Outline

Comparison of Central Projections

1. **Equidistant Projection**
   - **Coordinates**: Uses local Cartesian coordinates (x, y) on each cube face.
   - **Characteristics**: Grid spacing is constant in projection plane ($\Delta x = \Delta y$).

2. **Equiangular Projection**
   - **Coordinates**: Uses central angles (α, β) as independent variables.
   - **Characteristics**: Results in a more **uniform grid distribution** compared to equidistant.





Mapping: Cube to Sphere (Staircase Layout)

1. Logical Layout (Staircase)

- The six faces ($P_1 \dots P_6$) are arranged in a specific **staircase pattern**.
- Arrows indicate local coordinate orientation (x, y) and connectivity.

2. Gnomonic Projection Mapping from local face coordinates (x, y) to spherical surface (X, Y, Z):

Transformation Formula

Let $\mathbf{r}_c$ be a point on the cube face ($r^2 = a^2 + x^2 + y^2$):

$$(X, Y, Z) = \frac{R}{r}(a, x, y) \quad \text{(example for face } P_1\text{)} \tag{1}$$

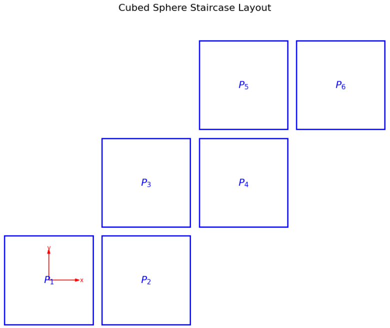
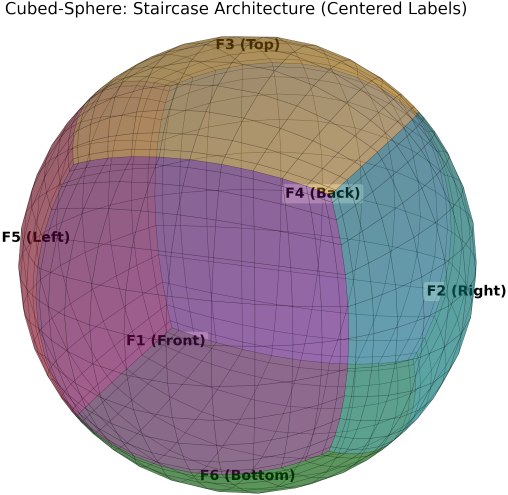


Figure: Connectivity and orientation of the six cube faces ($P_1 \dots P_6$).

The Cubed Sphere Grid Structure



1. Geometric Foundation: Cubed-Sphere

Cubed-Sphere Projection

Based on the equiangular projection, the surface of the sphere is divided into six identical regions. The local coordinates (x, y) on each face P_i are related to the equiangular coordinates (α, β) by:

$$x = a \tan\alpha, \quad y = a \tan\beta, \quad (\alpha, \beta) \in [-\frac{\pi}{4}, \frac{\pi}{4}] \tag{2}$$

Global-to-Local Mapping

A point on the sphere (X, Y, Z) is mapped to the local face using the radial distance r and a mapping vector $\mathbf{v}_i$ specific to each face:

$$(X, Y, Z) = \frac{R}{r}\mathbf{v}_i, \quad \text{where } r = \sqrt{a^2 + x^2 + y^2} \tag{3}$$

The vector $\mathbf{v}_i$ determines the orientation (e.g., for F_1, $\mathbf{v}_1 = (a, x, y)$).

2. Coordinate Transformation Logic

To perform numerical operations, we establish a relationship between the spherical coordinates (λ, θ) and the local projection coordinates.

Geometric Consistency

- The mapping ensures geometrical continuity across face boundaries.
- *Note:* For Face 5, the mapping vector is corrected to $(y, -a, -x)$ to ensure the local coordinate system aligns with the staircase structure (Eq. 3).

Role of Transformation Matrix

- Arbitrary tangent vectors on the sphere are mapped to the projection plane via a transformation matrix A.
- The Metric Tensor is defined as $G = A^T A$.
- The Jacobian determinant is derived from $J = \sqrt{\det(G)}$.

3. Derivation of Matrix A

General Definition (Jacobian Matrix): According to the chain rule and geometry, the transformation matrix A is derived as the product of rotation, spherical derivatives, and projection scaling (Eq. 4):

$$A = R \underbrace{\begin{bmatrix} \cos\theta & 0 \\ 0 & 1 \end{bmatrix}}_{\text{Spherical Metric}} \underbrace{\begin{bmatrix} \frac{\partial\lambda}{\partial x} & \frac{\partial\lambda}{\partial y} \\ \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial y} \end{bmatrix}}_{\text{Coord. Change}} \underbrace{\begin{bmatrix} a\sec^2\alpha & 0 \\ 0 & a\sec^2\beta \end{bmatrix}}_{\text{Projection Deriv.}} \tag{4}$$

Resulting Matrices

For example, P_1 results in:

$$A_{P1} = \begin{bmatrix} R\cos^2\lambda\cos\theta\sec^2\alpha & 0 \\ -R\sin\lambda\cos\lambda\sin\theta\cos\theta\sec^2\alpha & R\cos\lambda\cos^2\theta\sec^2\beta \end{bmatrix} \tag{5}$$

4.Jacobian Solution

The Jacobian $J = \sqrt{\det(G)}$ represents the area expansion factor.

ρ

We define a geometric parameter ρ to simplify the expressions:

$$\rho = \sqrt{1 + \tan^2\alpha + \tan^2\beta} \tag{6}$$

It relates to the spherical angles (e.g., for P_1, $\rho^2 = \sec^2\lambda \cdot \sec^2\theta$).

Analytical Formula

By substituting the coordinate definitions into the determinant of A, all six faces converge to a single, unified analytical solution (Eq. 7):

$$J(\alpha, \beta) = \frac{R^2}{\rho^3 \cos^2\alpha \cos^2\beta} \tag{7}$$

5. Advection Equation

1. Standard Conservation Law

The conservation law for a scalar ϕ on the sphere is given by:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = 0 \tag{8}$$

2. Numerical Formulation (Split Form)

To ensure numerical stability and minimize aliasing errors, we approximate the advection term using the **skew-symmetric split form**:

$$\mathbf{u} \cdot \nabla \phi = \underbrace{\frac{1}{2}\nabla \cdot (\phi \mathbf{u})}_{\text{Flux Term}} + \underbrace{\frac{1}{2}\mathbf{u} \cdot \nabla \phi}_{\text{Advective Term}} - \underbrace{\frac{1}{2}\phi(\nabla \cdot \mathbf{u})}_{\text{Correction Term}} \tag{9}$$

Note: The Correction Term is negligible for non-divergent flow ($\nabla \cdot \mathbf{u} \approx 0$) but improves conservation.

6.Expansion of Terms

Transforming the split terms into the computational (α, β) space using the Jacobian J:

$$\textbf{Flux Term} = \frac{1}{J}\left[\frac{\partial}{\partial\alpha}(Ju_\alpha\phi) + \frac{\partial}{\partial\beta}(Ju_\beta\phi)\right] \tag{10}$$

$$\textbf{Advective Term} = u_\alpha\frac{\partial\phi}{\partial\alpha} + u_\beta\frac{\partial\phi}{\partial\beta} \tag{11}$$

$$\textbf{Correction Term} = \phi\underbrace{\left\{\frac{1}{J}\left[\frac{\partial}{\partial\alpha}(Ju_\alpha) + \frac{\partial}{\partial\beta}(Ju_\beta)\right]\right\}}_{\nabla\cdot\mathbf{u}\approx 0} \tag{12}$$

- The final time-stepping equation combines these terms: $\frac{\partial\phi}{\partial t} = -\left(\frac{1}{2}\text{Flux} + \frac{1}{2}\text{Adv.} - \frac{1}{2}\text{Corr.}\right)$

9. Numerical Stability: The SAT Method

To ensure stability and continuity across element boundaries, we use the **Simultaneous Approximation Term (SAT)** method.

SAT Formulation (Upwind Scheme)

Combining the flux difference and the metric correction, the penalty term P applied to the RHS is:

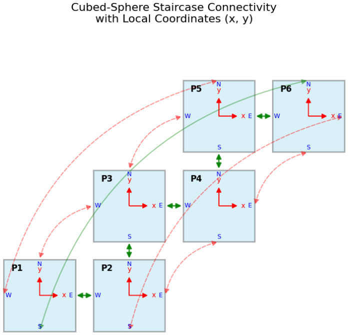
$$P = \frac{1}{w} \cdot \underbrace{\left[\frac{V_n - |V_n|}{2} \right]}_{\text{Upwind Switch}} \cdot (\phi_{outer} - \phi_{inner}) \tag{13}$$

Key Definitions:

- **Upwind Switch** ($\frac{V_n - |V_n|}{2}$):
  - **Outflow ($V_n \geq 0$):** Becomes 0. Physics relies on internal solution; no penalty needed.
  - **Inflow ($V_n < 0$):** Becomes V_n. Forces ϕ_{inner} to approach the neighbor's value ϕ_{outer}.
- **Weight Correction (w)**:
  - w represents the **Legendre-Gauss-Lobatto (LGL) weight**.

11. Staircase Topology & Exchange Algorithm

1. **Staircase Topology**
   - **Structure:** 6 Panels (P1-P6) in a step-like arrangement.
   - **Local Coords:** Red arrows indicate (x, y) axes.
   - **Connections:**
     - Green: Standard match.
     - Red: Requires **index reversal**.



2. Exchange Algorithm Logic

Step 1: Lookup

```
nbr, side, rev = conn[face][edge]
```

Step 2: Extraction Slice neighbor data based on side index:

- West (0): `data[0, :]`
- East (1): `data[-1, :]`
- South (2): `data[:, 0]`
- North (3): `data[:, -1]`

Step 3: Reverse Correction If geometric mismatch exists (`rev=True`), flip the array to align points:

```
if rev:
    boundary = boundary[::-1]
```

12. Full Connectivity Table Definition

The complete topology logic for all 6 faces (P1-P6) is defined below. **Note:** **True** indicates that the neighbor's boundary array must be reversed.

Panels 1 – 3 (Lower/Middle)

Face	Side	Neighbor (Face, Side)	Rev
P1	West	P5 (Face 4), North	**True**
	East	P2 (Face 1), West	False
	South	P6 (Face 5), North	False
	North	P3 (Face 2), West	**True**
P2	West	P1 (Face 0), East	False
	East	P4 (Face 3), South	**True**
	South	P6 (Face 5), East	**True**
	North	P3 (Face 2), South	False
P3	West	P1 (Face 0), North	**True**
	East	P4 (Face 3), West	False
	South	P2 (Face 1), North	False
	North	P5 (Face 4), West	**True**

Panels 4 – 6 (Upper/Top)

Face	Side	Neighbor (Face, Side)	Rev
P4	West	P3 (Face 2), East	False
	East	P6 (Face 5), South	**True**
	South	P1 (Face 0), East	**True**
	North	P5 (Face 4), South	False
P5	West	P3 (Face 2), North	**True**
	East	P6 (Face 5), West	False
	South	P4 (Face 3), North	False
	North	P1 (Face 0), West	**True**
P6	West	P5 (Face 4), East	False
	East	P2 (Face 1), South	**True**
	South	P4 (Face 3), East	**True**
	North	P1 (Face 0), South	False

14. Time Integration: LSRK45 Algorithm

To advance the solution from time t to $t + \Delta t$, we employ the **5-stage, 4th-order Low-Storage Runge-Kutta (LSRK45)** method.

Why LSRK45?

- **High Accuracy:** 4th-order accuracy minimizes time discretization errors.
- **Low Memory:** Uses only **2 storage registers** (U and dU) instead of the 4-5 arrays required by classical RK4 schemes. This is critical for large-scale grid problems.

The 2N-Storage Algorithm Scheme: For each time step, iterate through 5 stages ($j = 1 \dots 5$) :

$$\text{1. Compute RHS:} \quad R = \mathcal{L}(U^{(j-1)}, t_n + c_{j-1}\Delta t) \tag{14}$$

$$\text{2. Update Residual:} \quad dU^{(j)} = A_j dU^{(j-1)} + \Delta t \cdot R \tag{15}$$

$$\text{3. Update Solution:} \quad U^{(j)} = U^{(j-1)} + B_j dU^{(j)} \tag{16}$$

Time Step Definition (Δt): The time step is constrained by the CFL condition and scales with N^{-2}:

$$\Delta t = \frac{\text{CFL}}{2\nu_{\max}} \frac{2}{N^2} \tag{17}$$

15. Numerical Setup & Initial Condition

Initial Condition: Gaussian Bell

The scalar field ϕ is initialized as:

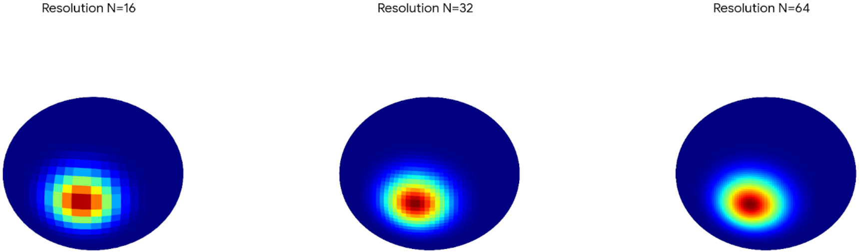
$$\phi = h_0 \exp\left(-\left(\frac{r_d}{r_0}\right)^2\right) \tag{18}$$

where r_d is the great-circle distance from the center (λ_0, θ_0).

Simulation Parameters

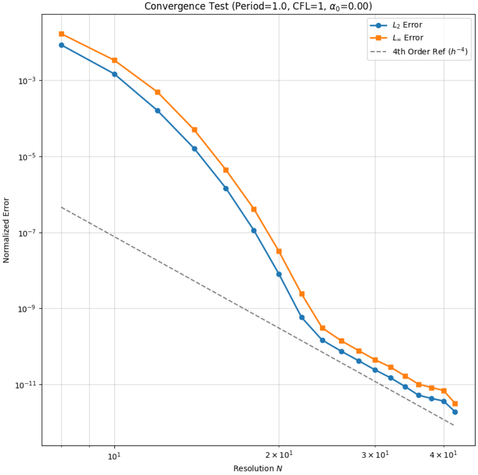
Parameter	**Value**
Test Case	Solid Body Rotation
Period (T)	1.0 (Full Revolution)
Time Step (Δt)	Based on CFL $= 0.5$
Flow Angle (α_0)	$\pi/4$ rad
Bell Height (h_0)	1.0
Bell Width (r_0)	$R/3$

Grid Resolution Comparison (Initial Field):



16. Convergence Test: Moving Bump

To verify the spectral accuracy, we simulate a moving Gaussian bump on the sphere for one full revolution ($T = 1.0$).



N	L2 Error	Order	Linf Error	Order
8	8.63-3	-	1.70-2	-
10	1.49-3	3.93	3.42-3	3.59
12	1.61-4	6.11	4.95-4	5.30
14	1.63-5	7.42	5.03-5	7.42
16	1.46-6	9.03	4.45-6	9.08
18	1.14-7	10.83	4.16-7	10.07
20	8.07-9	12.56	3.19-8	12.19
22	5.81-10	13.80	2.41-9	13.54
24	1.45-10	7.97	3.05-10	11.88
(Time Error Dominates)				
26	7.41-11	4.19	1.40-10	4.87
30	2.37-11	4.01	4.39-11	4.03
36	5.14-12	4.64	1.00-11	4.40
40	3.61-12	1.50	6.84-12	1.76
42	1.88-12	6.69	3.14-12	7.96

References

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

M. H. Carpenter and C. A. Kennedy.
Fourth-order 2N-storage Runge-Kutta schemes.
NASA Technical Memorandum, 109112, 1994.

Thank You

for your attention

Source Code Available at GitHub:

https://github.com/Funtrollor/Cube_sphere_tansport_equation