ELSEVIER

Computers & Geosciences 33 (2007) 1022–1035

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

High-order Galerkin methods for scalable global atmospheric models

Michael N. Levy[a,*], Ramachandran D. Nair[b], Henry M. Tufo[c]

[a]*Department of Applied Mathematics, University of Colorado at Boulder, 526 UCB Boulder, CO 80309-0526, USA*
[b]*Institute for Mathematics Applied to Geosciences (IMAGe), The National Center for Atmospheric Research, P.O. Box 3000 Boulder, CO 80307-3000, USA*
[c]*Computational & Information Systems Laboratory (CISL), The National Center for Atmospheric Research, P.O. Box 3000 Boulder, CO 80307-3000, USA*

Received 10 July 2006; received in revised form 6 November 2006; accepted 5 December 2006

Abstract

Three different high-order finite element methods are used to solve the advection problem—two implementations of a discontinuous Galerkin and a spectral element (high-order continuous Galerkin) method. The three methods are tested using a 2D Gaussian hill as a test function, and the relative L_2 errors are compared. Using an explicit Runge–Kutta time stepping scheme, all three methods can be parallelized using a straightforward domain decomposition and are shown to be easily and efficiently scaled across multiple-processor distributed memory machines. The effect of a monotonic limiter on a DG scheme is demonstrated for a non-smooth solution. Additionally, the necessary geometry for implementing these methods on the surface of a sphere is discussed.

Keywords: High-order methods; Discontinuous Galerkin methods; Spectral element methods; Cubed sphere; Transport equation; Atmospheric modeling

1. Introduction

With the rise in popularity of distributed memory parallel processing, high-order methods such as spectral element (SE or continuous Galerkin) methods and discontinuous Galerkin (DG) methods are becoming more common in atmospheric models (Taylor et al., 1997; Nair et al., 2005a). The advantages of these local methods are high-order accuracy, parallel efficiency, and geometric flexibility, including adaptive mesh refinement capability.

Traditional models use low-order finite difference methods or rely on global methods employing spectral transforms, which have dominated global climate modeling for the past two decades (Washington and Parkinson, 2005). Unfortunately, such numerical methods are not well-suited to exploit the massively parallel computers; for example, the spectral transform methods cannot take advantage of the computer power found in distributed memory parallel computers because of the non-local communication.

*Corresponding author.
E-mail address: levym@colorado.edu (M.N. Levy).

0098-3004/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2006.12.004

Conventional global atmospheric models based on latitude–longitude spherical geometry have polar singularities due to mesh convergence. Eulerian grid-point models based on such grid systems not only have severe stability restrictions in the polar regions, but also require non-local polar filtering which greatly impedes efficient communication on a massively parallel machine. To address these issues, atmospheric modelers recently reintroduced the cubed sphere geometry for global modeling (Rančić et al., 1996; Ronchi et al., 1996), based on the earlier work of Sadourny (1972). The cubed sphere geometry partitions the sphere into six identical subdomains by mapping a cube onto the spherical surface and is free from the polar singularities. The subdomains (faces of the cubed sphere) can be further partitioned into non-overlapping elements which are ideally suited for high-order Galerkin methods (Dennis et al., 2005).

The SE method was introduced by Patera (1984). The main difference between SE and the finite element methods available at the time was that Patera used high-order elements, whereas prior to his research low-order (linear or quadratic) elements were used. This led to a change in the interaction at the boundary of elements to enforce the continuity restriction. Exponential convergence of the spectral element method is seen in both Patera (1984) and Harkin (1995); an extensive review of SE methods can be found in Karniadakis and Sherwin (1999) and Deville et al. (2002). DG methods may be considered as a hybrid approach, combining finite element and finite volume methods, and became popular after the work of Cockburn and Shu (1989). They combined the classical DG method with a total-variation bounded Runge–Kutta time stepping scheme along with the DG spatial discretization. DG methods retain all the properties of SE methods, and in addition are inherently conservative. For long-term time integration conservation properties are very crucial, therefore DG methods are more attractive than SE methods in climate modeling (Nair and Tufo, 2006). A detailed review of the DG methods and their application can be found in Cockburn and Shu (2001).

Galerkin methods have been shown to efficiently scale onto large distributive memory machines. Dennis et al. (2005) implemented a DG method on $\mathcal{O}(1000)$ processes and Loft et al. (2001) implemented a SE method on $\mathcal{O}(10\,000)$ processes; results from both indicate that the methods will scale well on even larger machines. Since parallel machines continue to grow, these methods are becoming more attractive.

Advection plays a major role in atmospheric dynamics. We consider the advection equation in Cartesian and spherical geometry to evaluate the DG scheme. In this paper, the main focus is given to the DG schemes and we compare two variants of a DG method. In addition, a SE method is used as a reference for convergence and scalability in Cartesian geometry. All three solvers are introduced in detail in Section 2, with a discussion of scalability in Section 2.5. Section 3 discusses techniques for implementing a DG method on the surface of a sphere using the cubed sphere geometry.

2. Two-dimensional Galerkin methods

In two dimensions, the conservative transport equation is given as follows:

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathbf{F}(U) = S(U), \tag{1}$$

where $U = U(x^1, x^2, t)$, $(x^1, x^2) \in \Omega$, $t \in [0, T]$, $\mathbf{F}(U) = (F(U), G(U))$ is the flux function, $S(U)$ is the source term, and $\nabla = (\partial/\partial x^1, \partial/\partial x^2)$. For the given initial conditions $U(x^1, x^2, t = 0) = U_0(x^1, x^2)$ for all $(x^1, x^2) \in \Omega$, the solution $U(x^1, x^2, t = T)$ is sought. The first step in implementing a DG or SE method is to partition Ω into N_e non-overlapping elements Ω_{ij}. For this study, we consider a periodic rectangular domain (Ω) that consists of rectangular elements Ω_{ij} such that:

$$\Omega_{ij} = \{(x^1, x^2) : x^1 \in [x^1_{i-1/2}, x^1_{i+1/2}], x^2 \in [x^2_{j-1/2}, x^2_{j+1/2}]\},$$

where $i \in \{1, \ldots, N_i\}$ and $j \in \{1, \ldots, N_j\}$, so there are $N_e = N_i \times N_j$ elements tiled as shown in Fig. 1.

For the approximate solution $U_h(x^1, x^2, t)$ in a finite-dimensional function space V_h, a weak DG formulation of Eq. (1) is required. This is done by substituting U_h for U, multiplying the equation by an arbitrary test function $\phi_h = \phi_h(x^1, x^2) \in V_h$, and integrating over each element:

$$\int_{\Omega_{ij}} \frac{\partial U_h}{\partial t} \phi_h \, d\Omega + \int_{\Omega_{ij}} (\nabla \cdot \mathbf{F}(U_h)) \phi_h \, d\Omega = \int_{\Omega_{ij}} S(U_h) \phi_h \, d\Omega.$$

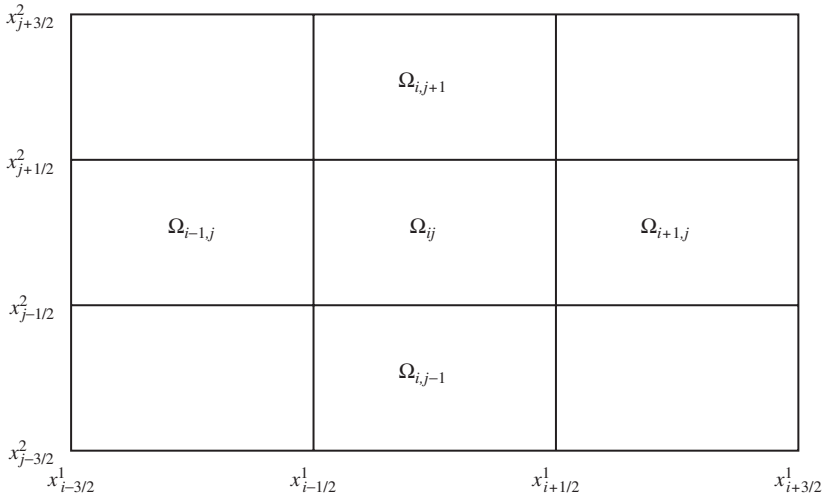

Fig. 1. Rectangular domain partitioned into elements.

Integrating the second term by parts yields:

$$\int_{\Omega_{ij}} (\nabla \cdot \mathbf{F}(U_h))\phi_h \, d\Omega = \int_{\Gamma_{ij}} (\mathbf{F}(U_h) \cdot \hat{\mathbf{n}})\phi_h \, ds - \int_{\Omega_{ij}} (\mathbf{F}(U_h) \cdot \nabla\phi_h) \, d\Omega,$$

where Γ_{ij} is the boundary of Ω_{ij} and $\hat{\mathbf{n}}$ represents the normal (outward) vector to Γ_{ij}. Thus the weak form corresponding to Eq. (1) can be rewritten as

$$\int_{\Omega_{ij}} \frac{\partial U_h}{\partial t}\phi_h \, d\Omega = \int_{\Omega_{ij}} S(U_h)\phi_h \, d\Omega + \int_{\Omega_{ij}} (\mathbf{F}(U_h) \cdot \nabla\phi_h) \, d\Omega - \int_{\Gamma_{ij}} (\mathbf{F}(U_h) \cdot \hat{\mathbf{n}})\phi_h \, ds. \quad (2)$$

In the DG method, functions in V_h are continuous on the interior of each element, but continuity across the element boundaries is not required. In this case, $\mathbf{F}(U_h)$ is not uniquely defined along element boundaries and instead is replaced with a numerical flux. A wide range of flux formulas are available (Cockburn and Shu, 2001), for simplicity we consider the Lax–Friedrichs flux given by

$$\hat{F}(U_h^-, U_h^+) = \tfrac{1}{2}[(\mathbf{F}(U_h^+) + \mathbf{F}(U_h^-)) \cdot \hat{\mathbf{n}} - \bar{\alpha}(U_h^+ - U_h^-)], \quad (3)$$

where U_h^+ and U_h^- are the right and left limits, respectively, of U_h approaching the element boundary and $\bar{\alpha}$ is an upper bound on the absolute value of the Flux Jacobian, $|\mathbf{F}'(U_h)|$. This interaction is schematically shown in Fig. 2.

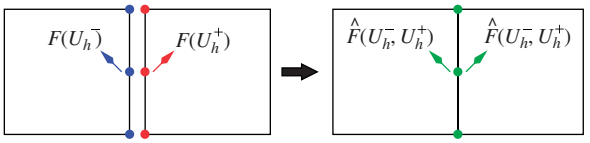

Fig. 2. Communication between element boundaries in the DG method, where the flux is the only "communicator".

The numerical flux resolves the discontinuity and provides the only mechanism by which two adjacent elements interact. Therefore, for the DG methods, parallel communication is built with the boundary flux operations (Baggag et al., 1999; Dennis et al., 2005).

Interchanging the time differentiation with the spatial integral and using the numerical flux from Eq. (3), the weak Galerkin formulation takes the form:

$$\frac{\partial}{\partial t}\int_{\Omega_{ij}} U_h\phi_h \, d\Omega = \int_{\Omega_{ij}} S(U_h)\phi_h \, d\Omega + \int_{\Omega_{ij}} (\mathbf{F}(U_h) \cdot \nabla\phi_h) \, d\Omega - \int_{\Gamma_{ij}} \hat{F}(U_h)\phi_h \, ds. \quad (4)$$

2.1. Local coordinate systems

The integrals in (4) need to be evaluated in an efficient manner, and their accuracy is crucial to

maintaining the overall order of accuracy of the Galerkin scheme. It is therefore important to have a common computational grid on which every element can be mapped, allowing numerical integrals to be calculated using Gauss quadrature rules. For rectangular elements the domain $[-1,1]\times[-1,1]$ is taken to be the *reference element* (or computational grid). Denote the Cartesian coordinate directions x^1 and x^2 as x^v, with $v\in\{1,2\}$. Let $x_i^v=\frac{1}{2}(x_{i+1/2}^v+x_{i-1/2}^v)$ and $\Delta x_i^v=x_{i+1/2}^v-x_{i-1/2}^v$, and define the local coordinates ξ^v for points in Ω_{ij} by $\xi^v=2(x^v-x_i^v)/\Delta x_i^v$ so that $\xi^v\in[-1,1]$. Thus the variables ξ^1 and ξ^2 are projections of x^1 and x^2 into the reference element.

In order to perform the numerical integration efficiently, a Gauss–Lobatto–Legendre (GLL) quadrature rule is employed on the reference element. In addition to allowing accurate integral computation, the GLL grid includes the two endpoints of the interval so points that lie on the boundary between two elements are included in the numerical grid on both elements. Details on computing the GLL nodes can be found in Appendix A.1.

It is also necessary to choose a basis set for V_h, and to properly define U_h. Two different bases for the DG method are discussed, one of which is easily adapted to the spectral element method.

2.2. Spatial discretization for the discontinuous Galerkin method

The two implementations of the DG method discussed in this section are mathematically equivalent—they differ only in the choice of basis functions for the finite-dimensional space the solution lies in. This leads to differences in calculating inner products and norms, but both bases span the same space.

2.2.1. Modal expansion

Denoting the kth-degree Legendre polynomial by $L_k(x)$, let the tensor-products $L_\ell(\xi^1)L_m(\xi^2)$ $(\ell,m\in\{0,1,\ldots,N_g\})$ form a basis for V_h. The approximate solution $U_h(\xi^1,\xi^2,t)$ can be expressed in terms of the basis functions as follows:

$$U_h(\xi^1,\xi^2,t)=\sum_{\ell=0}^{N_g}\sum_{m=0}^{N_g}\tilde{U}_{\ell m}(t)L_\ell(\xi^1)L_m(\xi^2), \tag{5}$$

where $\tilde{U}_{\ell m}$ are the spectral coefficients associated with the Legendre transform,

$$\tilde{U}_{\ell m}(t)=\frac{(2\ell+1)(2m+1)}{4}\times\int_{-1}^{1}\int_{-1}^{1}U(\xi^1,\xi^2,t)L_\ell(\xi^1)L_m(\xi^2)\,\mathrm{d}\xi^2\,\mathrm{d}\xi^1.$$

With this terminology, the left-hand side of Eq. (4) can be re-written as

$$\frac{\partial}{\partial t}\int_{\Omega_{ij}}U_h\phi_h\,\mathrm{d}\Omega=\left(\frac{\Delta x_i^1\Delta x_j^2}{4}\right)\frac{\mathrm{d}}{\mathrm{d}t}\int_{-1}^{1}\int_{-1}^{1}\sum_{\ell=0}^{N_g}\sum_{m=0}^{N_g}\tilde{U}_{\ell m}(t)L_\ell(\xi^1)\times L_m(\xi^2)\phi_h(\xi^1,\xi^2)\,\mathrm{d}\xi^2\,\mathrm{d}\xi^1,$$

where ϕ_h is an arbitrary test function in V_h, and defined to be

$$\phi_h(\xi^1,\xi^2)=L_p(\xi^1)L_q(\xi^2);\quad p,q\in\{0,1,\ldots,N_g\}.$$

Using the orthogonality property of Legendre polynomials,

$$\int_{-1}^{1}\int_{-1}^{1}L_\ell(\xi^1)L_m(\xi^2)L_p(\xi^1)L_q(\xi^2)\,\mathrm{d}\xi^1\,\mathrm{d}\xi^2=\frac{4}{(2\ell+1)(2m+1)}\delta_{\ell p}\delta_{mq},$$

where δ_{ij} is the Kronecker delta (i.e., $\delta_{ij}=1$ if $i=j$ and 0 otherwise), Eq. (4) can be further simplified and written as the ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{U}_{\ell m}(t)=M_{ij\ell m}[I_{\mathbf{F}}+I_\Gamma+I_S], \tag{6}$$

where

$$M_{ij\ell m}=\frac{(2\ell+1)(2m+1)}{\Delta x_i^1\Delta x_j^2},$$

$$I_{\mathbf{F}}=\int_{\Omega_{ij}}F(U_h)L_\ell'(\xi^1)L_m(\xi^2)\,\mathrm{d}\Omega+\int_{\Omega_{ij}}G(U_h)L_\ell(\xi^1)L_m'(\xi^2)\,\mathrm{d}\Omega,$$

$$I_S=\int_{\Omega_{ij}}S(U_h)L_\ell(\xi^1)L_m(\xi^2)\,\mathrm{d}\Omega,$$

$$I_\Gamma=-\int_{\Gamma_{ij}}\hat{F}(U_h)L_\ell(\xi^1)L_m(\xi^2)\,\mathrm{d}s.$$

Note that the *mass matrix* $\mathbf{M}_{ij}=[M_{ij\ell m}]^{-1}$ associated with the discretization in Ω_{ij} can be easily inverted, and for the global discretization in Ω, the

mass matrix $\mathbf{M}$ is block-diagonal. The surface integrals $I_{\mathbf{F}}$ and I_S and the boundary (contour) integral I_Γ appearing in the right-hand side of the ODE (6) are evaluated by employing the GLL quadrature (Nair et al., 2005b). Pre-computing the mass matrix, as well as Legendre polynomial values and its derivatives (L'_m) on the GLL grid, greatly simplifies the solution process.

2.2.2. Nodal expansion

The nodal basis set contains the N_g-degree Lagrange–Legendre polynomials, with Lagrange nodes on the GLL quadrature points ($\{\xi_0, \ldots, \xi_{N_g}\}$). Fig. 3 shows the plots of the modal and nodal basis set for $N_g = 4$. The modal basis, seen in Fig. 3a, consists of the GLL polynomials up to degree N_g (representing different modes), but the nodal basis (Fig. 3b) contains polynomials with fixed degree N_g.

The orthogonality of the nodal basis set is established in a discrete manner by exploiting a property of the Lagrange polynomials (Deville et al., 2002). The nodal basis functions are denoted as $h_i(\xi)$, $i \in \{0, \ldots, N_g\}$, and defined to be

$$h_i(\xi) = \frac{(\xi - 1)(\xi + 1)L'_{N_g}(\xi)}{N_g(N_g + 1)L_{N_g}(\xi_i)(\xi - \xi_i)}, \quad \xi \in [-1, 1].$$

Since h_i is a Lagrange polynomial it follows that $h_i(\xi_j) = \delta_{ij}$. Using the GLL quadrature discussed in Section 2.1, these polynomials form a (discrete) orthogonal basis for V_h:

$$\int_{-1}^{1} h_i(\xi)h_j(\xi)\,\mathrm{d}\xi \approx \sum_{k=0}^{N_g} w_k h_i(\xi_k)h_j(\xi_k) = w_i\delta_{ij}, \tag{7}$$

with quadrature weights w_i defined explicitly in Appendix A.1. However, the quadrature rule is only exact for polynomials of degree $2N_g - 1$ or lower, so the integration is not exact.

For 2D application the tensor-products $h_\ell(\xi^1)$ $h_m(\xi^2)$, with $\ell, m \in \{0, \ldots, N_g\}$, form a basis set. As in Section 2.2.1, on Ω_{ij} the test function and the approximate solution U_h are expressed in terms of the basis set:

$$U_h(\xi^1, \xi^2, t) = \sum_{\ell=0}^{N_g}\sum_{m=0}^{N_g} U_{\ell m}(t)h_\ell(\xi^1)h_m(\xi^2), \tag{8}$$

where $U_{\ell m}(t) = U_h(\xi_\ell, \xi_m, t)$, avoiding the need to transform from physical to spectral space. As with the modal formulation, Eq. (8) allows the left-hand side of Eq. (4) to be re-written as

$$\frac{\partial}{\partial t}\int_{\Omega_{ij}} U_h\phi_h\,\mathrm{d}\Omega = \left(\frac{\Delta x_i^1 \Delta x_j^2}{4}\right)\frac{\mathrm{d}}{\mathrm{d}t}\int_{-1}^{1}\int_{-1}^{1}\sum_{\ell=0}^{N_g}\sum_{m=0}^{N_g} U_{ij\ell m}(t)h_\ell(\xi^1) \times h_m(\xi^2)\phi_h(\xi^1, \xi^2)\,\mathrm{d}\xi^2\,\mathrm{d}\xi^1.$$

The discrete orthogonality displayed in Eq. (7) extends to two dimensions as well:

$$\int_{-1}^{1}\int_{-1}^{1} h_\ell(\xi^1)h_m(\xi^2)h_p(\xi^1)h_q(\xi^2)\,\mathrm{d}\xi^1\,\mathrm{d}\xi^2 \approx w_\ell w_m\delta_{\ell p}\delta_{mq}, \quad p, q \in \{0, \ldots, N_g\}.$$

Utilizing the above orthogonality relation and Eq. (8), the weak formulation in Eq. (4) can be simplified analogously to the modal discretization

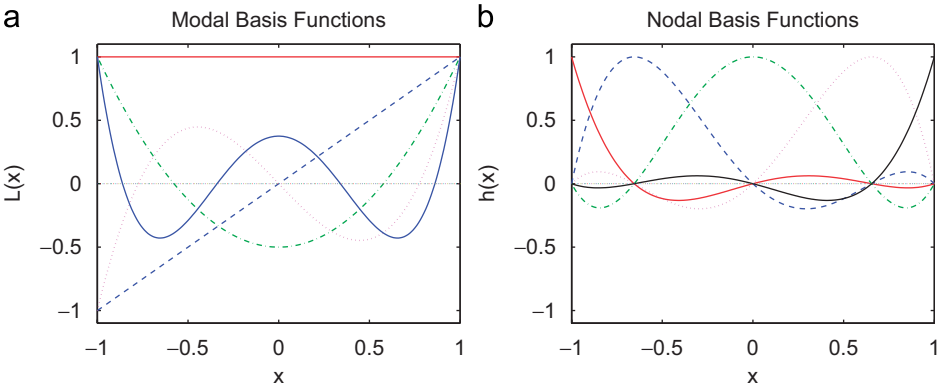

Fig. 3. Plots of: (a) the modal basis set (Legendre polynomials) having degree up to $N_g = 4$; and (b) the nodal basis (GLL–Lagrange polynomials) with degree $N_g = 4$.

in Eq. (6):

$$\frac{\mathrm{d}}{\mathrm{d}t}U_{\ell m}(t) = M_{ij\ell m}[I_{\mathbf{F}} + I_{\Gamma} + I_S],$$
$$M_{ij\ell m} = \frac{4}{w_\ell w_m \Delta x_i^1 \Delta x_j^2}, \tag{9}$$

where $M_{ij\ell m}$ form the elements of the inverted mass matrix $\mathbf{M}_{ij}$ associated with the nodal discretization in Ω_{ij}. The integrals $I_{\mathbf{F}}$, I_{Γ} and I_S appearing in Eq. (9) are similar to those defined in Eq. (6), but the Legendre polynomials L_ℓ and L_m, as well as their derivatives L'_ℓ and L'_m are replaced by Lagrange polynomials (h_ℓ and h_m) and their derivatives, respectively. This computational procedure can be simplified by pre-computing the mass matrix, and the derivatives h'_ℓ and h'_m can be computed using the algorithm provided in Appendix A.2.

2.3. Spatial discretization for the spectral element method

Unlike the DG method, the spectral element method uses an advective form to solve Eq. (1) and does not rely on conservation laws (Deville et al., 2002). However, the SE discretization is similar to that of the nodal DG method. The difference comes from discretizing Eq. (2): for the spectral element method, the space V_h (where the numerical solution is found) is restricted to include only functions that are continuous across element boundaries. Continuity is ensured by averaging the coefficients of the Lagrange polynomials that are non-zero on the shared boundaries, and the flux term in the equation is zero. It should be noted that a four-way average must be taken at corners (points where four elements come together). Fig. 4 illustrates the communication required for the SE method. Further applications and implementation details of SE methods can be found in Deville et al. (2002).

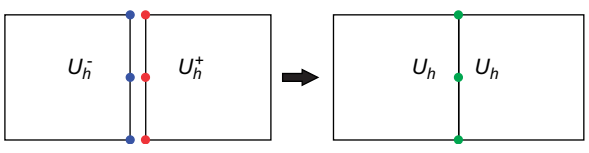

Fig. 4. Communication between element boundaries in the spectral element method. C^0 continuity is enforced along the boundaries of the element.

2.4. Time integration

The semi-discretized ODEs in Eqs. (6) and (9) can be written in the following general form:

$$\frac{\mathrm{d}}{\mathrm{d}t}U(t) = \mathscr{L}(U) \quad \text{in } (0, T),$$

where $\mathscr{L}$ denotes the spatial DG discretizations on the GLL grid. A variety of numerical integration schemes are available to solve such an equation. For the present study, we employ an explicit third-order Runge–Kutta scheme that belongs to a class of strong stability preserving (SSP) Runge–Kutta schemes (Gottlieb et al., 2001). Letting $U^n = U(t)$ and $U^{n+1} = U(t + \Delta t)$, the three-stage time integration scheme can be written in the following manner:

$$U^{(1)} = U^n + \Delta t \mathscr{L}(U^n),$$
$$U^{(2)} = \tfrac{3}{4}U^n + \tfrac{1}{4}U^{(1)} + \tfrac{1}{4}\Delta t \mathscr{L}(U^{(1)}),$$
$$U^{n+1} = \tfrac{1}{3}U^n + \tfrac{2}{3}U^{(2)} + \tfrac{2}{3}\Delta t \mathscr{L}(U^{(2)}).$$

2.5. Parallelization

Since local finite element methods partition the domain into elements with minimal interactions, the domain decomposition across parallel processes is straightforward. Each process can be assigned an element or group of neighboring elements—for the DG methods the only communication needed occurs during the flux computation stage, and the spectral element method requires communication to calculate averages across element boundaries. Therefore, in the spectral element method, each element needs to receive data from each of its eight neighbors (rather than just the four neighbors sharing an edge of the boundary).

In all of these methods, inter-element communication is computed along each of an elements boundaries three times per time step (as required by the third-order RK time integration scheme), so elements should be evenly distributed among the processes. Uneven distribution will cause processes with fewer elements to be idle while processes with more elements finish the necessary computations at each stage.

2.6. Numerical results

The test case discussed in this section is the solid-body rotation of a Gaussian hill centered at (x_c^1, x_c^2)

given by the initial condition

$$U_0(x^1, x^2) = a_0 \exp[-b_0[(x^1 - x_c^1)^2 + (x^2 - x_c^2)^2]],$$

with parameters $a_0 = 1$ and $b_0 = 5$ used in collecting data. Additionally, the test case uses a flux function of the form $\mathbf{F}(U) = (u^1 U, u^2 U)$, where (u^1, u^2) is the wind field $\mathbf{v}$, and no source term ($S(U) = 0$). Tests are run on the periodic domain $\Omega = [-\pi, \pi] \times [-\pi, \pi]$ with $\mathbf{v} = (-\pi x^2, \pi x^1)$. For convergence tests, a hill centered at $(0, 0)$ is used to maintain continuity across the periodic boundaries, so the test function is in C^0 but not C^1.

Analytically, the solution at any time t is $U(x^1, x^2, t) = U_0(x^1 \cos(\pi t) + x^2 \sin(\pi t), -x^1 \sin(\pi t) + x^2 \cos(\pi t))$, a counter-clockwise rotation of the initial condition. When $t = 2$, U_0 has made one full rotation around the origin, so $U(x^1, x^2, t = 2) = U_0(x^1, x^2)$.

Fig. 5 shows a hill centered at $(-\pi/2, 0)$ at $t = 0$ and 2, solved using the modal DG implementation.

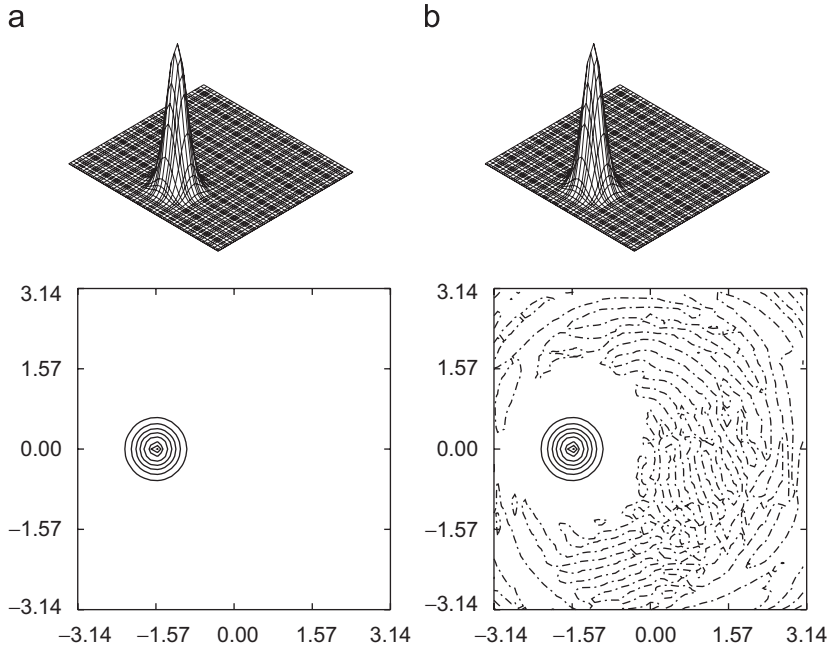

Fig. 5. Gaussian hill centered at $(-\pi/2, 0)$: (a) initially ($t = 0$) and (b) after one full revolution ($t = 2$) using the modal DG method with $N_e = 225$ and $N_g = 5$. Contours range from 0.15 to 0.99 for the hill, and zero values are shown as well (dash-dot line).

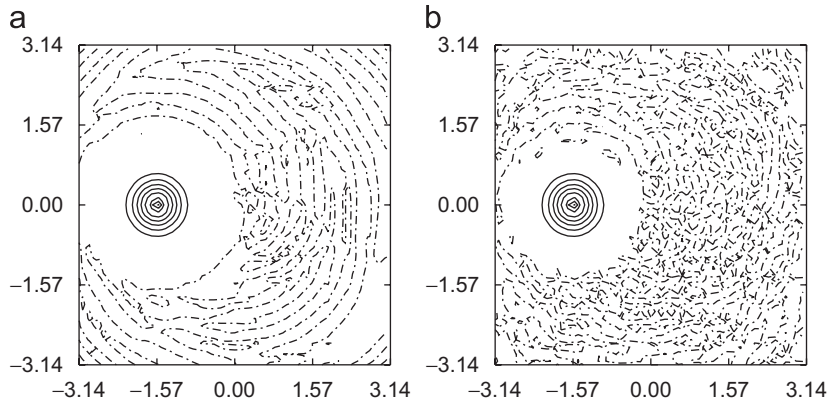

Fig. 6. Gaussian hill centered at $(-\pi/2, 0)$ after one full revolution ($t = 2$) using: (a) the nodal DG method; and (b) the spectral element method; $N_e = 225$ and $N_g = 5$. Contours range from 0.15 to 0.99 for the hill, and zero values are shown as well (dash-dot line).

The contour plot includes lines where $U = 0$, showing slight oscillations in the solution that are not noticeable in the 3D plot. Fig. 6 shows the contour plots of the numerical solutions using the nodal DG expansion and the spectral element method.

2.6.1. Monotonic limiter for discontinuous Galerkin methods

Before studying the C^0 test case, it is prudent to discuss how the DG method treats problems with discontinuous initial conditions. Consider the solid-body rotation of the characteristic function over a circle of radius r centered at (x_c^1, x_c^2). This function can be written in a piecewise manner:

$$U_0(x^1, x^2) = \begin{cases} 1 & (x^1 - x_c^1)^2 + (x^2 - x_c^2)^2 < r^2, \\ 0 & \text{otherwise.} \end{cases}$$

Fig. 7a shows the initial condition—the circle is centered at $(-\pi/2, 0)$ with a radius of $1/\sqrt{2}$. Fig. 7b shows the numerical solution at $t = 2$ using the nodal DG implementation; due to Gibbs phenomenon, there are spurious oscillations near the discontinuity. Monotonicity, however, is an important property for solutions to transport schemes commonly used in atmospheric modeling because atmospheric variables such as the relative humidity need to be both monotonic and positivity-preserving. In order to produce solutions without oscillations, monotonic limiters are often employed.

The *minmod* slope limiters are effective for second- and third-order DG methods (Cockburn and Shu, 2001). A minmod limiter reduces the order of the approximating polynomial in the elements containing the shock or discontinuity to first-order, but elsewhere the solution is not modified. Limiting the spurious oscillations for the high-order DG methods ($N_g > 3$), on the other hand, is a challenging problem and is an active research topic. Employing monotonic limiters designed for relatively low-order DG methods is appealing, but then the high-order accuracy of the solution is lost (Iskandarani et al., 2005).

The basic problem with existing limiters is that when the element order becomes higher, the ability to control spurious oscillations decreases and adversely affects the quality of the solution. High-order limiters such as the WENO class of limiters exist (Qiu and Shu, 2005); however, they are computationally expensive and do not parallelize efficiently.

Comparing and detailing different limiters are beyond the scope of this paper; a review of limiters and their properties can be found in Cockburn and Shu (2001) and Iskandarani et al. (2005). However, the DG advection combined with a WENO limiter is demonstrated using the deformational flow test-problem (idealized cyclogenesis) considered in Cheruvu et al. (2007). Using an initial condition of $U_0(x^1, x^2) = -\tanh[(x^2 - x_c^2)/\delta]$ and velocity field of $\mathbf{v} = (-\omega x^2, \omega x^1)$, the analytic solution at time t is

$$U(x^1, x^2, t) = -\tanh\left[\frac{x^2 - x_c^2}{\delta}\cos(\omega t) - \frac{x^1 - x_c^1}{\delta}\sin(\omega t)\right],$$

where $\omega = v_T/r$ is the angular velocity, $v_T = 3\sqrt{3}/2\,\mathrm{sech}^2(r)\tanh(r)$ is the normalized tangential velocity, r is the radial length of vortex, and (x_c^1, x_c^2) is the vortex center. The parameter $\delta = 0.05$ generates a non-smooth spiral-like structure as the solution (see Fig. 8).

The numerical solution was computed with the DG method and the modal basis functions on a

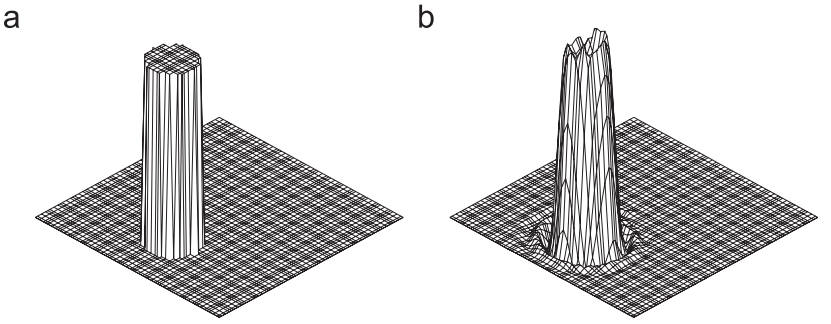

Fig. 7. Discontinuous profile centered at $(-\pi/2, 0)$: (a) initially; and (b) after one full revolution ($t = 2$) using the nodal DG method; $N_e = 225$ and $N_g = 5$.

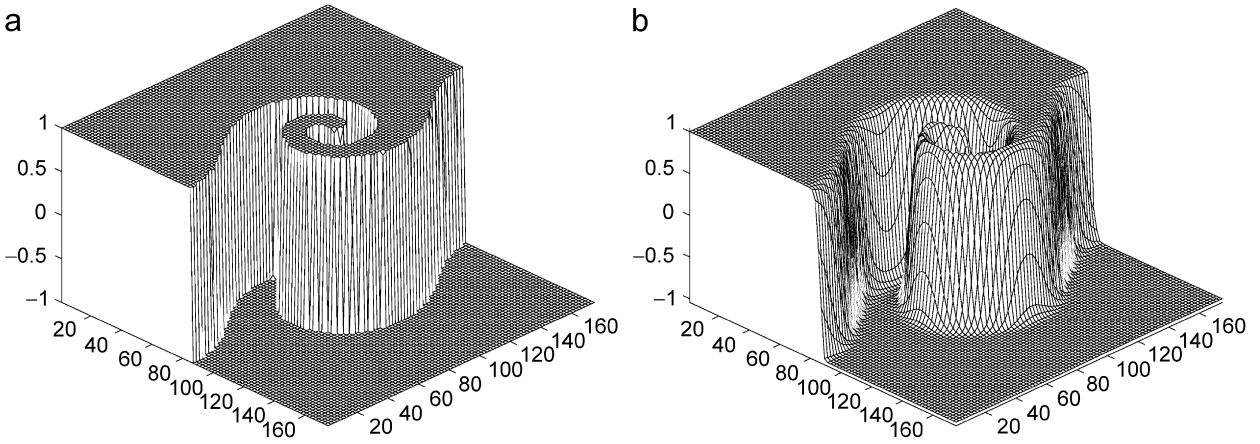

Fig. 8. (a) Analytic solution for the deformational flow problem at time $t = 3$ units; (b) corresponding numerical solution obtained by the DG method coupled with a WENO limiter.

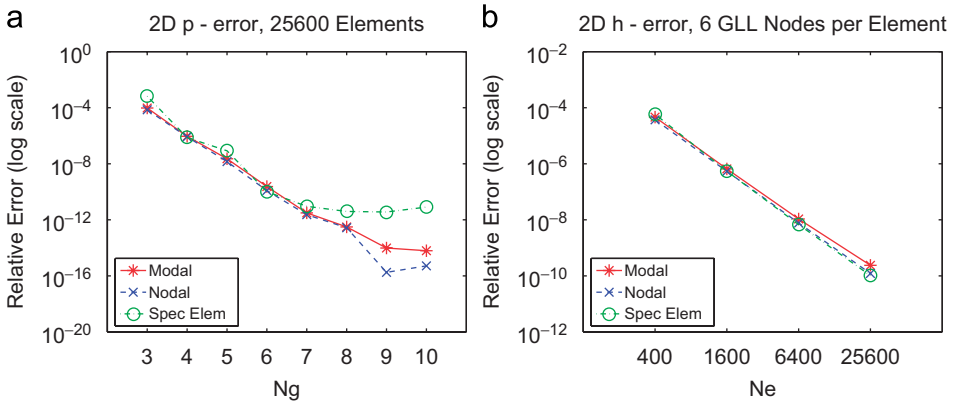

Fig. 9. (a) p-error plots for the 2D test function with $N_e = 25\,600$; (b) h-error plots for the 2D test function with $N_g = 6$.

40×40 mesh of elements, each with a 4×4 GLL grid. A third-order WENO limiter, consisting of a 5×5 element stencil, was used at every stage of RK integration (Qiu and Shu, 2005). The DG solution at time $t = 3$ is shown in Fig. 8b, and is free of any spurious oscillations. As compared to the exact solution (Fig. 8a), the numerical solution captures the vortex center but the vortex walls are smoothed.

2.6.2. Convergence

The relative L_2-error between $U(x^1, x^2, t = 2)$ and $U_0(x^1, x^2)$ is used to measure the accuracy of the numerical method. The error is given by the formula

$$\varepsilon = \left[\frac{\int_\Omega [U(x^1, x^2, 2) - U_0(x^1, x^2)]^2 \, d\Omega}{\int_\Omega [U_0(x^1, x^2)] \, d\Omega}\right]^{1/2},$$

and the integrals are calculated using the GLL quadrature on each element.

As expected, increasing the size of the grid improves the numerical solution. The way the domain is partitioned, the number of nodes can be increased by either using a finer GLL grid or by using more elements to cover the domain. The former is the p-error, and can be seen in Fig. 9a; the latter is the h-error, seen in Fig. 9b. From the charts in Fig. 9 it is clear that all three methods perform comparably on small domains—however, because our test function is not in C^1, the spectral element method levels off at error of order $\mathcal{O}(10^{-12})$, whereas the error in the DG methods leveled off at machine precision, or $\mathcal{O}(10^{-14})$.

2.6.3. Timing results

The method of parallelization discussed in Section 2.5 is ideal for distributed memory parallel computers, since all computations are done locally.

Therefore the MPI library is used to parallelize the code, and two methods are used to measure the efficiency. The first technique used is strong scaling, where the total work done remains the same as the number of processes increases, and the second technique is weak scaling, where the work done on each process remains the same as the number of processes increases. The strong scaling experiments were run on a 64 node cluster arranged in a 2D torus. Each node contained two 2.4 GHz Intel Xeon processors and 2 GB of RAM, but only one processor per node was used to increase the memory available to each process. The weak scaling experiments were run on a 1024 node BlueGene/L cluster arranged in a 3D torus. Each node contained two dual-core 700 MHz PowerPC 440 processors and 512 MB of RAM—however, one processor per node was dedicated to managing the MPI communication between nodes.

Strong scaling, plotted in Fig. 10a, is measured by speed-up, the ratio of runtime with one process to runtime with p processes. Ideally, the work will be split evenly among processes and doubling the number of processes would halve the runtime. This would result in a speed-up value of p, and is referred to as *linear speed-up*, shown as a solid black line. When the speed-up is less than p, such as the case when $N_e = 400$, it is labeled as *sublinear*. This occurs when the communication phases requires a significant amount of clock time relative to the computation phases, so processes sit idle while data is transferred. When the speed-up exceeds p, such as when $N_e = 6400$, it is *superlinear*. This occurs when dividing the work among processes allows better memory management because each process is being given less data—so a large problem may require working out of a pagefile for small process counts, but will all fit in RAM for larger counts.

Weak scaling, plotted in Fig. 10b, is measured by looking at the number of Mflops per second each process maintains. Since the amount of work is being held constant, Mflops per second is inversely proportional to runtime—while a constant Mflops per second is desired, the communication phases will result in some idle clock cycles. This is noticeable in the decrease in Mflops/proc/sec going from one process to two, compared to the fairly constant measurements for all the runs on multiple processes.

The two plots in Fig. 10 show how well this algorithm scales, especially for large problems. While there was a 16% decrease in Mflops per process per second in moving from one process to two, increasing the process count beyond two does not affect the efficiency of this code. Further, large problems require a lot of memory to store all the necessary data, and using more compute nodes allows access to more memory.

3. Cubed sphere geometry

Typically, position and the velocity vector are defined on the surface of a sphere as functions of latitude and longitude (θ and λ). However, the poles are singularities and present problems for most numerical methods. The cubed sphere avoids those issues by inscribing the sphere with a cube and using a central (gnomic) projection from the sphere to the cube.

The faces of the cube are labeled from 1 to 6 as in Fig. 11, and then the cube is oriented such that the intersection of the equator and the prime meridian

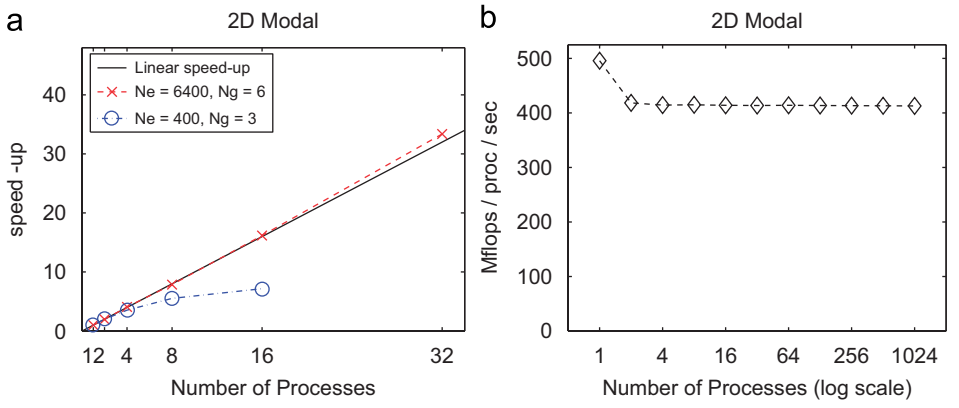

Fig. 10. Scaling plots: (a) strong scaling (20 000 timesteps with $N_e = 400$ and 40 000 timesteps with $N_e = 6400$); and (b) weak scaling (30 000 timesteps with 100 elements per process and $N_g = 6$).

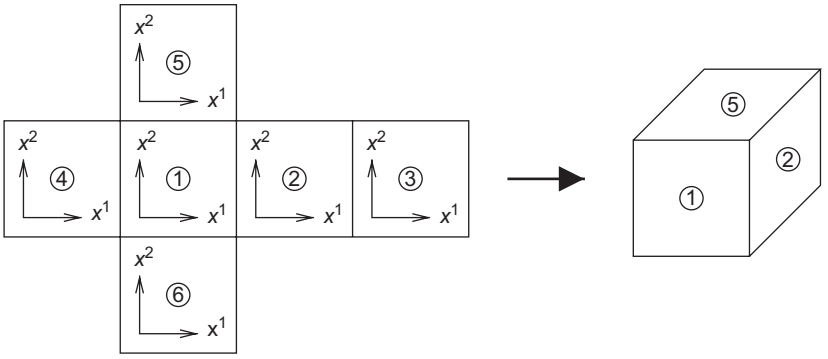

Fig. 11. Orientation of the faces of the cubed sphere and their coordinate systems.

is projected to the center of face 1, and the North Pole is projected to the center of face 5. Each face has a local Cartesian coordinate system, (x, y), with $x, y \in [-a, a]$ where $a = R/\sqrt{3}$ and R is the radius of the sphere. Previous research (Rančić et al., 1996; Nair et al., 2005a) has shown that superior results are obtained using an equi-spaced element grid on the (x^1, x^2) coordinate system, where $x = a \tan x^1$ and $y = a \tan(x^2)$ and $x^1, x^2 \in [-\pi/4, \pi/4]$.

Define $\mathbf{a}_1$ and $\mathbf{a}_2$ as the covariant base vectors $\partial \mathbf{r}/\partial x^1$ and $\partial \mathbf{r}/\partial x^2$, respectively, where $\mathrm{d}\mathbf{r} = R\cos\theta\,\mathrm{d}\lambda\,\hat{\mathbf{e}}_\lambda + R\,\mathrm{d}\theta\,\hat{\mathbf{e}}_\theta$ is a small displacement on the surface, $\hat{\mathbf{e}}_\lambda$ is the unit vector in the east–west direction, and $\hat{\mathbf{e}}_\theta$ is the unit vector in the north–south direction. Let $\mathbf{v}(\lambda, \theta)$ be the horizontal velocity on the sphere—the covariant components are given by $u_1 = \mathbf{v} \cdot \mathbf{a}_1$ and $u_2 = \mathbf{v} \cdot \mathbf{a}_2$ and the contravariant components are expressed as $\mathbf{v} = u^1\mathbf{a}_1 + u^2\mathbf{a}_2$.

The metric tensor of the transformation is defined as $g_{ij} = \mathbf{a_i} \cdot \mathbf{a_j}$. Covariant and contravariant components are related via g_{ij} by the relationships $u_i = g_{ij}u^j$ and $u^i = g^{ij}u^j$, where $g^{ij} = (g_{ij})^{-1}$.

On all six faces of the cube,

$$g_{ij} = \frac{R^2}{\rho^4 \cos^2 x^1 \cos^2 x^2} \times \begin{bmatrix} 1 + \tan^2 x^1 & -\tan x^1 \tan x^2 \\ -\tan x^1 \tan x^2 & 1 + \tan^2 x^2 \end{bmatrix}, \quad (10)$$

where $\rho = (1 + \tan^2 x^1 + \tan^2 x^2)^{1/2}$. Let $g = \det(g_{ij})$, so $\sqrt{g} = R^2/(\rho^3 \cos^2 x^1 \cos^2 x^2)$. The transformations between the contravariant velocity components and the spherical velocity components are

$$A \begin{bmatrix} u^1 \\ u^2 \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} \quad \text{and} \quad A^{-1} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u^1 \\ u^2 \end{bmatrix}$$

where $A^{\mathrm{T}}A = g_{ij}$.

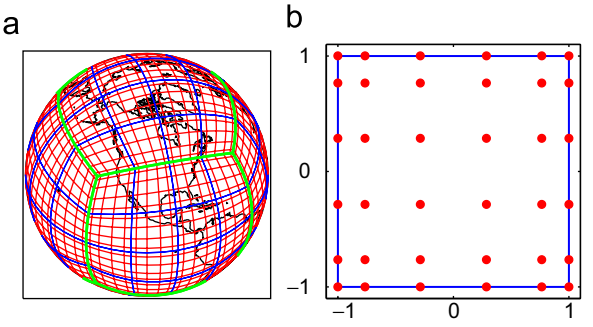

Fig. 12. (a) Earth tiled with 96 elements (blue lines), each with a 6-node GLL grid (thin red lines). Thick green lines outline the cube faces; (b) 6 × 6 GLL grid on the reference element.

A can be interpreted as the transformation matrix between the cube and the sphere, given by

$$A = R \begin{bmatrix} \cos\theta\,(\partial\lambda/\partial x^1) & \cos\theta\,(\partial\lambda/\partial x^2) \\ \partial\theta/\partial x^1 & \partial\theta/\partial x^2 \end{bmatrix}.$$

3.1. Advection on a cubed sphere

In curvilinear coordinates, without the source term, Eq. (1) takes the following form:

$$\frac{\partial U}{\partial t} + \frac{1}{\sqrt{g}}\frac{\partial}{\partial x^1}[u^1\sqrt{g}\,U] + \frac{1}{\sqrt{g}}\frac{\partial}{\partial x^2}[u^2\sqrt{g}\,U] = 0,$$

where U is the advecting field and $\mathbf{v}$ is the surface wind. Multiplying by $\sqrt{g}$, which has no time dependence, yields

$$\frac{\partial}{\partial t}[\sqrt{g}U] + \frac{\partial}{\partial x^1}[u^1\sqrt{g}\,U] + \frac{\partial}{\partial x^2}[u^2\sqrt{g}\,U] = 0. \quad (11)$$

Note that Eq. (11) looks remarkably similar to Eq. (1), advecting $\sqrt{g}\,U$ rather than U with $\mathbf{F}(\sqrt{g}\,U) = (u^1\sqrt{g}\,U, u^2\sqrt{g}\,U)$ and $S(\sqrt{g}\,U) = 0$.

Rectangular elements are arranged on the sphere, as in Fig. 12. The only computational difference between the 2D implementation and the cubed sphere implementation is in the flux calculations/SE averaging along the edges of the cube (the calculations are identical on each face). In the 2D algorithm, all the elements used the same coordinate system, while each face in the cubed sphere method has its own. For example, Fig. 13 shows the boundary shared by face 2 (an equatorial face) and face 6 (the southern face). A velocity vector given in terms of $\mathbf{a}_1$ and $\mathbf{a}_2$ on face 6 is not the same vector in terms of $\mathbf{a}_1$ and $\mathbf{a}_2$ on face 2—so for the boundary flux calculations, velocity needs to be mapped to spherical coordinates and then mapped back onto the neighboring face.

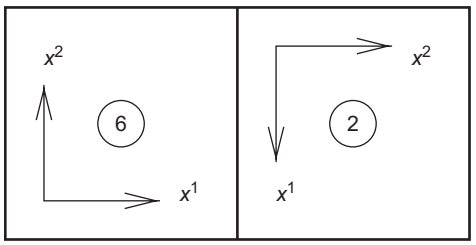

Fig. 13. Boundary between faces 2 and 6 on the sphere. x^1 and x^2 are not aligned on the two faces.

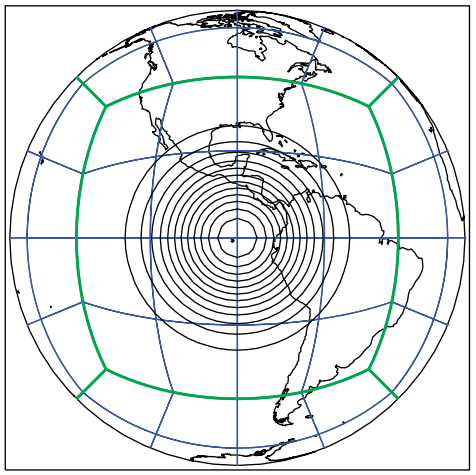

Fig. 14. Contour plot of $U_0(\lambda,\theta)$, with contours ranging from 0 to 6000.

3.2. Convergence results

The solid-body rotation of a Gaussian hill is again used as a test case. The initial condition on the surface of the sphere is defined in terms of the 3D Cartesian coordinates as

$$U_0(\lambda,\theta) = a_0 \exp[-b_0[(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2]],$$

where $a_0 = 6000$ is the amplitude of the hill, $b_0 = 10/R^2$ defines the width of the hill, and x, y, and z are related to λ and θ by

$$(x,y,z) = (R\cos\theta\cos\lambda, R\cos\theta\sin\lambda, R\sin\theta).$$

The hill is centered at $\lambda = 3\pi/2$, $\theta = 0$ ($x_c = 0$, $y_c = -R$, $z_c = 0$), as seen in Fig. 14.

Following the recommendation in Williamson et al. (1992), the wind field is defined as

$$\begin{bmatrix} u \\ v \end{bmatrix} = u_0 \begin{bmatrix} \cos\theta\cos\alpha + \sin\theta\cos\lambda\sin\alpha \\ -\sin\lambda\sin\alpha \end{bmatrix},$$

where α is the flow orientation parameter, defined as the angle between the axis of rotation and the polar axis and u_0 is the wind velocity. As in Nair et al. (2005a), let $\alpha = \pi/4$ so that the hill passes over two

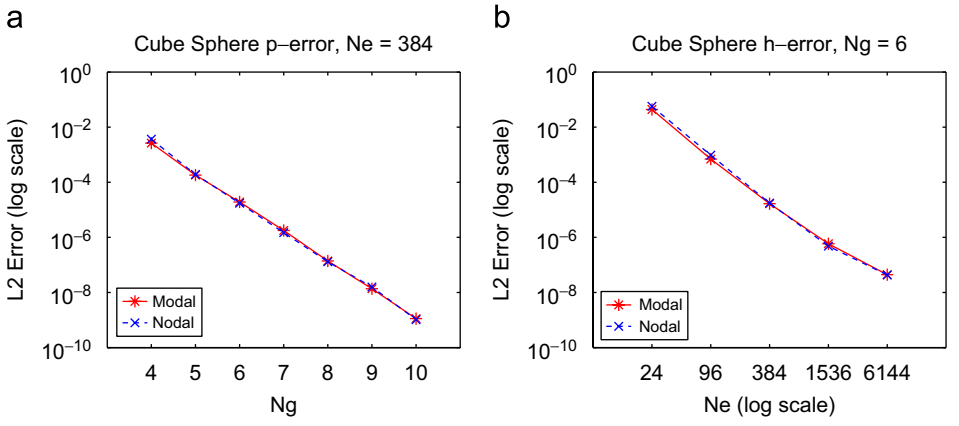

Fig. 15. (a) *p*-error and (b) *h*-error for the cubed sphere test problem.

cube edges in making a complete rotation, and let $u_0 = 2\pi R/(12\,\text{days})$, so that one full rotation takes 12 days to complete. Denoting t in units of seconds, then, it is clear that the analytic solution is $U(\lambda, \theta, t = 1\,036\,800) = U_0(\lambda, \theta)$.

As in the 2D case, Fig. 15 shows the L_2 norm of both the p-error and the h-error associated with changing the size of the grid. In this case, the relative error was computed by integrating over the sphere rather than integrating over each cube face. In either dimension, both methods show high-order convergence and produce error of the same order on identical grid sizes.

4. Summary and conclusions

Three high-order finite element methods have been compared using an explicit third-order total-variation bounded Runge–Kutta time stepping scheme and the conservative transport equation. All three methods exhibited exponential decay in the L_2 error, but at high orders, the spectral element method was not as accurate as the discontinuous Galerkin methods. For small grids, all three methods solved the equation with comparable error.

The modal and nodal discretizations of the DG method produced roughly equivalent error for a given N_e and N_g, but the computational cost was two to five times greater for the modal code (depending on the grid size) because each time step requires three transformations from spectral space to physical space and back. The spectral element method ran faster than either DG method because averaging over element boundaries is cheaper than calculating the boundary flux.

For non-smooth initial conditions, the DG methods generate spurious oscillations due to Gibbs phenomenon. The role of monotonic limiters on DG schemes was briefly discussed, demonstrating the effectiveness of the WENO limiter in removing such oscillations.

Acknowledgments

Computer time was provided by NSF ARI Grant #CDA-9601817, NSF MRI GRANT #CNS-0420873, NSF MRI Grant #CNS-0420985, NSF MRI Grant #CNS-0421498, NASA AIST Grant #NAG2-1646, DOE SciDAC Grant #DE-FG02-04ER63870, NSF sponsorship of the National Center for Atmospheric Research, the University of Colorado, and a grant from the IBM Shared University Research (SUR) program.

Additionally, the authors would like to thank two anonymous reviewers for their comments and suggestions.

Appendix A

A.1. Generating the GLL grid and weights

The (N_g+1)-node GLL grid ($\{x_0, \ldots, x_{N_g}\}$) is defined on the interval $[-1, 1]$ and contains the points $\{-1\}$ and $\{1\}$, as well as the $(N_g - 1)$ roots of the first derivative of the N_g-degree Legendre polynomial.

The Legendre polynomials are a sub-class of the family of Jacobi polynomials, $P_n^{\alpha,\beta}$—specifically, $L_{N_g}(x) = P_{N_g}^{0,0}$. It follows from the properties of Jacobi polynomials that $L'_{N_g}(x) = \lambda_{N_g} P_{N_g-1}^{1,1}/2$, where $\lambda_{N_g} = N_g(N_g + 1)$ is a constant. Therefore, the roots of $L'_{N_g}(x)$ can be found by computing the roots of $P_{N_g-1}^{1,1}$. This can be accomplished using the roots of the Chebyshev polynomial $[T_{N_g-1}(x) = P_{N_g-1}^{-1/2,-1/2}(x)]$ as initial estimates in the *Newton–Rhapson iteration* (Karniadakis and Sherwin, 1999).

Newton–Rhapson iteration: As found in Appendix B of Karniadakis and Sherwin (1999), this method is used to determine the $N_g - 1$ roots of $L'_{N_g}(x)$. Denote these roots by x_γ, with x_1 the smallest and x_{N_g-1} the largest ($x_0 = -1$ and $x_{N_g} = 1$).

Start with $r_\gamma =$ the γth-smallest root of $T_{N_g-1}(x)$, so $r_\gamma = \cos((2\gamma - 1)\pi/(2N_g - 2))$.

If $\gamma \neq 1$, then $r_\gamma = (r_\gamma + x_{\gamma-1})/2$.

Do the following until error (δ) is within tolerance or a maximum number of iterations have been exceeded:

$$s = \sum_{i=1}^{\gamma-1} 1/(r_\gamma - x_i),$$

$$\delta = -L'_{N_g}(r_\gamma)/(L''_{N_g}(r_\gamma) - sL'_{N_g}(r_\gamma)),$$

$$r_\gamma = r_\gamma + \delta,$$

$$x_\gamma = r_\gamma.$$

$L'_{N_g}(r_\gamma)$ and $L''_{N_g}(r_\gamma)$ are found via recursion relations in Appendix A of Karniadakis and Sherwin (1999).

GLL quadrature weights: GLL integration, where

$$\int_{-1}^{1} f(x)\,\mathrm{d}x = \sum_{\gamma=0}^{N_g} f(x_\gamma) w_\gamma$$

is exact when $f(x)$ is polynomial of degree $\leqslant 2N_g - 1$. The weights are defined by

$$w_\gamma = \frac{2}{(N_g + 1)N_g[L_{N_g}(x_\gamma)]^2}.$$

A.2. Calculating derivatives using a nodal expansion

In the nodal expansion of the DG algorithm, $h_i'(\xi)$ appears only in integration terms. Using a GLL quadrature, therefore, h_i' is only evaluated at the GLL nodes. Again, Karniadakis and Sherwin (1999) provides the necessary algorithm:

$$h_j'(\xi_i) = \begin{cases} \dfrac{-(N_g+1)N_g}{4}, & i=j=0, \\ \dfrac{(N_g+1)N_g}{4}, & i=j=N_g, \\ 0, & 1 \leqslant i=j \leqslant N_g-1, \\ \dfrac{L_{N_g}(\xi_i)}{L_{N_g}(\xi_j)(\xi_i-\xi_j)}, & i \neq j. \end{cases}$$

Note that the recursion relations mentioned in Appendix A.1 must be used to calculate $L_{N_g}(\xi)$.

References

Baggag, A., Atkins, H., Keyes, D., 1999. Parallel implementation of the discontinuous Galerkin method. NASA/CR-1999-209646 ICASE Report No. 99-35, 8pp.

Cheruvu, V., Nair, R.D., Tufo, H.M., 2007. A spectral finite volume transport scheme on the sphere. Applied Numerical Mathematics, 57, in press, doi:10.1016/j.apnum.2006.09.008.

Cockburn, B., Shu, C.-W., 1989. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework. Mathematics of Computation 52 (186), 411–435.

Cockburn, B., Shu, C.-W., 2001. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. Journal of Scientific Computing 16 (3), 173–261.

Dennis, J., Levy, M., Nair, R., Tufo, H., Voran, T., 2005. Towards an efficient and scalable discontinuous Galerkin atmospheric model. IEEE International Parallel & Distributed Processing Symposium, vol. 14(14), p. 257a.

Deville, M.O., Fisher, P.F., Mund, E.H., 2002. High-order Methods for Incompressible Fluid Flow, Cambridge University Press, Cambridge, 499pp.

Gottlieb, S., Shu, C.W., Tadmore, E., 2001. Strong stability preserving high-order time integration methods. SIAM Review 43, 89–112.

Harkin, A., 1995. A Legendre spectral element method for the rotational Navier–Stokes equations. Master's Thesis, Massachusetts Institute of Technology, 41pp.

Iskandarani, M., Levin, J., Choi, B.-J., Haidvogel, D., 2005. Comparison of advection schemes for high-order h–p finite element and finite volume methods. Ocean Modelling 10, 233–252.

Karniadakis, G.E., Sherwin, S.J., 1999. Spectral/hp Element Methods for CFD. Oxford University Press, New York, NY, 390pp.

Loft, R.D., Thomas, S.J., Dennis, J.M., 2001. Terascale spectral element dynamical core for atmospheric general circulation models. ACM/IEEE Supercomputing 2001 Conference, p. 26.

Nair, R.D., Tufo, H.M., 2006. A scalable high-order dynamical core for climate modeling. Proceedings of International Conference on Mesoscale Process in Atmosphere, Ocean, and Environment Systems. IMPA 2006, IITD, New Delhi, India.

Nair, R.D., Thomas, S.J., Loft, R.D., 2005a. A discontinuous Galerkin global shallow water model. Monthly Weather Review 133, 876–888.

Nair, R.D., Thomas, S.J., Loft, R.D., 2005b. A discontinuous Galerkin transport scheme on the cubed-sphere. Monthly Weather Review 133, 814–828.

Patera, A.T., 1984. A spectral element method for fluid dynamics: laminar flow in a channel expansion. Journal of Computational Physics 54, 468–488.

Qiu, J., Shu, C.-W., 2005. Runge–Kutta discontinuous Galerkin method using WENO limiters. SIAM Journal on Scientific Computing 26 (3), 907–929.

Rančić, M., Purser, R.J., Mesinger, F., 1996. A global shallow-water model using an expanded spherical cube: Genomic versus conformal coordinates. Royal Meteorological Society Quarterly Journal 122, 959–982.

Ronchi, C., Iacono, R., Paolucci, P.S., 1996. The "cubed sphere": a new method for the solution of partial differential equations in spherical geometry. Journal of Computational Physics 124, 93–114.

Sadourny, R., 1972. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. Monthly Weather Review 100 (2), 136–144.

Taylor, M., Tribbia, J., Iskandarani, M., 1997. The spectral element method for the shallow water equations on the sphere. Journal of Computational Physics 130, 92–108.

Washington, W.M., Parkinson, C.L., 2005. An Introduction to Three-dimensional Climate Modeling. University Science Books, Sausalito, CA, 353pp.

Williamson, D.L., Drake, J.B., Hack, J.J., Jakob, R., Swarztrauber, P.N., 1992. A standard test set for numerical approximations to the shallow water equations in spherical geometry. Journal of Computational Physics 102, 211–224.