Optimized Icosahedral Grids: Performance of Finite-Difference Operators and Multigrid Solver

ROSS P. HEIKES, DAVID A. RANDALL, AND CELAL S. KONOR

Department of Atmospheric Science, Colorado State University, Fort Collins, Colorado

(Manuscript received 14 August 2012, in final form 22 April 2013)

ABSTRACT

This paper discusses the generation of icosahedral hexagonal–pentagonal grids, optimization of the grids, how optimization affects the accuracy of finite-difference Laplacian, Jacobian, and divergence operators, and a parallel multigrid solver that can be used to solve Poisson equations on the grids. Three different grid optimization methods are compared through an error convergence analysis. The optimization process increases the accuracy of the operators. Optimized grids up to 1-km grid spacing over the earth have been created. The accuracy, performance, and scalability of the multigrid solver are demonstrated.

1. Introduction

All early global atmosphere models used longitude–latitude grids. Near the poles, the grid spacing on such a grid is much larger in the meridional direction than the zonal direction, and the horizontal area associated with a grid cell is much smaller than at lower latitudes. The small zonal grid spacing drastically reduces the size of the maximum time step that is compatible with linear computational stability. These issues are generically referred to as the ''pole problem.'' Longitudinal filters can be applied near the poles to allow longer time steps (e.g., Arakawa and Lamb 1977), but the filters have undesirable side effects and do not scale well to high resolution on parallel computers.

Spectral methods were used by Bourke et al. (1977) to eliminate the pole problem for rapidly propagating inertia–gravity waves. Spectral models suffer from the ''Gibbs phenomenon'' induced by sharp topography (Hoskins 1980). In addition, spectral advection does not work well for positive-definite scalars (e.g., Williamson and Rasch 1994), and in recent years spectral advection has been almost universally replaced by grid-based advection schemes.

Corresponding author address: Dr. Ross Heikes, Department of Atmospheric Science, Colorado State University, Fort Collins, CO 80523-1371.
E-mail: ross@atmos.colostate.edu

DOI: 10.1175/MWR-D-12-00236.1

Alternative grids have also been proposed to solve the pole problem. A quasi-spherical design derived from the icosahedron was used by Walther Bauersfeld in 1923 for the construction of the dome of the Zeiss planetarium located in the city of Jena, Germany. Later, Buckminster Fuller constructed a series of ''geodesic'' domes (Rothman, 1989). Icosahedral grids were first introduced to the earth sciences in the pioneering work of Vestine et al. (1963). A few years later, Sadourny et al. (1968) and Williamson (1968) independently proposed the use of icosahedral grids for atmospheric models. They took different approaches, however. Sadourny et al. (1968) proposed icosahedral hexagonal–pentagonal (IHP) grids (Fig. 1a), and Williamson (1968) proposed icosahedral triangular grids (Fig. 1b). The Nonhydrostatic Icosahedral Atmospheric Model (NICAM; Satoh et al. 2008), the models developed at Colorado State University (CSU; Heikes and Randall 1995a,b, hereafter HR95a,b, respectively; Randall et al. 2002), the operational global numerical weather prediction model of the German Weather Service (GME; Majewski et al. 2002), and Flow-Following Finite-Volume Icosahedral Model (FIM; Lee and MacDonald, 2009) all use IHP grids. Further discussion of IHP grids is given by Miura and Kimoto (2005) and Thuburn (1997). The icosahedral triangular grid has been used in Icosahedral Nonhydrostatic General Circulation Model (ICON; Bonaventura et al. 2005; Bonaventura and Ringler 2005; Wan 2009), although the most recent version of the triangular grid that allows regional grid refinement is not based on an

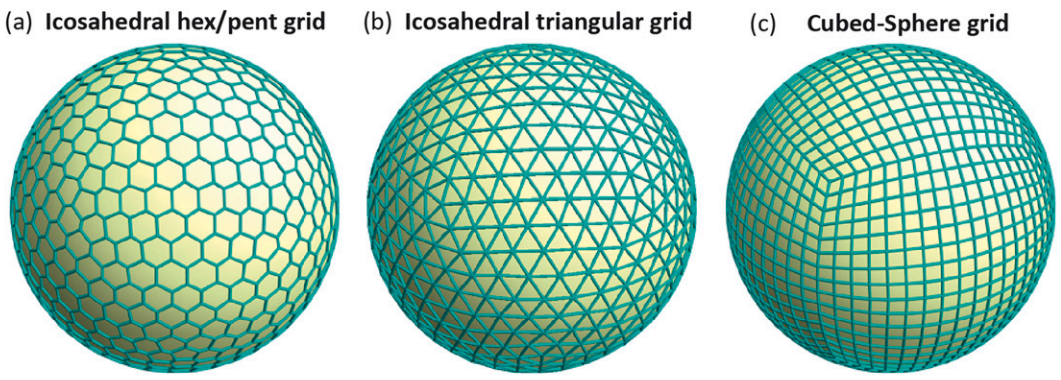
(a) Icosahedral hex/pent grid
(b) Icosahedral triangular grid
(c) Cubed-Sphere grid

FIG. 1. (a) Icosahedral hexagonal/pentagonal grid, (b) icosahedral triangular grid, and (c) cubed-sphere grid.

icosahedron. Another version of ICON with an IHP grid is under development (B. Stevens, 2011 and 2012, personal communication). The Model for Prediction Across Scales (MPAS; Skamarock et al. 2012) can be configured to use an IHP grid, although it has been designed with more general grids in mind.

Sadourny (1972) also proposed the cubed-sphere grid (Fig. 1c), which is widely used today (Rancic et al. 1996; Ronchi et al. 1996). Versions of it have been implemented at the Massachusetts Institute of Technology (MIT; Adcroft et al. 2004), Geophysical Fluid Dynamics Laboratory (GFDL; Putman and Lin 2007), and the National Center for Atmospheric Research (NCAR; Mishra et al. 2011). The cubed-sphere grid can be thought of as a spherical version of the planar Cartesian grid, and schemes designed for the Cartesian grid can easily be adapted to the cubed-sphere grid.

Additional grids are discussed in the recent review article by Staniforth and Thuburn (2012).

In this paper, we explain the motivation for and describe the construction of a family of optimized IHP grids. We discuss grid generation and optimization, the convergence of three important operators as the grid spacing is reduced, and the solution of global elliptic problems. A key goal of our work has been to enable the use of very high-resolution IHP grids on massively parallel computers. We explore the convergence properties of three finite-difference operators on successively finer versions of the grids. We also discuss the computational performance of a parallel multigrid solver that uses the grids.

In many respects, this paper is an updated and extended version of HR95a,b, although the icosahedral grid presented here is not twisted hemispherically along the equator[1] as in HR95a. The optimization algorithm presented here, which we call tweaking, is a modified version of the method used by HR95b. We have modified the cost function to obtain more accurate operators. We compare the tweaking optimization with a version of the spring dynamics optimization introduced by Tomita et al. (2001, 2002) and with a version of the centroidal voronoi tessellation (CVT) introduced by Du et al. (1999) and Ju et al. (2011).

The paper is organized as follows. Section 2 presents a comparison of properties of the Cartesian, icosahedral, and triangular grids on a plane. The generation of the icosahedral grid is discussed in section 3. We then present analyses of the convergence of finite-difference Laplacian, Jacobian, and divergence operators on the unoptimized grid. In section 4, we compare three different optimization algorithms, and present error analyses for the optimized grids. Section 5 presents the two-dimensional multigrid solver and discusses convergence and scaling to very fine grids. A summary and conclusions are presented in section 6.

This paper is the first of a sequence. The second paper describes and presents results from a shallow-water model and a three-dimensional hydrostatic model, both based on the grid described here. The third paper describes a three-dimensional nonhydrostatic generalization of the model. The grids and codes will be made available as supplemental material to the second paper.

2. A comparison of three planar grids

We first consider idealized planar grids, which are free from variations in the size, shape, and orientation of the grid cells. As is well known, only three regular polygons tile the plane: equilateral triangles, squares, and hexagons. Figure 2 shows planar grids made up of each of these three possible polygonal elements.

On the triangular and square grids, some of the neighbors of a given cell lie directly across cell walls, while

[1] While the hemispheric twisting results in symmetry across the equator, it is a complication that we now choose to avoid.

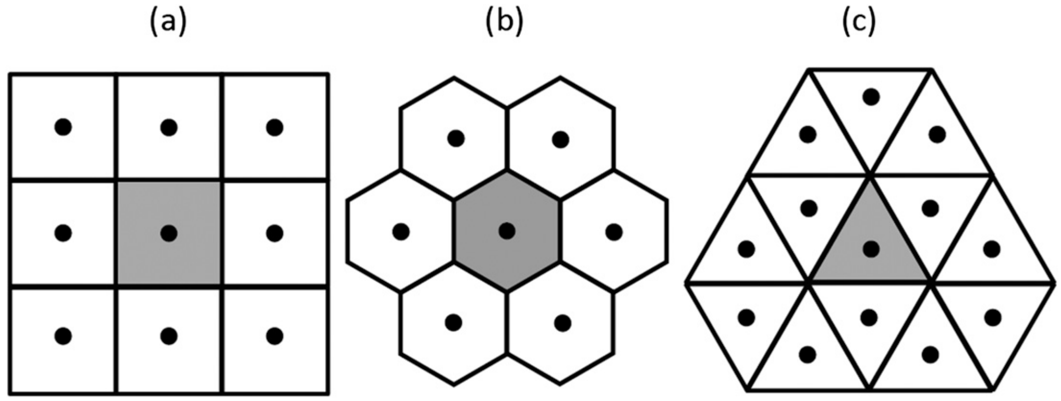

FIG. 2. (a) Cartesian, (b) hexagonal, and (c) triangular grids on a plane.

others lie across cell vertices. As a result, finite-difference operators constructed on these grids tend to use ''wall neighbors'' and ''vertex neighbors'' in different ways. For example, the simplest second-order finite-difference approximation to the gradient, on a square grid, uses only wall neighbors; vertex neighbors are ignored. It is certainly possible to construct finite-difference operators on square grids (and triangular grids) in which information from all neighboring cells is used [e.g., the Arakawa Jacobian, as discussed by Arakawa (1966)], but the operators can be cumbersome.

Hexagonal grids, in contrast, have the property that all neighbors of a given cell lay across cell walls; there are no vertex neighbors. In this sense, hexagonal grids are quasi-isotropic. As a result, the most natural finite-difference operators on hexagonal grids treat all neighboring cells in the same way; the discrete operators are as symmetrical and isotropic as possible.

On the other hand, both square grids and triangular grids can be nested, but hexagonal grids cannot because it is not possible to construct large hexagons from smaller hexagons. This can be viewed as a disadvantage of hexagonal grids. A second disadvantage is that, for a given number of grid cells per unit area, the distance between hexagon centers is slightly larger than the distances between the centers of triangles or squares.

With triangular and hexagonal grids, there can be an imbalance in the number of centers, walls, and corners. In particular, hexagonal grids have twice as many cell corners as cell centers and 3 times as many cell walls as cell centers. If the variables of a model are distributed over the centers, walls, and corners, computational modes can easily occur.

This mismatch in the degrees of freedom can be avoided by using ''hexagonal A'' staggering, in which the mass and the horizontal wind vector are both predicted at cell centers, but this again permits computational modes, due to averaging in the mass-convergence term of the continuity equation and the pressure-gradient force term of the momentum equation. NICAM (Satoh et al. 2008) and FIM (Lee and MacDonald, 2009) use grids like this. In global cloud resolving simulations with NICAM, it is observed that there are a high number of single-gridcell clouds, which is consistent with the existence of the computational mode associated with the A-grid (H. Muira 2010, personal communication).

With hexagonal C staggering, in which the mass is defined at cell centers and the normal components of the winds are predicted on the cell walls, the horizontal wind has three degrees of freedom for each degree of freedom in the mass; this permits a computational mode in the wind field. Although there are ways to minimize the amplitudes and deleterious effects of computational modes (see Thuburn 2008), they can be avoided altogether by the use of the ''Z staggering.''

With the Z staggering, in which the mass, the vertical component of the vorticity, and the horizontal divergence are defined at cell centers without staggering, there is no room for computational modes (Randall 1994). Additional advantages of the Z staggering are that it gives a good simulation of geostrophic adjustment (Randall 1994), and that direct prediction of the vorticity facilitates the use of an accurate form of the discrete vorticity equation.

We choose an IHP grid because of its good homogeneity and isotropy, and we are then led to choose Z staggering because it avoids computational modes while giving a good simulation of geostrophic adjustment and enabling an accurate form of the vorticity equation. The price we pay for using Z staggering is that we must solve a pair of Poisson equations on each time step. The method that we use for this is discussed in section 5. We use the Z-grid staggering in the hydrostatic and nonhydrostatic models to be discussed in the forthcoming papers mentioned above.

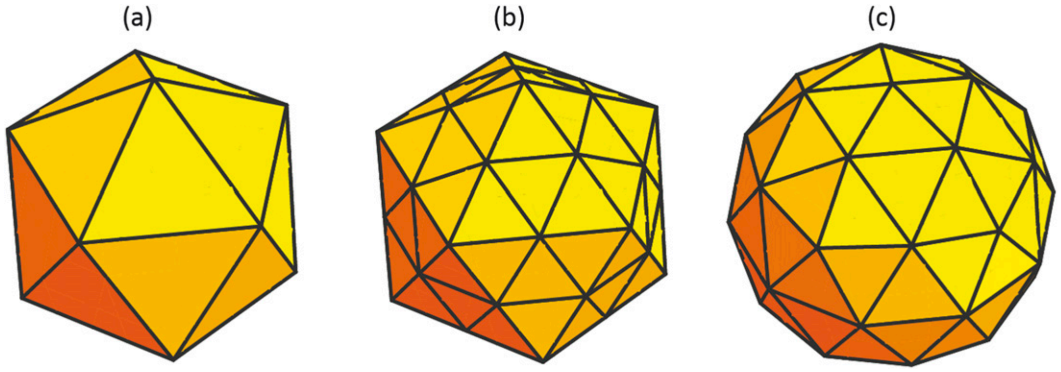

FIG. 3. Bisections and projections during the generation of the raw grid. (a) An icosahedron inscribed in the sphere. (b) Bisection of each edge of the icosahedron to form four new triangular faces. (c) Projection of the new vertices onto the sphere.

3. Grid generation

a. The "raw" grid

The icosahedron has 20 triangular faces and 12 vertices (Fig. 3a). Five triangles come together at each vertex. The icosahedron can be inscribed inside a sphere, which can represent the (approximately) spherical Earth. The icosahedron can be oriented so that two of its vertices align with Earth's North and South Poles of rotation.

In our grid-generation algorithm, each face of the icosahedron is subdivided into four new triangular faces by bisecting the edges of the triangles (Fig. 3b).[2] The vertices of the new triangles are projected onto the surface of the sphere, creating 80 new triangular faces (Fig. 3c). This bisection procedure can be repeated as many times as necessary to obtain the desired resolution. The edges of the triangular faces are then projected to the surface of the sphere, where they define spherical triangles.

We call the (spherical) triangular grid generated through this process the raw grid. The resolution of the raw grid can be specified by the number of recursive bisections needed to generate it. For example, grids 0, 1, and 2 have 12, 42, and 162 vertices, respectively. If N and G are the number of grid cells and the grid level, respectively, we can write

$$N = 5 \times 2^{2\times(G-1)+3} + 2. \quad (3.1)$$

Using this terminology, the raw grid shown in Fig. 3c corresponds to $G = 1$, which we will shorten to G1.

[2] Trisection and various other strategies are also possible but will not be discussed here.

b. From the raw grid to the IHP grid

The grid cells of our unoptimized IHP grid are the Voronoi cells centered at the vertices of the corresponding raw grid. The Voronoi cell associated with a particular vertex of the raw grid is, by definition, the set of all points closer to that vertex than to any other vertex. It follows that the corners of the Voronoi cells, called Voronoi corners, are equidistant from three or more vertices—three in the case of the IHP grids (and four in the case of a Cartesian grid). For discussions of Voronoi grids on the sphere, see Okabe et al. (2000), Augenbaum and Peskin (1985), and HR95a. The dual grid associated with the hexagonal and pentagonal Voronoi cells consists of Deluanay triangles (e.g., Bonaventura and Ringler 2005). The model discussed in this paper does not make use of the Delaunay triangles.

In Fig. 4a, the vertices of the raw grid are represented by the blue dots. Consider the spherical triangle formed by connecting three neighboring vertices of the raw grid, as represented by the green lines in Fig. 4a. The corners of the Voronoi cells are represented by the green dots in Fig. 4a. As mentioned above, these corners are equidistant from the three neighboring vertices. The locations of the Voronoi corners can be determined using Eq. (8) of HR95a. The IHP grid cells are constructed by connecting neighboring Voronoi corners with segments of great circles, as shown by the blue lines in Fig. 4b. The areas of the grid cells are determined taking into account that they are patches of a sphere. Twelve pentagonal grid cells are generated, with their centers at the 12 vertices of the original icosahedron. All of the other cells are hexagonal. The pentagons and hexagons are unavoidably distorted in the construction of an IHP grid, so that they have slightly varying shapes and sizes.

We use the term "arc" to denote a segment of a great circle. The Voronoi cells have the important property

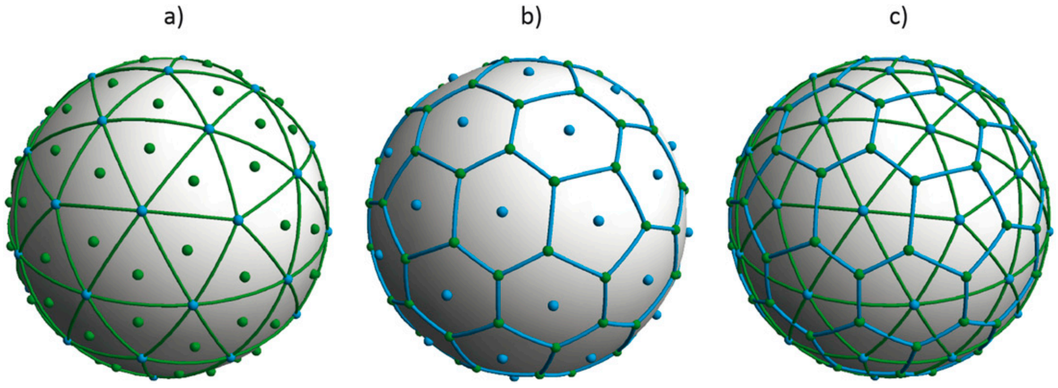

FIG. 4. Generation of an IHP grid from a raw grid for the case of G1 resolution. (a) The blue dots are the vertices of the raw grid. The Voronoi corners, represented by the green dots, are equidistant from the three neighboring vertices of the raw grid, and so lie at the centers of the triangular cells of the raw grid. (b) The hexagonal–pentagonal cells of the icosahedral grid are obtained by connecting the Voronoi corners. The dots are the cell centers. (c) The triangular cells of the raw grid and the hexagonal and pentagonal cells of the IHP grid are both shown. The vertices of the raw grid correspond to the centers of the IHP grid cells.

that the ‘‘cell walls’’ (i.e., the arcs connecting the Voronoi corners) are perpendicular bisectors of the intersecting arcs, called ‘‘grid segments,’’ which connect the centers of the two grid points that they separate. The converse is not true, however; as can be seen in Fig. 4c, the grid segments do not necessarily bisect the cell walls. This has important consequences, which are discussed in section 3c and in section 4.

As discussed above, our grid is truly spherical: we use great circles to determine the distances between points on the IHP grid, and the curved surface of the sphere to determine the cell areas. This ensures that the shape and total area of the grid are independent of resolution. If two geodesic grids of different resolutions are used to represent the atmosphere and Earth's surface (i.e., the ocean and the land surface), the total surface area seen by the atmosphere grid is the same as the total area seen by the surface grid. This ensures, for example, that the upward flux of energy or mass or momentum from the surface grid can exactly match the corresponding upward flux of energy entering the atmosphere from below.

The alternative to our spherical grid can be called a ‘‘faceted’’ grid, in which the linear distances between the points on the grid are used instead of great-circle distances, and the planar areas between the corner points are used to determine the cell areas, as if the sphere was replaced by a faceted jewel. A faceted grid converges toward a spherical grid as the resolution increases.

For a technical reason explained in section 4c, we use the spherical grid up to G11 and faceted grids for G12 and G13. At such high resolutions, the differences between the faceted and spherical grids are very minor.

There are three important differences between the planar hexagonal and spherical IHP grids. (i) The regular hexagons have uniform shapes and sizes, while the spherical IHP grid has a mixture of hexagons and pentagons of varying sizes. The ratio of the smallest to largest cell sizes is about 78% on the raw grid with medium and high resolutions. (ii) The distances between the grid points are uniform on the regular hexagonal grid, but not on the spherical IHP grid. (iii) The grid segments bisect the cell walls on the regular hexagonal grid, but not on the spherical IHP grid. Recall that, because of the use of Voronoi corners, the arcs connecting the corners are already perpendicular bisectors of the grid segments on the spherical IHP grid, as on the regular hexagonal grid. In section 4, we will show that the spherical IHP grid can be optimized to yield improved error convergence properties.

c. *Convergence of selected finite-difference operators on the raw grid*

We now analyze the truncation errors of three second-order finite-difference operators on the spherical IHP grid:

$$\text{Laplacian operator: } (\nabla^2\beta)_0 \equiv \frac{1}{A_0}\sum_{i=1}^{n} d_{0,i}\left(\frac{\beta_i-\beta_0}{\ell_{0,i}}\right), \tag{3.2}$$

Jacobian operator:

$$[J(\alpha,\beta)]_0 \equiv \frac{1}{A_0}\sum_{i=1}^{n} d_{0,i}\frac{\alpha_i+\alpha_0}{2}\left[\frac{(\beta_C)_i-(\beta_C)_{i-1}}{d_{0,i}}\right] \tag{3.3}$$

and

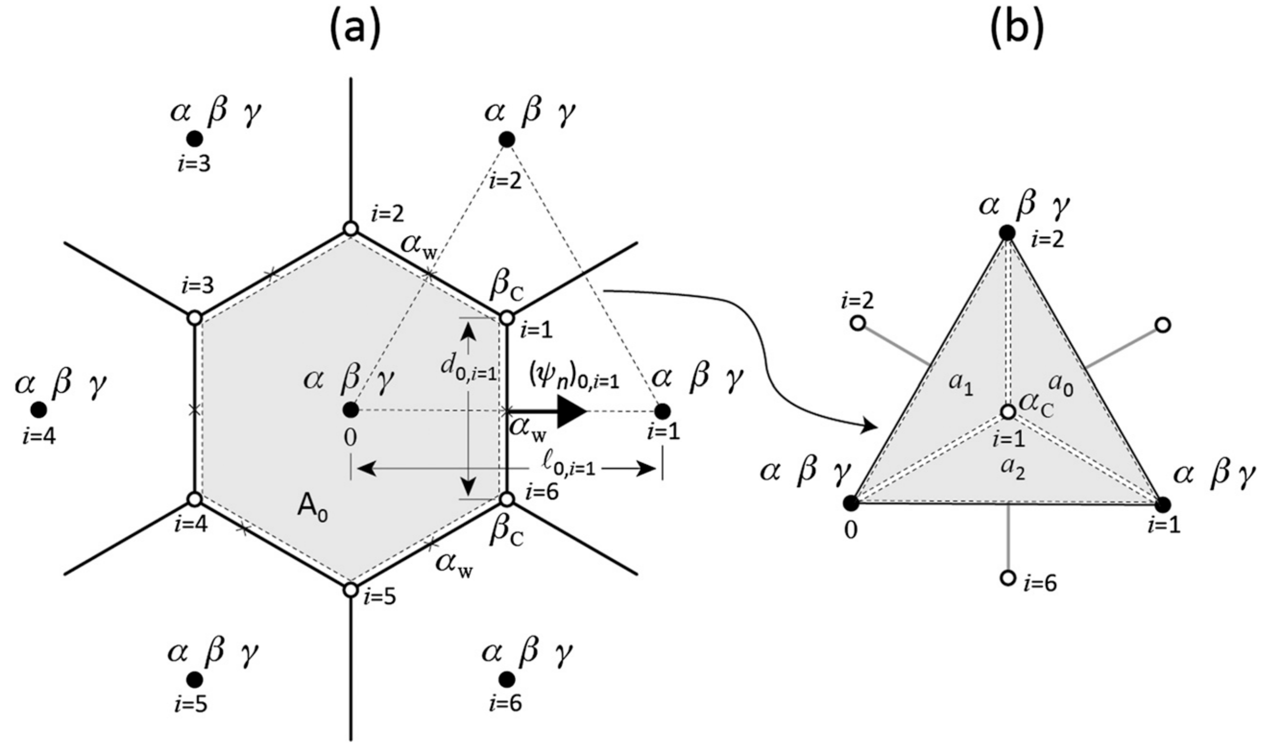

FIG. 5. An illustration of the portion of the hexagonal grid, grid indexing, and distribution of variables for (a) a hexagonal cell and (b) a triangunal region at the corner.

Divergence operator:

$$[\nabla \cdot (\alpha \boldsymbol{\Psi})]_0 \equiv \frac{1}{A_0} \sum_{i=1}^{n} d_{0,i} (\alpha_w)_{0,i} (\psi_n)_{0,i} . \qquad (3.4)$$

In Eq. (3.2), β is an arbitrary two-dimensional scalar expressed at cell centers, as shown in Fig. 5a; A_0 is the area of the cell 0; n is the number of edges (or walls) and corners of the polygon, $n = 6$ and $n = 5$ denote hexagons and pentagons, respectively; $d_{0,i}$ is the length of the arc edges separating cells 0 and i; and $\ell_{0,i}$ is the length of the arcs connecting the grid points 0 and i (i.e., the grid distance). In Eq. (3.3), α is another two-dimensional arbitrary scalar expressed at cell centers as shown in Fig. 5a, and $(\beta_C)_i$ is the value of β interpolated to the cell corners using

$$(\beta_C)_i \equiv \frac{a_0 \beta_0 + a_i \beta_i + a_{i+1} \beta_{i+1}}{a_0 + a_i + a_{i+1}} . \qquad (3.5a)$$

In Eq. (3.5a), a_i is the area of the triangular region shown in Fig. 5b. The expression on the right-hand side of Eq. (3.5a) interpolates β from the three surrounding centers to the corner by fitting a two-dimensional linear function of space passing through the three cell centers (Fig. 5b).

An alternative to Eq. (3.5a) is

$$(\beta_C)_i \equiv \frac{\beta_0 + \beta_i + \beta_{i+1}}{3} , \qquad (3.5b)$$

which was proposed by Sadourny et al. (1968), and Masuda and Ohnishi (1986). The Jacobian given by Eq. (3.5b) satisfies the conditions $J(\alpha, \beta) = -J(\beta, \alpha)$, $\overline{\alpha J(\alpha, \beta)} = 0$, and $\overline{\beta J(\alpha, \beta)} = 0$, where an overbar denotes a global average. With these properties, the discrete Jacobian conserves enstrophy and kinetic energy for two-dimensional nondivergent flow (Arakawa 1966). The alternative form, Eq. (3.5a), does not have the same conservation properties, but has higher local accuracy, as shown later in this section.

Now consider the divergence operator. In Eq. (3.4), $\boldsymbol{\Psi}$ is an arbitrary vector, $(\psi_n)_{0,i}$ is the component of $\boldsymbol{\Psi}$ that is normal to the cell wall separating cells 0 and i, and $(\alpha_w)_{0,i}$ is the interpolated value of α from the cell centers to walls, for which we use $(\alpha_w)_{0,i} \equiv (1/2)(\alpha_i + \alpha_0)$ for this case. Positive values of the normal vector point out of the cells as shown in Fig. 4a, and $(\psi_n)_{0,i} \equiv -(\psi_n)_{i,0}$ according to this definition. We define $(\psi_n)_{0,i}$ as a gradient given by $(\psi_n)_{0,i} \equiv (\beta_i - \beta_0)/\ell_{0,i}$.

In these analyses, we prescribe test functions $\alpha(\lambda, \varphi) \equiv \cos^3 \varphi \sin 5\lambda$, where λ and φ are the longitude and latitude, respectively, and $\beta(\lambda, \varphi) \equiv -(a^2/2) \cos^3 \varphi \cos 3\lambda$, where a is the radius of the earth. The true solutions are obtained by analytically computing the Laplacian, Jacobian, and divergence of these functions. For each operator at each grid point, the truncation error is computed as the difference between the analytical and numerical solutions. The L_2- and L_∞-norm errors, which are also called the RMS and maximum errors, respectively, are calculated for each operator using the resolutions from G4 to G13, and the results are plotted in Fig. 6. We define $L_2 \equiv [(1/A_{\mathrm{Earth}})\sum_{i=1,N} A_i(\phi_i^{\mathrm{approx}} - \phi_i^{\mathrm{true}})^2]^{1/2}$, where $A_{\mathrm{Earth}} \equiv \sum_{i=1,N} A_i$, N is the number of cells on the sphere and ϕ is an arbitrary variable, and $L_\infty \equiv \underset{i=1,N}{\mathrm{Max}}(|\phi_i^{\mathrm{approx}} - \phi_i^{\mathrm{true}}|)$. Note that the interpolation formula given by Eq. (3.5a) is used in the calculations of the Jacobian in these analyses, unless otherwise specified. Figures 6a,c show that the L_2-norm (or RMS) errors with the Laplacian and Jacobian operators follow a first-order convergence. The L_2-norm errors with the divergence operator follow a second-order convergence for low resolutions and a first-order convergence for high resolutions. The L_∞-norm (or maximum) errors for all these operators show no convergence at all, as shown in Figs. 6b,d,f. The hexagonal grid on a plane gives second-order convergence with the same operators.

4. Grid optimization

We now compare the convergence properties of three different grid-optimization algorithms. The first is the "tweaking" algorithm introduced by HR95b, the second is the "spring dynamics" algorithm introduced by Tomita et al. (2001, 2002), and the third is CVT introduced by Du et al. (1999) and Ju et al. (2011). A discussion of the various aspects of grid optimization can also be found in Miura and Kimoto (2005). We have created tweaked grids up to G13, which has a grid spacing of approximately 1 km on the earth. Optimization of high-resolution grids is computationally expensive and requires highly efficient codes, but in principle it only has to be done once. We are making both the tweaking codes and the tweaked grids available to the community. The discussion below focuses on the performance of the finite-difference Laplacian, Jacobian, and divergence operators with the optimized grids.

a. The tweaking algorithm

The tweaking optimization algorithm was introduced by HR95b, and is briefly described in the appendix of this paper. The goal of the optimization is to minimize the distance λ between the midpoint of the cell wall and the point where the grid segment intersects the cell wall, by displacing the grid points (or cell centers) relative to those of the raw grid (see Fig. 7). The algorithm tries to globally minimize a cost function, which is proportional to $(\lambda/d)^4$. Since $\lambda = 0$ for the planar hexagonal grid (section 3b), we can say that the tweaking algorithm tries to globally minimize the difference (iii) between the spherical and planar hexagonal grids as discussed at the end of section 3b. This point will be made clear with discussion in the next two paragraphs.

Table 1 shows some basic properties of the tweaked and unoptimized IHP grids. (The properties of the unoptimized IHP grids are shown in parentheses.) Through the tweaking the ratio of the smallest to the largest grid sizes (fifth column) and the maximum of λ/d (last column), where d is the length of the grid wall, is made closer to one. However, the tweaking does not improve the ratio of the shortest to longest grid distances (fourth column). The grid distances on the tweaked grid are slightly less uniform than those on the raw grid.

The L_2- and L_∞-norm errors for each operator on the tweaked grid are shown in Fig. 8. Truncation errors are reduced overall compared to the unoptimized grid. Moreover, error convergence rates with the tweaked grid are substantially improved. The L_2-error (or mean error) convergence rate of the divergence operator is almost second order, and it is close to second order for the Laplacian and Jacobian operators. The L_∞-error (or maximum error) convergence rate of the three operators is at least first order.

The Jacobian results presented in Fig. 8 are obtained by using the interpolation Eq. (3.5a). In Fig. 9, we present the L_2- and L_∞-norm error analyses for the Jacobian operator using the interpolation equation in Eq. (3.5b), which maintains the important conservation properties mentioned above. The L_2-norm error converges through a first-order slope, as shown in Fig. 9a, while its counterpart converges through a nearly second-order slope (Fig. 8c). The L_∞-norm error does not indicate any convergence with Eq. (3.5b), as shown in Fig. 9b, while its counterpart converges through a nearly first-order slope (Fig. 8d). Since the L_∞-norm error does not satisfy either the minimum accuracy requirement or consistency, we do not use the conservative Jacobian operator in the divergence equation. The conservative Jacobian expressed by the combination of Eqs. (3.3) and (3.5b) may produce inaccurate solutions of the divergence equation. Note that we use a highly accurate enstrophy-damping advection scheme in the vorticity equation instead of the Jacobian operator.

The spring dynamics algorithm is discussed in detail by Tomita et al. (2001, 2002), Tomita and Satoh (2004), and Satoh et al. (2008). The primary purpose of this optimization is to homogeneously distribute the grid points throughout the globe. The algorithm determines the steady state solution of the dynamical system through

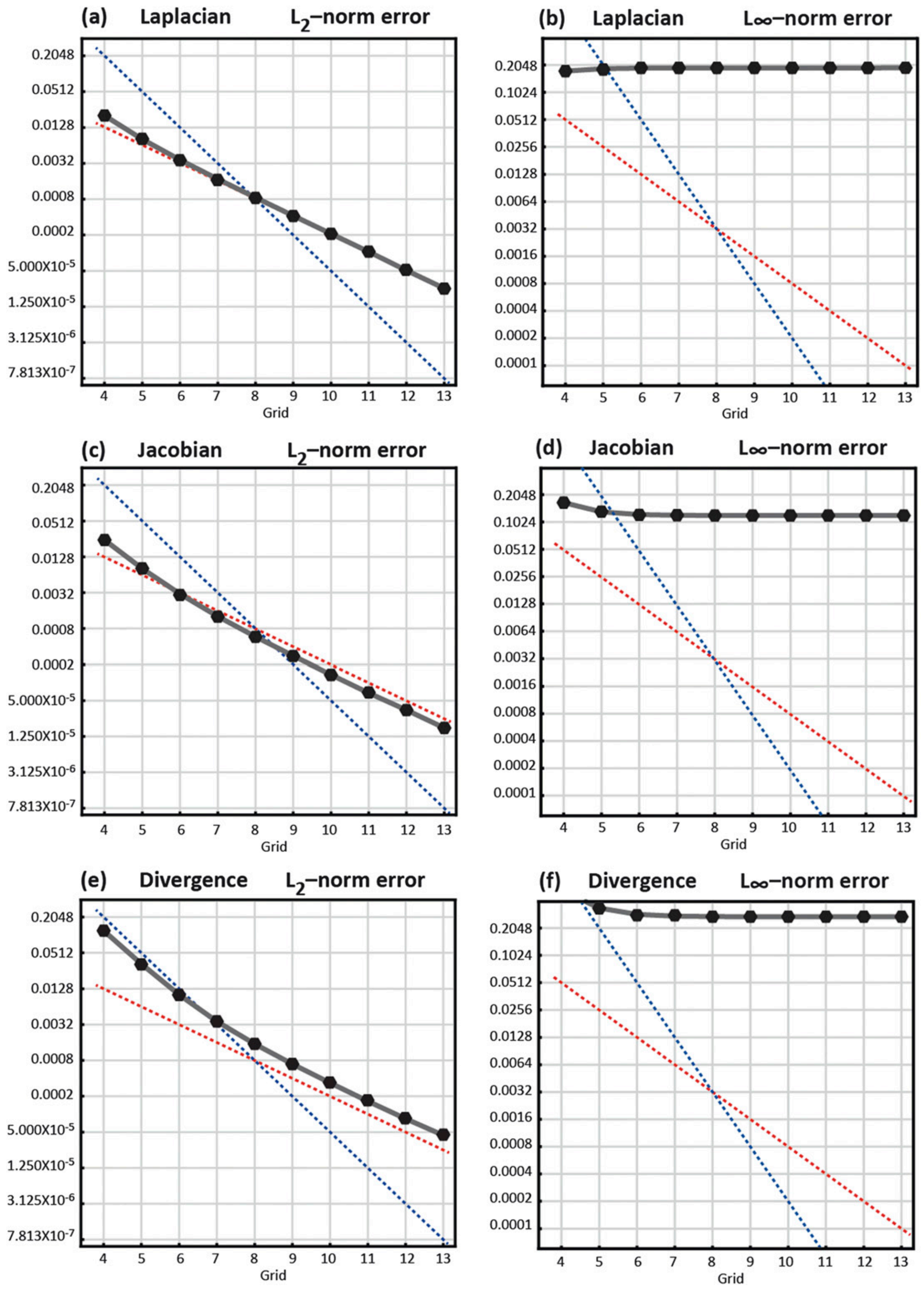

FIG. 6. (left) L_2-norm and (right) L_∞-norm errors for (a),(b) Laplacian; (c),(d) Jacobian; and (e),(f) divergence operators on the raw grid. Red and blue dashed lines indicate the slope of first- and second-order convergences, respectively. The Jacobian operator uses the interpolation given by Eq. (3.5a).

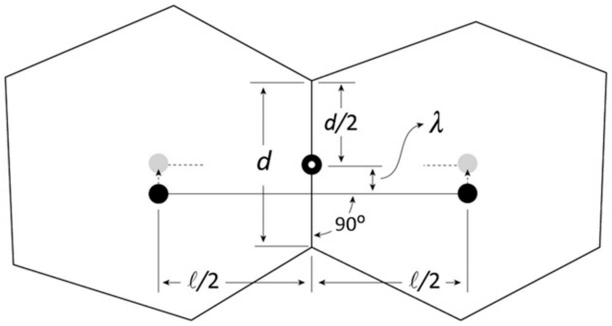

FIG. 7. An illustration of the tweaking algorithm on a couple of neighboring cells. The cell centers (solid black circles) are moved to their new positions (gray circles) to satisfy $\lambda = 0$. The cell wall already bisects the line connecting the cell centers at a right angle because of the use of Voronoi corners.

a numerical integration, in which the grid points on the raw grid are connected to each other with springs. Nonuniform horizontal resolution can be achieved by allowing the "spring constant" to vary in space, and this is a major motivation for the approach. Here we consider only spatially uniform spring constants.

We repeated the calculations described by Tomita et al. (2001) and Tomita et al. (2002) with the two values of the tuning parameter, $\kappa = 0.8$ and $\kappa = 1.1$, which correspond to the tuning parameter β in the original papers. The notation is changed to avoid confusion with β used in section 3c. Larger values of κ give more homogeneous distributions of grid points. It appears that there is a practical upper limit for κ; our algorithm was stable only up to $\kappa = 1.1$, while the highest value of κ used by Tomita et al. (2001, 2002) was 1.2. We stopped the integrations when the maximum displacement of grid points between the time steps becomes less than or equal to 0.3×10^{-4} m. After the spring grid points were located, Tomita and colleagues selected the centroids of the triangular regions as the cell corners. In our implementation, on the other hand, the cell corners were obtained by using the Voronoi principle, as for the tweaked grid.

Table 2 shows some basic properties of the spring grid, obtained using $\kappa = 1.1$ and raw grids up to G12. Through the spring dynamics, the ratio of the smallest to the largest grid sizes (fifth column) and the ratio of the shortest to the longest grid distances (fourth column) do not change significantly. Although not shown here, the cell sizes are distributed much more smoothly on the spring grid than on the unoptimized and tweaked grids, which are shown in Table 1. There is an improvement in the maximum of λ/d (last column) compared to the raw grid although the improvement is not as great as that obtained by tweaking.

Figure 10 shows L_2- and L_∞-norm errors for each operator on the spring grid, obtained with $\kappa = 1.1$ and $\kappa = 0.8$. We apply the spring dynamics optimization up to G10, which is sufficient for a comparison of the results with those from the raw and tweaked grids. Truncation errors are reduced overall, compared to the raw grid, with both $\kappa = 1.1$ and $\kappa = 0.8$. The L_2-error (or mean error) convergence rate of the divergence operator is almost second order, and is between the first and second orders for the Laplacian and Jacobian operators. There is a small improvement in the mean error for $\kappa = 1.1$, relative to $\kappa = 0.8$. The L_∞-error (or maximum error) convergence rate of the three operators is less than first order, but it is still quite a bit better than the convergence rate on the raw grid. Compared to $\kappa = 0.8$, the use of $\kappa = 1.1$ appears more effectively reduce the maximum errors and the convergence rates, although it has little effect on the mean error. The convergence rates are

TABLE 1. Some properties of the tweaked and raw grids. The raw grid properties are shown in the parentheses. Averaged grid distance is the arithmetic average of the maximum and minimum of grid distances.

Grid	No. of grid points N	Avg grid distance ℓ (km)	Ratio of shortest to longest grid distance (%)	Ratio of smallest to largest grid size (%)	Max of λ/d (%)	Avg λ/d (%)
G0	12	6699.1	100 (100)	100 (100)	0.0 (0.0)	0.0 (0.0)
G1	42	3709.8	88.1 (88.1)	88.5 (88.5)	9.9714 (9.9714)	5.0061 (5.0061)
G2	162	1908.8	82.0 (84.8)	91.6 (84.2)	5.8020 (9.9718)	3.6172 (3.6700)
G3	642	961.4	79.8 (83.9)	94.2 (76.3)	3.0933 (9.6888)	2.0437 (2.1255)
G4	2562	481.6	79.0 (83.7)	94.8 (74.1)	1.6020 (9.6758)	1.0699 (1.1363)
G5	10 242	240.9	78.7 (83.6)	95.0 (73.6)	0.8168 (9.6726)	0.5447 (0.5867)
G6	40 962	120.4	78.6 (83.6)	95.2 (73.4)	0.4128 (9.6718)	0.2743 (0.2980)
G7	163 842	60.2	78.6 (83.6)	95.2 (73.4)	0.2075 (9.6714)	0.1375 (0.1501)
G8	655 362	30.1	78.6 (83.6)	95.3 (73.4)	0.1041 (9.6715)	0.0688 (0.0753)
G9	2 621 442	15.0	78.6 (83.6)	95.3 (73.4)	0.0522 (9.6715)	0.0344 (0.0377)
G10	10 485 762	7.53	78.6 (83.6)	95.3 (73.4)	0.0260 (9.6715)	0.0172 (0.0189)
G11	41 943 042	3.76	78.6 (83.6)	95.3 (73.4)	0.0131 (9.6715)	0.0086 (0.0094)
G12	167 772 162	1.88	78.6 (83.6)	95.3 (73.4)	0.0065 (9.6715)	0.0043 (0.0047)
G13	671 088 642	0.94	78.6 (83.6)	95.3 (73.4)	0.0056 (9.6715)	0.0021 (0.0023)

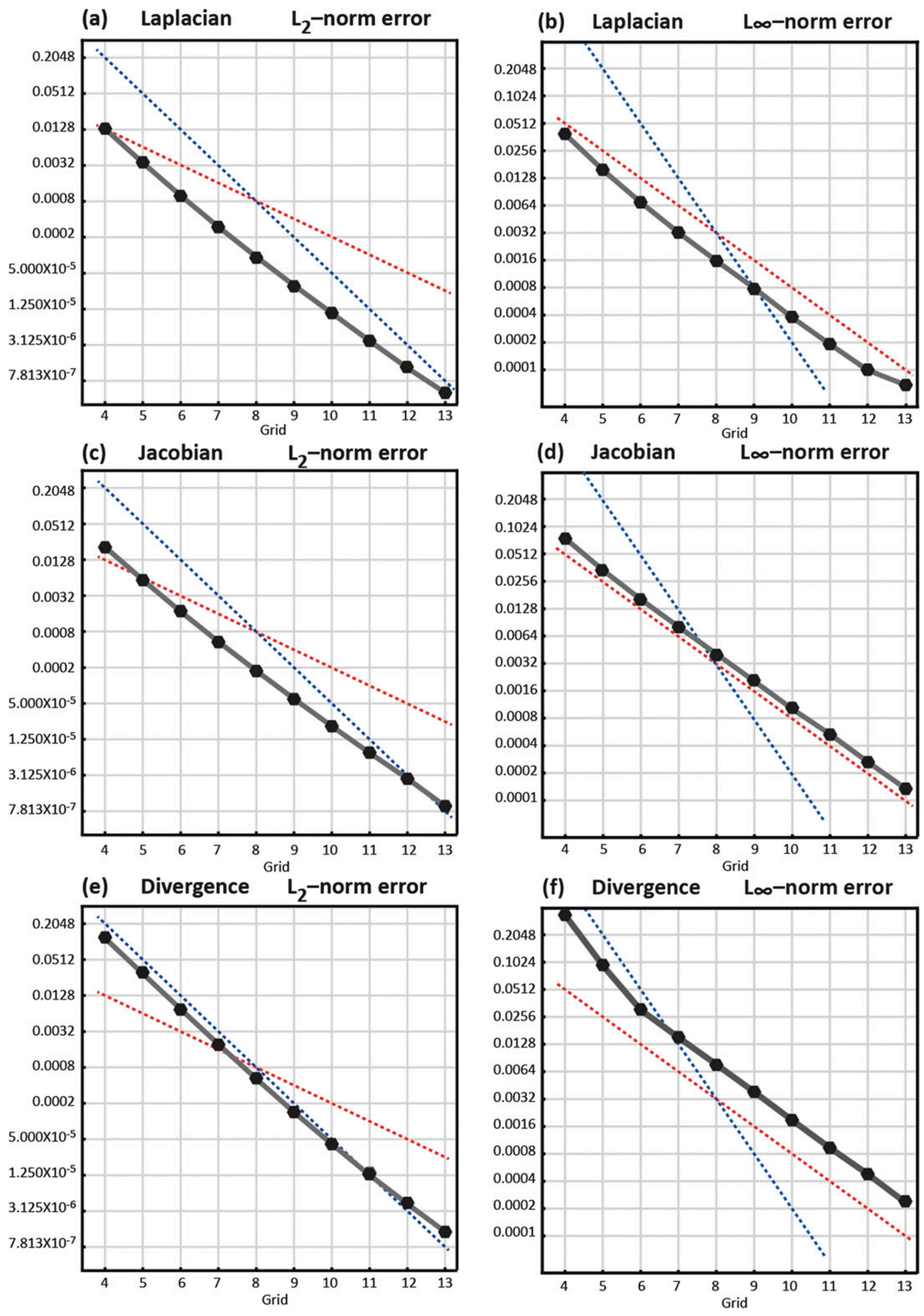

FIG. 8. As in Fig. 6, but on the tweaked grid.

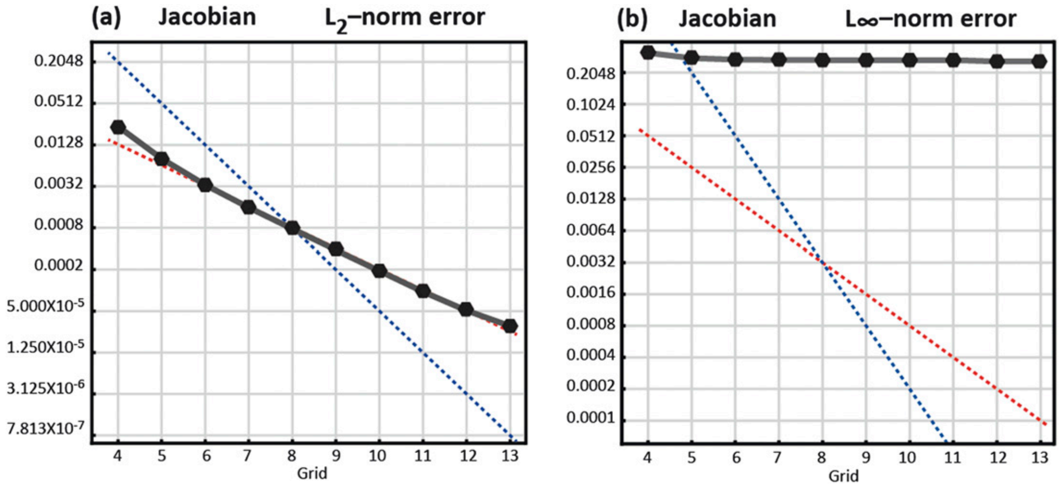

FIG. 9. As in Fig. 7, but for the Jacobian operator with the interpolation given by Eq. (3.5b).

slower overall than those obtained with the tweaked grid.

b. Comparison of tweaking with the CVT algorithm

The CVT algorithm is discussed in detail by Du et al. (1999), and its application to a sphere is described by Ju et al. (2011). The primary purpose of the CVT optimization is to make the centers of the Voronoi cells coincide with their centroids (or barycentric centers), using an iterative procedure. Unlike the tweaking algorithm, the CVT algorithm does not involve a global constraint.

We generated our CVT grids using an algorithm based on Lloyd's method as described by Ju et al. (2011). We then confirmed that our CVT grids are virtually identical to those generated by Ju et al. (2011) through the comparison of the grid statistics tabulated in Table 10.1 of Ju et al. (2011). Table 3 shows some basic properties of the CVT grids and unoptimized IHP grids. (The properties of the unoptimized grids are given in parentheses.) Through the CVT algorithm, the maximum values of λ/d are greatly improved relative to those of the unoptimized grid. However, the optimization does not improve the ratio of the shortest to longest grid distances (fourth column). The grid distances on the optimized grid are slightly less uniform than those on the unoptimized IHP grid.

Figure 11 shows L_2- and L_∞-norm errors for each operator on CVT grids up to G10, which is sufficient for a comparison of the results with those from the unoptimized IHP, as well as tweaked and spring grids. Truncation errors are reduced overall, compared to the unoptimized grid. These error characteristics are comparable to those obtained by the spring grid with $\kappa = 1.1$. The L_2-error (or mean error) convergence rate of the divergence operator is almost second order, and is between the first and second orders for the Laplacian and Jacobian operators. The L_∞-error (or maximum error) convergence rates of the three operators are less than first order, but still quite a bit better than on the unoptimized grid. The convergence rates are lower overall

TABLE 2. As in Table 1, but for the spring grid obtained with $\kappa = 1.1$ for resolutions up to G10.

Grid	No. of grid points N	Avg grid distance ℓ (km)	Ratio of shortest to longest grid distance (%)	Ratio of smallest to largest grid size (%)	Max of λ/d (%)	Avg λ/d (%)
G0	12	6699.1	100 (100)	100 (100)	0.0 (0.0)	0.0 (0.0)
G1	42	3937.6	88.1 (88.1)	88.5 (88.5)	9.9714 (9.9714)	5.0061 (5.0061)
G2	162	2033.5	85.7 (84.8)	84.5 (84.2)	8.7992 (9.9718)	3.1710 (3.6700)
G3	642	1016.5	84.3 (83.9)	81.4 (76.3)	7.9470 (9.6888)	1.6847 (2.1255)
G4	2562	509.3	82.7 (83.7)	78.6 (74.1)	7.4475 (9.6758)	0.8558 (1.1363)
G5	10 242	254.6	81.5 (83.6)	78.6 (73.6)	6.8878 (9.6726)	0.4327 (0.5867)
G6	40 962	127.3	80.5 (83.6)	74.5 (73.4)	6.2919 (9.6718)	0.2181 (0.2980)
G7	163 842	63.6	79.5 (83.6)	73.1 (73.4)	5.6742 (9.9714)	0.1095 (0.1501)
G8	655 362	31.8	78.8 (83.6)	72.1 (73.4)	5.0305 (9.6715)	0.0548 (0.0753)
G9	2 621 442	15.9	78.2 (83.6)	71.4 (73.4)	4.3481 (9.6715)	0.0274 (0.0377)
G10	10 485 762	7.96	78.8 (83.6)	70.9 (73.4)	3.6060 (9.6715)	0.0137 (0.0189)

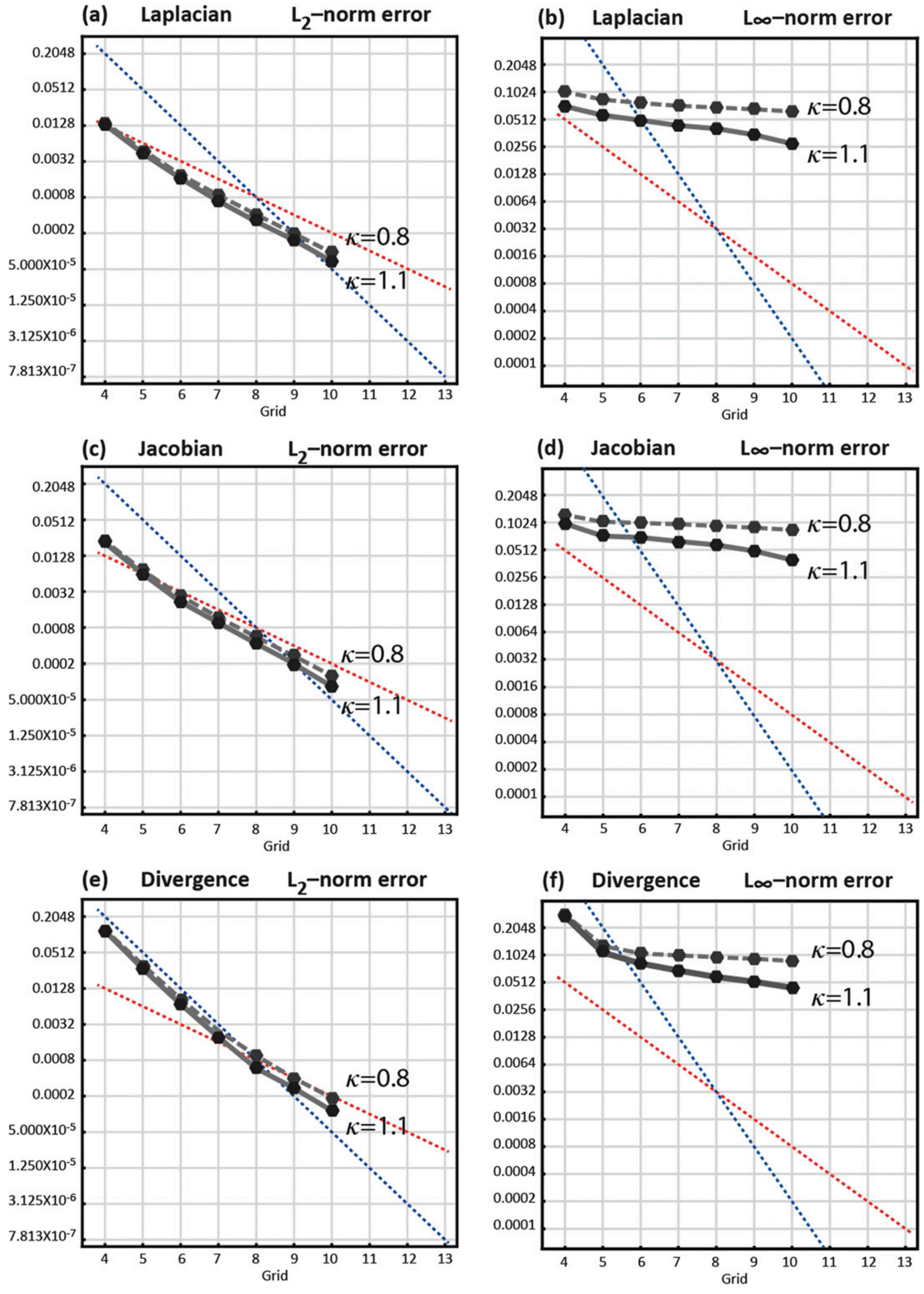

FIG. 10. As in Fig. 6, but for the spring grid obtained using $\kappa = 1.1$ (solid lines) and $\kappa = 0.8$ (dashed lines). Note that κ corresponds to b in Tomita et al. (2001) and Tomita et al. (2002).

TABLE 3. As in Table 1, but for the CVT up to G10.

Grid	No. of grid points N	Avg grid distance ℓ (km)	Ratio of shortest to longest grid distance (%)	Ratio of smallest to largest grid size (%)	Max of λ/d (%)	Avg λ/d (%)
G0	12	6699.1	100 (100)	100 (100)	0.0 (0.0)	0.0 (0.0)
G1	42	3709.8	88.4 (88.1)	88.5 (88.5)	9.9714 (9.9714)	5.0061 (5.0061)
G2	162	1905.8	85.8 (84.8)	83.9 (84.2)	9.1214 (9.9718)	3.1710 (3.6700)
G3	642	959.2	81.0 (83.9)	79.5 (76.3)	8.7016 (9.6888)	1.6847 (2.1255)
G4	2562	480.4	81.0 (83.7)	75.1 (74.1)	8.7649 (9.6758)	0.8558 (1.1363)
G5	10 242	240.3	78.6 (83.6)	70.6 (73.6)	8.8001 (9.6726)	0.4327 (0.5867)
G6	40 962	120.1	76.3 (83.6)	66.5 (73.4)	8.8093 (9.6718)	0.2181 (0.2980)
G7	163 842	60.0	74.1 (83.6)	62.6 (73.4)	8.8115 (9.9714)	0.1095 (0.1501)
G8	655 362	30.0	71.8 (83.6)	59.0 (73.4)	8.8120 (9.6715)	0.0548 (0.0753)
G9	2 621 442	15.0	69.4 (83.6)	55.4 (73.4)	8.8121 (9.6715)	0.0274 (0.0377)
G10	10 485 762	7.50	67.2 (83.6)	50.1 (73.4)	8.8121 (9.6715)	0.0137 (0.0189)

than those obtained with the tweaked grid, but comparable to those of the spring grid.

c. Concluding remarks on grid optimization

The results presented above demonstrate that icosahedral grids must be optimized in order to obtain good convergence properties. Tweaking yields a more uniform cell size distribution than the spring dynamics and CVT optimization. Tweaking tends to place the largest hexagonal cells close to the smallest cells, which are the pentagons. As mentioned above, the Voronoi principle is used to locate the cell corners in all optimization algorithms. It appears that the tweaking algorithm yields the best overall error convergence properties, of the three optimizations tested here. CVT optimization produces "well optimized" grids, in the sense that the error convergence rates are almost second order, away from the pentagons, but it fails at and near the pentagons, where the maximum errors are large and do not converge with increasing resolution.

The tweaking algorithm is the most computationally demanding of the three methods, but of course the optimization only has to be performed once. We are making the FORTRAN-90 tweaking code and the tweaked grid data up to G13 available to the community as supplements to this paper. We are also providing the code that generates the raw and unoptimized IHP grids.

With the G12 and G13 grids, we used faceted grids, as defined in section 3b, because the inverse cosine function used to generate the G12 and G13 spherical grids required the use of very high precision (128 bits) for accurate results. This made the optimization of the spherical G12 and G13 grids impractical because of the required CPU time and memory size. We overcame the difficulty by the use of faceted G12 and G13 grids generated with 64-bit precision. This technical problem can undoubtedly be resolved in the future. We repeat that we have used spherical grids for resolutions up to and including G11.

5. Multigrid solver

a. Description of the solver

We have developed a parallel multigrid solver that works on our geodesic grids. In our models, the solver is used to solve the Poisson equations to obtain the streamfunction and velocity potential from the predicted vorticity and divergence, respectively. The multigrid and conjugate gradient methods are the two most common methods used for the solution of such elliptic equations. The reader is referred to Fulton et al. (1986) and Zhou and Fulton (2009) for additional information about the multigrid method, and Smolarkiewicz and Margolin (1994), Smolarkiewicz et al. (1997), and Skamarock et al. (1997) for the conjugate gradient method. Although we are not aware of any comprehensive comparison of these methods in the context of atmospheric modeling, it is our belief that the multigrid method yields optimal computational scalability to very large grids. This is our main reason for choosing it.

We consider the two-dimensional elliptic equation on a global domain given by

$$\nabla_H^2 \gamma = F, \tag{5.1}$$

where γ and F are two-dimensional functions of longitude (λ) and latitude (φ). We discretize Eq. (5.1) using the finite-difference Laplacian in Eq. (3.2) on the icosahedral grid shown in Fig. 5 as

$$\frac{1}{A_0}\sum_{i=1}^{n} d_{0,i}\left(\frac{\gamma_i - \gamma_0}{\ell_{0,i}}\right) = F_0, \tag{5.2}$$

where γ_0 and γ_i are the numerical solutions at grid point 0 and neighboring i obtained by the multigrid solver and $F_0 \equiv F(\lambda_0, \varphi_0)$ is the discrete value of F calculated at grid point 0.

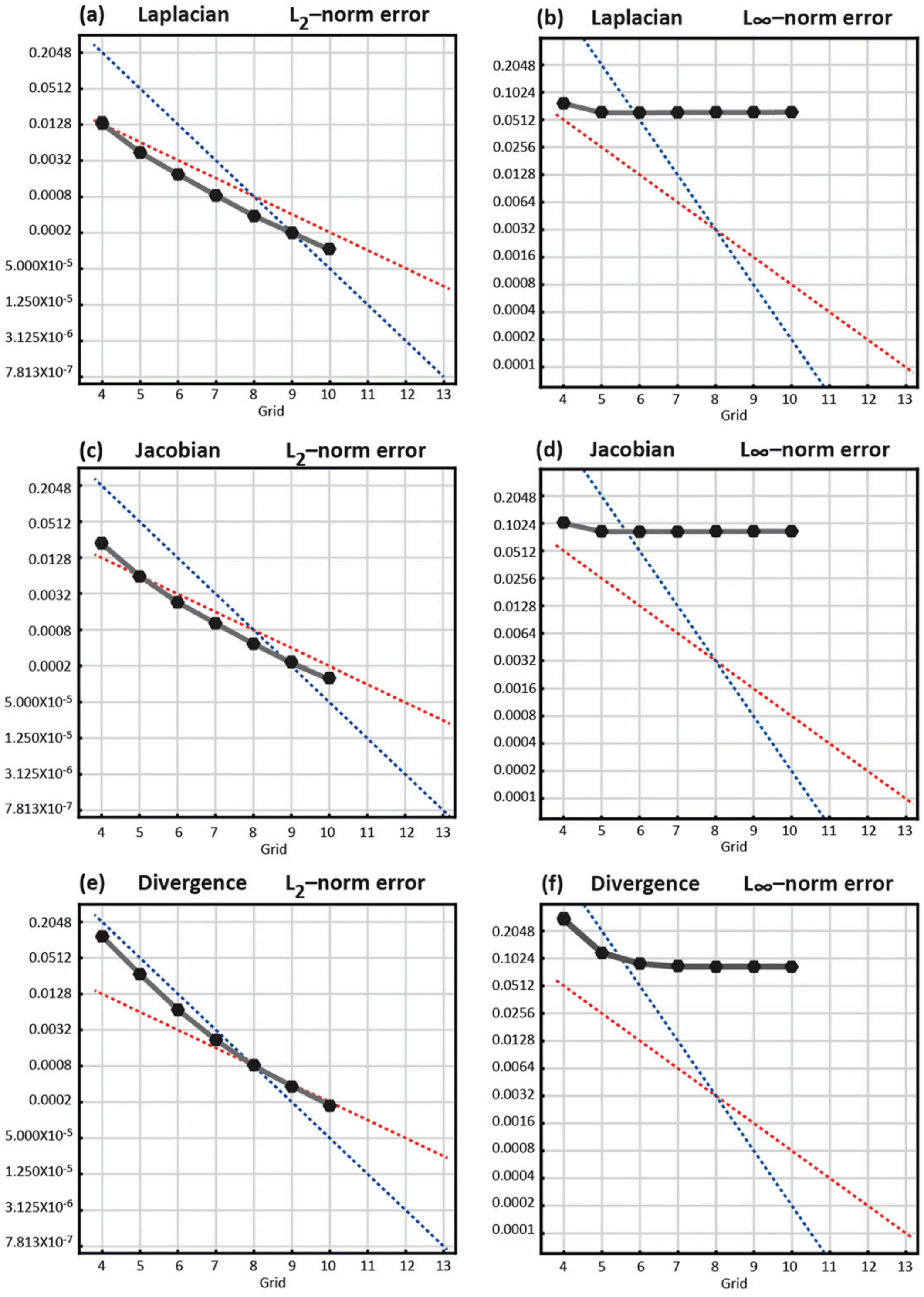

FIG. 11. As in Fig. 6, but for the CVT grid.

The multigrid solver finds γ through iterations on the native grid and coarser grids to maximize accuracy and efficiency. The rationale this is that the iterations on the native (high resolution) grid cannot effectively and efficiently reduce the errors in the large-scale (or smooth) fields. Therefore, coarser grids are used to reduce the large-scale errors, on which less iteration is enough to reduce the error than those on the native grid. For more

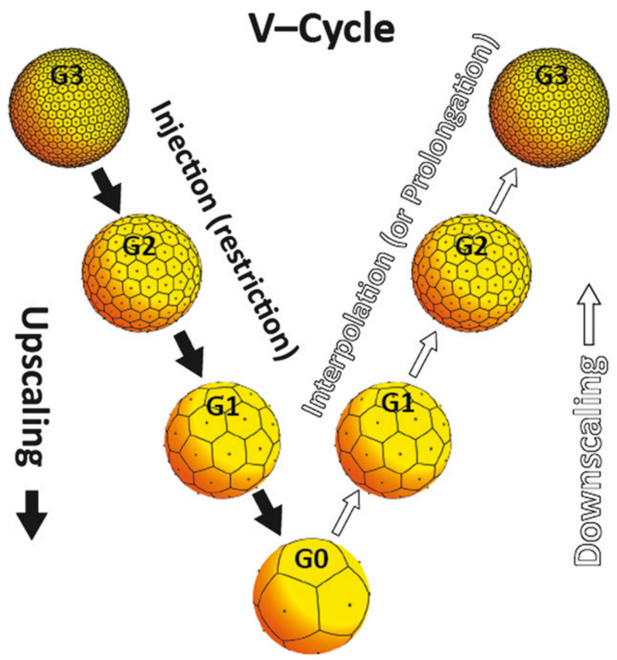

FIG. 12. A schematic illustration of the V cycle for G3.

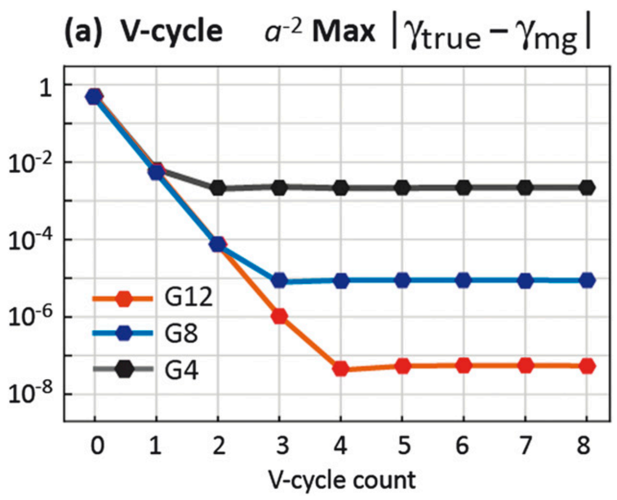

FIG. 13. Maximum error between the true and multigrid solutions as a function of V-cycle counts. The a is the radius of the earth.

information about the multigrid method, see Brandt (1977), Fulton et al. (1986), Briggs et al. (2000), Trottenberg et al. (2001), and Yavneh (2006). We use V cycles, as schematically illustrated in Fig. 12. The solid arrows on the downward side of the V cycle indicate the injections (or restrictions), in which the residual (R) [i.e., the difference between the left and right hand sides of Eq. (5.2)], and ''deviations (γ')'' (i.e., the values of γ required to satisfy the residual), are ''transferred'' to a lower resolution grid, where convergence is advanced through a few iterations. The term ''transfer'' is used because the grid points of a lower-resolution grid always coincide with the points of the higher-resolution grid, so that no interpolation is needed for injections. The grid points of the 12 pentagons are the same for all resolutions, so that no interpolations are needed for these grid points either. The injections continue until G0 is reached. On the upward or ''prolongation'' side of the V cycle, R and γ' are interpolated to higher resolutions using simple interpolations from neighboring points, which is a deviation from the algorithm described by HR95a (p. 1866). Then iterations are performed to correct γ'. Several of these V cycles are needed to achieve convergence.

We investigated the convergence rate as a function of the number of V cycles, for tweaked grids G4 to G12. The test function used was $\gamma(\lambda, \varphi) \equiv -(a^2/2)\cos^3\varphi\cos 3\lambda$, which gives $F(\lambda, \varphi) = 6\cos^3\varphi\cos 3\lambda$. We compared the ''true'' solution γ_{true} with the solution from the multigrid, which is denoted by γ_{mg}. The right-hand side of Eq. (4.2) was determined using $F_0 \equiv F(\lambda_0, \varphi_0) = 6\cos^3\varphi_0\cos 3\lambda_0$. The error was defined by $|\gamma_{\text{true}} - \gamma_{\text{mg}}|$. Figure 13 shows the maximum error as a function of the V-cycle count, for G4, G8, and G12. The error decreases with increasing resolution, but the number of V cycles required for convergence also increases with resolution. To show the order of accuracy, we plot the errors as functions of the resolution for the V cycle in Fig. 14. The slope is almost second order for the low resolutions (between G4 and G8), and slightly better for the high resolutions (between G8 and G12). These results are encouraging.

To examine the sensitivity of the multigrid solutions to the grid optimization, we repeated the solutions discussed above on the unoptimized grid as well. The solutions with the optimized and unoptimized grids showed no significant difference. This is not surprising because the errors in the elliptic equation solutions are dominated by the large-scale errors, which are not affected by the grid-scale differences between the optimized and unoptimized grids.

b. Computational performance of the multigrid solver

The FORTRAN code for implementing the two-dimensional multigrid solver described above has been parallelized using a domain-decomposition method. The global grid is divided into process domains with equal numbers of grid points, and each domain is enlarged by one row of ghost points to carry the information from neighboring domains. Purple lines in Fig. 15a denote the borders of these domains. To illustrate the domain decomposition during the grid injections, we use the example shown in Fig. 15. In this example, the native resolution is G3, and we use 40 processes with 40 domains.

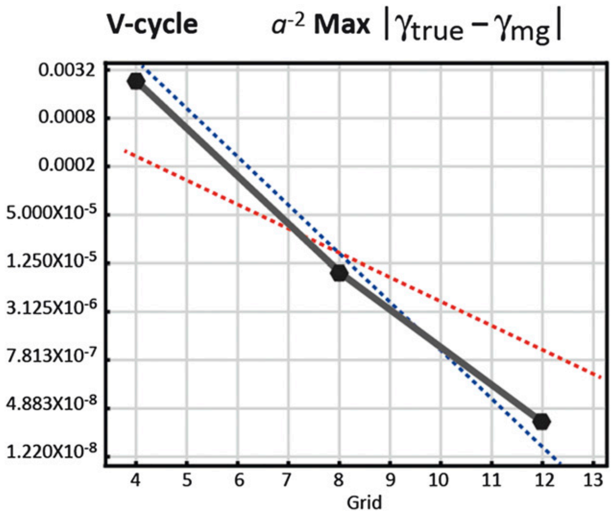

FIG. 14. Convergence of errors with the V-cycled multigrid as a function of grid resolution. The red and blue lines show the slopes of the first and second-order error convergences, respectively. The a is the radius of the earth.

During the injections, if the number of grid points per process is greater than or equal to 16, the number of processes used for the calculations is not changed from the previous injection step, as shown in Fig. 15b. If the number of grid points per process is less than 16, then domains are merged to form new domains with 16 grid points or more each, and the number of processes is reduced accordingly. Figure 15c illustrates an example of domain merging. Four-cell domains are merged together to form 16-cell domains (bordered by cyan lines), and we use only 10 processes out of 40 to perform the calculations. The relaxation scheme used in the multigrid solver is based on an underrelaxed Jacobi solver. We tested several relaxation coefficients with different test functions. The reader is referred to Briggs et al. (2000, p. 9) for the definition of the relaxation coefficient ω. Based on these tests, we concluded that there is no single universal coefficient yielding the fastest convergence in every case. Nevertheless, we found that $\omega = 0.75$ yields the fastest convergence in many cases. Thus we selected this value of the coefficient for use in our model.

We now examine the strong parallel scalability of the multigrid solver. Strong scalability is defined as the change of wall-clock time with the number of processes for a fixed problem size, as identified by the resolution level. Figure 16 shows the time (seconds) needed to execute one V cycle for a 192-layer application, as a function of the number of processes for the resolutions G4 to G13. The multigrid solver is tested with 192 layers because we intend to use the solver in a multilayer model with a deep vertical domain. The thin red lines show the slope of "perfect scalability" for a given resolution; with perfect scalability, the wall-clock time is cut in half when the number of processes is doubled. Small grids, such as G4 and G5, yield very poor strong scalability. The strong scalability improves as the problem size increases. For example, the G11 solution, which corresponds to 4-km grid spacing, scales reasonably well at least up to 81 920 processes.

6. Summary and conclusions

In this paper, we have discussed the generation and optimization of IHP grids, the accuracy of finite-difference Laplacian, Jacobian, and divergence operators applied on the grids, and the performance of a two-dimensional multigrid elliptic solver designed for use on the grids.

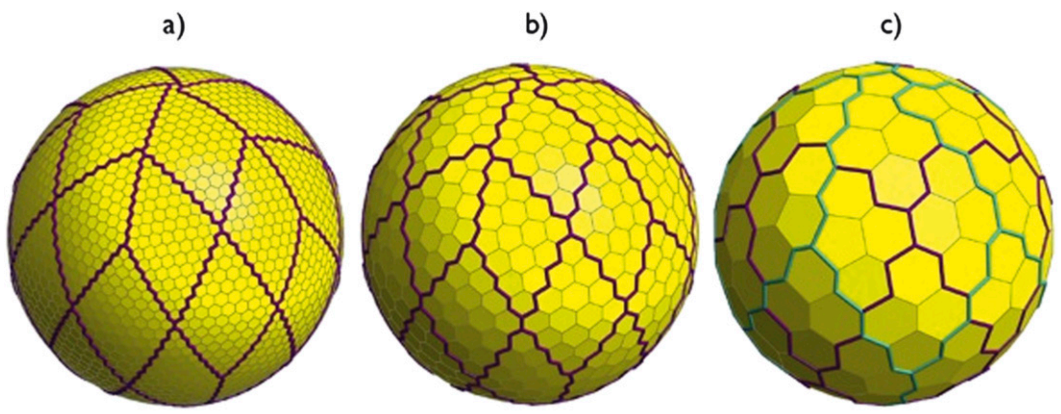

FIG. 15. A schematic depiction of the domain decompositions used in the parallelization of the injection (or restriction) steps of the multigrid with G3 native resolution and 40 processes. The purple and cyan colored lines are the borders of the process domains.

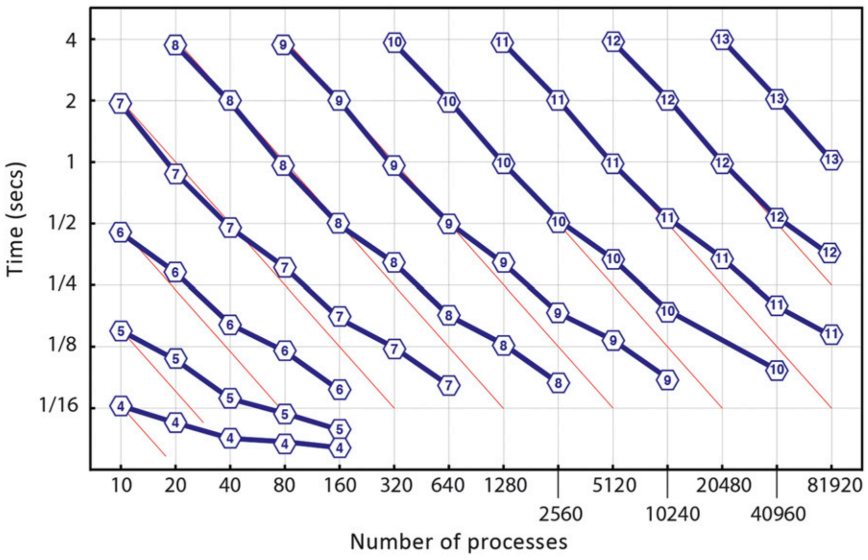

FIG. 16. Strong parallel scalability of the V-cycled multigrid for the native resolutions G4–G13. At each thick mark on the axis and ordinate, the number of processes and the time (seconds) are doubled, respectively. The slope of the red thin lines indicate perfect scalability. Time is measured over one V cycle for the case of 192 model layers. The computations were performed on Hopper, a Cray XE6 computer at the National Energy Research Scientific Computing Center (NERSC).

Our raw grid generation algorithm starts by bisecting the triangular faces of the icosahedron inscribed. The resulting vertices rare projected onto the sphere. The recursive bisections and projections continue until the desired resolution is reached. Projecting the edges of the triangles onto the sphere completes the generation of the grid. The vertices or corners of the spherical triangles form the center points of the hexagonal–pentagonal cells, which are Voronoi cells with respect to those center points.

Optimization of the grid is needed to obtain good convergence of finite-difference operators. Analyses of truncation errors of the finite-difference Laplacian, Jacobian, and divergence operators on the raw icosahedral grids show a nearly first-order convergence rate for the RMS (L_2-norm) errors and almost no convergence of the maximum (L_∞-norm) errors. These rates are much slower than the desired second-order convergence, which is obtained (as expected) using a regular hexagonal grid on a plane.

We have tested three different optimization algorithms that can be applied to unoptimized IHP grids to improve the error convergence rates of the three operators. The tweaking algorithm tries to minimize the distance between the midpoint of the cell wall and the point that the grid segments intersect the cell wall by moving the grid points (or cell centers) of the raw grid. The spring dynamics algorithm tries to homogenize the distances between the grid points, or cell centers. The CVT optimization makes the Voronoi cell centers coincide with the barycentric centers of the cells. All three optimization algorithms reduce the errors and improve the convergence rates. The tweaking optimization produces nearly first- and second-order convergence rates for the L_∞- and L_2-norms. The spring grid and CVT optimizations produce slightly slower convergence rates for the L_2-norms, relative to the tweaked grid. However, both the spring grid and CVT optimizations produce very poor convergence rates for the L_∞-norms.

The tweaking optimization algorithm presented here uses a different cost function than that of HR95b. The new tweaked grid produces more accurate operators and smoother simulations.

Both the code used to generate the raw grid and the tweaking optimization code are being made available as supplements to this paper. The grids themselves are also being made available.

Finally, we have designed and demonstrated a parallel multigrid solver that scales reasonably well up to 81 920 processes. We are making the solver code available to the community as a supplement to this paper.

Acknowledgments. The authors have greatly benefited from a long-term collaboration with Professor Akio Arakawa of UCLA. This research is funded by the U.S. DOE under Cooperative Agreement DE-FC02-06ER64302 to Colorado State University, the DOE under DE-SC07050, NSF Grant AGS-1062468, and NSF Science and Technology Center for Multi-Scale Modeling of Atmospheric Processes, which is managed by Colorado State University under Cooperative Agreement ATM-0425247. Calculations were performed at the National Energy Research Scientific Computing Center (NERSC) and ISTeC Cray at Colorado State University.

APPENDIX

The Tweaking Optimization Algorithm

Consider the function f: $\mathbb{R}^n \rightarrow \mathbb{R}$ defined as

$$f(\mathbf{x}) = \sum_{n=1}^{\text{all cells}} \sum_{i=1}^{\text{cell walls}} \left(\frac{\lambda_{n,i}}{d_{n,i}}\right)^4, \tag{A.1}$$

where $\mathbf{x}$ is a vector with n degrees of freedom that describes the position of cell centers in the global grid. For a given $\mathbf{x}$ we calculate the Voronoi corners, which determine $\lambda_{n,i}$ and $d_{n,i}$. The function sums the distance $\lambda_{n,i}$ normalized with the length of a cell wall $d_{n,i}$ for each cell wall over all cells. The exponent causes larger ratios to have an increased influence relative to smaller ratios in the global sum. Our goal is to minimize this function over all possible configurations of grid points, that is, to find

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \forall \quad \mathbf{x} \in \mathbb{R}^n. \tag{A.2}$$

This function is designed to optimize a particular property of the grid. Similar functions could be constructed to optimize other properties of the grid using the approach described below.

We minimize the function f using quasi-Newton optimization methods, constructing a sequence $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots\}$ of approximations that converge to the optimal solution. The starting point, $\mathbf{x}_0$, is the raw grid. Quasi-Newton methods, and in particular, the BFGS method used here, are described in detail by Fletcher (1987).

The quadratically truncated Taylor series $f(\mathbf{x})$ expanded about the kth iterate $\mathbf{x}_k$ is given by

$$f(\mathbf{x}_k + \delta) \approx f(\mathbf{x}_k) + \mathbf{g}_k^{\mathrm{T}}\delta + \frac{1}{2}\delta^{\mathrm{T}}\mathbf{B}_k\delta, \tag{A.3}$$

where $\delta = \mathbf{x} - \mathbf{x}_k$, $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$, and $\mathbf{B}_k$ is the Hessian matrix evaluated at $\mathbf{x}_k$. The Hessian matrix $\mathbf{B}$ is the square matrix of second-order partial derivatives where the (i, j)th element is given by $\partial^2 f(\mathbf{x})/\partial x_i \partial x_j$. The gradient of the Taylor series is

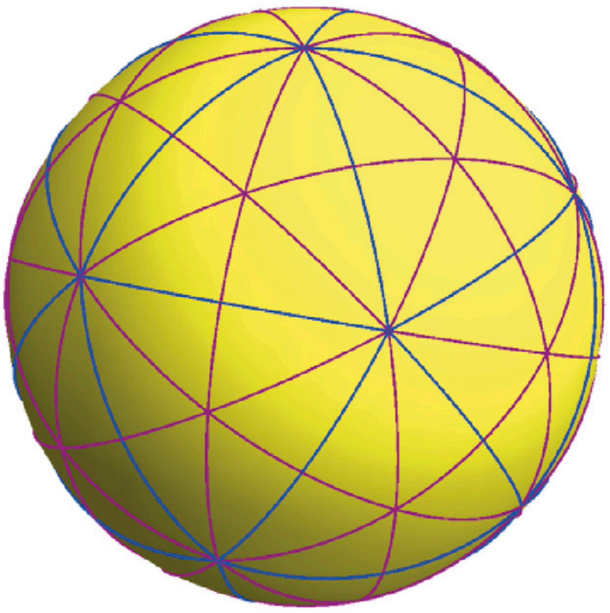

FIG. A1. Partitioning of the grid to reduce degrees of freedom in the optimization.

$$\nabla f(\mathbf{x}_k + \delta) = \mathbf{g}_k + \mathbf{B}_k\delta. \tag{A.4}$$

A local minimum of the Taylor series is characterized by a zero gradient. Setting the gradient to zero gives the conventional Newton step $\delta = \mathbf{B}_k^{-1}\mathbf{g}_k$.

In quasi-Newton methods, $\mathbf{B}_k^{-1}$ is approximated directly by a symmetric positive definite matrix that is corrected from iteration to iteration using only first-order derivative information. This avoids the need for second-order derivatives to calculate the Hessian. If we define $\delta_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\gamma_k = \mathbf{g}_{k+1} - \mathbf{g}_k$, then $\mathbf{H}_{k+1}$ is chosen so that $\mathbf{H}_{k+1}\gamma_k = \delta_k$. There are several ways to choose $\mathbf{H}_{k+1}$. We use the so-called Broyden–Fletcher–Goldfarb–Shanno (BFGS) formula, which is given by

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \left(1 + \frac{\gamma_k^{\mathrm{T}}\mathbf{H}_k\gamma_k}{\delta_k^{\mathrm{T}}\gamma_k}\right)\frac{\delta_k\delta_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}}\gamma_k} - \left(\frac{\delta_k\gamma_k^{\mathrm{T}}\mathbf{H}_k + \mathbf{H}_k\gamma_k\delta_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}}\gamma_k}\right). \tag{A.5}$$

The algorithm implemented in the optimization code is

(i) determine a search direction $\mathbf{s}_k$ with $\mathbf{s}_k = -\mathbf{H}_k\mathbf{g}_k$,
(ii) perform a line search along $\mathbf{s}_k$ giving $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{s}_k$,
(iii) update $\mathbf{H}_k$ giving $\mathbf{H}_{k+1}$,
(iv) repeat until the convergence criterion is met.

For a very high-resolution grid, it is impossible to store the dense $n \times n$ matrix associated with $\mathbf{H}_k$, so we use the limited-memory BFGS, or ‘‘L-BFGS,’’ algorithm pioneered by Nocedal (1980). With L-BFGS only a few vectors are required to represent $\mathbf{H}_k$.

We reduce the problem size by taking advantage of certain symmetries of the grid. The 20 spherical triangles of the projected icosahedron are all identical, and reflectively symmetric. This implies that each spherical triangle can be partitioned into six self-similar pieces. Figure A1 shows the global grid partitioned into 120 ($=20 \times 6$) pieces. The optimization is applied to only one small triangular piece. The positions of the grid points in the remaining pieces are then determined through reflections.

REFERENCES

Adcroft, A., J.-M. Campin, C. Hill, and J. Marshall, 2004: Implementation of an atmosphere–ocean general circulation model on the expanded spherical cube. *Mon. Wea. Rev.,* **132,** 2845–2863.

Arakawa, A., 1966: Computational design for long-term numerical integration of the equations of fluid motion. Two-dimensional incompressible flow. Part I. *J. Comput. Phys.,* **1,** 119–143.

——, and V. R. Lamb, 1977: Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in Computational Physics,* J. Chang, Ed., Vol. 17, Academic Press, 173–265.

Augenbaum, J. M., and C. S. Peskin, 1985: On the construction of the Voronoi mesh on a sphere. *J. Comput. Phys.,* **14,** 177–192.

Bonaventura, L., and T. Ringler, 2005: Analysis of discrete shallow-water models on geodesic Delaunay grids with C-type staggering. *Mon. Wea. Rev.,* **133,** 2351–2373.

——, L. Konrblueh, T. Heinze, and P. Ripodas, 2005: A semi-implicit method conserving mass and potential vorticity for the shallow water equations on the sphere. *Int. J. Numer. Methods Fluids,* **47,** 863–869.

Bourke, W., B. McAvaney, K. Puri, and R. Thurling, 1977: Global modelling of atmospheric flow by spectral methods. *Methods in Computational Physics,* J. Chang, Ed., Vol. 17, Academic Press, 267–353.

Brandt, A., 1977: Multi-level adaptive solutions to boundary value problems. *Math. Comput.,* **31,** 333–390.

Briggs, W. L., V. E. Henson, and S. F. McCormick, 2000: *A Multigrid Tutorial.* 2nd ed. SIAM, 193 pp.

Du, Q., V. Faber, and M. Gunzburger, 1999: Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Rev.,* **41,** 637–676.

Fletcher, R., 1987: *Practical Methods of Optimization*. 2nd ed. Wiley & Sons, 436 pp.

Fulton, S. R., P. E. Ciesielski, and W. Schubert, 1986: Multi-grid methods for elliptic problems: A review. *Mon. Wea. Rev.,* **114,** 943–959.

Heikes, R. P., and D. A. Randall, 1995a: Numerical integration of the shallow water equations on a twisted icosahedral grid. Part I: Basic design and results of tests. *Mon. Wea. Rev.,* **123,** 1862–1880.

——, and ——, 1995b: Numerical integration of the shallow water equations on a twisted icosahedral grid. Part II: A detailed description of the grid and an analysis of numerical accuracy. *Mon. Wea. Rev.,* **123,** 1881–1887.

Hoskins, B. J., 1980: Representation of the Earth topography using spectral harmonics. *Mon. Wea. Rev.,* **108,** 111–115.

Ju, L., T. Ringler, and M. Gunzburger, 2011: Voronoi tessellations and their application to climate and global modeling. *Numerical Techniques for Global Atmospheric Models,* P. H. Lauritzen et al., Eds., Lecture Notes in Computational Science and Engineering, Vol. 80, Springer-Verlag, 313–342.

Lee, J.-L., and A. E. MacDonald, 2009: A finite-volume icosahedral shallow-water model on a local coordinate. *Mon. Wea. Rev.,* **137,** 1422–1437.

Majewski, D., and Coauthors, 2002: The operational global icosahedral-hexagonal gridpoint model GME: Description and high-resolution tests. *Mon. Wea. Rev.,* **130,** 319–338.

Masuda, Y., and H. Ohnishi, 1986: An integration scheme of the primitive equations model with an icosahedral-hexagonal grid system and its application to the shallow water equations. *Short- and Medium-Range Numerical Weather Prediction,* T. Matsuno, Ed., Japan Meteorological Society, 317–326.

Mishra, S. K., M. A. Taylor, R. D. Nair, P. H. Lauritzen, H. M. Tufo, and J. J. Tribbia, 2011: Evaluation of the HOMME dynamical core in the aquaplanet configuration of NCAR CAM4: Rainfall. *J. Climate,* **24,** 4037–4055.

Miura, H., and M. Kimoto, 2005: A comparison of grid quality of optimized spherical hexagonal-pentagonal geodesic grids. *Mon. Wea. Rev.,* **133,** 2817–2833.

Nocedal, J., 1980: Updating quasi-Newton matrices with limited storage. *Math. Comput.,* **35,** 773–782.

Okabe, A., B. Boots, K. Sugihara, and S. N. Chiu, 2000: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* 2nd ed. John Wiley & Sons, 667 pp.

Putman, W., and S.-J. Lin, 2007: Finite-volume transport on various cubed-sphere grids. *J. Comput. Phys.,* **127,** 55–78.

Rancic, M., R. J. Purser, and F. Mesinger, 1996: A global shallow-water model using an expanded spherical cube: Gnomic versus conformal coordinates. *Quart. J. Roy. Meteor. Soc.,* **122,** 959–982.

Randall, D. A., 1994: Geostrophic adjustment and the finite-difference shallow-water equations. *Mon. Wea. Rev.,* **122,** 1371–1377.

——, T. D. Ringler, R. P. Heikes, P. Jones, and J. Baumgardner, 2002: Climate modeling with spherical geodesic grids. *Comput. Sci. Eng.,* **4,** 32–41.

Ronchi, C., R. Iacono, and P. S. Paolucci, 1996: The "Cubed sphere": A new method for the solution of partial differential equations in spherical geometry. *J. Comput. Phys.,* **124,** 93–114.

Rothman, T., Ed., 1989: Geodesics, domes, and spacetime. *Science à la Mode: Physical Fashions and Fictions,* Princeton University Press, 67–123.

Sadourny, R., 1972: Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Mon. Wea. Rev.,* **100,** 136–144.

——, A. Arakawa, and Y. Mintz, 1968: Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere. *Mon. Wea. Rev.,* **96,** 351–356.

Satoh, M., T. Matsuno, H. Tomita, H. Miura, T. Nasuno, and S. Iga, 2008: Nonhydrostatic Icosahedral Atmospheric Model (NICAM) for global cloud resolving simulations. *J. Comput. Phys.,* **227,** 3486–3514.

Skamarock, W. C., P. K. Smolorkiewicz, and J. B. Klemp, 1997: Preconditioned conjugate-residual solvers for Helmholtz equations in nonhydrostatic models. *Mon. Wea. Rev.,* **125,** 587–599.

——, J. B. Klemp, M. G. Duda, L. Fowler, S.-H. Park, and T. D. Ringler, 2012: A multiscale nonhydrostatic atmospheric model using centroidal Voronoi tessellations and C-grid staggering. *Mon. Wea. Rev.,* **140,** 3090–3105.

Smolarkiewicz, P. K., and L. G. Margolin, 1994: Variational solver for elliptic problems in atmospheric flows. *Appl. Math. Comput. Sci.,* **4,** 527–551.

——, V. Grubisic, and L. G. Margolin, 1997: On forward-in-time differencing for fluids: Stopping criteria for iterative solutions of anelastic pressure equations. *Mon. Wea. Rev.,* **125,** 647–654.

Staniforth, A., and J. Thuburn, 2012: Horizontal grids for global weather and climate prediction models: A review. *Quart. J. Roy. Meteor. Soc.,* **138,** 1–26, doi:10.1002/qj.958.

Thuburn, J., 1997: A PV-based shallow-water model on a hexagonal-icosahedral grid. *Mon. Wea. Rev.,* **125,** 2328–2347.

——, 2008: Numerical wave propagation on the hexagonal C-grid. *J. Comput. Phys.,* **227,** 5836–5858.

Tomita, H., and M. Satoh, 2004: A new dynamical framework of nonhydrostatic global model using the icosahedral grid. *Fluid Dyn. Res.,* **34,** 357–400.

——, M. Tsugawa, M. Satoh, and K. Goto, 2001: Shallow water model on a modified icosahedral geodesic grid by using spring dynamics. *J. Comput. Phys.,* **174,** 579–613.

——, M. Satoh, and K. Goto, 2002: An optimization of the icosahedral grid modified by spring dynamics. *J. Comput. Phys.,* **183,** 307–331.

Trottenberg, U., C. W. Oosterlee, and A. Schuller, 2001: *Multigrid.* Academic Press, 631 pp.

Vestine, E. H., W. L. Sibley, J. W. Kern, and J. L. Carlstedt, 1963: Integral and spherical harmonic analysis of the geomagnetic field for 1955.0, Part 2. *J. Geomagn. Geoelectr.,* **15,** 73–89.

Wan, H., 2009: Developing and testing a hydrostatic atmospheric dynamical core on triangular grids. Ph.D. dissertation, Max-Planck Institute for Meteorology, Hamburg, Germany, 153 pp.

Williamson, D. L., 1968: Integration of the barotropic vorticity equation on a spherical geodesic grid. *Tellus,* **20,** 642–653.

——, and P. J. Rasch, 1994: Water vapor transport in the NCAR CCM2. *Tellus,* **46A,** 34–51.

Yavneh, I., 2006: Why multigrid methods are so efficient. *Comput. Sci. Eng.,* **8,** 12–22.

Zhou, G., and S. R. Fulton, 2009: Fourier analysis of multigrid methods on hexagonal grids. *SIAM J. Sci. Comput.,* **31,** 1518–1538.