Voronoi, Delaunay, and Block-Structured Mesh Refinement for Solution of the Shallow-Water Equations on the Sphere

HILARY WELLER
National Centre for Atmospheric Science Climate, Department of Meteorology, University of Reading, Reading, United Kingdom

HENRY G. WELLER
OpenCFD, Ltd., Reading, United Kingdom

AIME FOURNIER
National Center for Atmospheric Research,* Boulder, Colorado

(Manuscript received 17 December 2008, in final form 12 June 2009)

ABSTRACT

Alternative meshes of the sphere and adaptive mesh refinement could be immensely beneficial for weather and climate forecasts, but it is not clear how mesh refinement should be achieved. A finite-volume model that solves the shallow-water equations on any mesh of the surface of the sphere is presented. The accuracy and cost effectiveness of four quasi-uniform meshes of the sphere are compared: a cubed sphere, reduced latitude–longitude, hexagonal–icosahedral, and triangular–icosahedral. On some standard shallow-water tests, the hexagonal–icosahedral mesh performs best and the reduced latitude–longitude mesh performs well only when the flow is aligned with the mesh. The inclusion of a refined mesh over a disc-shaped region is achieved using either gradual Delaunay, gradual Voronoi, or abrupt 2:1 block-structured refinement. These refined regions can actually degrade global accuracy, presumably because of changes in wave dispersion where the mesh is highly nonuniform. However, using gradual refinement to resolve a mountain in an otherwise coarse mesh can improve accuracy for the same cost. The model prognostic variables are height and momentum collocated at cell centers, and (to remove grid-scale oscillations of the A grid) the mass flux between cells is advanced from the old momentum using the momentum equation. Quadratic and upwind biased cubic differencing methods are used as explicit corrections to a fast implicit solution that uses linear differencing.

1. Introduction

Adaptive and variable-resolution modeling of the atmosphere is of increasing interest due to the potential benefits to, for example, regional weather and climate prediction, local resolution of convection and orography, and cyclone tracking. These techniques have been under investigation for atmospheric modeling for some time (e.g., Staniforth and Mitchell 1978; Berger and Oliger 1984; Skamarock et al. 1989; Dietachmayer and Droegemeier 1992; Skamarock and Klemp 1993; Fiedler and Trapp 1993) but are still mostly research oriented rather than operational (e.g., Jablonowski et al. 2006; Läuter et al. 2007; St-Cyr et al. 2008), a noteworthy exception being Bacon et al. (2000). The problems with variable-resolution modeling are less well publicized than the successes. A brash and sweeping generalization would be that high-order methods for unstructured meshes (such as discontinuous Galerkin or spectral element) are expensive (e.g., Läuter et al. 2008) whereas low-order methods such as lower-order finite volumes and elements are insufficiently accurate (e.g., Läuter et al. 2005), especially where the mesh changes resolution (e.g., Lanser et al. 2000).

Spatial changes in resolution can degrade the accuracy globally, particularly of a low-order model, if high resolution is placed arbitrarily to test the numerics rather than to refine a feature of interest (St-Cyr et al. 2008).
Alternatively, high local resolution can increase the accuracy, particularly of a higher-order model, if, for example, either external forcing or orographic forcing are significantly spatially localized (Fournier et al. 2004). Local resolution is particularly beneficial for resolving features that would otherwise be smaller than the stencil or element used for high-order differencing. It is thought that changes in resolution can degrade global accuracy due to spurious wave reflection and scattering (Vichnevetsky 1987; Vichnevetsky and Turner 1991), a problem that is partly rectified by using gradual rather than abrupt mesh refinement.

Low-order models have difficulties in maintaining balances between tightly curved fields with many non-zero derivatives, and the resulting spurious imbalance leads to the break down of balanced but physically unstable jets (e.g., Jablonowski and Williamson 2006; Läuter et al. 2008). These problems do not occur when the flow is aligned with, for example, a latitude–longitude mesh. Some models on hexagonal or triangular meshes are only second-order accurate where the mesh is uniform and revert to first-order accuracy where the mesh is nonuniform. The resulting errors have motivated the development of algorithms to adjust icosahedral meshes to be as uniform as possible, for example using spring dynamics (Tomita et al. 2002) or deformations to reduce mesh skewness (Heikes and Randall 1995).

High-order models have overcome some of the problems described above but some problems have been exacerbated. Läuter et al. (2008) described the problems with representing a barotrurally unstable jet with both a low-order (k = 2) and a high-order (k = 6) discontinuous Galerkin (DG) model. The low-order model does not respect the tight variations in the balanced flow whereas, once realistic barotropic instabilities develop, the high-order model creates spurious oscillations. Further increases in the order of accuracy do not produce more accurate results for the same number of degrees of freedom because then the span of the points required to fit a higher-order polynomial exceeds the width of the jet. Oscillations are also produced when using the spectral-transform method to simulate this test case without diffusion (Galewsky et al. 2004) and using the spectral-element method (St-Cyr et al. 2008).

St-Cyr et al. (2008) and Jablonowski and Williamson (2006) compare models with different mesh types: latitude–longitude, the cubed sphere, and icosahedral, on various two- and three-dimensional test cases. Some useful conclusions are drawn concerning the different numerical methods and the different meshes. For example, the cubed-sphere and icosahedral meshes can trigger spurious wavenumber 4 and 5 patterns, respectively. However, the numerical schemes on the different mesh structures were very different. In contrast, here we will compare these different meshes and different refinement patterns all with the same numerical model.

To our knowledge, there has been no systematic study of the advantages and disadvantages of different refinement techniques within the context of spherical shallow-water simulation. Block-structured 2:1 refinement has been popular because structured-mesh models can be nested explicitly (e.g., Skamarock and Klemp 1993) so that finer meshes can use shorter time steps. This type of nesting is used with explicit time stepping, which has severe time-step restrictions, based on the speed of the fastest waves. Implicit two-way nesting with longer variable time steps has been used by Almgren et al. (1998) for incompressible flows. Block-structured refinement has also been popular as multigrid algorithms for space and time can in principle be used (e.g., Ruge et al. 1995). Unstructured meshes are particularly popular for ocean modeling since coastlines and bottom topography can be represented accurately (e.g., Pain et al. 2005) but they have also been used for atmospheric modeling in order to refine gradually (e.g., Bacon et al. 2000; Läuter et al. 2007). The potential for unstructured meshes to refine anisotropically in an entirely flexible manner and align with the flow has not yet been applied to the global atmosphere.

Here, we plan to compare the accuracy of various block-structured and unstructured meshes and refinement patterns. The model is described in section 2 and the meshes are introduced in section 3, with a more detailed description provided in the appendix. Shallow-water test cases of global steady-state geostrophic flow, flow over an isolated mountain, and a barotropically unstable jet are presented in section 4 and conclusions are drawn on the suitability of different meshes and the current modeling framework in section 5.

2. Discretization of the shallow-water equations on polygons

The shallow-water equations on a rotating sphere have been implemented in the open-source C++ library OpenFOAM (Open Field Operation and Manipulation; information online at www.opencfd.co.uk), to create AtmosFOAM [which has in part been described by Weller and Weller (2008)].

a. The shallow-water equations

The shallow-water equations (SWEs) are written in 3D vector flux form on a sphere:

$$\frac{\partial h \mathbf{U}}{\partial t} + \nabla \cdot (h \mathbf{U} \otimes \mathbf{U}) = -2\Omega \times h \mathbf{U} - gh \nabla (h + h_0) + \mathbf{f}$$

(1)
and
\[
\frac{\partial h}{\partial t} + \nabla \cdot (h \mathbf{U}) = 0, \tag{2}
\]
where \( \mathbf{U} \) is the 3D velocity vector, \( \nabla \) is the 3D gradient, \( \nabla \cdot \) is the 3D divergence operator, \( h \) is the height of the fluid surface above the solid surface, \( h_0 \) is the height of the solid surface above a spherical reference height, \( \Omega \) is the angular-velocity vector of the sphere, \( g \) is the acceleration magnitude due to gravity, \( \mathbf{r} \) is the position vector on the surface of the sphere, \( \mathbf{r} = r|\mathbf{r}| \), and \( \mu \) is the Lagrange multiplier that constrains the motion to follow the sphere (Côté 1988). (The tensor product of two column vectors \( \mathbf{U} \) is defined as \( \mathbf{U} \otimes \mathbf{U} = \mathbf{U} \mathbf{U}^T \), where \( \mathbf{U} \) and \( \mathbf{U}^T \) are multiplied using standard matrix multiplication to form a \( 3 \times 3 \) tensor.)

b. The finite-volume formulation on polygons

OpenFOAM uses the finite-volume technique on 3D polyhedral meshes in Cartesian coordinates. The shallow-water model AtmosFOAM uses a spherical shell of polygons in Cartesian coordinates. Rather than using the Lagrange multiplier to constrain the velocity to the surface of the sphere, we remove the radial component of the momentum during each iteration \([ h \mathbf{U} \rightarrow h \mathbf{U} - (h \mathbf{U} \cdot \mathbf{r}) \mathbf{r} ] \), similar to Heinze and Hense (2002).

The prognostic variables are the three components of the cell-average momentum \( h \mathbf{U} \), and the cell average height \( h \) (Fig. 1). The mass flux between cells \( \phi \) is advanced from the previous time step by solving the momentum equation as described by Rhie and Chow (1983) and is considered a quasi-prognostic variable. This removes grid-scale oscillations of the A grid (Arakawa and Lamb 1977). Using local coordinate mappings for every cell individually in order to solve just two components of momentum per cell as in Läuter et al. (2008) would be more efficient, but this has not yet been implemented. We will first give an overview of the solution algorithm and then more details about the precise discretization.

\[
\frac{V}{\delta t} \left[ (h \mathbf{U})^0 - (h \mathbf{U})^0 \right] + \sum_f (h \mathbf{U} \otimes \mathbf{U})^n_f \cdot \mathbf{S} = -V(2 \mathbf{\Omega} \times h \mathbf{U})^n - gh \sum_f (h + h_0)^n_f \mathbf{S}, \tag{3}
\]

where \( V \) is the cell volume, \( \delta t \) is the time step, \( \sum_f \) means summation over all the faces of a cell, the subscript \( f \) indicates face-average quantities, \( \mathbf{S} = \mathbf{S}_f \) is the outward-pointing vector normal to a face with the magnitude of the area of that face (Fig. 1), and \( \alpha \) is the blending factor between the old and new time steps, that is, \( h^n = \alpha h^t + (1 - \alpha)h^n \). Here, \( \alpha = \frac{1}{2} \) gives the second-order, Crank–Nicholson time discretization. We have used \( \alpha = 0.55 \) to improve the stability, but this also reduces the temporal order of accuracy.

Each of the three components of the momentum equation is solved separately with other components.

1) OVERVIEW OF THE SOLUTION ALGORITHM

(i) The cell-centered momentum equation is linearized about the solution from the previous iteration and each component solved implicitly.

(ii) The momentum equation is also discretized on the faces and is combined with the continuity equation to form a modified Helmholtz equation to predict the height \( h \) and the face fluxes \( \phi \). This allows time steps much longer than the Courant–Friedrichs–Lewy (CFL) restriction based on the gravity wave speed.

(iii) Two iterations are performed for each time step with, higher-order discretization, nonlinear, and Coriolis terms updated for the second iteration to improve the accuracy and convergence.

The advantage of solving each component of the momentum equation implicitly is in the resulting stability of the centered, nondiffusive differencing without the need for explicit diffusion or limiters. The algorithm is not monotonic, so limiters would be needed for tracers or on a case where the height approaches zero.

2) DISCRETIZATION OF THE MOMENTUM EQUATION

We integrate the momentum equation between time steps 0 (previous) and 1 (current) using Gauss’s theorem applied to the divergence term and the height gradient:
lagged (i.e., most up-to-date value used from before the start of the implicit solution). A lagged term at time \( \delta t \) is denoted using superscript \( t(1) \) and a lagged term at time \( \alpha \delta t \) is denoted by superscript \( t(\alpha) \), where \( h^{t(\alpha)} = \alpha h^{t(1)} + (1 - \alpha)h^0 \). The nonlagged terms with superscripts 1 and \( \alpha \) at times \( \delta t \) and \( \alpha \delta t \) are the terms that are solved implicitly. The Coriolis and height-gradient terms on the right-hand side of Eq. (3) are all lagged. The nonlinear advection term, \( \sum_f (hU \otimes U)_f^a \cdot S \), is linearized by replacing \( (hU)_f^a \cdot S \) with \( \phi^{t(\alpha)} \). The remaining \( U_f^a \) factor in this term is handled implicitly by interpolating from the cell-average momentum and height to the face-average velocity:

\[
U_f^a = \frac{\lambda(hU)^a}{h^{t(\alpha)}} + \frac{(1 - \lambda)(hU)_N^a}{h^{t(\alpha)}} + C_{\text{hot}}^{t(\alpha)},
\]

where \( \lambda = \lambda_f \) is the linear interpolation factor used to interpolate between the cell center and the neighbor cell center \( N \) on the other side of face \( f \), and \( C_{\text{hot}}^{t(\alpha)} \) are the higher-order terms used to correct the linear interpolation. The method for finding the stencil and calculating the higher-order terms is given in section 2c. Equation (4) is substituted into the nonlinear advection term and nonlagged, superscripted \( \alpha \) terms are replaced with the appropriate combination of old and new time steps to give the discretized nonlinear advection term:

\[
\sum_f (hU \otimes U)_f^a \cdot S = \alpha A^{t(1)}(hU)^1 + (1 - \alpha)A^0(hU)^0 + \alpha B^1 + (1 - \alpha)B^0 + C_{\text{hot}}^{t(\alpha)},
\]

where

\[
A^k = \sum_f (\lambda \phi^k)/h^k \quad \text{and} \quad B^k = \sum_f (1 - \lambda) \phi^{(k)}[(hU)_N^h/(h_N^a)^k], \quad k = 0, 1.
\]

Here, \( B^1 \) contains all the intercell dependencies that will be solved simultaneously, that is, all the off-diagonal elements of a matrix that operates on a column of \( hU^1 \) components over all cells. We now substitute the nonlinear advection, Eq. (5), into the discretized momentum Eq. (3) and rearrange to give all terms involving \( (hU) \) at the new time step on the left-hand side:

\[
(hU)^1 \left[ \frac{V}{\delta t} + \alpha A^{t(1)} \right] + \alpha B^1 = (hU)^0 \left[ \frac{V}{\delta t} - (1 - \alpha)A^0 \right] - (1 - \alpha)B^0 - C_{\text{hot}}^{t(\alpha)} - 2V\Omega \times (hU)^{t(\alpha)} - gh^{t(\alpha)} \sum_f (h + h_0)^{t(\alpha)} S.
\]

Equation (6) is written as three, sparse matrix equations, one matrix for each component of \( (hU)^1 \). These matrices have dimensions \( N_{\text{cells}} \times N_{\text{cells}} \) for a mesh with \( N_{\text{cells}} \) cells and \( N_N \) nonzero off-diagonal elements per row for cells with \( N_N \) face neighbors (sides). For a CFL limit of about one, the matrix is diagonally dominant and is solved with a biconjugate gradient iterative solver with incomplete LU preconditioning.

3) SIMULTANEOUS SOLUTION FOR THE FLUXES AND HEIGHTS

The continuity Eq. (2) is discretized using Gauss’s divergence theorem:

\[
D = \frac{V - (1 - \alpha)\delta t A^0}{V + \alpha \delta t A^{t(1)}}, \quad E = \delta t \left[ \frac{B + C_{\text{hot}} + 2V\Omega \times (hU)^{t(\alpha)}}{V + \alpha \delta t A^{t(1)}} \right], \quad \text{and} \quad F = \frac{Vgh^{t(\alpha)}}{V + \alpha \delta t A^{t(1)}},
\]

We have not used the lagged cell-centered discretization of the height gradient from Eq. (6) to derive Eq. (8). Instead, the height gradient has been rediscritized at the face, making the algorithm staggered and the computational molecule compact:

\[
\nabla_f h^a = \frac{(h_N^a - h^a)}{\delta x} + G_{\text{hot}}^{t(\alpha)},
\]

where \( \delta x \) is the distance between the cell center and the cell center of the neighbor \( N \) on the other side of face \( f \).
and \( G^{(1)}_{hot} \) contains corrections for higher-order terms, which are lagged (see section 2c).

To remove the possible inconsistency that arises because we have separate diagnostic equations for the cell-centered momentum and the fluxes, the old time-level flux is interpolated from the old time-level momentum following Rhie and Chow (1983):

\[
\psi^0 = (hU)^0 \cdot S. \tag{10}
\]

Because this interpolation is done at the old time step rather than the current time step, most of the spurious computational mode of the Arakawa A grid is removed. Remaining grid-scale oscillations are avoided by the use of upwind-biased cubic differencing for \( C_{hot}^{(a)} \) as will be described in section 2c.

Here, \( \phi \) and \( h \) are solved simultaneously by substituting Eqs. (8) and (9) into the continuity Eq. (7) to create the “pressure equation,” a modified Helmholtz equation. This is rearranged so that nonlagged terms involving \( h^1 \) are on the left-hand side:

\[
\frac{V}{\delta t^2} (h^1 - h^0) - \alpha \sum_f F_f \nabla h^0 |S| = \text{explicit terms}. \tag{11}
\]

Since the higher-order corrections for \( \Psi_f \) are lagged, as are the nonlinear parts of the advection and the Coriolis force, the matrix equation for \( h^1 \) has only \( N_N \) nonzero off-diagonal terms per row for cells with \( N_N \) face neighbors (sides). The matrix is solved using an incomplete Cholesky preconditioned conjugate-gradient iterative solver.

c. Quadratic and cubic differencing

We now define how the corrections for the higher-order terms \( [C_{hot} \text{ and } G_{hot} \text{ in sections 2(b)2 and 2(b)3}] \) are calculated. First, a stencil of cells surrounding each face is found. To make the discretization on polyhedral meshes simple, cell-volume average quantities are approximated by the cell-center value and face-area averages are approximated by face-center values. These approximations are second-order accurate.

First, we define a local coordinate system centered on the face center with \( x \) in the direction of the face normal (Fig. 2). Interpolations and face-normal gradients are calculated using a 2D quadratic expression:

\[
\psi = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2. \tag{12}
\]

These are the terms that can be fitted accurately using stencils shaped as in Fig. 2. Stevens and Bretherton (1996) used more symmetric stencils and so included different additional terms from a 3D cubic. The interpolant of field \( \Psi \) at the face \( \Psi_f \) and the face-center gradient normal to the face \( \nabla \Psi_f \) are then given by

\[
\Psi_f = a_0 \quad \text{and} \quad \nabla \Psi_f = a_1. \tag{14}
\]

These polynomials are fit over three different types of stencils:

(a) On nontriangular meshes, we fit the quadratic over a stencil centered on the face we interpolate onto (Fig. 2a). This stencil consists of the cells on either side of the face and all face neighbors of these cells.

(b) For the divergence in the momentum equation (for which we use some additional cubic terms), we use an upwind-biased stencil (Fig. 2b). We first find the one or two faces of the upwind cell that are opposite the face we are interpolating onto. The upwind face is defined to be the face with the maximum \( \psi_c \).

Fig. 2. The 2D stencils of cells for interpolating onto the gray face. The stencils are expanded out from the cells marked by circles.
face-center positions with respect to the local coordinate system. If the face with the second-largest $c \cdot c_j$ is within 90% of the maximum, then this face is defined as an upwind face also. (Upwind faces are marked as railway tracks in Fig. 2b.) The stencil then consists of the cells on either side of the upwind faces and all face neighbors of these cells.

(c) On meshes of triangles, the stencil of a face consists of the cell upwind of the face and all point neighbors of this cell (Fig. 2c). The stencil needs to be this big so as to include a cell directly upwind of the upwind cell.

Depending on the mesh topology, these stencils will have more cells than unknowns in the quadratic or cubic so the system is overspecified. Following Lashley (2002), we find a least squares fit using singular value decomposition with the immediate upwind and downwind cells of the face weighted by a factor of 1000 relative to the other cells of the stencil so that the fit is most accurate near the face. (We have not found the results to be sensitive to this weighting factor, with values ranging from 10 to $10^9$.) The singular value decomposition needs to be done only once per cell for a fixed mesh at the beginning of the simulation, leaving just one multiply operation per cell of each stencil per time step to calculate an interpolation.

3. Meshes with and without local refinement

We create four different mesh structures at various resolutions both with and without local mesh refinement:
hexagonal icosahedral, triangular icosahedral, cubed sphere, and reduced latitude–longitude. Some of the coarsest of each of these that include a disc of refined mesh are shown in Fig. 3. The refined disc has twice the resolution of its surroundings, has a radius of 258, and is centered at 308N, 908E.

We create quasi-uniform hexagonal–icosahedral meshes using the mesh generator of Thuburn (1997). Triangular–icosahedral meshes are simply the dual of the hexagonal meshes with hexagon and pentagon centers becoming triangle vertices.

The nonuniform Delaunay meshes of triangles (Fig. 3d) are generated by the algorithm described in the appendix based on a “desired resolution” field defined over the globe. The nonuniform Voronoi meshes of polygons (Fig. 3a) are the Voronoi duals of the meshes of triangles. That is, the center of the circle enclosing each triangle becomes a vertex in the mesh of polygons. These meshes consist entirely of pentagons, hexagons, and heptagons.

All the edges of the cubed-sphere mesh without refinement sweep out an equal angle as described by Fournier et al. (2004). The 2:1 refinement as shown in Fig. 3b is achieved by treating the large squares that are adjacent to two small squares as if they had five edges, two of which are in line.

The reduced latitude–longitude meshes reduce the number of longitudes by a factor of 2 when the mesh spacing reaches half of that at the equator. There is also a polygonal cell at each pole in order to improve the accuracy of the cross-polar flow (Fig. 3c). The coenings toward the poles and the refined disc are achieved in the same way as for the cubed sphere.

Some statistics of all the meshes that we use are given in Table 1: the total number of cells, total number of faces between cells, the distance between adjacent cell centers at the equator (δx eq), and, for the reduced latitude–longitude meshes, the latitudes where the number of longitudes reduces by a factor of 2. The distance between cell centers is not necessarily a good indicator of accuracy or computational cost but is given for ease of comparison with other models. Cost depends on the number of cells, the number of faces, and the number of cells in each stencil.

We have not computed results on full latitude–longitude meshes for two reasons; first, because we have not implemented the necessary polar filter and, second, because the numerical schemes as implemented behave badly if a very large number of cells meet at one point or meet at one polygonal cell at each pole.

4. Shallow-water test cases


We first look at the accuracy and efficiency of representing steady-state geostrophically balanced flow [test case 2 in Williamson et al. (1992)] on the different mesh structures defined in section 3. We show results both
with and without the arbitrary circular refinement pattern of radius 25° at 30°N. All simulations use a 15-min time step for consistency with case 5 in section 4b. This time step gives very low-flow CFL numbers, $\delta t |\phi|/|\nabla\phi| \delta x$, no greater than 0.2 on the finest meshes. We are therefore comparing only spatial truncation errors between different meshes, not temporal truncation errors. Figure 4 shows the height errors after 5 days of steady-state flow rotated 45° from the geometric north pole for resolutions of all of the mesh structures that imply similar

![Fig. 4. Total height contours and shaded height errors after 5 days for global steady-state geostrophic flow rotated 45° from north (Williamson et al. 1992). The time step is 15 min for all meshes.](image)
computational costs, both with and without a refined disc. The maximum errors of around 10 m should be compared with the total equator-to-pole height difference of 1900 m on these moderate resolution meshes. However, for all of the structures, mesh refinement degrades the solution (and at extra cost). In contrast, degree-7 spectral-element discretization with local refinement increases the global accuracy even when the refinement is placed arbitrarily rather than based on local errors (St-Cyr et al. 2008).

The rotated hexagonal and triangular meshes appear not to project strongly onto the error patterns (Figs. 4a and 4g), unlike the rotated cubed-sphere and reduced latitude–longitude meshes (Figs. 4c and 4e). When refined discs are introduced, wave trains of errors form downstream, increasing the global errors despite the

**FIG. 5.** (left) Normalized root-mean-square or $\ell_2$ error norms for the global steady-state geostrophic flow after 5 days and the flow over a mountain after 15 days [test cases 2 and 5 of Williamson et al. (1992)] as a function of CPU time and (right) the number of cells on various mesh structures: (a) steady state, not rotated, (b) steady state, rotated 45°, and (c) isolated mountain test case.
improved resolution of part of the domain. This is in line with Vichnevetsky (1987) and Vichnevetsky and Turner (1991) who explain the modifications to wave dispersion when resolution changes.

Using the cubed-sphere mesh, errors are generated as the flow alternates between being aligned with and at $45^\circ$ to the mesh (Fig. 4c). Otherwise, the cubed-sphere mesh gives good accuracy in comparison to the reduced latitude–longitude meshes. Cube edges and cube corners can also slightly increase the errors even for much higher-order methods [e.g., order-15 24-element computation of scalar advection in Fig. 7 of Taylor et al. (1997)].

The mesh-coarsening patterns toward the poles of the reduced latitude–longitude mesh lead to large errors because so many coarsening boundaries occur within a few cells of each other. For this mesh, the arbitrary circular refinement pattern makes little difference to the errors. It should be noted that a Fourier filter can be used with a reduced latitude–longitude mesh, which may improve accuracy. This has not been done here.

The mesh of triangles is more prone to grid-scale noise, especially at the centers of the flow (i.e., inside the lowest height contours in Figs. 4g and 4h), where the wind is very light and the upwinding introduces little numerical diffusion. The staggered component of the algorithm presented in section 2b(3) is not beneficial for representing geostrophically balanced flow on triangles; a solution for a discretized geostrophic balance does not exist on the triangular C grid because there are not enough velocity degrees of freedom (J. Thuburn 2009, personal communication; information online at www.met.rdg.ac.uk/~sws02hs/Newton/Thuburn_waves.pdf).

The different mesh structures have different computational costs for the same number of cells or for the same nominal resolution because of the differing numbers of faces per cell and the differing stencil sizes for the higher-order differencing (see section 2c). The important metric is the accuracy for the computational cost. We plot the normalized root-mean-square or $L_2$ error norm [as defined in Williamson et al. (1992)] as a function of computational cost (CPU time) and as a function of the number of cells for each mesh structure and for various resolutions in Figs. 5a and 5b for this case both unrotated with respect to the geometric north pole, and rotated $45^\circ$. Lines showing first- and second-order convergence with resolution are also shown on these graphs, assuming that both the CPU time and total number of cells are proportional to $1/\delta x^2$ for a constant time step. Results on all mesh structures converge with second-order accuracy. These graphs show the superior accuracy for cost of using hexagons and the degradation in the solution when an arbitrary refinement pattern is added. The meshes of triangles appear to give poor results when comparing the accuracy for a number of cells, but this is merely because it takes many more triangles to fill the space than other cells. The results are cheap per triangle, so when comparing accuracy for cost, triangles give similar results to the meshes of quadrilaterals.

When the flow is not rotated with respect to the geometric north pole, the reduced latitude–longitude mesh without a refinement pattern performs as well as the hexagonal mesh because additional errors are not introduced as the flow does not pass over the coarsening patterns; rather, it travels parallel to them. However, when rotated, reduced latitude–longitude meshes give the worst accuracy for cost. This demonstrates how results can be misleadingly good on test cases with flow aligned with the mesh.

The cost on each mesh structure using the quadratic and cubic fit schemes described in section 2c can be understood by considering the size of the stencil necessary for the schemes. For upwind differencing from cell centers onto face centers, 8 cells are included in the stencil on regular square parts of the meshes and 10 on regular hexagonal regions, but 13 triangles are used so as to include a well-placed upwind cell of the upwind cell. This can make the solution on triangles more expensive than the solution on hexagons for meshes with the same number of faces (as opposed to the same number of cells). However, meshes of hexagons lead to seven nonzero elements per row in the matrix to be solved whereas triangles lead to four, so the matrices resulting from the meshes of triangles will be cheaper to solve.

The stencil sizes can explain why meshes of hexagons are more cost effective than triangles. But based purely on the stencil size and number of nonzero elements per matrix row, meshes of squares should be more cost effective than hexagons. However, to mesh the surface of a sphere with squares, larger distortions must be introduced than using icosahedra, such as at cube corners or latitude–longitude-coarsening boundaries. Alternatively, for a full latitude–longitude mesh, very thin cells are used toward the poles, which increases cost. This then explains the superior behavior of the hexagonal meshes, particularly over the rotated meshes of squares.

b. Williamson et al.’s (1992) case 5: Flow over an isolated mountain

It may be beneficial to use local mesh refinement over orography. This is tested using the four mesh structures and different refinement patterns given in section 3, on Williamson et al.’s (1992) test case 5—westerly flow over an isolated mountain. Much of the benefit of using high resolution for orography is in capturing small-scale
FIG. 6. Height errors after 15 days for the westerly flow over a midlatitude mountain (Williamson et al. 1992). Time step is 15 min for all models. Errors are calculated as differences from the initial conditions in comparison with STSWM T426 differences from the initial conditions. Dark-shaded contours mark 2 m every 4 m. Light-shaded dashed contours mark −2 m every −4 m.
Table 2. Conservation errors for test case 5 of Williamson et al. (1992): shallow-water westerly flow over a midlatitude mountain after 15 days. Values from other models are estimated from an examination of published graphs and are therefore not precise.

<table>
<thead>
<tr>
<th>Model</th>
<th>Grid and resolution</th>
<th>Space order</th>
<th>$\delta t$ (min)</th>
<th>Normalized mass ($10^{-6}$)</th>
<th>Normalized energy ($10^{-6}$)</th>
<th>Potential vorticity ($s^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtmosFOAM</td>
<td>208 km</td>
<td>2</td>
<td>15</td>
<td>$4.1 \times 10^{-10}$</td>
<td>$-5.1 \times 10^{-6}$</td>
<td>$3.2 \times 10^{-11}$</td>
</tr>
<tr>
<td>STSWM (Jakob et al. 1993)</td>
<td>Lat–lon T63</td>
<td>15</td>
<td>15</td>
<td>$6.2 \times 10^{-11}$</td>
<td>$4.6 \times 10^{-6}$</td>
<td>$5.0 \times 10^{-17}$</td>
</tr>
<tr>
<td>AtmosFOAM</td>
<td>84 km</td>
<td>2</td>
<td>0.75</td>
<td>$1.1 \times 10^{-10}$</td>
<td>$-1.8 \times 10^{-6}$</td>
<td>$6.8 \times 10^{-21}$</td>
</tr>
<tr>
<td>Spectral element (Taylor et al. 1997)</td>
<td>Cubed sphere</td>
<td>8</td>
<td>0.75</td>
<td>$&lt;10^{-8}$</td>
<td>$-4 \times 10^{-9}$</td>
<td>$&lt;10^{-10}$</td>
</tr>
<tr>
<td>AtmosFOAM</td>
<td>250 km</td>
<td>2</td>
<td>15</td>
<td>$1.6 \times 10^{-11}$</td>
<td>$1.0 \times 10^{-6}$</td>
<td>$5.3 \times 10^{-11}$</td>
</tr>
<tr>
<td>Spectral element (Thomas and Loft 2002)</td>
<td>Cubed sphere</td>
<td>8</td>
<td>?</td>
<td>$&lt;10^{-11}$</td>
<td>$-10^{-6}$</td>
<td>$-10^{-17}$</td>
</tr>
<tr>
<td>AtmosFOAM</td>
<td>250 km</td>
<td>2</td>
<td>30</td>
<td>$8.3 \times 10^{-11}$</td>
<td>$-1.7 \times 10^{-5}$</td>
<td>$1.3 \times 10^{-10}$</td>
</tr>
<tr>
<td>PV (Thuburn 1996)</td>
<td>Hexagons 1–3</td>
<td>15</td>
<td>15</td>
<td>$-1.6 \times 10^{-11}$</td>
<td>$-9.4 \times 10^{-5}$</td>
<td>$9.1 \times 10^{-11}$</td>
</tr>
<tr>
<td>AtmosFOAM</td>
<td>120 km</td>
<td>2</td>
<td>1</td>
<td>$10^{-9}$</td>
<td>$-4^{-4}$</td>
<td>$5 \times 10^{-7}$</td>
</tr>
<tr>
<td>ICON (Bonaventura and Ringler 2005)</td>
<td>Triangles 1–2</td>
<td>15</td>
<td>15</td>
<td>$4.7 \times 10^{-11}$</td>
<td>$2.8 \times 10^{-6}$</td>
<td>$4.2 \times 10^{-11}$</td>
</tr>
<tr>
<td>AtmosFOAM</td>
<td>156 km</td>
<td>2</td>
<td>15</td>
<td>$4.7 \times 10^{-11}$</td>
<td>$2.8 \times 10^{-6}$</td>
<td>$4.2 \times 10^{-11}$</td>
</tr>
<tr>
<td>MMFV (Chen and Xiao 2008)</td>
<td>Cubed sphere</td>
<td>4</td>
<td>?</td>
<td>0</td>
<td>$-4 \times 10^{-6}$</td>
<td>—</td>
</tr>
<tr>
<td>DG (Läuter et al. 2008)</td>
<td>180 km</td>
<td>$k = 6$</td>
<td>$\delta t_{CFL}$</td>
<td>0</td>
<td>$5.5 \times 10^{-8}$</td>
<td>—</td>
</tr>
</tbody>
</table>

Diabatic effects, such as orographically induced rainfall, which are not present in the shallow-water equations.

Simulations on all meshes use a time step of 15 min in order to compare with the spectral-model solution at triangular truncation 63 (T63) Jakob et al. (1993). This time step gives very low flow CFL numbers—no greater than 0.2 on the finest meshes. T63 corresponds to a grid-point resolution of 1.875°, or 210 km, at the equator.

The reference solution and errors after 15 days are shown in Fig. 6 for the spectral-model [the Spectral Transform Shallow-Water Model (STSWM); Hack and Jakob (1992)] at truncation T63 and for the eight AtmosFOAM meshes that have resolutions similar to T63 but with and without double resolution around the mountain. The reference solution is calculated at T426 (0.28° or 31 km at the equator) using a version of STSWM revised by P. Ripodas of the Deutscher Wetterdienst (information online at http://www.icon.enes.org/swm/stswm). The errors shown in Fig. 6 are calculated as differences from the initial conditions in comparison to differences from the initial conditions of the reference solution. This means that errors in the initial conditions represented at lower resolution are not included, which means that no spectral ringing is seen in the spectral model errors. The spectral model results are interpolated from the T426 grid points onto the AtmosFOAM meshes using the bicubic interpolation code also available from the Deutscher Wetterdienst. The bicubic interpolation from grid points onto cell centers is less accurate than evaluating the solution at AtmosFOAM cell centers directly from spectral coefficients, but it does avoid generating spectral ringing. The accuracy of the bicubic interpolation is limited to second order by assuming that the cell average value is represented by the cell center value.

The most accurate AtmosFOAM results in Fig. 6 come from using uniform hexagonal or uniform cubed-sphere meshes. With these configurations, errors are similar to STSWM at T63. The reduced latitude–longitude mesh gives larger errors toward the north pole and the triangular mesh gives larger errors globally (despite having similar computational cost). Errors appear to increase slightly for all mesh structures when extra resolution is added around the mountain, and these increased errors come at the additional cost of computing the solution on more cells. Ringler et al. (2008) also found that local resolution does not benefit the global solution when solving this case with low-order finite volumes.

To compare the accuracy for the cost and the accuracy for the number of cells of the different mesh structures more rigorously, we plot the $L2$ error norm (Williamson et al. 1992) as a function of the CPU time and number of cells in Fig. 5c for different resolutions of each of the mesh structures. Lines showing first- and second-order convergence are also shown. For the graph with respect to the number of cells we also include the STSWM rerun at T21, T42, and T63 assuming 32 × 64, 64 × 128, and 96 × 192 grid points, respectively. These are the number of grid points used at these truncations but typically spectral-transform models use 50% fewer grid points to achieve the same accuracy than does a low-order finite-volume model (Williamson 2008). However, the right-hand side of Fig. 5c shows that AtmosFOAM achieves better accuracy than STSWM at T42 and T63 for the
same number of cells for grid points. It is only at T21 that
STSWM outperforms most of the AtmosFOAM meshes.
AtmosFOAM results generally show second-order con-
vergence with resolution, apart from at high resolution
(high computational cost) where both AtmosFOAM and
STSWM are converging more slowly to the T426 solution.
At low resolution (short CPU time), local refinement
of the mountain gives better price performance only for
the hexagonal and triangular meshes that include grad-
ual rather than abrupt refinement. However, once the
resolution is increased, the mountain is sufficiently well
resolved and the additional errors from the refinement
pattern lead to inferior price performance.
The hexagonal mesh is most cost effective, followed
by the cubed sphere, then the reduced latitude–longitude,
and then the triangular icosahedral mesh. The reduced
latitude–longitude mesh does relatively well in this case
because the flow is not rotated in comparison to the
geometry and so much of the flow is aligned with the
mesh. We have not plotted errors of STSWM as a func-
tion of CPU time on the same graph as the two codes
were not run on the same computer and OpenFOAM is
a 3D code running a 2D simulation, so there is consid-
erable overhead in taking the 2D projection of the 3D
operators.
As would be expected of a spectral model, the results
are particularly accurate at low resolution. However,
as the resolution is increased, the large-scale features
are adequately represented by the lower-order finite-
volume model and the forcing from the cone-shaped
mountain is also better represented.
Conservation of mass, total energy, and potential
vorticity have been compared with some other models
and the results are shown in Table 2. Where possible, the
mesh and time step that we use for AtmosFOAM match
the other published models. The AtmosFOAM conser-
vation is within the range of the other models. Some of
the smaller numbers will be influenced by round-off
errors and different models were run with different
machine precisions.

This test case gives interesting results because the initial conditions are very nearly in balance but physically unstable, and so the solution is sensitive to the initial conditions but also very sensitive to geometrically varying truncation errors. St-Cyr et al. (2008) present the results of this test case without explicit diffusion using a spectral-element, seventh-order-accurate model on a cubed-sphere mesh and a block-structured finite volume on a latitude–longitude mesh, both with and without adaptive mesh refinement. Using both models on uniform meshes, good results are obtained once the resolution reaches 140 km (1.25°). At coarser resolutions the spectral-element model creates a spurious “wavenumber-4 signal invoked by the computational cubed-sphere mesh” (St-Cyr et al. 2008, p. 1916) and the finite-volume model tends to smooth out the wave. The finite-volume model has the advantage that the mesh is aligned with the flow and so does not trigger any spurious barotropic instability.

We use this test case to examine the impacts of local mesh refinement and different mesh structures on the triggering of barotropic waves. It is possible to adaptively refine meshes so that regions of high vorticity are always well resolved and the jet does not pass over changes in resolution as in St-Cyr et al. (2008) and Weller (2009), but this is not what we are doing. Instead, we examine the influence of a statically refined disc that may be motivated by, for example, the need for a local weather forecast or the resolution of a locally produced tracer.

The vorticity after 6 days on the hexagonal–icosahedron, cubed-sphere, and reduced latitude–longitude meshes with and without a disc of local refinement of radius 25° centered at 30°N, 90°E are shown in Fig. 7. (Results using meshes of triangles are not computed because, from sections 4a and 4b, triangles appear to have no benefits over polygons.) The base resolution for each mesh structure is 120–140 km (≈1.25°) with 60–70 km (≈0.625°) in the refined region. We also show results using the reduced latitude–longitude mesh rotated 30°. These are compared with an AtmosFOAM reference solution calculated on a 35-km (≈0.3125°) reduced latitude–longitude mesh (which is similar to the inviscid T341 solution of (Galewsky et al. 2004) but without the Gibb’s oscillations).

At 1.25° resolution, all of the meshes apart from the unrotated uniform latitude–longitude mesh produce spurious barotropic instability due to changes with the longitude of the mesh structure. These are worst on the cubed-sphere mesh on which the spuriously triggered waves are larger than the wave at 270° longitude, which is supposed to have grown the most. The problem with the unrotated cubed-sphere mesh when using this finite-volume model is that the jet tends to follow the mesh rather than the lines of constant latitude. More satisfactorily, for both hexagonal meshes, the latitude–longitude mesh with refinement and the rotated latitude–longitude mesh, the spurious barotropic waves have grown less than the real barotropic wave. These results highlight the lack of useful information from using this case to demonstrate the merits of a numerical method if results are only shown on a latitude–longitude mesh aligned with the flow. Also of note, only the unrotated reduced latitude–longitude mesh without refinement can maintain the zonal symmetry of balanced initial conditions without the perturbation to trigger the wave (not shown).

These results suggest that if maintaining steady-state and unstable jets at any orientation are important, this is not possible with a second-order-accurate model at moderate resolution. Higher-order accuracy or higher resolution must be used as for the seventh-order-accurate spectral-element results presented by St-Cyr et al. (2008). However, Läuter et al. (2008) say of this case: “For the days 4–6 strong gradients develop in the regions of instability and benefit the low-order [schemes].”

5. Summary and conclusions

We have presented AtmosFOAM, which solves the rotating shallow-water equations on any mesh of the sphere. AtmosFOAM uses a second-order finite-volume formulation with prognostic variables of height and momentum collocated at cell centers. To remove grid-scale oscillations of the A grid, the mass flux between cells is advanced from the old momentum using the momentum equation. Quadratic and upwind-biased cubic differencing are used as explicit corrections to an implicit solution that uses linear differencing. This keeps the matrices involved in the implicit solution sparse and diagonally dominant and, hence, cheap to solve while exploiting some accuracy gains of higher-order schemes.

Much of our analysis of these results is quite negative, but the AtmosFOAM results have also been compared with those of a selection of other published results, and AtmosFOAM compares well. For each published model, we compare AtmosFOAM on a similar mesh and with the same or a longer time step.

We present results of some standard test cases on the cubed-sphere and reduced latitude–longitude meshes, both with and without block-structured, static 2:1 refinement, and on hexagonal and triangular icosahedral meshes and Voronoi meshes of polygons and Delaunay
meshes of triangles with gradual static local refinement. The latitude–longitude meshes give very good accuracy for cost when the flow is aligned with the mesh. Otherwise, the hexagonal–icosahedral meshes of polygons are the most cost effective. If a local reduction of truncation errors is required, then this can be achieved with local refinement, but this introduces errors where the mesh changes resolution, and this can actually degrade the global accuracy. This confirms the results of St-Cyr et al. (2008), who found increased global errors due to the inclusion of an unnecessarily refined mesh in a low-order finite-volume model. However, this problem is alleviated when St-Cyr et al. (2008) test a higher-order spectral-element model.

We present solutions of the barotropically unstable jet test case of Galewsky et al. (2004) and find that mesh irregularities can trigger barotropic instability of a similar magnitude to those triggered by the initial perturbation, for the mesh resolution of about 1.25° that we use for this case. Using an unrotated reduced latitude–longitude mesh at 1.25° gives very good results for this case, but rotating the mesh, including a 2:1 refined region, or using a hexagonal–icosahedral mesh triggers spurious waves that are nearly as big as the correctly initialized wave. When the barotropically unstable jet is discretized on a cubed sphere, the initial jet very nearly follows the mesh lines. Subsequently, the jet tends to follow the cubed-sphere mesh lines, which dip toward the equator at the cube corners. This triggers larger barotropic instabilities than the perturbations in the initial conditions.

Gradual Voronoi and Delaunay mesh refinements lead to better accuracy than abrupt block-structured 2:1 refinement. However, our mesh-generating algorithms are limited in how gradual they can make the refinement. Factor-of-2 increases in resolution tend to occur over two or three cells. We would like to create meshes of polygons with more gradual mesh grading, which might mean using spherical–centroidal Voronoi tessellations or relaxing the Delaunay constraint to allow more anisotropic cells and refining based on tensor rather than scalar criteria. This may improve the accuracy gains possible from including local mesh refinement.

Acknowledgments. Author HW acknowledges support from NCAS Climate and is particularly grateful to Julia Slingo for her support. We are grateful to OpenCFD for OpenFOAM and for parallelizing some of the additional algorithms described here. Particular thanks are given to Mattijs Janssens at OpenCFD and Graham Macpherson at Strathclyde University. Author AF thanks M. Taylor for useful discussions and the University of Reading’s Department of Meteorology for visiting scientist support.

APPENDIX

Delaunay Mesh Generation

Delaunay meshes in general consist of nonoverlapping triangles in which the circle that goes through the three vertices of a triangle does not enclose a vertex of any other triangle. This leads to triangles that are as close as possible to equilateral for a given set of vertices, which may have computational advantages. Delaunay triangulations are computed using the Computational Geometry Algorithms Library (CGAL; information online at http://www.cgal.org).

The vertices and mesh are distributed on the sphere using a simple algorithm that appears to share some features of a spherical–centroidal Voronoi tessellation (e.g., Ringler et al. 2008), but a direct comparison is the subject of future work. Our algorithm works as follows:

(i) Start from an initial set of points and find the Delaunay triangulation using CGAL. At every step of the algorithm, for every point insertion, removal, or movement, CGAL recomputes the Delaunay triangulation based on local changes.

(ii) Read in a field describing the desired (isotropic) mesh spacing as a function of position over the globe. This controls the ultimate mesh spacing more directly than the density function as described in Ringler et al. (2008), although our technique may not be optimal.

(iii) Loop through all the vertices on the sphere and remove some from regions that are too densely packed. This is decided as follows:

- We consider vertex \( v \), where the desired resolution is \( \delta_v \).
- Vertex \( v \) is connected to \( n \) other vertices with edges of length \( \delta_i \). If \( (1/n)\sum_i \delta_i^2 < 0.7\delta_v^2 \), then vertex \( v \) is removed. (The factor 0.7 is empirically beneficial.)

(iv) Loop through all the triangle edges on the sphere and add a vertex at the midpoint if the edge length, \( \delta_e \), is greater than \( \frac{2}{3} \) times as long as the desired edge length, \( \delta_d \), at the edge midpoint. This criterion means that the edge lengths after adding vertices are always closer to \( \delta_v \). This loop is repeated until no more vertices are added.

(v) Remove vertices that are only connected to exactly four other vertices. The vertices removed are associated with edges shorter than the surrounding edges (although they are not short enough to be removed in step iii). This step removes squares from the Voronoi dual mesh.

(vi) For vertices that are directly connected to eight or more other vertices, add another vertex close to
the vertex under consideration. This removes octagons and polygons with more sides from the Voronoi dual. Procedures v and vi do not guarantee that there are no squares, octagons, or polygons with more sides in the final Voronoi mesh, as more can be created as others are removed. However, in practice performing steps v and vi once does remove these shapes.

(vii) Each triangle edge is assumed to be a critically damped spring with unstretched length equal to the desired mesh resolution at the location of the center of the edge. These springs undergo five relaxation iterations following Tomita et al. (2002).

For the accuracy and stability of the shallow-water equation solver, we want meshes with gradual refinement. The mesh of triangles in Fig. 3d was created using this algorithm and a desired resolution field consisting of $\delta_2 = 300$ km inside a disc of radius 20° and $\delta_4 = 700$ km outside this radius (Fig. A1a), which is then expanded outward and smoothed (Fig. A1b). The resulting mesh (Fig. 3d) is not as smooth as the desired resolution because of the tendency of the Delaunay algorithm to make equilateral triangles. Smoother grading might be achieved using a definition of the anisotropic mesh spacing and weighted Delaunay triangulation or by using spherical–centroidal Voronoi tessellations.

REFERENCES


