JOURNAL OF COMPUTATIONAL PHYSICS **124,** 93–114 (1996)
ARTICLE NO. 0047

The "Cubed Sphere": A New Method for the Solution of Partial Differential Equations in Spherical Geometry

C. RONCHI,*,[1] R. IACONO,* AND P. S. PAOLUCCI†

*Unità di Modellistica Numerica, ENEA, Rome, Italy and †Ape100 Group, INFN, Rome, Italy

Received January 27, 1995; revised July 24, 1995

A new gridding technique for the solution of partial differential equations in spherical geometry is presented. The method is based on a decomposition of the sphere into six identical regions, obtained by projecting the sides of a circumscribed cube onto a spherical surface. By choosing the coordinate lines on each region to be arcs of great circles, one obtains six coordinate systems which are free of any singularity and define the same metric. Taking full advantage of the symmetry properties of the decomposition, a variation of the composite mesh finite difference method can be applied to couple the six grids and obtain, with a high degree of efficiency, very accurate numerical solutions of partial differential equations on the sphere. The advantages of this new technique over both spectral and uniform longitude–latitude grid point methods are discussed in the context of applications on serial and parallel architectures. We present results of two test cases for numerical approximations to the shallow water equations in spherical geometry: the linear advection of a cosine bell and the nonlinear evolution of a Rossby–Haurwitz wave. Performance analysis for this latter case indicates that the new method can provide, with substantial savings in execution times, numerical solutions which are as accurate as those obtainable with the spectral transform method.

1. INTRODUCTION

Due to the intrinsic curvature properties of the sphere, the choice of an appropriate coordinate system is crucial in allowing the formulation of an efficient and accurate numerical method for solving partial differential equations in spherical geometry. Although arguably the most natural coordinates for representing phenomena on a sphere, spherical polar coordinates present several disadvantages (collectively known as the "pole problem") when used for numerical computations over the entire spherical surface. The coordinate singularity at the poles (longitude becomes multivalued) is generally reflected by the presence in the equations of terms that become unbounded. Furthermore, although the vector velocity is well defined everywhere, its polar spherical components are undefined at the poles. These facts prevent a straightforward implementation of finite-difference and spectral methods previously developed in Cartesian coordinates and they have been the main motivation for the development of alternative techniques for solving partial differential equations on the sphere (see *Browning et al.* [4] for a detailed discussion).

In one of his pioneering papers on large-scale numerical weather prediction, *Phillips* [16] put forward a list of desirable features that a mapping of the sphere and the numerical scheme based on it should have in order to be used with success for global forecasting purposes:

1. The mapping should be free of any singularities.
2. The mapping should preserve the general form of the hydrodynamic equations.
3. The physical grid defined on the spherical surface should be as close as possible to a regular equidistant grid.
4. The equator should be a coordinate line of the grid network.

We believe that two items should be added to this list in view of the developments in the field of global atmospheric modeling since *Phillips*' paper, namely:

5. The numerical scheme implemented should be competitive, both in terms of execution time and accuracy, with the spectral transform method.
6. The numerical algorithms used should be scalable and their communication requirements should allow an efficient implementation on massively parallel architectures.

For obvious reasons, the main impulse to the development of numerical methods for solving partial differential equations on the sphere has come from the fields of meteorology and atmospheric dynamics. The first approaches to global numerical weather forecasting were grid point methods based on the use of conformal projections, which preserve the angle between two intersecting curves. This property reflects on a greater symmetry of the equations of fluid motion when expressed in terms of Cartesian coor-

[1]Author to whom correspondence should be addressed at AMB/GEM/CLIM SP 110, ENEA-C.R. Casaccia, Via Anguillarese 301, 00060 S. Maria di Galeria, Rome ITALY; e-mail address: corrado@cicero.casaccia.enea.it.

0021-9991/96 $18.00

dinates on the map projection. But given the fact that no single conformal mapping can map the surface of the sphere on a finite section of the plane, only by combining several conformal projections is one able to represent without singularities the entire spherical surface. For example, *Phillips* [16–18] used two stereographic projections for the pole regions coupled with a Mercator projection for latitudes equatorward of some boundary latitude. In the regions of projection overlap the values of the boundary points for one coordinate system were obtained by interpolations using interior points of the adjacent coordinate system. Although this method was shown to provide good results with a careful definition of the finite difference scheme and of the interpolation procedure, it never gained popularity mainly because its overall complexity and cost [29].

In the late 1960s, a number of attempts were made to construct alternative methods based on a geometric decomposition of the spherical surface into several regions defined from central nonconformal projections of circumscribed regular polyhedrons [23, 28, 24]. As in the case of coupled conformal projections, this approach removes the problem of geometric singularities, but it has the additional advantage of allowing the construction of quasi-uniform spherical grids. Unfortunately, the use of polyhedral-gnomic projections was soon abandoned, either because of the insurgence of numerical instabilities at the internal boundaries or because of the inherent complications of the methods. In the 1970s, the strong advances in the spectral method [14] and in techniques for uniform longitude–latitude grids [1] further contributed to slow down the investigation of alternative numerical methods on the sphere.

It is, nevertheless, a fact that the numerical methods currently used to solve fluid flows in spherical geometry fall short of satisfying all six criteria listed above. In particular, although the spectral transform method is generally accepted as the basis of operational numerical weather prediction and global climate models, the computational cost of the Legendre transform [7] and the tendency to develop unphysical features in the numerically predicted fields [20] have recently contributed to question the predominance of the spectral method in global atmospheric modeling. On the other hand, grid point methods based on a uniform longitude–latitude representation of the sphere still suffer from restrictions on both accuracy and efficiency, essentially because of the problems associated with the structure of the coordinate system used. For example, the convergence of the meridians near the poles results in a clustering of grid points and imposes the use of extremely small time steps when employing explicit time differencing schemes. Alternatively, some filtering of the unstable short wavelengths must be applied at the highest latitudes in order to relax the restriction on the time step. Nevertheless, it is well known that such filtering procedures can severely degrade the accuracy of the numerical solution at all latitudes [19]. More fundamentally, the strongly nonuniform placement of the grid points in physical space produced by a uniform longitude–latitude grid poses by itself a limit to the degree of accuracy attainable by the numerical solutions.

In this paper, we tackle the pole problem by utilizing a new set of transformations, similar to the cubic–gnomic projections proposed by *Sadourny* [24]. We present a new gridding technique based on a decomposition of the sphere into six identical regions, obtained by projecting the sides of a circumscribed cube onto the spherical surface. Each of the six nonconformal mappings defined in this way is free of any singularity, defines the same metric, and allows the construction of a quasi-uniform grid.

In his study, *Sadourny* employed the cubic–gnomic projections as the simplest example of a decomposition of the sphere using circumscribed regular polyhedrons. As noted above, several years before, *Phillips* presented a method in which three conformal coordinate systems were coupled through an interpolation procedure. Nevertheless, *Sadourny* discarded the use of interpolations to couple the six numerical domains obtained by applying cubic–gnomic projections, based on the consideration that the design of a globally conservative scheme was extremely complicated by the use of interpolations. He, instead, opted for a coupling scheme based on the use of one-sided differencing formulae on the boundary points. Unfortunately, such a procedure caused a decay of accuracy at the internal boundaries and resulted in the development of perturbations with two-grid interval wavelengths. Given the poor results, no further tests using the cubic–gnomic projection have been carried out, until the present study.

Our method, which we named the ''cubed sphere,'' presents several innovations over the original attempt by *Sadourny*. First, the method we employ to couple the six coordinate systems is based on a variation of the composite mesh method introduced by *Starius* [25, 26]. Second, we calculate analytically the coordinate transformations and the corresponding metric tensors for the six projections. This allows one to explicitly write the equations to solve in the new coordinate systems and to avoid the numerical computation of the metric tensor elements. Third, we exploit to the fullest extent all the symmetry properties of the decomposition. This greatly simplifies the construction and the coupling of the different component meshes. Finally, we discuss the implementation of the ''cubed sphere'' technique on massively parallel computers and prove its efficiency for processing networks with 3D first neighbour connecting topologies.

One of the main new ingredients of our method over the one proposed by *Sadourny* is the use of the composite mesh method, which allows us to obtain globally stable

and accurate numerical solutions over the entire spherical surface. In the composite mesh method, partial differential equations are solved in regions with complicated geometry by representing the domain of interest with several overlapping grids, each of which can be defined from a metric and a coordinate system chosen to suit the particular structure of its boundary domain. Each region is then mapped onto a rectangular grid, where the equations can be solved using standard finite-difference techniques for regular meshes. The key to the stability of the method is generally believed to lie in the way the different grids are coupled together through the overlapping of all boundary points and through the interpolation procedure used to obtain the values for the time dependent variables at the stencil points (see *Henshaw* [10]). Compared to other boundary conforming techniques (such as the nested grid or refinement method) the composite mesh method has the advantage of not requiring a smooth and continuous matching of the different component grids and is therefore generally considered more flexible. On the other hand, the interpolation procedure and the need to evolve the equations also in overlap regions represent an additional burden, which can be a substantial fraction of the overall computational time. As we will see, the particular domain decomposition and gridding technique that we adopt minimize these drawbacks of the composite mesh method. In fact, we will show that the six component meshes can be stably and accurately coupled also by performing interpolations only in one dimension and by using a minimum amount of overlap, limited to the boundary lines between blocks. This is possible thanks to the remarkable symmetry properties of the coordinate systems and grids used on each of the six spherical regions (four equatorial and two polar).

Once the problem of how to lock together the different component meshes is solved, the task of coding the equations and debugging the finite difference version is completely equivalent to what would be done when using a single square grid. On the other hand, the use of six separate domains requires some additional care when trying to develop algorithms for global solvers, such as those typically needed in elliptic problems.

The gain in efficiency and simplicity of this procedure over a spectral code for hyperbolic-parabolic type of equations can be quite obvious. For example, the operation count for the Legendre transform (which scales with the cube of the number of grid points) is such that at high enough resolutions a grid point method (whose cost scales with the square of the number of grid points) will be substantially faster. It is *a priori* less evident whether the higher efficiency of grid point methods can be exploited to match, with comparable overall execution times, the accuracy of the spectral method. However, results of the classical Rossby–Haurwitz nonlinear test case for numerical approximations to the shallow water equations in spherical geometry will show that the ''cubed sphere'' method can achieve comparable or even higher accuracy than the spectral method, with substantial savings in execution time and using an equal total number of underlying grid points to cover the spherical surface.

The plan of the paper is the following. In Section 2, we will introduce the coordinate transformations, describe the metric defined on each region, give the expressions for the principal differential operators, and provide the transformation laws for vector functions between regions. In Section 3, we will describe the implementation of the composite mesh method as applied to our decomposition of the spherical surface. We will discuss its advantages both over more traditional approaches (spectral and finite-difference) and over the composite mesh method as applied to alternative mappings of the sphere, such as the one recently proposed by *Browning et al.* [4] based on two overlapping stereographic coordinate systems. In Section 4, we will show the results of two test problems (advection of a cosine bell over the poles and Rossby–Haurwitz waves) on single processor scalar/vector computers and compare the results of our integrations with those obtained from a code based on the spectral transform technique. Finally, in Section 5 we discuss the implementation of our new method on massively parallel systems and report preliminary performance data obtained on the APE100/Quadrics parallel computers [2, 3].

2. THE CUBED SPHERE

As we noted, the idea of decomposing a sphere into six identical regions, obtained by projecting the sides of a circumscribed cube onto a spherical surface, is not new. In the context of our work, the geometric decomposition of the spherical surface into six identical regions was suggested by one of us (P.S.P.) to allow efficient numerical simulations of atmospheric dynamics on the APE100/Quadrics massively parallel computers. We know of only one other study (*Russell and Eiseman* [22]), besides the original one by *Sadourny*, that applies such a decomposition to the solution of partial differential equations in spherical geometry.The main purpose of the work by *Russell and Eiseman* is the construction of a boundary conforming grid suitable for simulating ocean circulation dynamics on the Earth and is therefore quite different in scope from the present study. In fact, our goal is to develop a gridding technique that can be used to perform numerical computations for *any* problem which can be formulated as a set of partial differential equations on the sphere. In order to accomplish this, we will first analytically provide the coordinate transformations and the metric for each of the six regions (or ''blocks'' as we might also call them). Once the metric is known, it is a rather trivial (albeit cumbersome) exercise to derive all the needed differential

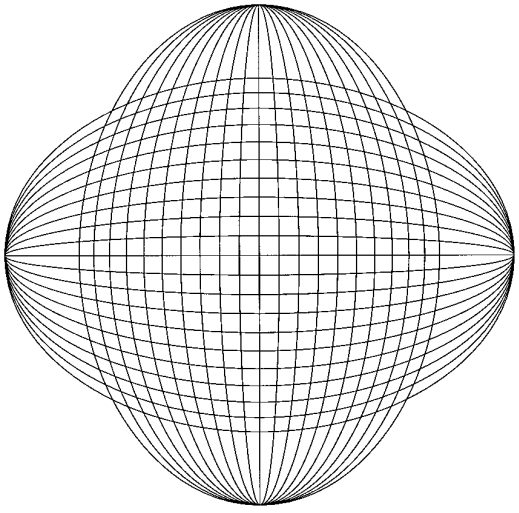

FIG. 1. Construction of one of the six meshes from two sets of angularly equidistant great circles.

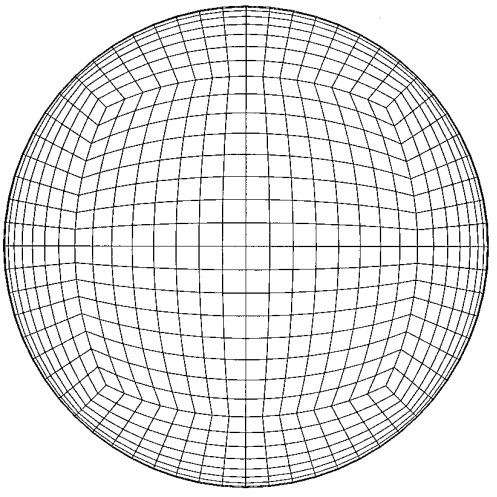

FIG. 2. View of the spherical surface after the procedure outlined in Fig. 1 has been repeated to produce each of the six component meshes.

operators and transformation laws for vector functions between regions.

A geometrically rigorous approach to the problem of how to cover a spherical surface with tiles shows that there are only five classes of inscribed polyhedrons, which produce a covering which is both regular and homogeneous: the tetrahedron, the cube, the octahedron, the dodecahedron, and the icosahedron. The covering can be obtained by a central projection of the sides and corners of the inscribed polyhedron on the spherical surface. There are several good reasons to pick the cube among the five possible regular polyhedrons. First, it is clear that, in terms of memory mapping, squares are the ideal domains to treat numerically. Furthermore, as we will see more in detail later, there is a very favourable balance between the number of internal boundaries and the number of operations to be performed on the boundary points in order to couple the six resulting grids.

Let us now introduce in a more intuitive and visual way the basic concept underlying the construction of the coordinate transformation for one of the regions. As shown in Fig. 1, the intersection of two sets of angularly equidistant great circles defines a gridded region, where each of the arcs of great circles can be associated to either a vertical or horizontal coordinate line. Notice that the coordinate system defined in this way is clearly nonorthogonal, since great circles intersect at 90° only along the equator. The fact that on the sphere locally parallel lines eventually intersect is a consequence of its intrinsic curvature properties and we will see that choosing a coordinate system that mirrors this behavior will turn out to be greatly advantageous. Repeating the procedure outlined in Fig. 1 six times and making the appropriate choices required to obtain regions of identical size, one is able to represent exactly the entire spherical surface (see Fig. 2). Following the notation introduced in Fig. 3, one can choose on each region the angular variables ξ and η to span the range $[-\pi/4, \pi/4]$ and so construct the angularly equidistant grids shown in Fig. 2.

We will now introduce the following auxiliary variables that allow us to write the expressions for the transformation laws and for the differential operators in a more concise form:

$$X \equiv \tan(\xi) \tag{1}$$

$$Y \equiv \tan(\eta) \tag{2}$$

$$\delta \equiv 1 + X^2 + Y^2 \tag{3}$$

$$C \equiv (1 + X^2)^{1/2} \tag{4}$$

$$D \equiv (1 + Y^2)^{1/2} \tag{5}$$

We can then write the transformation laws for the six regions as follows:

I (Equator),

$$X = y/x = \tan\phi$$

$$Y = z/x = 1/\tan\theta\cos\phi$$

$$r = (x^2 + y^2 + z^2)^{1/2} \tag{6}$$

$$\begin{pmatrix} A_\xi \\ A_\eta \end{pmatrix} = \begin{pmatrix} 0 & CD/\delta^{1/2} \\ -1 & XY/\delta^{1/2} \end{pmatrix} \begin{pmatrix} A_\theta \\ A_\phi \end{pmatrix} \tag{7}$$

II (Equator),

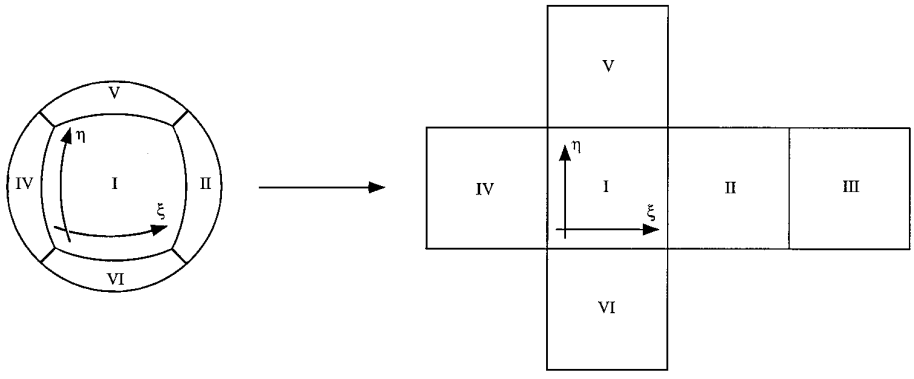

FIG. 3. The "cubed sphere" transformations map the spherical surface on the six sides of a cube, which are shown here opened on the plane. A regular square grid on each side is obtained as the image of the angularly equidistant grid shown in Fig. 2, with the angular variables ξ and η spanning the range $[-\pi/4, \pi/4]$.

$$X = -x/y = -1/\tan\phi$$
$$Y = z/y = 1/\tan\theta\sin\phi$$
$$r = (x^2 + y^2 + z^2)^{1/2} \tag{8}$$

III (Equator),

$$X = y/x = \tan\phi$$
$$Y = -z/x = -1/\tan\theta\cos\phi$$
$$r = (x^2 + y^2 + z^2)^{1/2} \tag{9}$$

IV (Equator),

$$X = -x/y = -1/\tan\phi$$
$$Y = -z/y = -1/\tan\theta\sin\phi$$
$$r = (x^2 + y^2 + z^2)^{1/2} \tag{10}$$

V (North Pole),

$$X = y/z = \tan\theta\sin\phi$$
$$Y = -x/z = -\tan\theta\cos\phi$$
$$r = (x^2 + y^2 + z^2)^{1/2} \tag{11}$$

$$\begin{pmatrix} A_\xi \\ A_\eta \end{pmatrix} = \frac{1}{(\delta-1)^{1/2}} \begin{pmatrix} DX & -DY/\delta^{1/2} \\ CY & CX/\delta^{1/2} \end{pmatrix} \begin{pmatrix} A_\theta \\ A_\phi \end{pmatrix} \tag{12}$$

VI (South Pole).

$$X = -y/z = -\tan\theta\sin\phi$$
$$Y = -x/z = -\tan\theta\cos\phi$$
$$r = (x^2 + y^2 + z^2)^{1/2} \tag{13}$$

$$\begin{pmatrix} A_\xi \\ A_\eta \end{pmatrix} = \frac{1}{(\delta-1)^{1/2}} \begin{pmatrix} -DX & DY/\delta^{1/2} \\ -CY & -CX/\delta^{1/2} \end{pmatrix} \begin{pmatrix} A_\theta \\ A_\phi \end{pmatrix}, \tag{14}$$

where $[\phi, \theta, r]$ are the longitude, colatitude, and radial coordinates of the standard polar system, (x, y, z) are the Cartesian coordinates, and

$$\mathbf{A} = A_\xi \hat{\mathbf{e}}_\xi + A_\eta \hat{\mathbf{e}}_\eta = A_\theta \hat{\mathbf{e}}_\theta + A_\phi \hat{\mathbf{e}}_\phi$$

is a generic vector on the sphere. The Jacobian matrices obtained from (6)–(14) determine the transformation rules between basis vectors and thereby the metric. We have chosen to employ on each block a set of unit base vectors $[\hat{\mathbf{e}}_\xi, \hat{\mathbf{e}}_\eta, \hat{\mathbf{e}}_r]$, such that $\hat{\mathbf{e}}_\xi \cdot \hat{\mathbf{e}}_\xi = \hat{\mathbf{e}}_\eta \cdot \hat{\mathbf{e}}_\eta = \hat{\mathbf{e}}_r \cdot \hat{\mathbf{e}}_r = 1$.

Notice that all equatorial regions have the same transformation matrix for vector quantities, given by Eq. (7). Note also that all coordinate transformations are free of any singularity and that in this sense there is no criteria to geometrically single out one region from another. In fact, each block with its assigned coordinate system can be obtained from any other through rigid rotations about at most two Cartesian coordinate axes. This is reflected in the desirable property that the metric tensor is the same on all regions:

$$\mathbf{g} = \begin{pmatrix} 1 & -XY/CD & 0 \\ -XY/CD & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{15}$$

Given two vectors $\mathbf{a} = [a_\xi, a_\eta, a_r]$ and $\mathbf{b} = [b_\xi, b_\eta, b_r]$, the metric employed defines the following scalar and cross products:

$$\mathbf{a} \cdot \mathbf{b} = a_\xi b_\xi + a_\eta b_\eta + a_r b_r - \frac{XY}{CD}(a_\xi b_\eta + a_\eta b_\xi) \tag{16}$$

$$\mathbf{a}\times\mathbf{b}=\frac{1}{\delta^{1/2}}[CD(a_\eta b_r-a_r b_\eta)+XY(a_r b_\xi-a_\xi b_r)]\hat{\mathbf{e}}_\xi + \frac{1}{\delta^{1/2}}[XY(a_\eta b_r-a_r b_\eta)+CD(a_r b_\xi-a_\xi b_r)]\hat{\mathbf{e}}_\eta + \frac{\delta^{1/2}}{CD}(a_\xi b_\eta-\mathrm{a}_\eta b_\xi)\hat{\mathbf{e}}_r, \tag{17}$$

where one can notice the presence of the cross terms due to the nonorthogonality of the basis vectors.

It is interesting to calculate the infinitesimal line elements along the ξ and η directions and the corresponding surface element. For infinitesimal displacements along one of the coordinate lines on a sphere of radius a, we have

$$dl_\xi=\frac{aD}{\delta}\frac{d\xi}{\cos^2\xi} \tag{18}$$

$$dl_\eta=\frac{aC}{\delta}\frac{d\eta}{\cos^2\eta} \tag{19}$$

$$dS=\frac{a^2}{\delta^{3/2}}\frac{d\eta\,d\xi}{\cos^2\eta\cos^2\xi}. \tag{20}$$

From these expressions it is clear that, while at the center of each region the grid is almost regular, deviations from a square mesh progressively grow towards the boundaries (see also Fig. 2). In particular, the ratio between the smallest surface element (obtained for $\eta=0$ and $\xi=\pm\pi/4$ and for $\eta=\pm\pi/4$ and $\xi=0$) and the surface element at the center of the block (*i.e.,* for $\eta=\xi=0$) is $\sqrt{2}/2\approx 0.71$. Going towards the four corners of a block, this ratio increases monotonically to about 0.77. This result points to one of the major advantages of the chosen coordinate system, *i.e.,* the fact that the meshes defined on each region span the surface of the sphere with an almost constant spatial resolution.

From the metric tensor it is now possible to derive the expressions for some of the most common differential operators in the coordinate systems defined on each of the six regions. The fact that all six regions share the same metric implies that all differential operators will have identical expressions on any of them. Any given PDE will then have exactly the same form on all regions, thereby simplifying noticeably the programming and debugging of the resulting finite difference codes.

Given a scalar field $f(\xi,\eta,r)$ and a vector field $\mathbf{V}(\xi,\eta,r)$, the gradient, laplacian, divergence, and curl are obtained from the following expressions:

$$\nabla f=\frac{1}{r}\left(D\frac{\partial f}{\partial\xi}+\frac{XY}{D}\frac{\partial f}{\partial\eta}\right)\hat{\mathbf{e}}_\xi + \frac{1}{r}\left(\frac{XY}{C}\frac{\partial f}{\partial\xi}+C\frac{\partial f}{\partial\eta}\right)\hat{\mathbf{e}}_\eta + \frac{\partial f}{\partial r}\hat{\mathbf{e}}_r \tag{21}$$

$$\nabla^2 f=\frac{\delta}{r^2}\left[C^{-2}\frac{\partial}{\partial\xi}\left(\frac{\partial f}{\partial\xi}\right)+D^{-2}\frac{\partial}{\partial\eta}\left(\frac{\partial f}{\partial\eta}\right)+\frac{2XY}{C^2D^2}\frac{\partial^2 f}{\partial\xi\partial\eta}\right]+\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f}{\partial r}\right) \tag{22}$$

$$\nabla\cdot\mathbf{V}=\frac{\delta^{3/2}}{rDC^2}\frac{\partial}{\partial\xi}\left(\frac{V_\xi}{\delta^{1/2}}\right)+\frac{\delta^{3/2}}{rCD^2}\frac{\partial}{\partial\eta}\left(\frac{V_\eta}{\delta^{1/2}}\right)+\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2V_r\right) \tag{23}$$

$$\nabla\times\mathbf{V}=\frac{1}{r}\left(\frac{XY}{\delta^{1/2}}\frac{\partial}{\partial r}rV_\xi-\frac{CD}{\delta^{1/2}}\frac{\partial}{\partial r}rV_\eta+\frac{\delta^{1/2}}{D}\frac{\partial V_r}{\partial\eta}\right)\hat{\mathbf{e}}_\xi + \frac{1}{r}\left(\frac{CD}{\delta^{1/2}}\frac{\partial}{\partial r}rV_\xi-\frac{XY}{\delta^{1/2}}\frac{\partial}{\partial r}rV_\eta-\frac{\delta^{1/2}}{C}\frac{\partial V_r}{\partial\xi}\right)\hat{\mathbf{e}}_\eta + \frac{\delta^{1/2}}{r}\left[\frac{XY}{CD}\left(\frac{1}{D}\frac{\partial V_\eta}{\partial\eta}-\frac{1}{C}\frac{\partial V_\xi}{\partial\xi}\right)-\frac{1}{D}\frac{\partial V_\xi}{\partial\eta}+\frac{1}{C}\frac{\partial V_\eta}{\partial\xi}\right]\hat{\mathbf{e}}_r. \tag{24}$$

3. A VARIATION OF THE COMPOSITE MESH METHOD

When faced with the task of numerically solving a PDE in a region with complicated geometry, a possible choice is to decompose the original domain into several subregions, each with a simpler structure (a sort of computational *Divide et Impera*). The obvious advantage of this procedure is that each subdomain can be designed to conform to the particular structure of its assigned boundary region, thereby simplifying both the imposition of boundary conditions and the construction of a global nonsingular coordinate system. In fact, each of the subdomains' coordinate systems can then be chosen such that the determinant of the Jacobian is bounded away from zero, which might be very hard (if not impossible) when trying to cover the entire region with a single coordinate system.

An equally obvious potential disadvantage of this strategy is that one is then faced with the nontrivial problem of how to lock together the different coordinate systems. The composite mesh method, introduced by *Starius* [25, 26], was developed with the aim of providing a procedure for coupling multiple coordinate systems to obtain globally stable and accurate solutions of PDEs in irregular domains. While for the details we direct the reader to more exhaustive treatments (e.g. [25, 26, 10, 6]), let us briefly discuss some of the key ingredients of the composite mesh method.

In a finite-difference scheme, values of dependent variables at stencil points are needed to approximate derivatives at the boundary points. When dealing with multiple coordinate systems, the question arises as to how to obtain

the values at the stencil points, since these points are not covered by the local coordinate system. If the set of coordinate systems was designed to cover the entire domain, all stencil points for one mesh must be interior points of a neighboring region covered by another coordinate system. It is therefore natural to suggest that the values at the stencil points be obtained by interpolations in the other coordinate system (*Phillips* [17]). In the case in which the coordinate systems are butted together, they will overlap solely on the boundary lines and one will need to perform interpolations only to obtain the values at the stencil points. Usually, however, a greater degree of overlap between the component meshes is considered necessary, based on both stability and accuracy considerations [26, 10]. Therefore, in practice the amount of overlap is taken to be of the order of a few mesh widths, and is generally kept minimal since all the work done on the overlap points is *de facto* an extra burden on the computational cycle.

The role of the overlap region in determining the stability and accuracy of the global solution was investigated mainly in the context of one-dimensional problems. In particular, *Henshaw* [10] showed that, for a given discretization of the mesh equations, the total error on the solution depends on the finite-difference truncation error, the interpolation truncation errors and the size of the overlap region. If the amount of overlap between component meshes is independent of the mesh size, he showed that in order to assure a global accuracy and convergence of the discretized equations it is *sufficient* to perform centered interpolations of the same order as the accuracy of the finite difference scheme employed.

Let us now see how these considerations carry over to our decomposition of the sphere.

As we saw, the coordinate lines of the six regions which we employ to represent the entire spherical surface are arcs of great circles. As a consequence of this choice, the coordinate lines of a given block join smoothly along one direction with those of the contiguous ones. For example, the vertical coordinate lines of the two equatorial blocks shown in Fig. 4 are both formed by arcs of great circles which originate from the North and South Poles and are, therefore, continuous along the equatorial direction. Let us now restrict our discussion to the case in which on each block one chooses the same mesh widths on both directions, i.e., the case when $N_\eta = N_\xi \equiv N$ on all blocks, where N_η and N_ξ are the number of grid points in the physical η and ξ directions of each block. We emphasize that this choice is not dictated by the "cubed sphere" decomposition or by the composite mesh method, but rather by our desire to optimize the overall computational efficiency of the algorithm. In fact, by using the same regular meshes on all blocks, one can exploit a number of symmetries between the different component meshes, which can greatly reduce the amount of computations required by the coupling scheme.

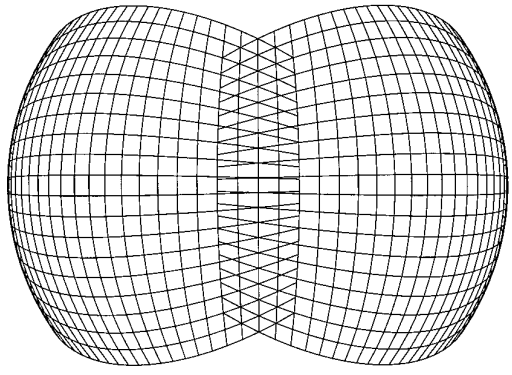

FIG. 4. View of two contiguous equatorial blocks and two of their stencils. The view is centered on the common vertical boundary line and shows the case of two grids with $N_\eta = N_\xi$ and $N_S = 2$. Notice that in this case, since both blocks use the same grid spacing, the horizontal coordinates of the stencil points of one grid coincide with the horizontal coordinates of the last two interior grid points of the contiguous block.

In the example of Fig. 4, by choosing the same mesh widths on both blocks, one obtains that the vertical grid lines of the two blocks will coincide in the overlap and stencil regions. Consequently, only a one-dimensional interpolation along the vertical η direction will be required to obtain the stencil values for all grid points on the coordinate lines defined by $\xi = \pm(\pi/4 + i\Delta)$, with $i = N_O, \ldots, N_O + N_S - 1$, where N_O is the number of grid points in the overlap region and Δ is the grid spacing. Similarly, for all equatorial blocks only a one-dimensional interpolation on interior points of a polar block will be required to obtain the stencil values for the grid points on the coordinate lines defined by $\eta = \pm(\pi/4 + j\Delta)$, with $j = N_O, \ldots, N_O + N_S - 1$.

It is also interesting to notice that, for meshes with $N_\xi = N_\eta = N$, all overlap points of the horizontal $\eta = \pm\pi/4$ and vertical $\xi = \pm\pi/4$ coordinate lines of any block fall exactly on interior grid points of an contiguous block (see Fig. 4). Therefore, if the amount of overlap is limited to the boundary lines between blocks $N_O = 1$, for each block one would need to perform one-dimensional interpolations only on $4\ N_S(N - 2)$ stencil points.

It should be noted that there is a geometrical limit on the amount of overlap that one can impose between two blocks, since the center point of any block is a singular point for the coordinate systems of the four contiguous ones. Consequently, $N_O + N_S < N/2$ is a strict condition, which cannot be violated and should not be approached if accuracy is to be maintained. In fact, as the amount of overlap increases, the values at the stencil points will be determined by interpolating on interior points which belong to increasingly smaller physical portions of the contiguous blocks.

Although an "orthodox" application of the composite

mesh method to our decomposition would dictate that the degree of overlap should lie in the range $1 < N_O < N/2 - N_S$, we find that equally stable and accurate solutions can be obtained even in the case of $N_O = 1$. This feature of our coupling procedure, together with the possibility of performing only one-dimensional interpolations on the stencil points, dramatically reduces the number of operations required to couple the six blocks.

To further qualify this statement, let us compare the number of interpolations required by our method with those needed to couple two polar stereographic coordinate systems, as proposed by *Browning et al.* [4]. Although we will consider for simplicity the case of minimum overlap, the generalization to an arbitrary degree of overlap is trivial and will not modify our conclusions. For a given size of the stencil region, N_S, the total number of interpolation points required by the ''cubed sphere'' algorithm is

$$I_{\mathrm{CS}} = 24\, N_S(N-2) \tag{25}$$

while those considered in the polar stereographic decomposition are

$$I_{\mathrm{PS}} = 2\pi N_S^2(1 + 2N_T/N_S), \tag{26}$$

where N_T is the number of grid points on the radius of the two polar stereographic equatorial circles.

The total number of grid points employed in the polar stereographic decomposition of the spherical surface is given by

$$N_{\mathrm{PS}} = 2\pi N_T^2$$

and an equivalent resolution can be achieved with the ''cubed sphere'' decomposition by taking on each block

$$N = (N_{\mathrm{PS}}/6)^{1/2} = (\pi/3)^{1/2} N_T$$

From (25) and (26) we then obtain

$$\frac{I_{\mathrm{PS}}}{I_{\mathrm{CS}}} = \frac{2\pi N_S^2(1 + 2N_T/N_S)}{24 N_S[(\pi/3)^{1/2} N_T - 2]} \approx \frac{1}{2}, \tag{27}$$

where we considered the limit $N_T \gg N_S$. This result reflects the fact that in the ''cubed sphere'' technique there are more boundary lines on which to interpolate, compared to the polar stereographic method. When estimating the ratio between the total number of operations performed on the interpolation points, one must however multiply (27) by the ratio between the number of operations required in the two schemes to interpolate on a single point. Given the fact that the ''cubed sphere'' method requires only one-dimensional interpolations, this ratio is equal, for the typical case of Lagrangian interpolations, to the order of the interpolation scheme. From (27), we then see that in a sixth-order accurate scheme, with the polar stereographic method one then performs about three times $(\frac{1}{2} \cdot 6)$ the number of operations required by the ''cubed sphere'' technique.

Finally, the efficiency of the ''cubed sphere'' technique can be further appreciated if one considers that, for the values of N_T and N_O, typically employed by *Browning et al.* [4], the amount of computations performed in the overlap region approached 50% of the overall cost of the simulation. As we will show in the next section, our method does not carry this additional burden, since we can obtain stable and accurate solutions also in the case of $N_O = 1$.

4. RESULTS OF TEST CASES ON SINGLE PROCESSOR COMPUTERS

Recently, *Williamson et al.* [30] proposed a suite of seven test cases for the evaluation of numerical methods intended for the solution of the shallow water equations in spherical geometry. The suite of test cases was designed for use in the evaluation of numerical methods proposed for atmospheric modeling and presents the major difficulties found in the horizontal aspects of three-dimensional codes.

While in a forthcoming paper we will discuss the performance of the ''cubed sphere'' technique on the entire suite of test cases and on the solution of elliptic type of equations [12], in this context we will restrict our analysis to the study of two hyperbolic problems: the advection of a structure of compact support and the evolution of a Rossby–Haurwitz wave.

One of the crucial issues which must be addressed by a numerical method based on a domain decomposition strategy concerns the effects caused by the artificial internal boundaries on the global accuracy and stability of the numerical solutions obtained. The linear advection of a structure of compact support, such as the cosine bell suggested in [30], by a wind field corresponding to solid body rotation is particularly suited to address this issue. In fact, while the cosine bell is linearly advected from one region to another of the spherical surface, any distortion of its shape due to the presence of artificial internal boundaries will be easily detected. Furthermore, if the advection takes place across the poles, this case will also test the ability of a given numerical method of solving the pole problem.

Since this first test concerns only the linear advective component of the shallow water equations, we also present a second case in which the full nonlinear structure of the equations is retained: the evolution of a Rossby–Haurwitz wave. Rossby–Haurwitz waves are analytic solutions of the nonlinear barotropic vorticity equation on the sphere [9] and since Phillips' first studies they have become one of the most commonly used tests for evaluating the perfor-

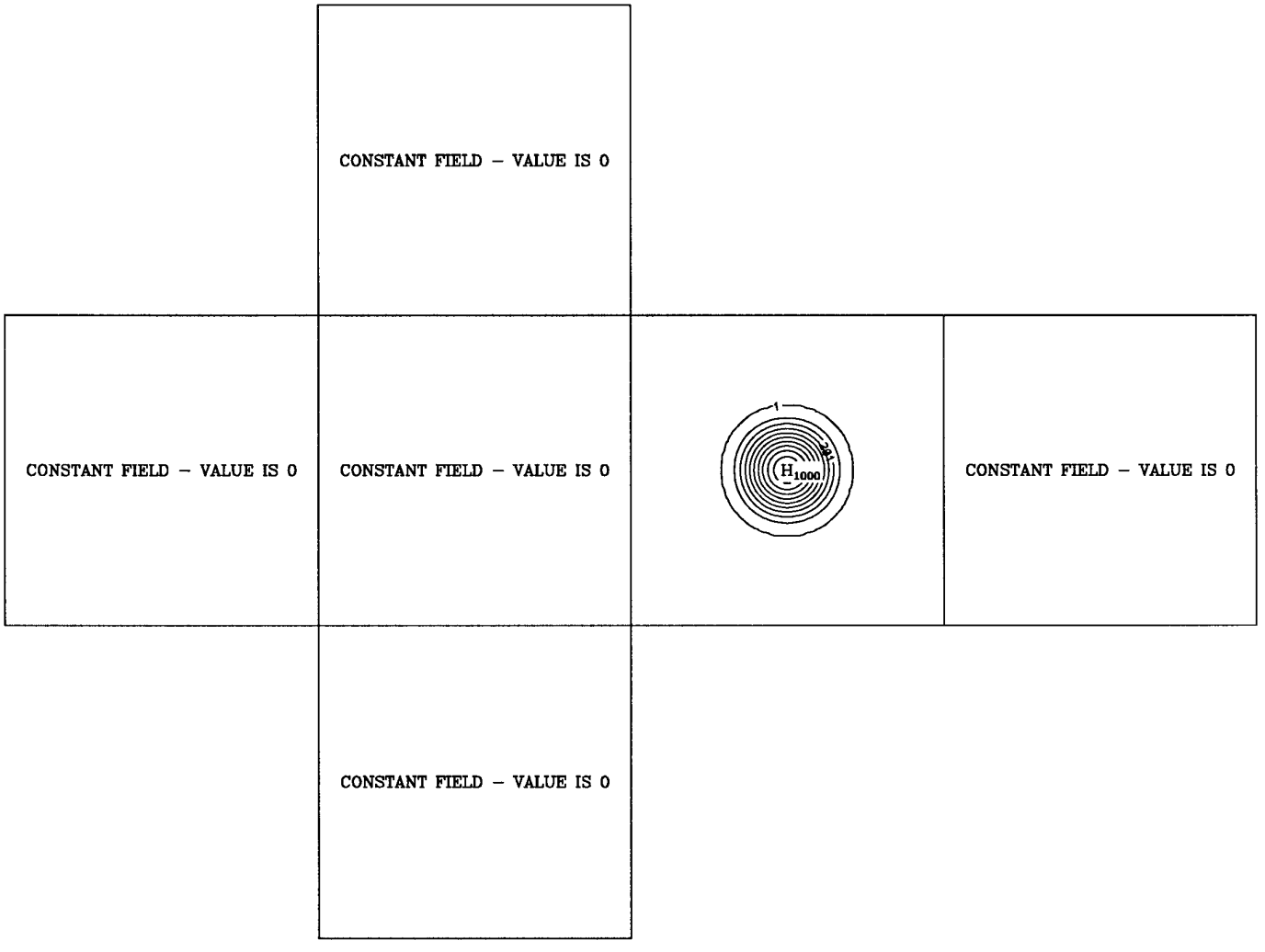

FIG. 5. Initial condition for the linear advection test case, as obtained on the six meshes of the "cubed sphere."

mance of numerical methods on the sphere. Differently from the linear advection case, no analytical expression is available for the exact solution of the shallow water equations with the Rossby–Haurwitz wave initial conditions. All error measurements for this test were, therefore, performed through a comparison with a reference solution obtained from a very high resolution run (T213) of the spectral transform code by *Hack and Jakob* [8].

Let us now give for the sake of completeness the advective form of the shallow water equations:

$$\frac{d\mathbf{v}}{dt} = -f\hat{\mathbf{k}} \times \mathbf{v} - g\nabla h \tag{28}$$

$$\frac{dh}{dt} + h\nabla \cdot \mathbf{v} = 0, \tag{29}$$

where h is the height of the free surface above a reference sphere, f is the Coriolis parameter, g is the gravitational constant, $\hat{\mathbf{k}}$ is the outward radial unit vector, and the substantial derivative is defined as usual by

$$\frac{d}{dt} \equiv \frac{\partial}{\partial t} + (\mathbf{v} \cdot \nabla). \tag{30}$$

In the first test case, a time independent nondivergent wind field is specified, which corresponds to a solid body rotation about an axis forming an arbitrary angle α with the polar axis of the spherical coordinate system. Since the advecting field is nondivergent, the equation for h reduces to the linear advection equation $dh/dt = 0$. In terms of the angular coordinates ξ and η in the "cubed sphere," we then obtain the equation

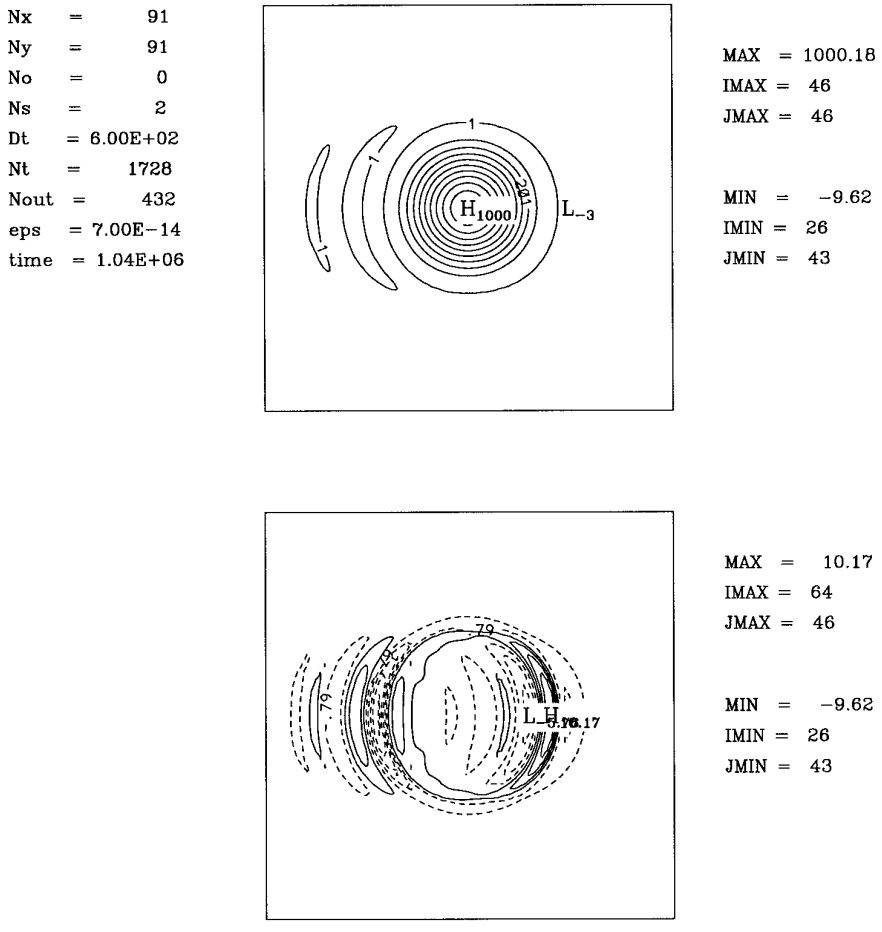

FIG. 6. Geopotential height (upper panel) after a full rotation (12 days, or 1728 time steps) for the case of equatorial advection ($\alpha = 0$). In the lower panel we show a contour plot of the error, calculated as the height difference from the known analytical solution.

$$\frac{\partial h}{\partial t} + \frac{\delta}{aC^2} u \frac{\partial h}{\partial \xi} + \frac{\delta}{aD^2} v \frac{\partial h}{\partial \eta} = 0, \tag{31}$$

where

$$u = \frac{u_0}{\delta} (C^2 \cos\alpha + Y \sin\alpha) \tag{32}$$

$$v = \frac{u_0}{\delta} (XY \cos\alpha - X \sin\alpha) \tag{33}$$

are the expressions for the wind field components in terms of the local coordinates of region I.

The initial condition is a cosine bell pattern placed at the equator [30] (see Fig. 5) with a central maximum value of 1000. The advecting wind velocity is $u_0 = 2\pi a/$ (12 days) $\approx$ 40 m/s.

In Figs. 6–8, we show the results of three runs using the "cubed sphere" method with 90 intervals on each of the sides of the six square meshes (referred to as a C91 resolution run, corresponding to exactly 1° grid spacing along the equator). In each figure, the upper contour plot shows the numerical solution after a full rotation and the lower contour plot displays the error, calculated as the height difference from the known analytical solution. The three runs differ only in the value used for the angle α of solid body rotation: $\alpha = 0, \pi/4, \pi/2$. In the first case (Fig. 6), the advection occurs along the equatorial band (starting in region II and passing through regions III, IV, I and then back to II), while for $\alpha = \pi/2$ (see Fig. 8) the bell crosses the south and north poles before returning to its original position after 12 days (i.e., going from region II, through regions VI, IV, V and then back to II). In the intermediate case of $\alpha = \pi/4$ (see Fig. 7), the advection occurs at an angle with respect to the sides of the square regions, so that during the time evolution the initial cosine bell pattern

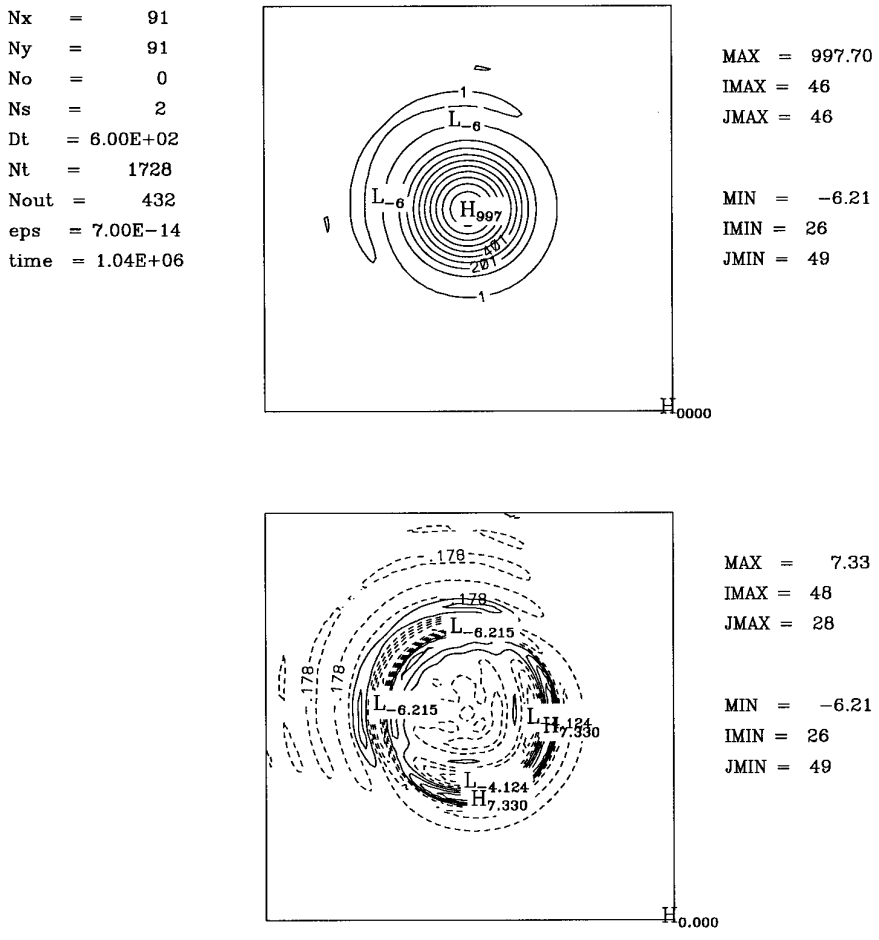

FIG. 7. Geopotential height (upper panel) after a full rotation (12 days, or 1728 time steps) for the case in which the advection occurs at an angle with respect to the sides of the square regions ($\alpha = \pi/4$). In the lower panel we show a contour plot of the error, calculated as the height difference from the known analytical solution.

will often be split and assigned to more than two neighboring regions (e.g., after one day parts of the bell will be found in regions II, III, and V).

The time integration was performed using an explicit third-order Adams–Bashforth scheme, while a centered fourth-order differencing scheme was used in space. The addition of an explicit hyperdiffusion term ($-\nabla^4$) was found to be necessary to free the numerical solutions from the small amplitude short scale noise introduced at the internal boundaries by the interpolation procedure. We stress that no hyperdiffusion was needed for shorter integrations (up to a full half rotation, corresponding to about six days or 864 time steps) or when using a higher degree of overlap between the component meshes. It was, furthermore, observed that the small amount of hyperdiffusion introduced in the case of $N_S = 2$ did not degrade the accuracy of the solution, while totally eliminating the short scale noise formed at the boundaries.

A first important conclusion can be drawn by comparing the error patterns of Fig. 6 with those of Fig. 8. The fact that, besides the expected different orientation of the patterns, there is no distinction between the two error plots is a striking indication that in the "cubed sphere" method the pole problem is totally absent. In fact, there is no intrinsic geometrical distinction between the polar regions and the equatorial ones and the only means for discriminating them are provided by the spatial dependence displayed by the coefficients or by the boundary conditions of the equations to be solved.

From Fig. 7 one can see that the error pattern for the $\alpha = \pi/4$ case has more structure than that of the other two cases, but nevertheless displays essentially the same accuracy, with errors at most of about 1% of the maximum central value.

The analysis of this first test case shows that the "cubed sphere" method is able to treat properly the presence of artificial internal boundaries. Smooth and accurate solutions can be obtained even after multiple crossings of the

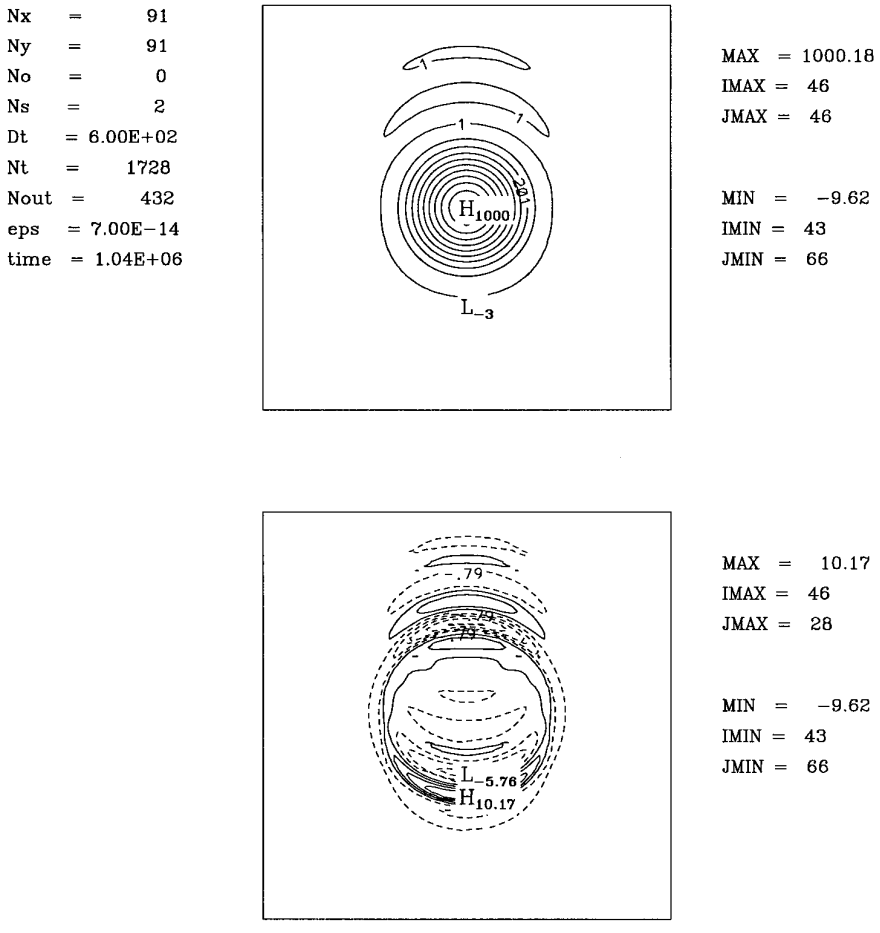

FIG. 8. Geopotential height (upper panel) after a full rotation (12 days, or 1728 time steps) for the case of polar advection ($\alpha = \pi/2$). In the lower panel we show a contour plot of the error, calculated as the height difference from the known analytical solution.

boundary lines, which proves that the interpolation procedure does not degrade the accuracy of the spatial differencing scheme.

Turning now to the Rossby–Haurwitz test case, in Fig. 9 (upper plot) we show the initial condition for the geopotential on a cylindrical projection. As noted above, in the absence of an exact analytical expression for the solution, we used the result of a T213 spectral transform code as the reference solution from which all error measurements were calculated. In order to test the convergence properties of the "cubed sphere" method and to compare its efficiency and accuracy to that of the spectral method, we chose to run the reference case and all tests with a very small time step of 40*s*. This assures that the space truncation error will dominate over the time truncation error at all the resolutions tested (we checked this *a posteriori* by comparing the results of runs using a longer time step). This is an essential condition that must be satisfied in order to perform a meaningful comparison of different methods of spatial discretization. Both the spectral code and the "cubed sphere" code used an explicit Assalin filtered leapfrog time integration scheme and used an explicit hyperdiffusion term ($-\nabla^4$) to prevent nonlinear instability effects from destroying the accuracy of the numerical solution. The "cubed sphere" code employed a sixth-order centered scheme for calculating spacial derivatives (i.e., $N_S = 3$).

The reference solution after 7 days of integration, corresponding to over 15,000 time steps, is shown in Fig. 9 (bottom plot). The numerical solutions obtained from the T63 and C56 runs and the corresponding error plots (calculated as the difference from the reference solution) are shown in Figs. 10 and 11. The solution for the C56 run is also shown on the six blocks in Fig. 12. In the first six rows of Table I, we compare the results of runs performed with the "cubed sphere" and the spectral transform methods using a comparable number of underlying grid cells to cover the spherical surface. At all resolutions tested, the "cubed sphere" and the spectral transform method yielded numerical solutions with comparable accuracy. However, the "cubed sphere" method allowed savings in execution

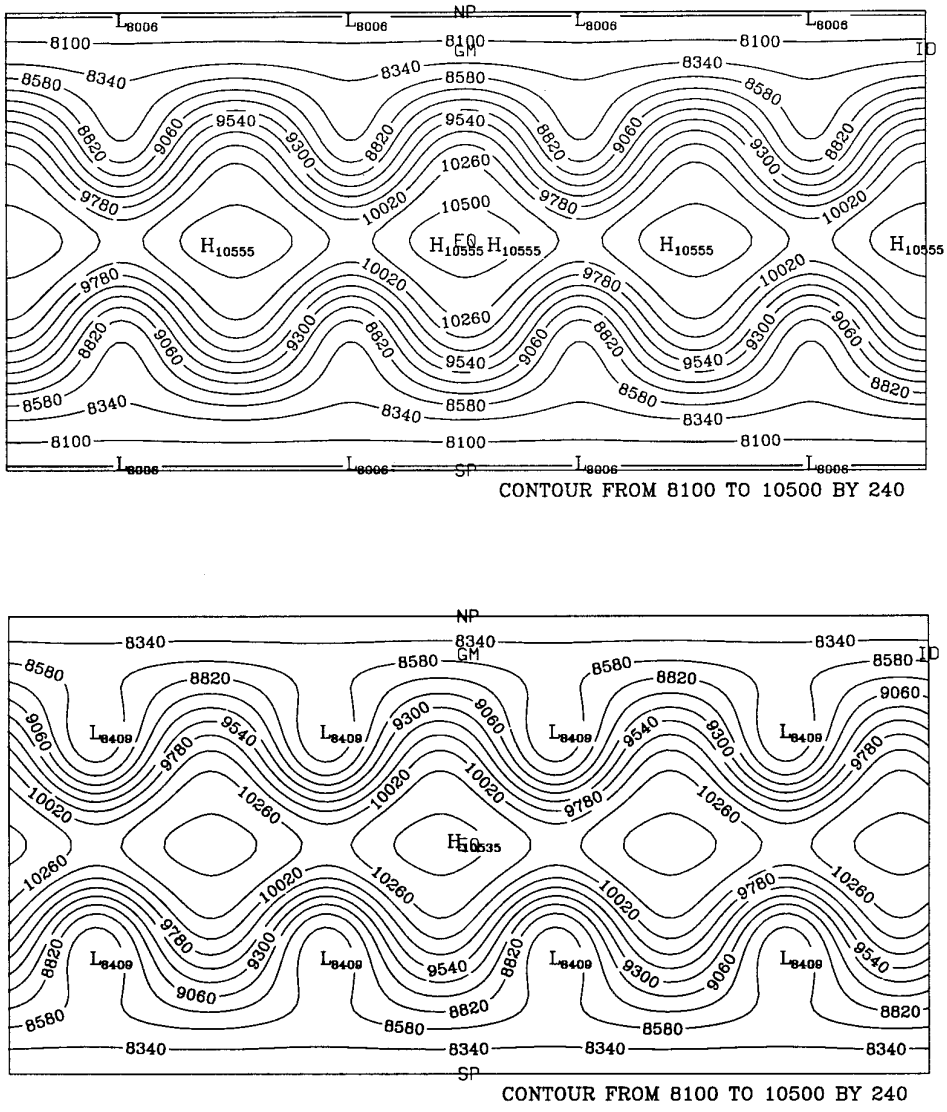

FIG. 9. Initial condition (upper plot) and reference solution after 7 days of integration (bottom plot) on the sphere in cylindrical projection for the geopotential of the Rossby–Haurwitz test case. The reference solution was obtained running a spectral transform code at a T213 truncation.

time which ranged from a factor 6.5 at the lowest resolution (T42 and C38), to over an order of magnitude at the T106 and C92 resolutions. The ''cubed sphere'' method required slightly more storage for low resolutions, while at the highest resolution it needed about 1.5 less memory than the spectral transform method.

It is a well known fact that one of the advantages of the spectral method is the possibility of implementing with relatively little cost a semi-implicit time-stepping algorithm, whereas the use of grid point based methods would entail the solution of an elliptic problem. In the context of the ''cubed sphere'' method, the need to satisfy the global communication requirements of elliptic solvers might somewhat reduce the gain in execution time over the spectral transform method obtained previously with the use of an explicit time-stepping scheme. On the other hand, the last two rows of Table I show that, for a given specified level of accuracy, the ''cubed sphere'' method can still save almost a factor 4 in execution time, employing an explicit time differencing scheme with a time step about three times smaller than that taken by the spectral transform method using a semi-implicit algorithm. This result is due to the fact that, in order to attain the specified accuracy, the ''cubed sphere'' method can run at a lower spatial resolution, while exploiting the advantage of a higher order explicit time stepping algorithm, such as the third-order Adams–Bashforth scheme. The use of an even longer time step by the spectral method would further

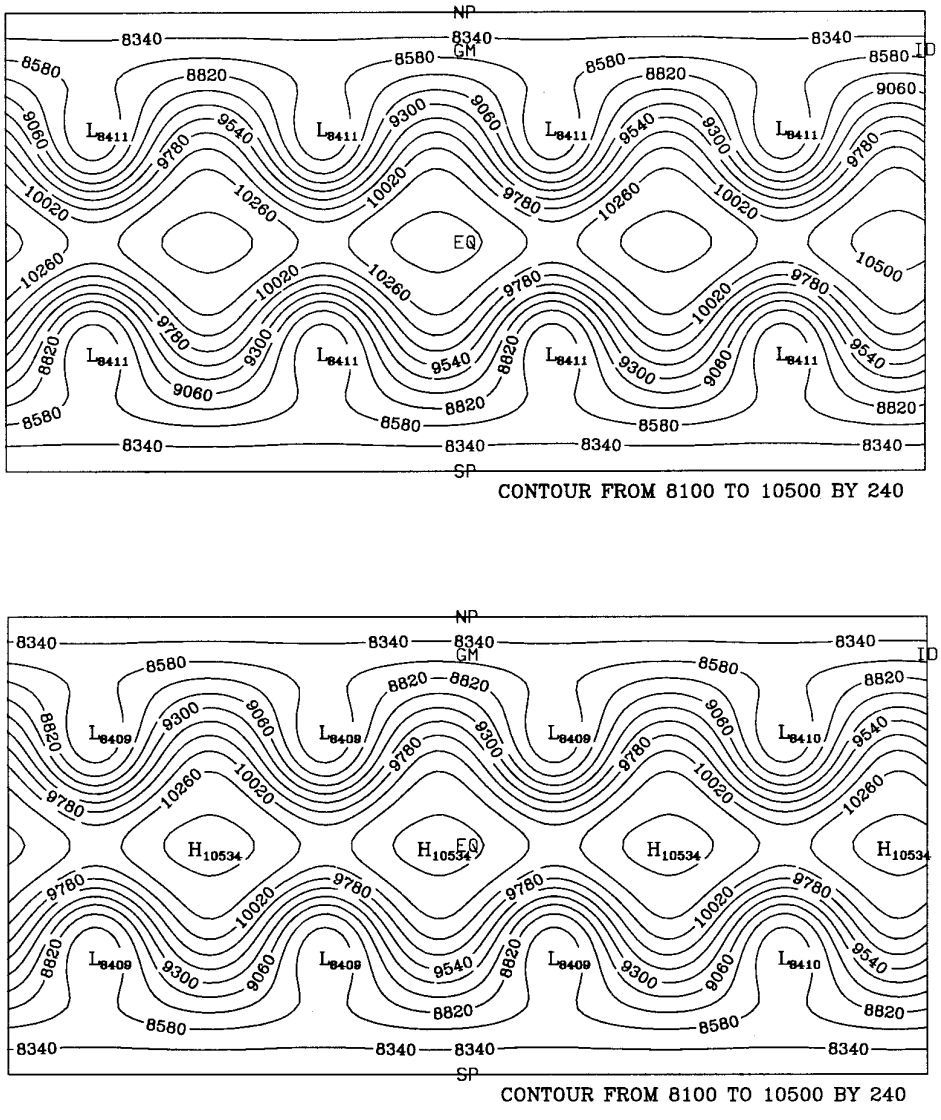

FIG. 10. Solution for the geopotential after 7 days of integration, corresponding to over 15,000 time steps, obtained from the spectral transform code at a T63 resolution (upper plot) and from the ''cubed sphere'' method at a C56 resolution (bottom plot).

degrade the accuracy of the numerical solution, up to the point of questioning the relevance of running at a T63 spatial resolution. While these results were obtained in the context of the Rossby–Haurwitz test case, their relevance to more realistic situations, such as those pertinent to the modeling of global atmospheric flows, should not be discarded. In fact, recent studies [4, 13] suggest that the role of time truncation errors in determining the global accuracy of the numerical solutions might be dominant over that played by the spatial truncation errors. We will further address this issue in a forthcoming paper on the implementation of the ''cubed sphere'' method in a full baroclinic atmospheric model.

In conclusion, the results of the linear advection and of the Rossby–Haurwitz test cases prove that the ''cubed sphere'' method can provide, with equivalent storage requirements, extremely accurate solutions using a fraction of the execution time employed by the spectral transform method.

5. THE "CUBED SPHERE" ON MASSIVELY PARALLEL ARCHITECTURES

The purpose of this section is to show how this new gridding technique can be implemented on any parallel computer supporting 2D and 3D first neighbour connecting topologies [11, 27]. In fact, there are several features of he ''cubed sphere'' technique which make it particularly

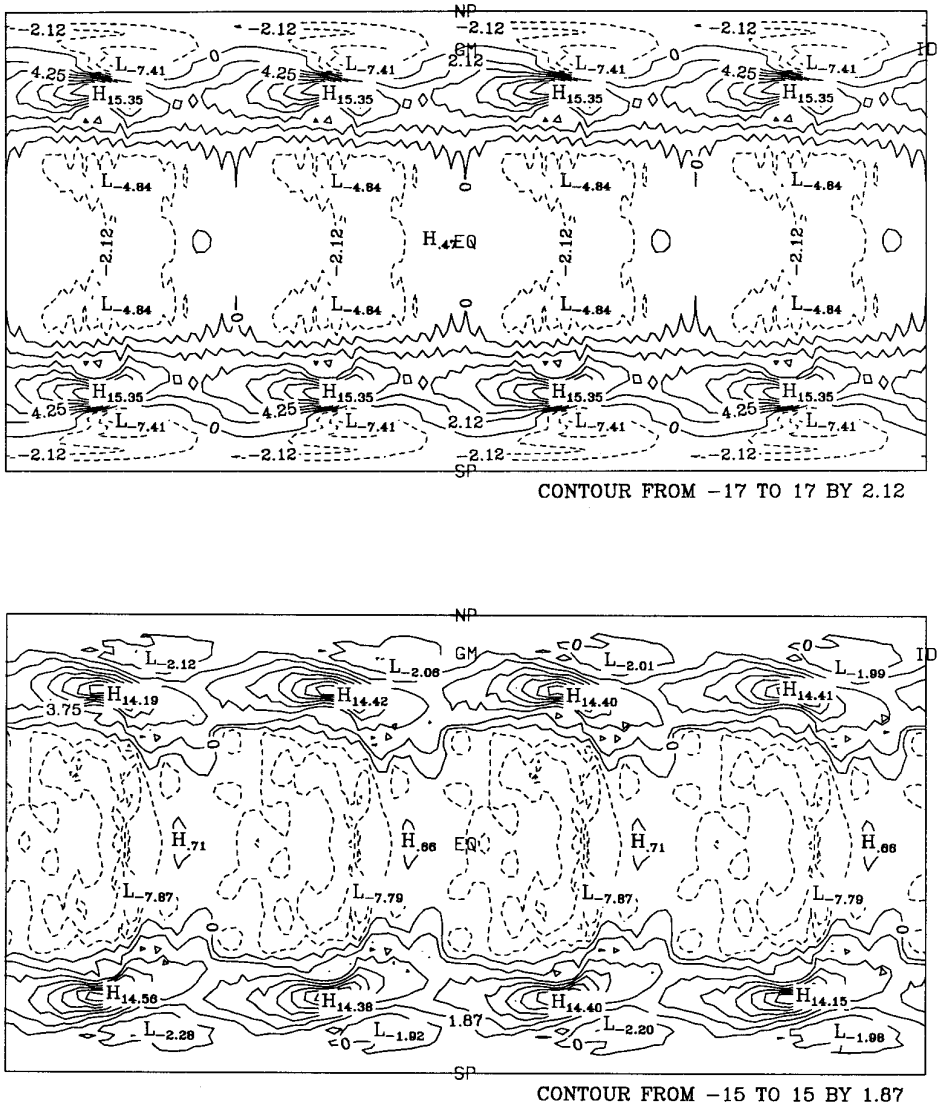

FIG. 11. Error plots for the solutions shown in Fig. 11. The error was calculated as the difference from the reference solution shown in Fig. 9.

amenable to an efficient implementation on parallel machines. Since all six regions have the same geometric structure and metric, the equations to be solved take exactly the same form on each region and require only one discretization. Furthermore, the treatment of boundary data on different blocks will be shown to require similar internode communications patterns on all regions.

Let us now first briefly review some background information and taxonomy on parallel architectures.

Massively parallel computers are commonly classified using two main criteria: the instruction execution mode and the available direct connecting pathways among the processing nodes [11, 27].

In a SIMD (single instruction multiple data) machine, all the processing nodes execute the same instruction acting on different data, typically stored on local memories associated with each processing node. On the other hand, MIMD (multiple instruction multiple data) computers are able to asynchronously execute a different program on each node. It is clear that efficient SIMD algorithms, such as our "cubed sphere" finite difference scheme, can achieve high computational efficiency also on machines supporting a MIMD execution mode.

The second architectural keystone of parallel computers is their internode communication hardware. The most widely adopted connecting topologies are crossbar networks, two- and three-dimensional meshes, multilevel switched networks, and hypercubes. Together with the execution mode, the processors mesh topology can strongly affect the parallel efficiency of a given algorithm.

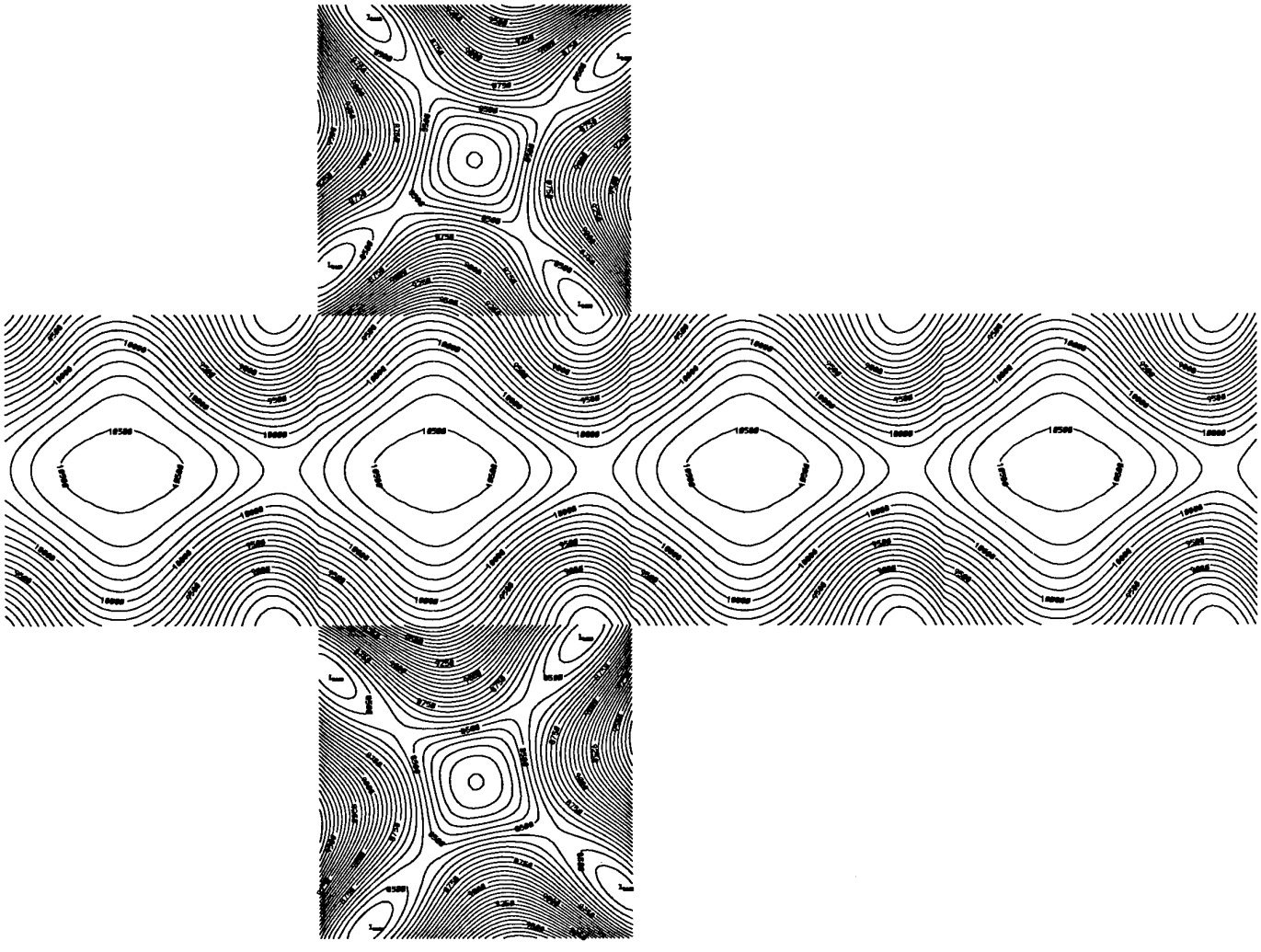

FIG. 12. Solution on the ''cubed sphere'' for the geopotential of the Rossby–Haurwitz test case after 7 days of integration (see also Fig. 10). Notice that although the geopotential must be continuous passing from one region to another, its derivative generally suffers a discontinuity across the boundaries.

We performed our parallelization experiments on APE100/Quadrics [2, 3], a parallel machine with peak performances scalable from 100 Gflops, obtained using 2048 processing nodes, down to 400 MFlops in its minimal 8-node configuration. The standard topology of APE100/Quadrics is a 3D toroidal mesh of nodes executing in SIMD mode. The basic unit (FPU Board) contains 8 nodes configured as a $2*2*2$ cube, and several basic units can be combined to obtain larger configurations (e.g., 512 nodes can be arranged in an $8*8*8$ mesh). The system provides synchronous communication among each node and its six nearest neighbours and each processing node may either fetch data residing on its own memory or access the memories of its six first neighbouring nodes. The $8*8*8$ configuration of Ape100/Quadrics, can be characterized by the following figures:

- peak arithmetic performances: 25.6 Gflops/s
- local memory bandwidth: 25.6 Gbytes/s
- first neighbouring nodes memory bandwidth: 6.4 Gbytes/s
- data memory: 2 Gbytes.

We remark that the 4 : 1 ratio between the local versus first neighbour memory access speeds is independent from the total number of nodes in the configuration.

As noted, standard APE100/Quadrics systems are arranged to provide periodic conditions on the machine mesh boundaries, so that the topology of the mesh is that of a 3D torus. This is attained by physically connecting all processors on the machine mesh boundaries which are required to be topologically contiguous in a torus. Through a different cabling it is nevertheless also possible to build a

TABLE I

Comparison between the Numerical Solutions Obtained after 7 Days of Integration of the Shallow Water Equations (with Rossby–Haurwitz Initial Conditions) Using Both the Spectral Transform Code and the "Cubed Sphere" Methods

	N_{cells}	$\Delta t(s)$	N_t	CPU(s)	L_2	L_{max}	$\varepsilon(m^4/s)$	Memory
*T*42	8192	40	15120	3383	1.15×10^{-3}	3.4×10^{-3}	5×10^{15}	108234
C38	8214	40	15120	520	8.4×10^{-4}	3.4×10^{-3}	5×10^{15}	172520
*T*63	18432	40	15120	7647	3.9×10^{-4}	1.4×10^{-3}	1×10^{15}	305235
C56	18150	40	15120	960	3.78×10^{-4}	1.4×10^{-3}	1×10^{15}	340928
*T*106	51200	40	15120	29384	1.11×10^{-4}	5.1×10^{-4}	1×10^{14}	1224618
C92	49686	40	15120	2543	2.26×10^{-4}	6.9×10^{-4}	1.5×10^{14}	844928
*T*63	18432	450	1344	701	9.0×10^{-4}	8.5×10^{-4}	1×10^{15}	305235
C48	13254	160	3780	195	9.0×10^{-4}	6.2×10^{-4}	3.3×10^{15}	172520

Note. In the upper row, N_{cells} indicates the number of grid point cells, Δt is the time step in seconds, N_t is the total number of time steps performed, CPU is the total cpu time in seconds employed on a single processor of a Cray YMP EL98, L_2 and L_{max} are measures of the global error calculated as suggested in [30], ε is the coefficient of the hyperdiffusion operator, and Memory is the required amount of storage in words. In the leftmost column we indicate the resolutions at which the runs were performed (T42, T63, T106 for the spectral code and C38, C48, C56, C92 for the "cubed sphere" method).

"spherical" machine composed of six groups of processing nodes directly satisfying the communication requirements of the "cubed sphere" topology. It will be sufficient to connect the boundaries of different groups of processing nodes according to the scheme outlined in Fig. 13. In what follows, we will evaluate the performance of the "cubed sphere" technique on both toroidally and spherically connected machines. Our analysis will consider simulations performed on both two- and three-dimensional meshes. Two-dimensional meshes can be distributed on a parallel machine with 3D first neighbour connecting topology by means of a software mapping algorithm, which allows us to contract two hardware dimensions into a single virtual dimension [5]. Through this algorithm, a first neighbour communication along the contracted direction requires a time which is exactly twice that of a standard communication between processing nodes connected by a hardware first neighbour communication channel. A 3D mesh is ob-

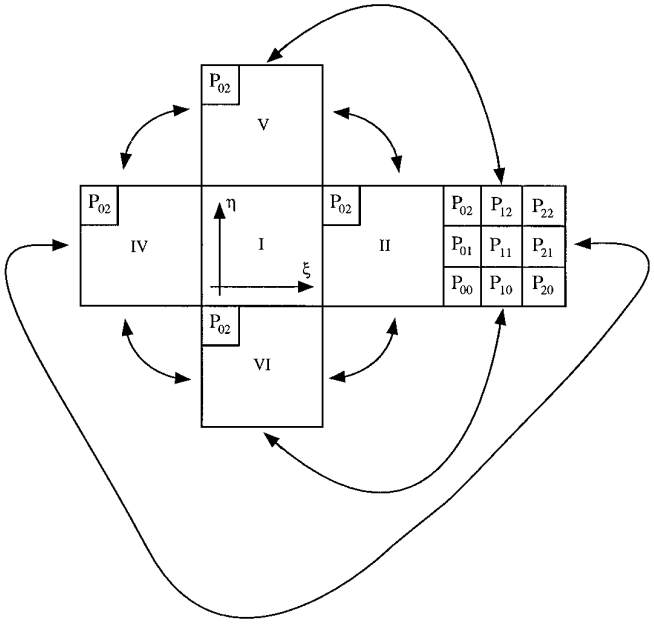

FIG. 13. Each side of the six squares in the "cubed sphere" must communicate its boundary data to the side which is contiguous in the spherical topology sense. The thick lines with arrows show the communications patterns required among the external boundaries of the six squares. These patterns indicate also the hardware connections needed to build a "spherical" parallel machine, composed of six groups of processing nodes directly satisfying the communication requirements of the spherical topology.

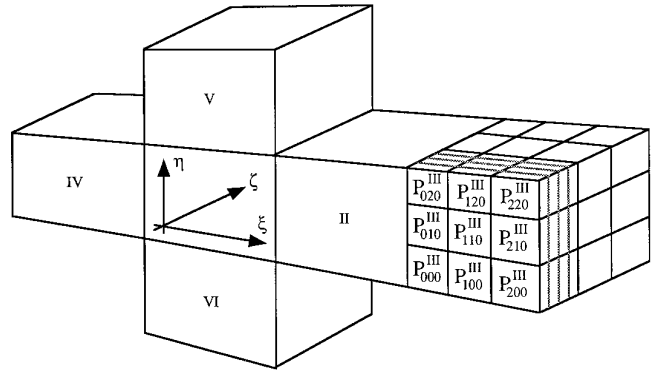

FIG. 14. Three-dimensional configuration of the "cubed sphere," obtained by overlaying several 2D spherical shells. While in the toroidally connected machine, every processing node evolves the same portion of each region (as shown in Fig. 13), in a "spherical" machine the boundary conditions are hardware-enforced and each processor can compute the evolution of a portion of a single block independently from the others, with a factor six saving in local memory requirements.

TABLE II

Theoretical Expressions for T_E, T_S, T_I, T_C for 2D Finite Difference Codes Executed on a 3D Toroidal Machine

	2D (toroidal)
T_E	$N_q N_f N_E N^2/(P_T^3 V_F)$
T_S	$2N_q N_f N_S N(2 + 1/P_T)/(P_T V_T)$
T_I	$2N_q N_f N_S N_I N(1 + 1/P_T)/(P_T V_F)$
T_C	$2N_q N_f N_S N(2 + 1/3 + 1/P_T)/(P_T V_T)$

Note. Here, $P_T = (P_{\text{tot}})^{1/3}$ is the number of processors along the side of a 3D toroidal mesh of processors, $N_q = 6$ is the number of blocks, N_f is the number of physical variables per grid point, N_E is the number of arithmetic operations per physical variable on each grid point performed every time step, N_I is the number of arithmetic operations required to obtain the interpolated value of a physical variable on each stencil point, V_F and V_T are, respectively, the effective speed of floating point operations and of data transfer between neighbouring processors.

tained by overlaying several 2D spherical shells. Therefore, the 3D extension of our method does not introduce any additional complications as far as the coupling algorithm between regions is concerned. Mapping a 3D block onto a 3D first neighbour hardware topology is an obvious task if in the vertical direction only local computations are required. In fact, in this case in the vertical r direction a first neighbour topology satisfies the communication requirements of finite difference schemes. Instead, if the in-node storage of all values along a vertical column is needed for an efficient computation along the vertical direction, the 3D mesh should be mapped through the same data distribution scheme used in the case of the 2D mesh.

The different topological structure of the toroidal and spherical cases is reflected in the way each of the six blocks will be assigned to different processors. In the toroidal case, every processing node evolves the same portion of each region. In the configuration of Fig. 13, processor P_{20} will perform the evolution of the right bottom corner of each block, processor P_{02} the evolution of the left upper corners, and so on. In the spherical machine the boundary conditions are hardware-enforced and each processor can then compute the evolution of a portion of a single block independently from the others. This fact points to one of the advantages of the spherical configuration over the standard toroidal one, i.e., the possibility to perform an efficient simulation with six times less local memory requirements.

TABLE III

Theoretical Expressions Calculated for 3D Codes on Both 3D Toroidal and Spherical Hardware Topologies

	3D (spherical)	3D (toroidal)
T_E	$N_f N_E N^2 N_\zeta/(P_S^3 V_F)$	$N_q N_f N_E N^2 N_\zeta/(P_T^3 V_F)$
T_S	$2N_f N_S N(N + 3N_\zeta)/(P_S^2 V_T)$	$2N_q N_f N_S N N_\zeta(2 + N/N_\zeta)/(P_T^2 V_T)$
T_I	$4N_f N_S N_I N N_\zeta/(P_S^2 V_F)$	$4N_q N_f N_S N_I N N_\zeta/(P_T^2 V_F)$
T_C	$2N_f N_S N(N + 3N_\zeta)/(P_S^2 V_T)$	$2N_q N_f N_S N N_\zeta(1 + 5P/6)/(P_T^2 V_T)$

Note. Here, $P_S = (P_{\text{tot}}/6)^{1/3}$ is the number of processors associated with the sides of each of the six blocks in a spherical machine. For explanations on all other symbols see the text or the caption of Table II. Notice that T_C is the only communication time which is severely affected by a different topological connection of the machine (toroidal vs. spherical).

TABLE IV

Measured Evolution and Communication Times for a Test Version of a 2D code on Two APE100/Quadrics Configurations (One with 512 Processing Nodes Arranged in an $8*8*8$ Toroidal Topology and a Second One with 8 Nodes on a $2*2*2$ Toroidal Lattice), Together with the Corresponding Theoretical Estimates.

		$8*8*8$			$2*2*2$
		2048	1024	256	256
T_E	th	0.47	0.12	0.0074	0.47
	ex	0.45	0.17	0.015	0.45
T_S	th	0.013	0.0065	0.0016	0.0082
	ex	0.020	0.011	0.005	0.016
T_C	th	0.015	0.0076	0.0020	0.0087
	ex	0.043	0.021	0.010	0.023

Note. Notice that from Table II one has that $T_I \approx T_S$ and that we used a typical assumption for the efficiency for floating point operations of 25%, corresponding to $V_F = 4V_T$.

Let us now consider the simulation of a system with a total number of grid points per region given by $N_\xi \cdot N_\eta \cdot N_\zeta$ on $N_1 \cdot N_2 \cdot N_3$ processing nodes (see Fig. 14). Within each region, every processing node has assigned a set $[i, j, k]$ of coordinates, where $i = 0, 1, ..., N_1 - 1$, $j = 0, 1, ..., N_2 - 1$, and $k = 0, 1, ..., N_3 - 1$, and must evolve the equations on $N_\xi/N_1 \cdot N_\eta/N_2 \cdot N_\zeta/N_3$ grid points.

The total execution time T required to advance the equations of one time step can be expressed as $T_E + T_S + T_I + T_C$, where each time corresponds to a specific phase of the algorithm:

- T_E is time required to evolve the $N_\xi/N_1 \cdot N_\eta/N_2 \cdot N_\zeta/N_3$ grid points assigned to each processing node. Here the computations assigned to each processor are performed in complete parallelism, gaining a $N_1 \cdot N_2 \cdot N_3$ speedup factor against a single processor machine, which must compute the evolution of all $N_\xi \cdot N_\eta \cdot N_\zeta$ grid points contained in the entire physical domain.
- T_S is the time required to copy a stencil of depth N_S from a neighbouring processor. This is strictly necessary only for the processors which lie entirely inside their sub-

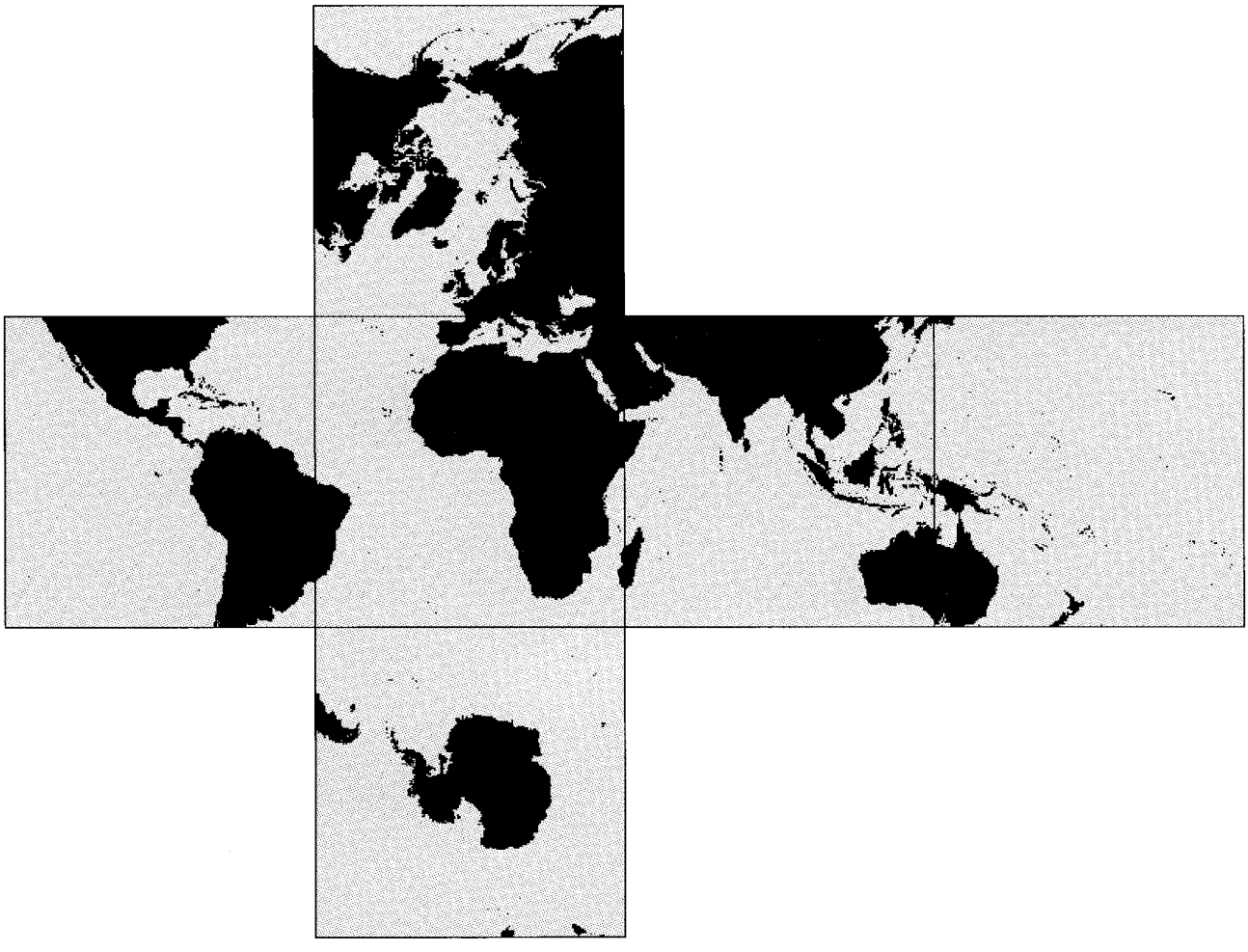

FIG. 15. Orography of the Earth represented on the "cubed sphere." Notice how the continents are split among the different blocks in order to realize the spherical topology.

group (e.g., processor P_{110} of Fig. 14). This step is absent (or, better, it is included in T_E) on serial/vector computers, but must be considered on parallel machines because the interprocessor communications are generally slower than local memory accesses. As already noted, in the case of APE100/Quadrics, the ratio between the local *vs.* first neighbour access times is $\approx \frac{1}{4}$.

• T_I is the time required to calculate through interpolation the stencil values of the time-dependent variables in the processing nodes placed at the boundaries of the six blocks. In fact, while all processors which lie entirely inside one of the regions can directly use the stencil values copied during the previous phase, the stencil data for the boundary processors can be obtained only by interpolating values stored on processors residing in different block subgroups (see Fig. 13). At the end of this phase, each line of boundary processors will have stored locally the correct stencil data needed by the contiguous (in the "spherical" topology sense) line of processors. For example, the right boundary line of processors of block V will have stored the stencil data needed by the top boundary line of processors of block II.

• T_C is the time required to transfer the stencil data (calculated during the interpolation phase) from one boundary line of processors to the line of processors in which it is needed. For example, in order to calculate the η derivative at the top boundary of block II, processor P_{120} needs to use the stencil data stored during the previous step into processor P_{210}.

We emphasize that of these four phases, only the second one (i.e., the stencil copy) is absent on serial machines.

It is clear that the contribution of T_E to the total execution time is dominant in sequential codes as the mesh is refined. On the other hand, if one fixes the number of grid points assigned to each processing node then T_E, T_S, and

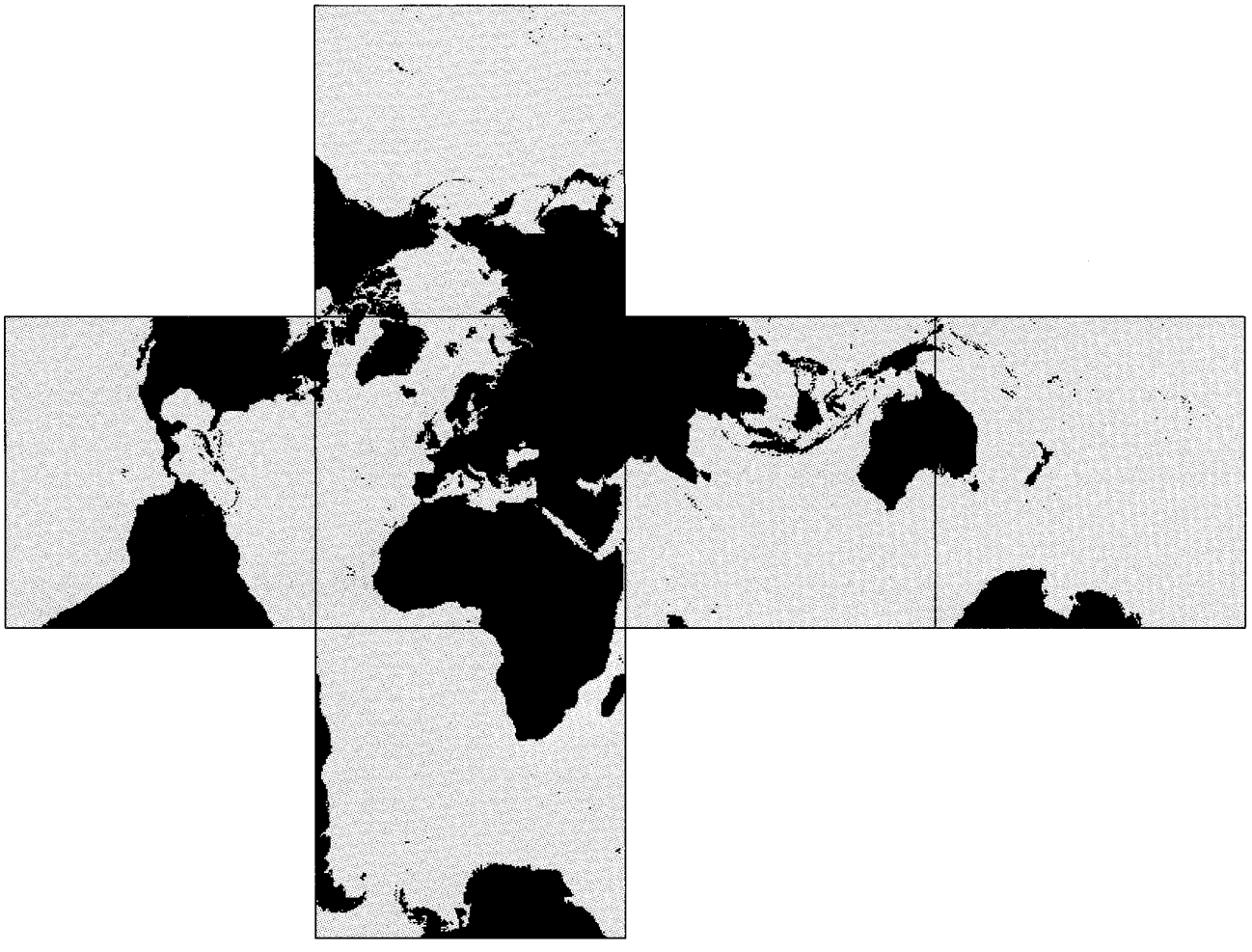

FIG. 16. Rotated orography of the Earth represented on the ''cubed sphere.'' A rigid rotation of the continental masses was applied in order to place Rome at the center of one of the blocks.

T_I will not change when increasing the total number of processors. These two facts suggest a good efficiency of the ''cubed sphere'' technique on massively parallel computers for large enough problems.

The details of the parallelization of the ''cubed sphere'' method and its communication requirements on machines supporting 2D and 3D first neighbour connecting topologies will be presented in a forthcoming paper [15]. However, we would like to conclude this section giving some quantitative estimates for the expected parallel efficiency of our algorithm (see also *Ronchi et al.* [21]).

In Table II we give the theoretical expressions for T_E, T_S, T_I, T_C for 2D finite difference codes executed on a 3D toroidal machine. In Table III, we provide the corresponding expressions calculated for 3D codes on both 3D toroidal and spherical hardware topologies. Finally, in Table IV, we give the measured communication times for a test version of a 2D finite-difference code on two APE100/Quadrics configurations (one with 512 processing nodes arranged in an $8*8*8$ toroidal topology and a second one with 8 nodes on a $2*2*2$ toroidal lattice), together with the corresponding theoretical estimates. Defining P_{tot} as the total number of processing nodes for a given computer, in these tables:

- $P_T = (P_{\text{tot}})^{1/3}$ is the number of processors along the side of a 3D toroidal mesh of processors;
- $P_S = (P_{\text{tot}}/6)^{1/3}$ is the number of processors associated with the sides of each of the six blocks in a spherical machine;
- $N_q = 6$ is the number of blocks;
- N_f is the number of physical variables per grid point;
- N_E is the number of arithmetic operations per physical variable performed every time step on each grid point;
- N_I is the number of arithmetic operations required to obtain the interpolated value of a physical variable on each stencil point;

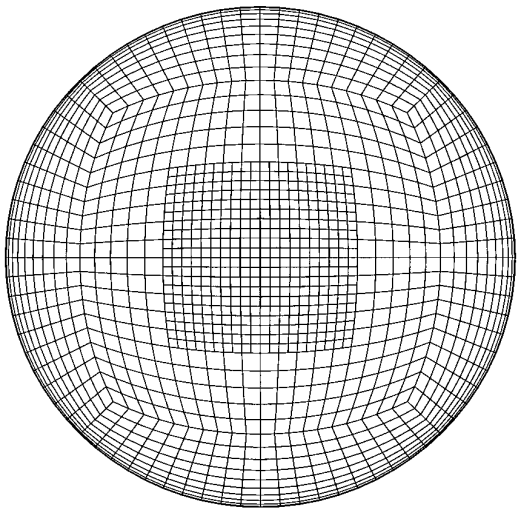

FIG. 17. Schematic representation of a nesting procedure that allows the resolution of small scale details on the regional level. This procedure can in principle be repeated until the necessary resolution is achieved.

• V_F and V_T are, respectively, the effective speed of floating point operations and of data transfer between neighbouring processors.

We define *balanced* as a parallel implementation of the ''cubed sphere'' technique if $T_E > T_I + T_S + T_C$.

Using this criterion and the expressions of Tables II and III, one can easily derive a sort of *breakeven* condition, which specifies the minimum grid size, N, needed to obtain a balanced parallel execution. For a 2D finite difference code with $N_S = 2$ and $N_I = 9$ executing on a toroidal APE100/Quadrics system with an effective typical ratio $V_F/V_T = 4$ corresponding to a 25% efficiency on floating point operations, one obtains the condition

$$N > 52 P_T^2 (2 + 1.3/P_T)/N_E.$$

For a typical 2D value of $N_E = 40$, this gives $N > 14$ on a machine with $P_{\text{tot}} = 8$ and $N > 180$ on a machine with $P_{\text{tot}} = 512$.

For the same values of N_S, N_I, and V_F/V_T, we obtain the following approximate breakeven condition for a 3D code executed on a 3D spherical computer,

$$N > 168 P_S/N_E,$$

having assumed $N_\zeta > 20$. This indicates that for a spherical machine the balance condition will be always satisfied for reasonable values of N_E^{3D}.

6. CONCLUSIONS

We presented the ''cubed sphere'' method, a new gridding technique for the solution of partial differential equations in spherical geometry. We adopted a domain decomposition strategy and represented the spherical surface as six coupled square regions. The six coordinate transformations allow the construction of a global quasi-uniform physical grid, while removing the presence of any geometrical singularity. The requirement of spherical topology uniquely determines the boundary conditions to impose on each region. Exploiting the particular symmetry properties of the decomposition, a very efficient variation of the composite mesh method can be applied to couple the six regions and obtain globally stable and smooth solutions. We presented the results of two standard test cases for numerical approximations to the shallow water equations in spherical geometry. The results of the first test case (linear advection of a cosine bell) prove that the ''cubed sphere'' method is able to treat properly the presence of artificial internal boundaries and that the interpolation procedure employed to couple the six meshes does not degrade the accuracy of the spatial differencing scheme. The second test case (Rossby–Haurwitz nonlinear wave) showed that the ''cubed sphere'' method can provide, with substantial savings in execution times, numerical solutions which are as accurate as those obtainable with the spectral transform method.

Finally, the communication requirements of the ''cubed sphere'' method were shown to allow an efficient and scalable implementation on massively parallel computers supporting 2D and 3D first neighbor connecting topologies.

These results suggest that the ''cubed sphere'' method could provide in many fields of applications in spherical geometry a valid alternative to the spectral method, both in terms of efficiency and accuracy. Of course, such a conclusion must be checked more exhaustively than has been done in the present study. These issues will be discussed in a forthcoming paper, in which the performance of the ''cubed sphere'' method will be analyzed in the context of the full suite of test cases suggested by *Williamson et al.* [30].

Following the introduction of a new methodology for solving partial differential equations in spherical geometry, it is natural to address the question of its potential applications. in the present paper, we were mainly interested in resolving the issues connected with the intrinsic curvature of the two-dimensional physical domain. Nevertheless, the generalization of our method to three dimensions will not introduce any additional complications as far as the coupling algorithm between regions is concerned. We were able to prove this fact explicitly in the context of the parallel implementation of the ''cubed sphere'' method, where the 3D mesh was obtained by overlaying several 2D spherical shells.

It is then quite natural to suggest among the many fields of application of our new method the numerical simulation atmospheric dynamics on a global scale. In addition, we believe that in the ''cubed sphere'' method one can easily

implement a nesting technique for the study of phenomena occurring also on a regional level. For example, because of the absence of any geometric singularity and thanks to the quasi-uniformity of the physical grid, the ''standard'' orography of the Earth (Fig. 15) can be equivalently represented as shown in Fig. 16, where the regions of interest are placed at the center of block I. One can then apply one or more grid refinements as shown in Fig 17, thereby obtaining a higher resolution of the dynamics on a regional scale. One of the clear advantages of this procedure lies in the natural way the boundary conditions imposed on the nested region can be obtained from the coarser global grid.

ACKNOWLEDGMENTS

The authors thank G. Browning for many lively and illuminating discussions and for allowing them to use his polar stereographic composite grid code. We also thank L. Bengtsson for pointing out, during the final stages of this work, the original paper by Sadourny on the cubic–gnomic decomposition of the sphere. We acknowledge R. Cannata for his help in checking the expressions for the differential operators and A. Rossi, E. Lombardi, and G. Pisacane for their invaluable assistance in visualizing the results. We also thank S. Pratesi for his efficient work on the preliminary 2D APE100/Quadrics communication routines and M. Torelli, M. Cosimi, and W. Rinaldi for their analysis on the hardware feasibility of the spherical machine. Two of us (C.R. and R.I.) thank the INFN APE100 group for the hospitality provided during the early stages of this project. Finally, we acknowledge the constant support and encouragement given to this work by N. Cabibbo. The computations were performed on Cray YMP and APE/Quadrics computers at the ENEA C.R. Casaccia and INFN Roma Computing Centers.

REFERENCES

1. A. Arakawa and V. R. Lamb, *Methods in Computational Physics,* Vol. 17, edited by J. Chang (Academic Press, New York, 1977), p. 173.
2. A. Bartoloni *et al*., *Int. J. Mod. Phys. C* **4,** 955 (1993).
3. A. Bartoloni *et al*., *Int. J. Mod. Phys. C* **4,** 969 (1993).
4. G. L. Browning, J. J. Hack, and P. N. Swarztrauber, *Mon. Weather Rev.* **117,** 1058 (1989).
5. N. Cabibbo and P. S. Paolucci, *Int. J. Mod. Phys. C*, to appear.
6. G. Chessire and W. D. Henshaw, *J. Comput. Phys.* **90,** 1 (1990).
7. ECMWF, ''Techniques for Horizontal Discretization in Numerical Weather Prediction Models,'' *Proceedings, Workshop held at ECMWF, November 2–4, 1987, European Centre for Medium-Range Weather Forecasts, 1988.*
8. J. J. Hack and R. Jakob, NCAR Technical Note NCAR/TN-343 +STR, National Center for Atmospheric Research, Boulder, CO, 1992 (unpublished).
9. B. Haurwitz, *J. Mar. Res.* **3,** 254 (1940).
10. W. D. Henshaw, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 1985.
11. K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing* (McGraw–Hill, Singapore, 1984).
12. R. Iacono and C. Ronchi, manuscript in preparation.
13. M. J. Naughton, G. L. Browning, and W. Bourke, *Mon. Weather Rev.* **121,** 1058 (1993).
14. S. A. Orszag, *Mon. Weather Rev.* **102,** 56 (1974).
15. P. S. Paolucci, S. Pratesi, A. Matteuzzi, R. Iacono, and C. Ronchi, manuscript in preparation.
16. N. A. Phillips, *J. Meteor. Soc. Japan,* 75th Anniversary Volume, **262** (1957).
17. N. A. Phillips, *Mon. Weather Rev.* **87,** 333 (1959).
18. N. A. Phillips, in *Proceedings, International Symposium on Numerical Weather Prediction, Tokyo, Japan, 1962* (Meteor. Soc. Japan, Tokyo, 1962), p 109.
19. R. J. Purser, *Mon. Weather Rev.* **116,** 1057 (1988).
20. P. J. Rasch and D. L. Williamson, *J. Geophys. Res.* **96,** 13123 (1991).
21. C. Ronchi, R. Iacono, and P. S. Paolucci, ''Proceedings, HPCN Europe '95, MIlan, Italy, May 3–5, 1995, in *Lecture Notes in Computer Science,* (edited by G. Goos, J. Hartmanis, and J. van Leeuwen).
22. W. S. Russel and P. R. Eiseman, *J. Comput. Phys.,* submitted.
23. R. Sadourny, A. Arakawa, and Y. Mintz, *Mon. Weather Rev.* **96,** 351 (1968).
24. R. Sadourny, *Mon. Weather Rev.* **100,** 136 (1972).
25. G. Starius, *Numer. Math.* **28,** 243 (1977).
26. Starius, *Numer. Math.* **35,** 241 (1980).
27. H. S. Stone, *High Performance Computer Architecture* (Addison Wesley, Reading, MA, 1990).
28. D. L. Williamson, *Tellus* **20,** 642 (1968).
29. D. L. Williamson, *GARP Publi. Ser. No. 17,* Vol. II, (GARP, Geneva, 1979), p. 51.
30. D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber, *J. Comput. Phys.* **102,** 211 (1992).