1997

Lagrange-Galerkin methods on spherical geodesic grids

Giraldo, F. X.

Journal of Computational Physics / Volume 136, Issue 1, 197-213
http://hdl.handle.net/10945/25512

JOURNAL OF COMPUTATIONAL PHYSICS **136,** 197–213 (1997)
ARTICLE NO. CP975771

Lagrange–Galerkin Methods on Spherical Geodesic Grids

Francis X. Giraldo[1]

Naval Research Laboratory, Monterey, California 93943

Received December 12, 1996; revised June 13, 1997

Lagrange–Galerkin finite element methods that are high-order accurate, exactly integrable, and highly efficient are presented. This paper derives generalized natural Cartesian coordinates in three dimensions for linear triangles on the surface of the sphere. By using these natural coordinates as the finite element basis functions we can integrate the corresponding integrals exactly thereby achieving a high level of accuracy and efficiency for modeling physical problems on the sphere. The discretization of the sphere is achieved by the use of a spherical geodesic triangular grid. A tree data structure that is inherent to this grid is introduced; this tree data structure exploits the property of the spherical geodesic grid, allowing for rapid searching of departure points which is essential to the Lagrange–Galerkin method. The generalized natural coordinates are also used for determining in which element the departure points lie. A comparison of the Lagrange-Galerkin method with an Euler–Galerkin method demonstrates the impressive level of high order accuracy achieved by the Lagrange–Galerkin method at computational costs comparable or better than the Euler–Galerkin method. In addition, examples using advancing front unstructured grids illustrate the flexibility of the Lagrange–Galerkin method on different grid types. By introducing generalized natural coordinates and the tree data structure for the spherical geodesic grid, the Lagrange–Galerkin method can be used for solving practical problems on the sphere more accurately than current methods, yet requiring less computer time.

1. INTRODUCTION

Advection governs the most important phenomena of atmospheric and ocean dynamics, namely the transport processes of the velocities. However, this process presents a formidable challenge for many numerical methods including finite elements. The difficulty lies in the lack of self-adjointness of the mathematical operator. One way to avoid this problem is by using a Lagrangian reference frame. This approach has many advantages, including sidestepping the CFL condition as well as increasing the accuracy of the scheme. This paper presents Lagrange–Galerkin methods for the advection equation on the sphere using geodesic triangulations. Lagrange–Galerkin methods have increased in popularity in the last 10 years because they offer increased accuracy and efficiency by virtue of their independence on the CFL condition. Detailed results and analyses are given in one-dimension in [13, 14] and in two dimensions on the plane in [2, 3, 17]. Very little work has been done on the weak Lagrange–Galerkin method on spherical domains; a thorough review of the literature reveals no published work in this subject. This paper assists in filling this gap in the literature. Spherical geodesic grids have been around for quite some time [20]. However, these methods have recently been rediscovered [9], as more and more researchers have begun to move away from spectral methods toward finite volumes, finite elements, and finite difference methods for spherical domains.

On spherical geometries, spherical coordinates appear to be the obvious coordinate system for formulating the problem; however, there are numerical singularities associated with the poles. This issue can be circumvented by either rotating the spherical coordinate system as in [11] or by mapping to Cartesian coordinates whenever we are in the vicinity of the poles as is done in [19]. This paper takes a different approach, namely, remaining in Cartesian space throughout [15]. This strategy offers some clear advantages.

First, the spherical geodesic grids are constructed in Cartesian space. While it is relatively inexpensive to compute and store the spherical coordinates as well, it is unnecessary and can be omitted. Second, in Cartesian space we can construct natural (or area) coordinates for triangles. This paper shows that these natural coordinates, although three-dimensional, can still be integrated exactly as in the two-dimensional planar case. Finally, the departure points for the Lagrange–Galerkin method need no special treatment as all operations are executed in Cartesian space where numerical singularities associated with the poles do not exist.

In Section 2, the model equation used in this study is presented. An Euler–Galerkin formulation is used as a comparison to the proposed Lagrange–Galerkin method. Section 3 presents the discretization of the equation in the Euler–Galerkin and Lagrange–Galerkin formulations. Section 4 introduces the generalized natural coordinate developed and discusses how these coordinates can be used

[1] This research was conducted while the author was an ONR/ASEE fellow at the Naval Research Laboratory. E-mail: giraldo@nrlmry.navy.mil.

0021-9991/97 $25.00

to obtain exact integrations of the finite element equations and their role in accelerating the searching process of the departure points. In Section 5, the spherical geodesic grid is discussed and the tree data structure developed for searching is introduced. Section 5 also describes the advancing front unstructured grids used to show the flexibility of the Lagrange–Galerkin method. Section 6 presents a numerical test case along with some error norms and conservation measures which illustrate the accuracy and conservation of the Lagrange–Galerkin method on both the spherical geodesic grids and the advancing front unstructured grids.

2. GOVERNING EQUATION

The conservative differential form of the advection equation is

$$\frac{\partial \varphi}{\partial t} + \nabla \cdot (\mathbf{u}\varphi) = 0, \tag{1}$$

where φ is some conserved variable, $\mathbf{u}$ is the velocity vector, and ∇ is the divergence operator. In spherical coordinates the equation appears as

$$\begin{aligned} &\frac{\partial \varphi}{\partial t} + \left[\frac{\tilde{u}}{a \cos\theta} \frac{\partial \varphi}{\partial \lambda} + \frac{\tilde{v}}{a} \frac{\partial \varphi}{\partial \theta} \right] \\ &\quad + \left[\varphi \left(\frac{1}{a \cos\theta} \frac{\partial \tilde{u}}{\partial \lambda} + \frac{1}{a} \frac{\partial \tilde{v}}{\partial \theta} - \frac{\tilde{v}}{a} \tan\theta \right) \right] = 0, \end{aligned} \tag{2}$$

where a is the radius of the sphere, $(\tilde{u}, \tilde{v})$ are the zonal and meridional velocity components, and (λ, θ) are the longitudinal and latitudinal coordinates. The first bracketed term represents the operator $\mathbf{u} \cdot \nabla\varphi$ and the second term represents $\varphi \nabla \cdot \mathbf{u}$. Since the first terms in each of the brackets become singular at the poles, it is preferable to use the Cartesian form

$$\frac{\partial \varphi}{\partial t} + \left[u \frac{\partial \varphi}{\partial x} + v \frac{\partial \varphi}{\partial y} + w \frac{\partial \varphi}{\partial z} \right] + \left[\varphi \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right] = 0 \tag{3}$$

which is the form used in this paper.

3. DISCRETIZATION

This section introduces the discretization of the conservative form of the advection equation in Cartesian coordinates by the Euler–Galerkin and Lagrange–Galerkin formulations. (The term Euler–Galerkin is used to refer to the conventional Bubnov–Galerkin finite element method, which when written semi-implicitly is equivalent to the Crank–Nicholson finite element method.) Two types of Lagrange–Galerkin formulations are considered: the direct and the weak methods. Both the Euler–Galerkin and Lagrange–Galerkin methods represent second-order accurate schemes in both space and time.

3.1. *Euler–Galerkin*

In Eulerian schemes the evolution of the system is monitored from fixed positions in space; consequently, they are the easiest methods to implement as all variable properties are computed at the grid points comprising the discretization of the domain. Discretizing (3) by an Eulerian finite element method, we arrive at the elemental equations

$$\int_\Omega \psi \frac{\partial \varphi}{\partial t} + \nabla \cdot (\mathbf{u}\varphi\psi) - \varphi \mathbf{u} \cdot \nabla\psi \, d\Omega = 0 \tag{4}$$

which can be written in matrix form as

$$M\dot{\varphi} - A\varphi = R,$$

where M is the mass matrix, A is the advection, and R is the boundary terms which are given by

$$M_{ij} = \int_\Omega \psi_i \psi_j \, d\Omega,$$

$$A_{ij} = \int_\Omega \sum_{k=1}^{\mathbf{nd}} [\nabla\psi_i \cdot (\mathbf{u}_k \psi_k) \psi_j] \, d\Omega,$$

and

$$R_i = -\int_\Gamma \mathbf{N} \cdot (\mathbf{u}\varphi\psi_i) \, d\Gamma,$$

where ψ are the finite element basis functions, $\mathbf{N}$ is the outward pointing normal vector of the boundaries, and i, j, and k all vary from 1 to **nd**. For linear, quadratic, and cubic triangles **nd** = 3, 6, and 10, respectively. Discretizing this relation in time gives the following family of algorithms

$$\begin{aligned} [M - \Delta t\, \theta A]\varphi^{n+1} = [M + \Delta t(1-\theta) A]\varphi^n \\ + \Delta t[\theta R^{n+1} + (1-\theta) R^n], \end{aligned} \tag{5}$$

where $\theta = 0, \frac{1}{2}$, and 1 give explicit, semi-implicit, and implicit schemes, respectively. However, this class of methods is dispersive and limited to small time steps in order to satisfy the CFL condition for stability. Lagrange methods, on the other hand, suffer from neither of these ailments. Two types of Lagrange–Galerkin methods will now be introduced: the direct and the weak methods.

3.2. *Direct Lagrange–Galerkin*

Lagrangian methods belong to the general class of upwinding methods. These methods incorporate characteristic information into the numerical scheme. The Lagrangian form of (1) is

$$\frac{d\varphi}{dt} = -\varphi \nabla \cdot \mathbf{u} \tag{6}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}, t), \tag{7}$$

where

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$$

denotes the total (or Lagrangian) derivative. Discretizing this equation by the direct Lagrange–Galerkin method yields the elemental equations

$$\int_\Omega \psi \frac{d\varphi}{dt} + \psi\varphi \nabla \cdot \mathbf{u}\, d\Omega = 0 \tag{8}$$

which can be written in matrix form as

$$M\dot{\varphi} + D\varphi = 0,$$

where M is the mass matrix and D is the divergence and they are given by

$$M_{ij} = \int_\Omega \psi_i \psi_j \, d\Omega$$

and

$$D_{ij} = \int_\Omega \sum_{k=1}^{\mathbf{nd}} [\psi_i \psi_j (\mathbf{u}_k \cdot \nabla \psi_k)] \, d\Omega.$$

Discretizing this relation in time gives the family of algorithms

$$[M - \Delta t\, \theta\, D]\varphi^{n+1} = [M + \Delta t(1 - \theta)\, D]\varphi_d^n, \tag{9}$$

where $\varphi^{n+1} = \varphi(\mathbf{x}, t + \Delta t)$ and $\varphi_d^n = \varphi(\mathbf{x} - \boldsymbol{\alpha}, t)$ are the solutions at the arrival and departure points, respectively, and $\boldsymbol{\alpha} = \mathbf{x}(t + \Delta t) - \mathbf{x}(t)$ denotes the trajectory vector. Integrating (7) by the midpoint rule yields

$$\boldsymbol{\alpha} = \Delta t\, \mathbf{u}\left(\mathbf{x}\left(t + \frac{\Delta t}{2}\right), t + \frac{\Delta t}{2}\right)$$

which defines a recursive relation for the departure points. Since in general we do not know the midpoint trajectory $\mathbf{x}(t + \Delta t/2)$, we must approximate it. There are many choices but one option is to approximate the midpoint trajectory via a second-order Taylor series expansion which then yields the trajectory relation

$$\boldsymbol{\alpha} = \Delta t\, \mathbf{u}\left(\mathbf{x} - \frac{\boldsymbol{\alpha}}{2}, t + \frac{\Delta t}{2}\right) \tag{10}$$

which is a recursive relation that typically requires between three and five iterations for convergence. Note that the midpoint trajectories will not fall on grid points and so they must be interpolated in some fashion. Linear interpolation is sufficient for this purpose but higher order interpolations are required for the conserved variable φ and at least third-order accurate interpolation is required in order to obtain a second-order scheme [7].

This method is called the direct Lagrange–Galerkin method because the method of weighted residuals is applied directly onto the Lagrangian form of the equations. This method is also known as the *semi-Lagrangian* method because the equations are solved in Lagrangian form but the departure points are chosen such that they arrive at grid points at the end of the time step.

Interpolation is required in this approach in order to obtain the unknown values at the departure points. For uniform grids it is possible to use cubic splines, Hermite, or Lagrange interpolation; for unstructured grids as in the spherical geodesic grids, interpolation is rather difficult and costly. In addition, the direct Lagrange–Galerkin method is nonconservative which may cause problems for long time integrations [7]. The weak Lagrange–Galerkin method discussed in the following section is exactly conservative because integration rather than interpolation is used to obtain the values at the departure points [14].

3.3. *Weak Lagrange–Galerkin*

In the weak Lagrange–Galerkin method we begin with the Eulerian form (1) and then apply the method of weighted residuals in order to find the adjoint operator. This leads to the integral equation

$$\int_\Omega \frac{\partial(\varphi\psi)}{\partial t} + \nabla \cdot (\mathbf{u}\varphi\psi) - \varphi\left[\frac{\partial \psi}{\partial t} + \mathbf{u} \cdot \nabla \psi\right] d\Omega = 0, \tag{11}$$

where the bracketed term represents the advection operator acting on the finite element basis functions. In this approach, the basis functions must satisfy this operator; therefore, the basis functions are functions of both space

and time. These conditions allow the bracketed term to vanish. The elemental equations can now be written in matrix form as

$$\frac{\partial(M\varphi)}{\partial t} = R,$$

where M is the mass matrix and R is the boundary terms which are defined as in the Eulerian case. Discretizing this relation in time gives the family of algorithms

$$[M]\varphi^{n+1} = [M_d^n]\varphi_d^n + \Delta t[\theta R^{n+1} + (1-\theta) R_d^n], \quad (12)$$

where

$$M_{d,ij}^n = \int_{\Omega_d^n} \psi_i \psi_j \, d\Omega_d^n,$$

$$R_{d,i}^n = -\int_{\Gamma_d^n} \mathbf{N} \cdot (\mathbf{u}\varphi\psi_i) \, d\Gamma_d^n,$$

and Ω_d^n denotes the Lagrangian element formed by the departure points of the finite element Ω. Once again, $\theta = 0, \frac{1}{2}$, and 1 yield explicit, semi-implicit, and implicit schemes, respectively. On the surface of the sphere no formal boundary conditions exist except those of periodicity. Since these conditions are already accounted for by virtue of the finite element connectivity matrix, the boundary vector R vanishes.

At this point, all of the Galerkin matrix equations have been derived in very general terms—in other words, no assumptions have been made about the finite element basis functions and so these equations are valid for any type of element and basis function. In general, the resulting finite element integrals need to be solved by numerical integration methods but, by choosing the basis functions and finite elements wisely, we can avoid numerical integration and instead solve the integrals exactly. The following section describes a set of basis functions for triangular finite elements in three dimensions which can be integrated exactly.

4. GENERALIZED NATURAL COORDINATES

In two dimensions on the plane, we can obtain the exact finite element integrals for any of the terms in the elemental equations presented thus far, provided that we use linear triangular elements. On the sphere, although the surface is actually three-dimensional we can still recover much of the same properties as in the planar case. After all, the surface of the sphere is quasi two-dimensional because there are only two independent space variables, while the third variable is restricted by the fact that the point must remain on the surface of the sphere. Let x and y be independent; then we may write z as

$$z = f(x, y) = \sqrt{a^2 - x^2 - y^2}.$$

Therefore, let us construct finite element basis functions on the surface of the sphere using linear triangular elements but in three dimensions. Natural coordinates can be derived for any simplex element by constructing them to be the linear interpolation functions within the element. The conditions to be satisfied by these interpolants on a triangle in three-dimensional space are

$$x = \psi_1 x_1 + \psi_2 x_2 + \psi_3 x_3$$

$$y = \psi_1 y_1 + \psi_2 y_2 + \psi_3 y_3$$

$$z = \psi_1 z_1 + \psi_2 z_2 + \psi_3 z_3$$

which just says that the coordinates within the triangular element are linearly dependent on the vertices of that element. When inverted, this system yields the general relation for the natural coordinates,

$$\psi_i = \frac{a_i x + b_i y + c_i z}{\text{deter} \triangle}, \quad (13)$$

where

$$a_i = y_j z_k - y_k z_j, \quad b_i = x_k z_j - x_j z_k, \quad c_i = x_j y_k - x_k y_j,$$

$$\text{deter} \triangle = \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix},$$

and

$$i, j, k = 1, ..., 3.$$

By using the definition of the natural coordinates (13) and the fact that the three nodes on each triangle define a plane,

$$\mathbf{N} \cdot (\mathbf{x} - \mathbf{x}_1) = 0,$$

where $\mathbf{N}$ is the outward pointing normal to the triangle and defined by

$$\mathbf{N} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix}, \quad (14)$$

it can be shown that the natural coordinates satisfy the condition

$$\psi_1(x, y, z) + \psi_2(x, y, z) + \psi_3(x, y, z) = 1.$$

This is a necessary condition for a consistent and monotonic interpolation. These natural coordinates can now be used as the finite element basis functions. By following the derivation of Sylvester's formula [4] we can derive the generalized formula extended to triangles in three-dimensional space,

$$\int \psi_1^\alpha \psi_2^\beta \psi_3^\gamma d\Omega = \frac{\text{cross} \triangle\, \alpha!\, \beta!\, \gamma!}{(\alpha + \beta + \gamma + 2)!}, \tag{15}$$

where

$$\text{cross} \triangle = |\mathbf{N}|$$

which is valid only for the finite element basis functions introduced in this section. This relation is almost identical to Sylvester's formula; the only exception is that 2Ω has been replaced by cross $\triangle$, where Ω is the area of the triangle. For the special case that the three-dimensional domain lies entirely on a plane, the generalized formula recovers Sylvester's formula. This relation can now be used to obtain exact integrals for all of the terms in the Euler–Galerkin and Lagrange–Galerkin formulations.

4.1. *Finite Element Matrices*

Using the generalized natural coordinates (13) and the generalized exact integral relation (15), we can now write closed form solutions for all of the finite element integrals presented in Section 3. The mass matrix is

$$M_{ij} = \frac{\text{cross} \triangle}{24} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix},$$

the advection matrices are

$$A_{ij}^u = \frac{\text{cross} \triangle}{24 \text{ deter} \triangle}$$

$$\begin{bmatrix} a_1(2u_1 + u_2 + u_3) & a_1(u_1 + 2u_2 + u_3) & a_1(u_1 + u_2 + 2u_3) \\ a_2(2u_1 + u_2 + u_3) & a_2(u_1 + 2u_2 + u_3) & a_2(u_1 + u_2 + 2u_3) \\ a_3(2u_1 + u_2 + u_3) & a_3(u_1 + 2u_2 + u_3) & a_3(u_1 + u_2 + 2u_3) \end{bmatrix},$$

$$A_{ij}^v = \frac{\text{cross} \triangle}{24 \text{ deter} \triangle}$$

$$\begin{bmatrix} b_1(2v_1 + v_2 + v_3) & b_1(v_1 + 2v_2 + v_3) & b_1(v_1 + v_2 + 2v_3) \\ b_2(2v_1 + v_2 + v_3) & b_2(v_1 + 2v_2 + v_3) & b_2(v_1 + v_2 + 2v_3) \\ b_3(2v_1 + v_2 + v_3) & b_3(v_1 + 2v_2 + v_3) & b_3(v_1 + v_2 + 2v_3) \end{bmatrix},$$

$$A_{ij}^w = \frac{\text{cross} \triangle}{24 \text{ deter} \triangle}$$

$$\begin{bmatrix} c_1(2w_1 + w_2 + w_3) & c_1(w_1 + 2w_2 + w_3) & c_1(w_1 + w_2 + 2w_3) \\ c_2(2w_1 + w_2 + w_3) & c_2(w_1 + 2w_2 + w_3) & c_2(w_1 + w_2 + 2w_3) \\ c_3(2w_1 + w_2 + w_3) & c_3(w_1 + 2w_2 + w_3) & c_3(w_1 + w_2 + 2w_3) \end{bmatrix};$$

and the divergence matrices are

$$D_{ij}^u = (a_1u_1 + a_2u_2 + a_3u_3) \frac{\text{cross} \triangle}{24 \text{ deter} \triangle} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix},$$

$$D_{ij}^v = (b_1v_1 + b_2v_2 + b_3v_3) \frac{\text{cross} \triangle}{24 \text{ deter} \triangle} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix},$$

$$D_{ij}^w = (c_1w_1 + c_2w_2 + c_3w_3) \frac{\text{cross} \triangle}{24 \text{ deter} \triangle} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix},$$

where $A_{ij} = A_{ij}^u + A_{ij}^v + A_{ij}^w$ and $D_{ij} = D_{ij}^u + D_{ij}^v + D_{ij}^w$. Because the finite element basis functions and the exact integral formula described in this section are general, we can apply the same procedure to obtain exact integrals for any finite element integral and not just those presented here. These definitions simplify the finite element integrals and eliminate the need for quadrature formulas which have been known to diminish the efficiency and stability properties of Galerkin methods, theoretically speaking. In the following section we describe how quadrature formulas can be eliminated completely from the Lagrange–Galerkin method.

4.2. *Quadrature vs Exact Integration*

The primary operation in the weak Lagrange–Galerkin method is integration, in contrast to interpolation, which is used in the direct method. However, the integrals on the right-hand side of (12) contain terms that are not defined exclusively on a grid (Eulerian) element. Instead, the Lagrangian elements defined by the departure points in gen-

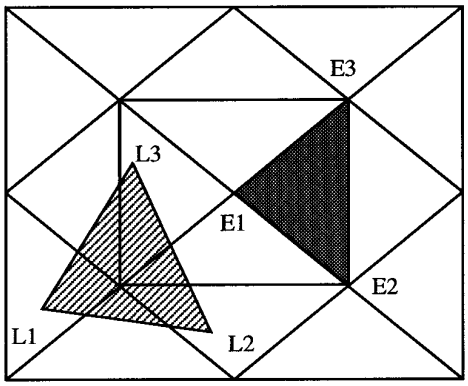

FIG. 1. Depiction of the Eulerian element (E1, E2, E3) and its corresponding Lagrangian element (L1, L2, L3).

eral will span across many grid elements as illustrated in Fig. 1. In most implementations of the Lagrange–Galerkin method, these integrals are computed by quadrature formulas. In [17], Priestley introduced a means of eliminating quadrature integration from the Lagrange–Galerkin method for planar two-dimensional problems. This was achieved by writing the finite element basis functions of the Lagrangian element in terms of the Eulerian (or grid) element basis functions. The Eulerian elements are the elements formed by the three nodes of the triangles comprising the grid. When we speak of a grid discretizing a domain, we are referring to the Eulerian elements. These elements remain fixed for all time, assuming adaptive grids are not used. In contrast, the Lagrangian elements are the triangular elements formed by the departure points of the vertices of the Eulerian elements. Since the departure point corresponding to a given grid point varies with time, the Lagrangian element consequently also varies with time. Exact integration is only possible by using the natural (or area) coordinates as the basis functions for the Eulerian and Lagrangian elements. It is then straightforward to obtain the integrals using Sylvester's formula. Note that as long as we use linear triangular elements we assume nothing. In other words, the Eulerian and Lagrangian elements are both triangles, albeit, in general not of the same size or shape (see Fig. 1). However, if we were to use quadratic or cubic elements, then we could not guarantee the Lagrangian element to be a triangle but we could decompose this polygon into its corresponding set of smaller triangles [18]. Once this has been accomplished, the exact integration method could then be applied. It is uncertain what affects this strategy would have on the solution because the Eulerian elements would be integrated as higher order elements (p refinement) while the Lagrangian elements would be decomposed into many smaller linear elements (h refinement).

This section illustrates that the same ideas used in two-dimensional planar domains can be extended to three dimensions by using the finite element natural coordinates (13) and the exact integral relation (15). In [17], the Lagrangian element is decomposed into triangular elements inside the Eulerian elements. This involves finding the points where the Lagrangian elements intersect other Eulerian elements. Upon storing all of these intersection points, we then proceed to triangulate the resulting subdomain. In each of these subtriangles spanning the total Lagrangian element, we can write the subtriangle Lagrangian basis functions as Eulerian basis functions, where these basis functions are the generalized natural coordinates. Since the interpolation of a given point belonging to both the Eulerian and Lagrangian elements must have the same value for consistency, we can construct the Lagrangian finite element basis functions by using the equalities

$$x = \psi_1^L x_1^L + \psi_2^L x_2^L + \psi_3^L x_3^L = \psi_1^E x_1^E + \psi_2^E x_2^E + \psi_3^E x_3^E$$
$$y = \psi_1^L y_1^L + \psi_2^L y_2^L + \psi_3^L y_3^L = \psi_1^E y_1^E + \psi_2^E y_2^E + \psi_3^E y_3^E$$
$$z = \psi_1^L z_1^L + \psi_2^L z_2^L + \psi_3^L z_3^L = \psi_1^E z_1^E + \psi_2^E z_2^E + \psi_3^E z_3^E$$

which can be written in matrix form as

$$\begin{bmatrix} x_1^L & x_2^L & x_3^L \\ y_1^L & y_2^L & y_3^L \\ z_1^L & z_2^L & z_3^L \end{bmatrix} \begin{Bmatrix} \psi_1^L \\ \psi_2^L \\ \psi_3^L \end{Bmatrix} = \begin{Bmatrix} \psi_1^E x_1^E + \psi_2^E x_2^E + \psi_3^E x_3^E \\ \psi_1^E y_1^E + \psi_2^E y_2^E + \psi_3^E y_3^E \\ \psi_1^E z_1^E + \psi_2^E z_2^E + \psi_3^E z_3^E \end{Bmatrix},$$

where (ψ_i^E, x_i^E) and (ψ_i^L, x_i^L) are the Eulerian and Lagrangian finite element basis functions and node points, respectively. Upon inverting this system, we obtain the expression for the Lagrangian finite element basis functions

$$\psi_i^L = \frac{a_i^L \psi_1^E + b_i^L \psi_2^E + c_i^L \psi_3^E}{\text{deter } \triangle^L}, \tag{16}$$

where

$$a_i^L = x_1^E \xi_i^L - y_1^E \eta_i^L + z_1^E \zeta_i^L, \quad b_i^L = x_2^E \xi_i^L - y_2^E \eta_i^L + z_2^E \zeta_i^L,$$
$$c_i^L = x_3^E \xi_i^L - y_3^E \eta_i^L + z_3^E \zeta_i^L, \quad \xi_i^L = y_j^L z_k^L - y_k^L z_j^L,$$
$$\eta_i^L = x_j^L z_k^L - x_k^L z_j^L, \quad \zeta_i^L = x_j^L y_k^L - x_k^L y_j^L$$

and

$$i, j, k = 1, ..., 3.$$

Note that we can now use (15) to obtain exact integrals for the Lagrangian elements. For example, the mass vector on the right-hand side of (12) can be written as

$$M^n_{d,ij}\varphi^n_{d,j} = \sum_{E=1}^{\mathbf{nel}} C \begin{Bmatrix} a_1^L(2\varphi_1^E + \varphi_2^E + \varphi_3^E) + b_1^L(\varphi_1^E + 2\varphi_2^E + \varphi_3^E) \\ \qquad + c_1^L(\varphi_1^E + \varphi_2^E + 2\varphi_3^E) \\ a_2^L(2\varphi_1^E + \varphi_2^E + \varphi_3^E) + b_2^L(\varphi_1^E + 2\varphi_2^E + \varphi_3^E) \\ \qquad + c_2^L(\varphi_1^E + \varphi_2^E + 2\varphi_3^E) \\ a_3^L(2\varphi_1^E + \varphi_2^E + \varphi_3^E) + b_3^L(\varphi_1^E + 2\varphi_2^E + \varphi_3^E) \\ \qquad + c_3^L(\varphi_1^E + \varphi_2^E + 2\varphi_3^E) \end{Bmatrix},$$

where

$$C = \frac{\text{cross } \triangle^E}{24 \text{ deter } \triangle^L}$$

and **nel** represents the number of grid elements contained within the Lagrangian element (see Fig. 1). However, an efficient grid generator is required in order to write the Lagrangian element in terms of its Eulerian components. For efficiency reasons, it is often simpler to use quadrature rules. This is not to say that the exact integration method is inefficient or unworthy of note. In fact, this approach is quite promising and should be further explored. For convenience we use a quintic quadrature rule in this study. The weak Lagrange–Galerkin method is exactly conservative but this is only theoretically true for the exact integration method. However, the results in [17] show that the numerical implementations of the exact and quadrature methods yield very similar results. In addition, the results in [8] also show this to be true for 2D advection on the plane. The only problem with the quadrature method is that it may suffer from instabilities for certain Courant numbers, theoretically speaking [12]. In practice, however, the method has not been known to fail [17]. In the next section, the spherical geodesic grid used in this paper is presented and the tree data structure developed for rapid searching is introduced.

5. SPHERICAL GEODESIC GRID

A spherical geodesic grid can be constructed by first defining a background icosahedron. This icosahedron is defined by the following 12 points [10]: one at each pole,

$$[\lambda_1, \theta_1] = \left[0, \frac{\pi}{2}\right], \quad [\lambda_{12}, \theta_{12}] = \left[0, -\frac{\pi}{2}\right];$$

five in the northern hemisphere,

$$[\lambda_i, \theta_i] = \left[\left(i - \frac{3}{2}\right)\frac{2\pi}{5}, 2 \arcsin\left(\frac{1}{2\cos\frac{3\pi}{10}}\right) - \frac{\pi}{2}\right]$$

for $i = 2, ..., 6$;

and five more in the southern hemisphere

$$[\lambda_i, \theta_i] = \left[(i-7)\frac{2\pi}{5}, -2 \arcsin\left(\frac{1}{2\cos\frac{3\pi}{10}}\right) - \frac{\pi}{2}\right]$$

for $i = 7, ..., 11$.

These 12 initial grid points are used to form 20 equilateral triangles which completely encompass the sphere. Each triangle may now be subdivided into four smaller triangles by bisecting each of the three edges of the current triangle. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the coordinates defining an edge. Then the midpoint node is $\mathbf{x}_4 = (\mathbf{x}_1 + \mathbf{x}_2)/2$. This new node must then be projected onto the surface of the sphere, and so it becomes

$$\mathbf{x}_4 = a\frac{\mathbf{x}_4}{|\mathbf{x}_4|},$$

where again a is the radius of the sphere. This process is repeated for each edge. Figure 2 depicts the subdivision (or refinement) process. The process of subdividing each triangle into four smaller triangles can be made efficient by creating an array containing all of the edge data, since each edge is shared by two elements. Call this integer array **iedge[1:nedge,1:4]**, where **nedge** are the number of edges in the grid. Locations 1 and 2 store the identification numbers of the two nodes defining the edge, and locations 3 and 4 store the identification numbers of the elements that share this edge. Then for each edge, we store its midpoint. Once this has been achieved, we can loop through

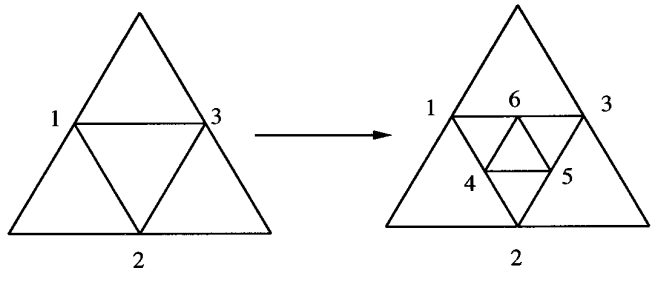

FIG. 2. Refinement of an element for the spherical geodesic grid.

TABLE I

The Spherical Geodesic Grid Parameters as Functions of Refinement Loop **n**

n	npoin	nelem	nedge	ntree	Time (s)
0	12	20	30	20	0.4
1	42	80	120	100	1.4
2	162	320	480	420	2.3
3	642	1280	1920	1700	3.5
4	2562	5120	7680	6820	5.3
5	10242	20480	30720	27300	10.0

Note. The parameters are given for up to five refinement loops along with **cpu** time.

each element and subdivide the element using the previously calculated midpoints corresponding to its three edges. The grids for refinement loops zero through five are illustrated in Figs. 5 through 10 and Table I shows the grid parameters for the five refinement loops along with the **cpu** time required to generate the grids.

5.1. *Tree Data Structure*

Since for each refinement of the spherical geodesic grid each triangle is subdivided into four triangles, this process forms a quadtree-like data structure. Therefore, we can define an integer array **itree[1:ntree,1:9]**, where **ntree** are the number of tree elements. Initially, the tree has 20 elements which coincide with the 20 triangular faces that define the background icosahedron. Locations 1–3 are reserved for the node identification numbers defining the elements. Location 4 contains the parent of the current element and locations 5–8 store the four children of the current element. Finally, location 9 stores the element identification number of the current triangle. This location is zero if this tree element is not an active element in the grid. This occurs, for example, after one refinement for the initial 20 elements. These initial triangular elements are no longer active elements in the grid but rather parents of four smaller triangles and so their location 9 is zero. The children, however, will have nonzero identification numbers because they are active elements of the grid. This data structure can now be used to accelerate the searching process which is required in order to determine the departure points for the Lagrange–Galerkin method.

5.2. *Searching*

The first step in the searching process involves finding in which of the initial 20 elements of the background icosahedron the departure point lies. These constitute the first 20 elements in the tree. Once this tree element has been isolated, we can branch through its children to determine which child owns the point. This process is continued until the tree element that claims the node has a location 9 that is nonzero.

Testing whether a point lies inside a triangular element on the sphere is not all that obvious. On the plane we can use the natural coordinates to determine the inclusion of a point with respect to an element. This is done by building the natural (area) coordinates and then if any of the areas are less than zero, then the point is outside the element. This means that the direction of the surface normal of this point with respect to two of the vertices of the triangular element points in an opposite direction to the normal of the triangle. We can use this same approach using the generalized natural coordinates (13). On a sphere, only the vertices of the triangle lie on the surface, but the rest of the plane does not. Conversely, the departure point will lie on the sphere and thus not on the plane of any of the triangles. Therefore, we must obtain the projection of the departure point onto the plane defined by the three vertices of the triangle. Recall that the equation of the plane is given by $\mathbf{N} \cdot (\mathbf{x} - \mathbf{x}_1) = 0$, where $\mathbf{N}$ is defined in (14). The equation of the vector passing through the origin and the departure point may be written parametrically as

$$\mathbf{x} = t\,\mathbf{x}_d,$$

where $\mathbf{x}_d$ is the departure point and t is the parametric variable which for $t = 0$ yields the origin and for $t = 1$ recovers the departure point. Substituting the parametric equation into the plane equation and simplifying yields

$$t_p = \frac{\mathbf{N} \cdot \mathbf{x}_d}{\mathbf{N} \cdot \mathbf{x}_1},$$

where t_p is the parametric value that defines the projection $\mathbf{x}_p$ of the departure point $\mathbf{x}_d$ onto the plane. Once this projection is obtained, we can then proceed with the natural coordinates. As in the planar case, an inclusion is guaranteed if all the normals point in the same direction. Figure 3 illustrates the case when a point lies inside a given ele-

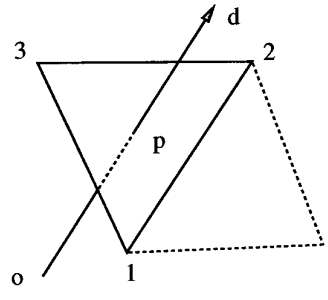

FIG. 3. Inclusion of a departure point.

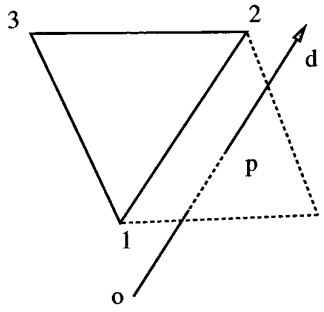

FIG. 4. Exclusion of a departure point.

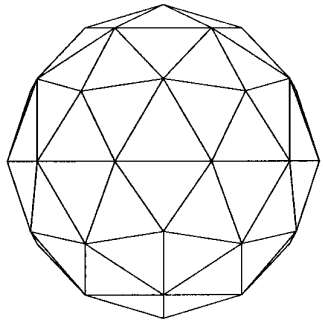

FIG. 6. The spherical geodesic grid after one refinement loop.

ment. Figure 4 shows that a projection of a point can be found on any given plane, but this point need not lie inside the triangle. The dotted lines in these figures represent the extension of the plane and the numbers *1*, *2*, *3* are the three nodes of the element, *o* is the origin, *p* is the projection, and *d* is the departure point. In the exclusion case, the normal defined by $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_p)$ points in the direction opposite to the normal of the triangle $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. Once the initial element is found among the first 20 tree elements, the search becomes a $\log_4$ **ntree** search, where the general expressions

$$\textbf{npoin(n)} = 4^{\mathbf{n}}(12) - 6\sum_{i=0}^{\mathbf{n}-1} 4^i, \quad \textbf{nelem} = 2(\textbf{npoin} - 2),$$

$$\textbf{nedge} = 3(\textbf{npoin} - 2), \quad \textbf{ntree} = 2\sum_{i=0}^{\mathbf{n}} (\textbf{npoin}_i - 2)$$

hold, where **n** are the number of refinement loops, **npoin** is the number of nodes, **nelem** is the number of triangular elements, **nedge** is the number of element edges, and **ntree** are the number of elements in the tree. The above expression for **npoin** is only defined for $\mathbf{n} \geq 1$, where for $\mathbf{n} = 0$ we set $\mathbf{npoin} = 12$ in order to recover the initial icosahedron. Table I illustrates these values for different values of **n** for up to five refinement loops. Also given are the **cpu** times required to generate each of the grids; these times include writing the output files. These grids not only have inherent data structures associated with them but are also extremely efficient for generating large grids. Delaunay [1] and advancing front methods for discretizing a sphere require much more computing time.

5.3. *Advancing Front Unstructured Grids*

Much work has been done on advancing front methods, specifically in the areas of computational fluid dynamics (and aerodynamics), where irregular geometries need to be discretized, say, over an airfoil or an entire aircraft. These methods were developed specifically for these reasons and, as a consequence, are very general (see [5, 6, 8]). However, these methods require an initial front or boundary as a starting point for the triangulation. In the case of the sphere, there are no physical boundaries as such. We can introduce a virtual boundary, say at the equator, and triangulate the northern and southern hemispheres independently and later unite the two hemispheres. In this study we use the equator as the virtual boundary but we can choose any great circle. Although the triangulations obtained with this approach are not as regular or as efficient as those obtained with the geodesic grids it must be pointed out that the advancing front method is not only a grid generator but an adaptive refinement method as well. In other words, it has the capabil-

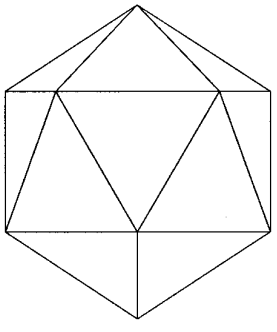

FIG. 5. The initial icosahedron (zero refinement loops).

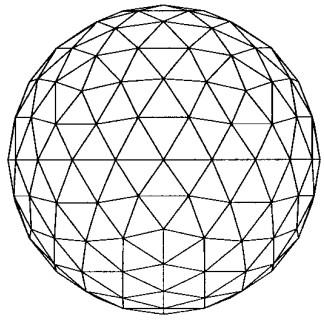

FIG. 7. The spherical geodesic grid after two refinement loops.

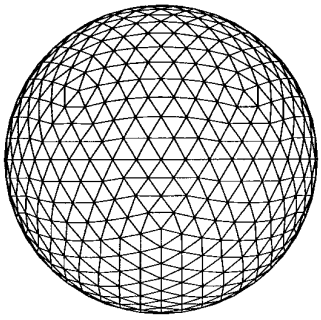

FIG. 8. The spherical geodesic grid after three refinement loops.

ity to dynamically alter the grid if the gradients change dramatically in regions of the sphere. For the purposes of this study, the advancing front method is used only to generate fixed unstructured grids. Obtaining accurate solutions on quasi-structured grids (spherical geodesic) and randomly generated unstructured grids (advancing front) would suggest that the Lagrange–Galerkin method can be used on any kind of grid, including adaptive grids.

The advancing front method used in this study is the spherical version of the method presented in [5, 6, 8]. In [6, 8] a two-dimensional planar advancing front method is described. In [5] a three-dimensional surface triangulator and fully three-dimensional method is described in detail. The advancing front method used in the current study is an *ad hoc* version of the surface triangulator which has been tailored for spherical geometries.

6. NUMERICAL EXPERIMENTS

Numerical experiments are performed on the advection equation on the sphere which is defined by (3). The initial condition is given as in [21] by the cosine wave

$$\varphi_o = \begin{cases} \dfrac{h}{2}\left(1 + \cos\dfrac{\pi r}{R}\right) & \text{if } r < R \\ 0 & \text{if } r \geq R \end{cases}$$

where $h = 100$, $r = a \arccos[\sin\theta_c \sin\theta + \cos\theta_c \cos\theta \cos(\lambda - \lambda_c)]$, $R = a$, and (λ_c, θ_c) is the initial location of the center of the cosine wave . In this study (λ_c, θ_c) is set to $(3\pi/2, 0)$ and the velocity field is assumed to be constant and given by

$$\tilde{u} = +\omega(\cos\theta\cos\alpha + \sin\theta\cos\lambda\sin\alpha)$$

$$\tilde{v} = -\omega\sin\lambda\sin\alpha,$$

where $\omega = 2\pi a/12$ days and α determines the axis of rotation of the flow with respect to the north pole. As an example $\alpha = 0$ yields flow along the equator, whereas $\alpha = \pi/2$ defines flow along a great circle passing through both poles. By using the mapping from spherical to Cartesian space

$$x = a\cos\theta\cos\lambda$$

$$y = a\cos\theta\sin\lambda$$

$$z = a\sin\theta,$$

where

$$\lambda = \arctan\left(\frac{y}{x}\right)$$

and

$$\theta = \arcsin\left(\frac{z}{a}\right),$$

we can write the initial conditions in terms of Cartesian coordinates. This results in the velocity field

$$u = -\tilde{u}\sin\lambda - \tilde{v}\sin\theta\cos\lambda$$

$$v = +\tilde{u}\cos\lambda - \tilde{v}\sin\theta\sin\lambda$$

$$w = +\tilde{v}\cos\theta,$$

along with the analytic solution

$$\varphi_{\text{exact}}(x, y, z, t) = \varphi_o(x - ut, y - vt, z - wt, t)$$

which is the solid body rotation of the cosine wave about the axis defined by α. The mapping from spherical to

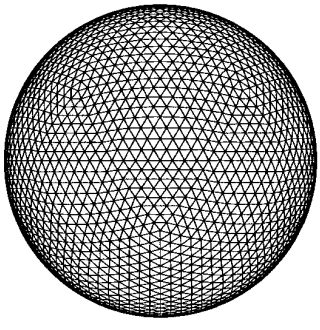

FIG. 9. The spherical geodesic grid after four refinement loops.

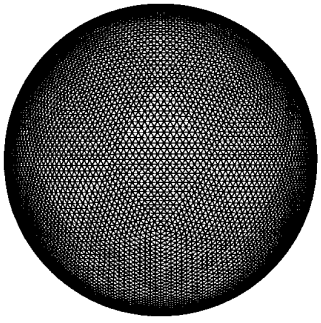

FIG. 10. The spherical geodesic grid after five refinement loops.

Cartesian is only done once at the beginning in order to define the problem. From then on, the problem is solved in Cartesian space. The L_2 error norm is defined in the standard way

$$\|e\|_{L_2} = \sqrt{\frac{\int_\Omega [\varphi(x,y,z,t) - \varphi_{\text{exact}}(x,y,z,t)]^2 \, d\Omega}{\int_\Omega [\varphi_{\text{exact}}(x,y,z,t)]^2 \, d\Omega}},$$

where Ω represents the surface of the sphere. In addition to the L_2 norm, we also use two more measures, namely, the first and second moments of the conservation variable which are defined as

$$M_1 = \frac{\int_\Omega \varphi(x,y,z,t) \, d\Omega}{\int_\Omega \varphi_{\text{exact}}(x,y,z,t) \, d\Omega}$$

and

$$M_2 = \frac{\int_\Omega \varphi(x,y,z,t)^2 \, d\Omega}{\int_\Omega \varphi_{\text{exact}}(x,y,z,t)^2 \, d\Omega}.$$

These values measure the conservation properties and dispersion-diffusion of the numerical methods, respectively. In the following sections, the results for the spherical geodesic and advancing front grids are presented.

6.1. *Spherical Geodesic Grids*

Tables II and III show the results obtained using the Euler–Galerkin and Lagrange–Galerkin methods on the spherical geodesic grid. The tables show accuracy and efficiency measures for different values of α. For the Eulerian method, the time step must be restricted such that the Courant number σ is less than one in order for the scheme to remain stable. On the other hand, the Lagrangian method has no such stability limitation, assuming there are no source terms, and so we can theoretically increase the Courant number without limit; in this study we only use Courant numbers in the neighborhood of two. A few caveats are in order concerning the stability and accuracy of Lagrange–Galerkin methods: while for pure advection there are no stability limitations on the time step, excessively large time steps will introduce errors into the trajectory computations, thereby diminishing the overall accuracy of the method. In addition, for equations containing source terms we are restricted by the time step due to ODE stability conditions which, while less stringent than PDE stability conditions, nonetheless must be obeyed. The results illustrated are obtained on the spherical geodesic grid with three refinement loops ($\mathbf{n} = 3$). The corresponding number of grid points, triangular elements, edges, and tree elements are given in Table I.

Tables II and III demonstrate two important points: that the generalized natural coordinates provide good solutions for both the Euler–Galerkin and weak Lagrange–Galerkin formulations and that the Lagrange–Galerkin method yields a solution that is one order of magnitude more accurate than its Eulerian counterpart. We can see from these tables that it hardly matters which axis we use as the center of our rotation because the end result is the same, as it should be. Not only is the Lagrange–Galerkin solution far more accurate but it achieves this higher level of accuracy without sacrificing efficiency.

By comparing the Lagrange–Galerkin method with $\sigma = 1.13$ to the Euler–Galerkin method with $\sigma = 0.56$ we see that the computing times are a bit higher for the former. However, when we increase the Courant number for the Lagrange–Galerkin method to $\sigma = 2.27$ we observe two things: the accuracy of the Lagrange–Galerkin method has increased and the computing time has decreased. In fact, the computing time is now less than the time required for the Euler–Galerkin method. The efficiency of the Lagrange–Galerkin method is achieved because the inherent tree data structure of the geodesic grid has been exploited. Without such a data structure, this algorithm would be prohibitively expensive. In addition to being highly accurate and efficient, the weak Lagrange–Galerkin method is shown here to be conservative which is an important improvement over the direct Lagrange–Galerkin method.

Figures 11 and 12 show the grid and contour plots after five revolutions from the viewpoint $(0, -1, 0)$ which is the location $(3\pi/2, 0)$ in spherical coordinates. In order to better understand the results we have taken slices of the contour plot along the latitudinal (keeping the longitude constant at $\lambda = 3\pi/2$) and longitudinal directions (keeping the latitude constant at $\theta = 0$). These curves pass through the center of the cosine hill. The results at these slices are given in Figs. 13 and 14.

TABLE II

The Results for the Spherical Geodesic Grid with **npoin** = 642 and Different α for the Euler–Galerkin Method for Up to Five Revolutions

Method	σ	α	Revs	L_2 Norm	φ_{max}	φ_{min}	M_1	M_2	Time (s)
Euler–Galerkin	0.56	0	1	0.0842	99.01	−4.87	1.0000	0.9998	86.0
			2	0.1506	99.40	−8.89	1.0000	0.9997	109.0
			3	0.2125	98.78	−10.61	1.0000	0.9998	124.1
			4	0.2717	97.77	−14.40	1.0000	0.9998	142.9
			5	0.3279	95.47	−18.20	1.0000	0.9998	162.1
Euler–Galerkin	0.56	$\pi/2$	1	0.0842	99.01	−4.87	1.0000	0.9998	86.0
			2	0.1506	99.40	−8.89	1.0000	0.9997	109.0
			3	0.2125	98.78	−10.61	1.0000	0.9998	124.1
			4	0.2717	97.77	−14.40	1.0000	0.9998	142.9
			5	0.3279	95.47	−18.20	1.0000	0.9998	162.1

Although the grids used for both methods are identical, the Euler–Galerkin method shows asymmetries in the contour plot, whereas the Lagrange–Galerkin method produces a symmetric solution (see Figs. 11 and 12). These differences are even more pronounced when viewed from the longitudinal and latitudinal slices. Figures 13 and 14 show that while the Euler–Galerkin solution has lost its symmetry, the Lagrange–Galerkin solution not only retains its symmetry but is also free from the oscillations that commonly plague higher order Eulerian methods. In order to suppress these oscillations, higher order Eulerian methods must use *ad hoc* methods such as TVD, MUSCL, or ENO schemes in conjunction with flux-limiting [5, 6]. These schemes automatically switch from higher order to first

TABLE III

The Results for the Spherical Geodesic Grid with **npoin** = 642 and Different α for the Lagrange–Galerkin Method for Up to Five Revolutions

Method	σ	α	Revs	L_2 Norm	φ_{max}	φ_{min}	M_1	M_2	Time (s)
Weak Lagrange–Galerkin	1.13	0	1	0.0078	99.81	−0.52	0.9994	1.0019	93.5
			2	0.0123	99.66	−0.72	0.9989	1.0039	119.4
			3	0.0164	99.53	−0.86	0.9984	1.0060	144.9
			4	0.0203	99.42	−0.98	0.9980	1.0082	170.6
			5	0.0240	99.33	−1.14	0.9976	1.0104	196.3
Weak Lagrange–Galerkin	1.13	$\pi/2$	1	0.0078	99.81	−0.52	0.9994	1.0019	93.5
			2	0.0123	99.66	−0.72	0.9989	1.0039	119.4
			3	0.0164	99.53	−0.86	0.9984	1.0060	144.9
			4	0.0203	99.42	−0.98	0.9980	1.0082	170.6
			5	0.0240	99.33	−1.14	0.9976	1.0104	196.3
Weak Lagrange–Galerkin	2.27	0	1	0.0070	99.84	−0.38	0.9997	1.0002	80.8
			2	0.0103	99.70	−0.46	0.9995	1.0006	93.6
			3	0.0130	99.55	−0.55	0.9993	1.0011	106.8
			4	0.0155	99.40	−0.69	0.9991	1.0016	119.5
			5	0.0179	99.25	−0.82	0.9989	1.0021	132.5
Weak Lagrange–Galerkin	2.27	$\pi/2$	1	0.0070	99.84	−0.38	0.9997	1.0002	80.8
			2	0.0103	99.70	−0.46	0.9995	1.0006	93.6
			3	0.0130	99.55	−0.55	0.9993	1.0011	106.8
			4	0.0155	99.40	−0.69	0.9991	1.0016	119.5
			5	0.0179	99.25	−0.82	0.9989	1.0021	132.5

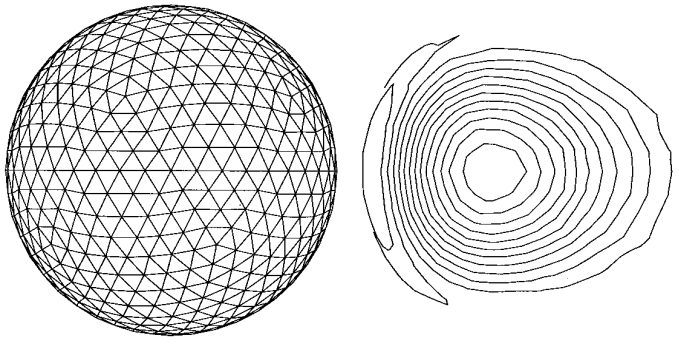

FIG. 11. The grid and contours for the Euler–Galerkin solution after five revolutions using the spherical geodesic grid. The Courant number is $\sigma = 0.56$, **npoin** = 642, and $\alpha = 0$.

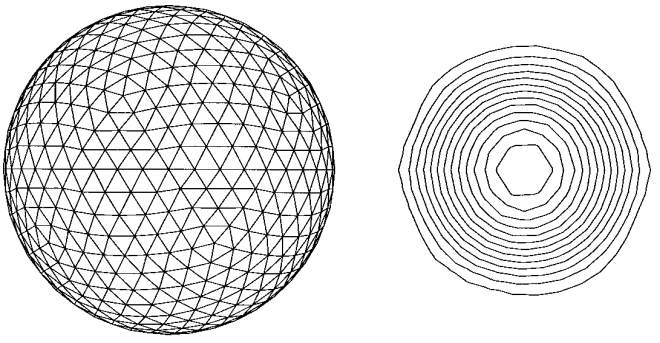

FIG. 12. The grid and contours for the Lagrange–Galerkin solution after five revolutions using the spherical geodesic grid. The Courant number is $\sigma = 1.13$, **npoin** = 642, and $\alpha = 0$.

order near strong gradients in order to avoid dispersion errors and remain monotonic. While neither the Euler–Galerkin nor the Lagrange–Galerkin methods are naturally monotonic, the Lagrange–Galerkin method exhibits far less dispersion than its Eulerian counterpart. In fact, this dispersion is almost negligible. For applications where preserving monotonicity is imperative, such as in the conservation of mass equation for Navier-Stokes and the precipitation in meteorological applications, the Lagrange–Galerkin method can be made monotonic by using principles similar to those used in TVD and FCT schemes [16].

Table IV shows the solutions for the spherical geodesic grid with one, two, and three refinement loops. The presentation of these results must be prefaced by noting that the matrix solver used to obtain these results does in no way represent the most efficient solver. In fact, the major portion of the **cpu** times reported are spent on the matrix decomposition of this particular solver. Although we have developed an incomplete Choleski conjugate gradient method (ICCG) with zero fill-in for the Lagrange–Galerkin method, we have used an LU decomposition for this study in order to use the same solver for both the Euler–Galerkin and Lagrange–Galerkin method. This allows for fair timing comparisons between the two methods. (The ICCG method cannot be used with the Euler–Galerkin method because the advection terms prevent the coefficient matrix from being symmetric positive-definite.)

The results in Table IV show that the Lagrange–Galerkin method is competitive in terms of efficiency with the Euler–Galerkin method. However, the differences in accuracy are astounding. As the grid becomes finer, the Lagrange–Galerkin method achieves even higher levels of accuracy than the Euler–Galerkin method. For **npoin** = 162, the Lagrange–Galerkin method is 10 times more accurate, but for the fine resolution grid with **npoin** = 2562,

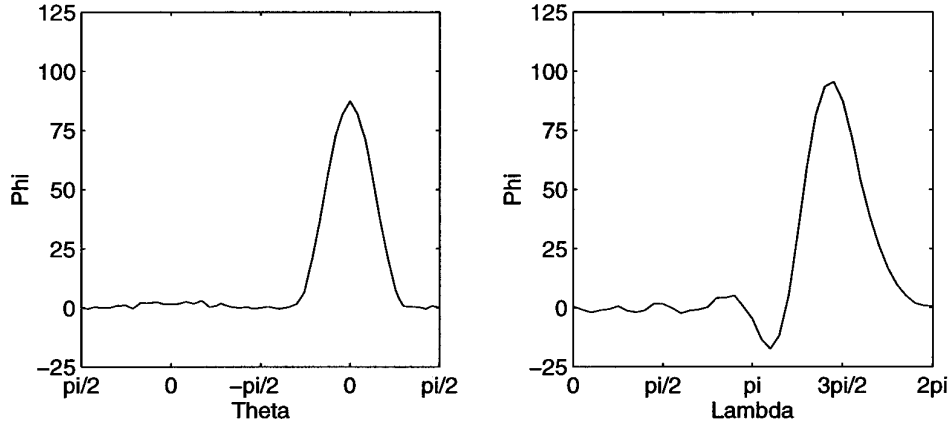

FIG. 13. Views of the cosine hill for the Euler–Galerkin solution after five revolutions using the spherical geodesic grid. The plot on the left shows the cosine hill slice taken at $\lambda = 3\pi/2$. The plot on the right shows the cosine hill slice taken at $\theta = 0$. The Courant number is $\sigma = 0.56$, **npoin** = 642, and $\alpha = 0$.

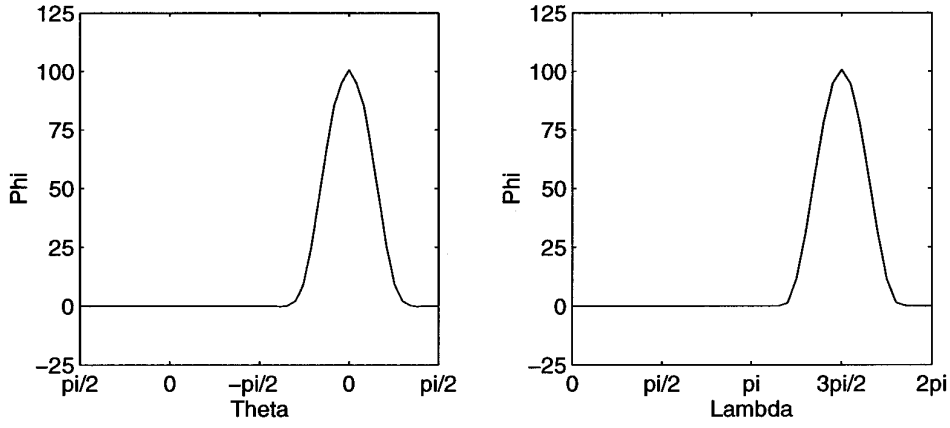

FIG. 14. Views of the cosine hill for the Lagrange–Galerkin solution after five revolutions using the spherical geodesic grid. The plot on the left shows the cosine hill slice taken at $\lambda = 3\pi/2$. The plot on the right shows the cosine hill slice taken at $\theta = 0$. The Courant number is $\sigma = 1.13$, **npoin** = 642, and $\alpha = 0$.

the Lagrange–Galerkin method is almost 20 times more accurate. In addition, the results show that the weak Lagrange–Galerkin method is also conservative. By using the inherent tree data structure and optimal matrix solvers, the Lagrange–Galerkin method can be used for solving practical problems on spherical geodesic grids accurately, conservatively, quasi-monotonically, and efficiently.

6.2. *Advancing Front Unstructured Grids*

Table V shows the results obtained using the Euler–Galerkin and Lagrange–Galerkin methods on the advancing front unstructured grids. Since advancing front grid generators cannot be constrained to produce a given number of grid points, a grid was selected that most closely resembled the spherical geodesic grid in number of grid points (**npoin** = 645 for the advancing front grid and **npoin** = 642 for the geodesic grid). We illustrate the results on this grid in order to show that the Lagrange–Galerkin method can be used on randomly generated unstructured grids on the sphere. This is important because it suggests that this method can be used in conjunction with adaptive grids. We have chosen to work primarily with the spherical geodesic grid because it offers inherent fast searching tools which the advancing front method does not. However, it is possible to construct a quadtree-like data structure for searching, but it is not inherent to the grid and must be generated independently.

For brevity, we only illustrate results for $\alpha = 0$. This table clearly shows that the Euler–Galerkin and Lagrange–Galerkin methods work well even for such random and disproportioned grids as those produced by the advancing front method. The aspect ratio of maximum to minimum lengths for the advancing front grid is 3.1, while for the geodesic grid it is 1.4. Large aspect ratios could conceivably cause problems for Eulerian methods because the Courant number is determined by the smallest edge. This edge can be considerably smaller than the majority of the edges in the grid, thereby restricting the time step

TABLE IV

The Results for the Spherical Geodesic Grid for Various **npoin** for the Euler–Galerkin and Lagrange–Galerkin Methods for Five Revolutions

Method	σ	**npoin**	L_2 Norm	$\varphi_{\max}$	$\varphi_{\min}$	M_1	M_2	Time (s)
Euler–Galerkin	0.56	162	0.7559	68.04	−32.18	1.0000	0.9982	3.2
		642	0.3279	95.47	−18.20	1.0000	0.9998	162.1
		2562	0.0949	98.88	−6.22	1.0000	0.9999	10651.3
Weak Lagrange–Galerkin	2.27	162	0.0690	98.02	−1.89	0.9950	1.0025	16.9
		642	0.0179	99.25	−0.82	0.9989	1.0021	132.5
		2562	0.0052	99.88	−0.36	0.9995	1.0014	8282.6

TABLE V

The Results for the Advancing Front Unstructured Grid with **npoin** = 645 for the Euler–Galerkin and Lagrange–Galerkin Methods for Up to Five Revolutions

Method	σ	α	Revs	L_2 Norm	$\varphi_{\max}$	$\varphi_{\min}$	M_1	M_2	Time (s)
Euler–Galerkin	0.74	0	1	0.0978	97.18	−5.27	1.0000	0.9995	87.1
			2	0.1643	95.22	−11.55	0.9999	0.9992	106.3
			3	0.2261	98.45	−12.15	0.9999	0.9989	127.9
			4	0.2850	99.70	−16.61	0.9999	0.9986	146.7
			5	0.3420	99.68	−19.37	0.9998	0.9983	166.2
Weak Lagrange–Galerkin	1.48	0	1	0.0175	98.16	−0.73	1.0002	0.9991	308.8
			2	0.0235	97.19	−1.02	1.0005	0.9987	547.4
			3	0.0283	96.47	−1.24	1.0008	0.9984	783.4
			4	0.0327	95.86	−1.40	1.0012	0.9982	1012.3
			5	0.0368	95.31	−1.54	1.0016	0.9981	1248.0

to prohibitively small values. This disadvantage becomes more pronounced with adaptive grids.

Figures 15 and 16 show the grid and contour plots for the two methods after five revolutions. Once again, the Euler–Galerkin method exhibits asymmetries in the solution produced by the dispersive nature of the method, whereas the Lagrange–Galerkin method yields a symmetric result.

Figures 17 and 18 show the results for longitudinal and latitudinal slices after five revolutions. The Euler–Galerkin solution suffers severe dispersion errors while the Lagrange–Galerkin method does not. This result confirms that the weak Lagrange–Galerkin method can be used successfully to obtain smooth (nondispersive) yet highly accurate solutions on the sphere. Furthermore, these high order accuracy solutions are independent of the grid types; they can be obtained on spherical geodesic or advancing front grids.

At this point, however, the advancing front approach is not yet practical because a data structure has not been developed for the searching operations. This is evident from the large **cpu** times reported in Table V for the Lagrange–Galerkin method. Once this data structure is constructed, the Lagrange–Galerkin method can be used in conjunction with adaptive unstructured grids on the sphere. This promises to be a potent combination as the adaptive grids increase the accuracy further, while the leniency of the CFL restriction for the Lagrange–Galerkin method allows a large fixed time step to be used throughout the grid adaptation. This differs from using adaptive grids with Eulerian methods because with these methods once the minimum grid size is decreased, the time step must also be decreased proportionally in order to satisfy the CFL condition. Since Lagrangian methods do not have this restriction they can be used quite efficiently with adaptive grid strategies.

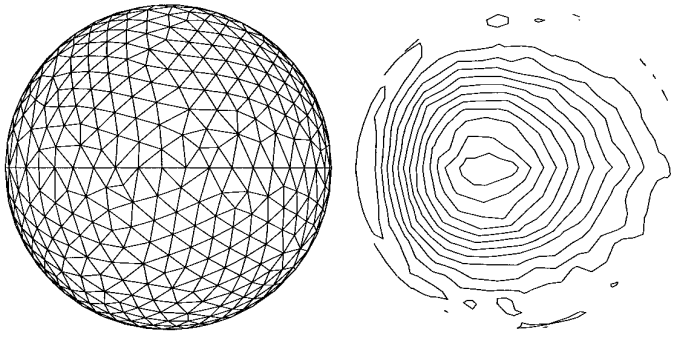

FIG. 15. The grid and contours for the Euler–Galerkin solution after five revolutions using the advancing front grid. The Courant number is $\sigma = 0.74$, **npoin** = 645, and $\alpha = 0$.

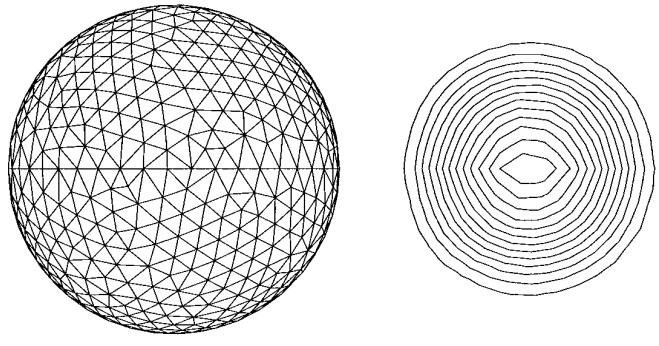

FIG. 16. The grid and contours for the Lagrange–Galerkin solution after five revolutions using the advancing front grid. The Courant number is $\sigma = 1.48$, **npoin** = 645, and $\alpha = 0$.

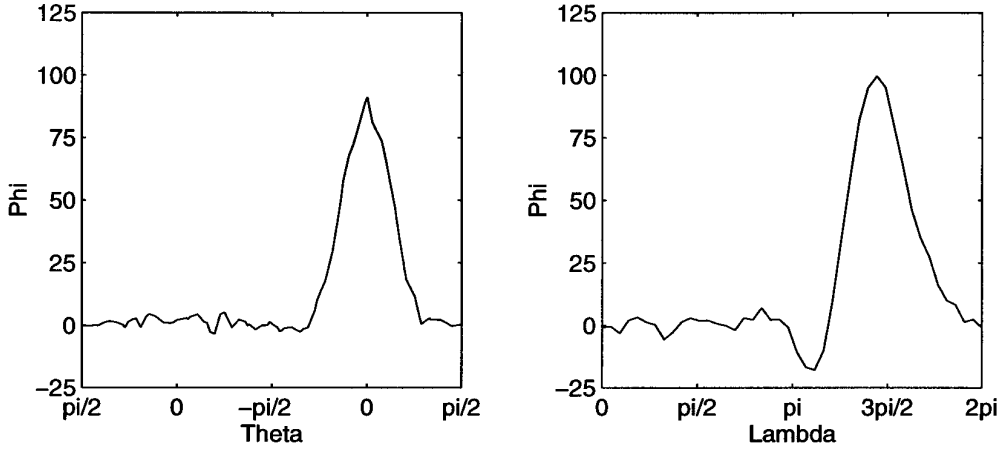

FIG. 17. Views of the cosine hill for the Euler–Galerkin solution after five revolutions using the advancing front grid. The plot on the left shows the cosine hill slice taken at $\lambda = 3\pi/2$. The plot on the right shows the cosine hill slice taken at $\theta = 0$. The Courant number is $\sigma = 0.74$, **npoin** = 645, and $\alpha = 0$.

7. CONCLUSIONS

Generalized natural Cartesian coordinates for triangular elements in three-dimensional space are presented. When these natural coordinates are used as the basis functions, exact integrals for all of the finite element terms can be obtained. This has significant implications not just for Eulerian methods but for Lagrangian methods as well, especially if exactly integrating Lagrange–Galerkin methods are to be explored. This paper describes how these ideas can be used to apply the exactly integrating Lagrange–Galerkin method on the sphere. The spherical geodesic grids are beginning to gain popularity and the tree data structure developed in this paper permits the extension of these grids from Eulerian numerical methods to Lagrange–Galerkin methods. Without such a data structure, the search operations required by the Lagrange–Galerkin method would be prohibitively expensive. The numerical experiments show that the Lagrange–Galerkin method can be used not just with the quasi-structured grids resulting from the spherical geodesic approach, but also with random and disproportionate unstructured grids such as those created by the advancing front method. This last finding is important because it suggests that the Lagrange–Galerkin method can be used in conjunction with adaptive grids; this combination should provide an even more accurate solution. Efficient advancing front grids on the sphere need to be explored. These grids can be used not just for adaptive grid refinement but in conjunction with the exactly integrating Lagrange–Galerkin method as well, since this method requires a grid generation step in order to integrate the elemental equations exactly.

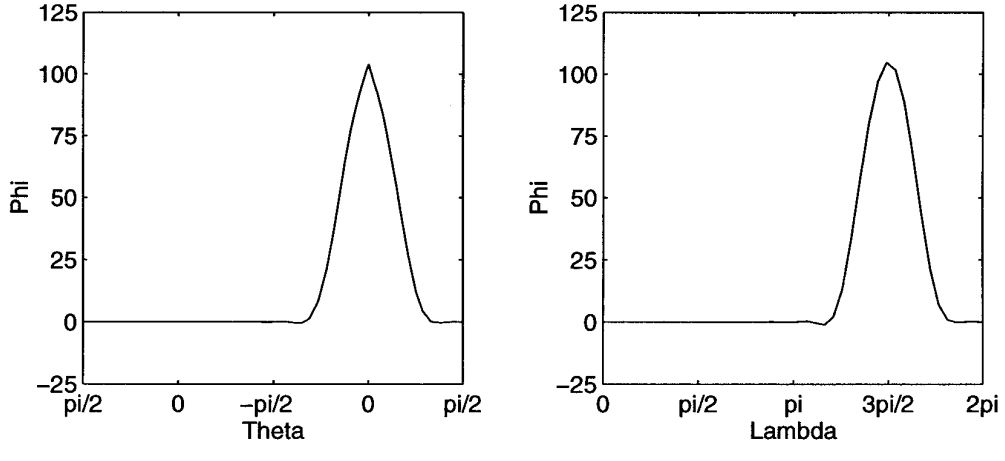

FIG. 18. Views of the cosine hill for the Lagrange–Galerkin solution after five revolutions using the advancing front grid. The plot on the left shows the cosine hill slice taken at $\lambda = 3\pi/2$. The plot on the right shows the cosine hill slice taken at $\theta = 0$. The Courant number is $\sigma = 1.48$, **npoin** = 645, and $\alpha = 0$.

ACKNOWLEDGMENTS

The support of the sponsor, the Office of Naval Research through program PE-0602435N, is gratefully acknowledged. I also thank Anabela Oliveira of the Oregon Graduate Institute for answering my questions about the Lagrange–Galerkin method in one dimension and Ross Heikes of Colorado State University for giving me the coordinates of the icosahedron. Special thanks go to Andrew Priestley of Geoquest Reservoir Technologies (formerly of the University of Reading) for discussing with me the intricacies of the Lagrange–Galerkin method.

REFERENCES

1. J. M. Augenbaum and C. S. Peskin, On the construction of the Voronoi mesh on a sphere, *J. Comput. Phys.* **14,** 177 (1985).
2. J. P. Benque and J. Ronat, Quelques difficultes des modules numeriques en hydralique, in *Computing Methods in Applied Sciences and Engineering,* Vol. V, edited by R. Glowinski and J. L. Lions (North-Holland, New York, 1982), p. 471.
3. J. P. Benque, G. Labadie, and J. Ronat, A new finite element method for Navier–Stokes equations coupled with a temperature equation, in *Proceedings of the Fourth International Symposium on Finite Element Methods in Flow Problems,* edited by T. Kawai (North-Holland, New York, 1982), p. 295.
4. T. J. Chung, *Finite Element Analysis in Fluid Dynamics* (McGraw–Hill, San Francisco, 1978), p. 77.
5. F. X. Giraldo, *A Space Marching Adaptive Remeshing Technique Applied to the 3D Euler Equations for Supersonic Flow,* Ph.D. dissertation, University of Virginia, 1995.
6. F. X. Giraldo, A finite volume high resolution 2D Euler solver with adaptive grid generation on high performance computers, in *Proceedings, Ninth International Conference on Finite Elements in Fluids* **2,** 1995, p. 1031.
7. F. X. Giraldo and B. Neta, A comparison of a family of Eulerian and semi-Lagrangian finite element methods for the advection–diffusion equation, in *Coastal Engineering 97, La Coruña, Spain, 1997.*
8. F. X. Giraldo, Efficiency and accuracy of Lagrange–Galerkin methods on unstructured adaptive grids, *Math. Modelling Sci. Comput.* **8** (1997).
9. R. Heikes and D. A. Randall, Numerical integration of the shallow water equations on a twisted icosahedral grid. Part I. Basic design and results of tests, *Mon. Weather Rev.* **123,** 1862 (1995).
10. R. Heikes, *personal communication,* 1996.
11. A. McDonald and J. R. Bates, Semi-Lagrangian integration of a gridpoint shallow water model on the sphere, *Mon. Weather Rev.* **117,** 130 (1988).
12. K. W. Morton, A. Priestley, and E. E. Süli, Stability of the Lagrange–Galerkin method with non-exact integration, *Math. Modelling Numer. Anal.* **22,** 625 (1988).
13. A. Oliveira, *A Comparison of Eulerian–Lagrangian Methods for the Solution of the Transport Equation,* Master's thesis, Oregon Graduate Institute of Science and Technology, 1994.
14. A. Oliveira and A. M. Baptista, A comparison of integration and interpolation Eulerian–Lagrangian methods, *Int. J. Numer. Methods Fluids* **21,** 183 (1995).
15. A. Priestley, The Taylor–Galerkin method for the shallow-water equations on the sphere, *Mon. Weather Rev.* **120,** 3003 (1992).
16. A. Priestley, A quasi-conservative version of the semi-Lagrangian advection scheme, *Mon. Weather Rev.* **121,** 621 (1993).
17. A. Priestley, Exact projections and the Lagrange–Galerkin method: A realistic alternative to quadrature, *J. Comput. Phys.* **112,** 316 (1994).
18. A. Priestley, All you ever wanted to know about exact projection and the Lagrange–Galerkin method but were afraid to ask, 1995, unpublished.
19. H. Ritchie, Semi-Lagrangian advection on a Gaussian grid, *Mon. Weather Rev.* **115,** 608 (1986).
20. D. L. Williamson, Integration of the barotropic vorticity equation on a spherical geodesic grid, *Tellus* **20,** 642 (1968).
21. D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber, A standard test set for numerical approximations to the shallow water equations in spherical geometry, *J. Comput. Phys.* **102,** 211 (1992).