Journal of Computational Physics **160,** 336–368 (2000)
doi:10.1006/jcph.2000.6469, available online at http://www.idealibrary.com on IDEAL®

# Lagrange–Galerkin Methods on Spherical Geodesic Grids: The Shallow Water Equations

Francis X. Giraldo

*Naval Research Laboratory, Monterey, California 93943*
E-mail: giraldo@nrlmry.navy.mil

Received April 30, 1999; revised February 14, 2000

The weak Lagrange–Galerkin finite element method for the 2D shallow water equations on the sphere is presented. This method offers stable and accurate solutions because the equations are integrated along the characteristics. The equations are written in 3D Cartesian conservation form and the domains are discretized using linear triangular elements. The use of linear triangular elements permits the construction of accurate (by virtue of the second-order spatial and temporal accuracies of the scheme) and efficient (by virtue of the less stringent CFL condition of Lagrangian methods) schemes on unstructured domains. Using linear triangles in 3D Cartesian space allows for the explicit construction of area coordinate basis functions thereby simplifying the calculation of the finite element integrals. The triangular grids are constructed by a generalization of the icosahedral grids that have been typically used in recent papers. An efficient searching strategy for the departure points is also presented for these generalized icosahedral grids which involves very few floating point operations. In addition a high-order scheme for computing the characteristic curves in 3D Cartesian space is presented: a general family of Runge–Kutta schemes. Results for six test cases are reported in order to confirm the accuracy of the scheme.

*Key Words:* finite element method; icosahedral grid; Lagrange–Galerkin; Runge–Kutta; shallow water equations; spherical geometry; unstructured grid.

## 1. INTRODUCTION

Due to the recent paradigm shift in large-scale computing from vector machines (having few but powerful processors) to distributed memory machines (having a multitude of less powerful processors) researchers have begun to explore methods other than the spectral method for solving the shallow water equations on the sphere. Besides not parallelizing well, the spectral method also suffers from the restriction that the grid be a longitude–latitude



0021-9991/00

grid which packs too many unnecessary points at the poles. By exploring other classes of methods, researchers are also free to choose other types of grids. For example, in [11] cubic gnomonic grids are used in conjunction with the spectral element method. In [6] icosahedral grids are employed with a finite differencing spatial discretization. In [1] a spiral triangular grid similar to the icosahedral grid is used with the weak Lagrange–Galerkin method. (For a complete review of grids for tiling the sphere see [10].) However, many of these new approaches continue to follow in the footsteps of the spectral method by writing the equations in spherical coordinates; the only exception is the finite difference method presented in [6], which solves the equations in a coordinate invariant form.

Although spherical coordinates seem to be the natural choice, they present many problems and associated computational costs. As we have mentioned previously, spectral methods on the sphere require the use of latitude–longitude Gaussian grids, which introduces too many redundant points around the poles; this situation is exacerbated as the resolution is increased. In addition, spherical coordinates result in singularities at the poles for all discretization methods except the spectral method (which uses Gaussian quadrature rules that do not contain the poles as quadrature points). The problem of singularities at the poles caused by spherical coordinates can be circumvented by applying rotation transformations (as is done in [11] from spherical to gnomonic space) or by using Cartesian coordinates in which case the equations no longer contain singularities at the poles without the need to introduce any type of rotation transformation.

In [2, 3, 9] the equations are solved in Cartesian rather than spherical coordinates. In [2] the Lagrange multiplier approach for transforming the shallow water equations from spherical to Cartesian coordinates is introduced. This idea is then used to construct a semi-Lagrangian shallow water model but on longitude–latitude grids, an unnecessary remnant of spectral methods. In [9] the Taylor–Galerkin method in 3D Cartesian space using icosahedral grids is presented. This work is the closest to that of our previous work presented in [3] in that the approach uses neither spherical coordinates to write the governing equations nor spectral methods to solve them. However, in [9], the work stops short of that in [3] by then mapping the 3D linear triangles to a local 2D space, an unnecessary step since all the computations can be carried out in 3D space thereby avoiding any type of mapping to a local space. In [3] we developed a weak Lagrange–Galerkin method in 3D Cartesian space using triangular linear basis functions on icosahedral grids. In this work, all of the computations take place in 3D Cartesian space, thus avoiding any local mappings or rotation transformations.

Using Cartesian coordinates avoids another obstacle encountered with spherical coordinates, that is, that explicit basis functions cannot be obtained in this space. By using 3D Cartesian coordinates, explicit basis functions can be constructed, which means that these functions (which are the natural or area coordinates) can be used for obtaining all of the finite element integrals in closed form and interpolating the values of the departure points required by all Lagrangian schemes. This is the approach taken in our current paper.

The Lagrange–Galerkin method has been shown to be very attractive for solving hyperbolic equations on the sphere [1, 3]. This method combines the *method of characteristics* with the finite element method. Because this method represents an integration of the equations along the direction in which information is propagating, namely the characteristics, the method is not only very accurate but also quite stable even for very large Courant numbers. The current paper is an extension of the ideas presented in [3] but for the full nonlinear shallow water equations. The Lagrange–Galerkin method is implemented on a new class of grids which contains the typical icosahedral grids. These new grids offer much

more flexibility in the construction of grids over the typical icosahedral grids. In addition, a searching strategy for locating departure points is introduced for these new grids. This searching strategy is much faster than the icosahedral quadtree-like data structure originally introduced in [3]. Finally, the high-order general family of Runge–Kutta schemes for computing the departure points on the plane introduced in [5] is extended to the sphere. This method is more accurate than the midpoint rule typically used in Lagrangian schemes (see [5] for a comparison of this method with the composite midpoint rule).

## 2. SHALLOW WATER EQUATIONS

The 2D spherical shallow water equations in conservation form are

$$\frac{\partial}{\partial t}\begin{bmatrix}\varphi\\ \varphi u\\ \varphi v\\ \varphi w\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\varphi u\\ \varphi u^2+\frac{1}{2}\varphi^2\\ \varphi uv\\ \varphi uw\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\varphi v\\ \varphi uv\\ \varphi v^2+\frac{1}{2}\varphi^2\\ \varphi vw\end{bmatrix}+\frac{\partial}{\partial z}\begin{bmatrix}\varphi w\\ \varphi uw\\ \varphi vw\\ \varphi w^2+\frac{1}{2}\varphi^2\end{bmatrix}$$
$$=\begin{bmatrix}0\\ f\left(\frac{z}{a}\varphi v-\frac{y}{a}\varphi w\right)+\mu x\\ f\left(\frac{x}{a}\varphi w-\frac{z}{a}\varphi u\right)+\mu y\\ f\left(\frac{y}{a}\varphi u-\frac{x}{a}\varphi v\right)+\mu z\end{bmatrix}$$

but note that if we move the pressure terms ($\frac{1}{2}\varphi^2$) to the right-hand side, we get

$$\frac{\partial}{\partial t}\begin{bmatrix}\varphi\\ \varphi u\\ \varphi v\\ \varphi w\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\varphi u\\ \varphi u^2\\ \varphi uv\\ \varphi uw\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\varphi v\\ \varphi uv\\ \varphi v^2\\ \varphi vw\end{bmatrix}+\frac{\partial}{\partial z}\begin{bmatrix}\varphi w\\ \varphi uw\\ \varphi vw\\ \varphi w^2\end{bmatrix}$$
$$=\begin{bmatrix}0\\ -\varphi\frac{\partial\varphi}{\partial x}+f\left(\frac{z}{a}\varphi v-\frac{y}{a}\varphi w\right)+\mu x\\ -\varphi\frac{\partial\varphi}{\partial y}+f\left(\frac{x}{a}\varphi w-\frac{z}{a}\varphi u\right)+\mu y\\ -\varphi\frac{\partial\varphi}{\partial z}+f\left(\frac{y}{a}\varphi u-\frac{x}{a}\varphi v\right)+\mu z\end{bmatrix}. \tag{1}$$

This system can now be written more compactly as

$$\frac{\partial\boldsymbol{\varphi}}{\partial t}+\nabla\cdot(\boldsymbol{\varphi}\mathbf{u})=\mathbf{S}(\boldsymbol{\varphi}) \tag{2}$$

where

$$\boldsymbol{\varphi}=\begin{bmatrix}\varphi\\ \varphi u\\ \varphi v\\ \varphi w\end{bmatrix},\qquad \mathbf{u}=\begin{bmatrix}u\\ v\\ w\end{bmatrix},\qquad \mathbf{S}(\boldsymbol{\varphi})=\mathbf{F}_{\mathrm{P}}+\mathbf{F}_{\mathrm{R}}+\mathbf{F}_{\mathrm{C}}, \tag{3}$$

$$\mathbf{F}_{\mathrm{P}}=-\varphi\nabla\varphi,\qquad \mathbf{F}_{\mathrm{R}}=-f(\mathbf{r}\times\varphi\mathbf{u}),\qquad \mathbf{F}_{\mathrm{C}}=\mu\mathbf{r},$$

and

$$f = \frac{2\Omega z}{a},$$

where $\Omega$ and $a$ are the rotation of the earth and its radius, respectively. In other words, $\mathbf{F}_P$ is the force due to pressure, $\mathbf{F}_R$ is the force due to the rotation of the earth (Coriolis), $\mathbf{F}_C$ is the force required to constrain the fluid particles to remain on the surface of the sphere, and $\mu$ is the Lagrange multiplier used to satisfy this condition. This term is best obtained in discrete form. If it is obtained from the continuous equations, then for the discrete equations the particles may no longer be guaranteed to remain on the surface of the sphere.

The equations are solved for the four conservation variables $\varphi$, $\varphi u$, $\varphi v$, and $\varphi w$. Clearly, we could also include other forcing functions and in this case they would be included within $\mathbf{S}(\boldsymbol{\varphi})$. Before proceeding to the discretization of the weak Lagrange–Galerkin method, let us first look at the discretization obtained by an Eulerian method. This will serve both for contrast and also because we require one Eulerian time step as the weak Lagrange–Galerkin method requires two known time levels ($t = 0$ and $t = \Delta t$) prior to its usage.

## 3. EULER–GALERKIN METHOD

Beginning with (2) we can define an Eulerian finite element method by taking the weak form

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

where $\psi$ denotes the basis functions. Integrating by parts (Green's theorem) such as

$$\nabla \cdot (\psi \boldsymbol{\varphi}\mathbf{u}) = \psi \nabla \cdot (\boldsymbol{\varphi}\mathbf{u}) + (\boldsymbol{\varphi}\mathbf{u}) \cdot \nabla \psi$$

yields

$$\int_\Omega \psi \frac{\partial \boldsymbol{\varphi}}{\partial t}\, d\Omega + \int_\Gamma \mathbf{n} \cdot \mathbf{u} \psi \boldsymbol{\varphi}\, d\Gamma - \int_\Omega \nabla \psi \cdot (\boldsymbol{\varphi}\,\mathbf{u})\, d\Omega - \int_\Omega \psi \mathbf{S}(\boldsymbol{\varphi})\, d\Omega = 0.$$

In the case of flow on a sphere, the second integral vanishes because the basis functions are chosen to be continuous across element interfaces and there are no physical boundaries; in other words,

$$\int_\Gamma \mathbf{n} \cdot \mathbf{u} \psi \boldsymbol{\varphi}\, d\Gamma = 0,$$

and we are left with

$$\int_\Omega \psi \frac{\partial \boldsymbol{\varphi}}{\partial t}\, d\Omega = \int_\Omega \nabla \psi \cdot (\boldsymbol{\varphi}\mathbf{u})\, d\Omega + \int_\Omega \psi \mathbf{S}(\boldsymbol{\varphi})\, d\Omega.$$

The resulting system of ordinary differential equations can now be written as

$$\frac{\partial \boldsymbol{\varphi}}{\partial t} = H(\boldsymbol{\varphi}),$$

which after integrating in time by a general family of Runge–Kutta schemes yields

$$\varphi^{k+1} = \varphi^n + \Delta t \beta H(\varphi)^{k-1},$$

where

$$\beta = \frac{1}{M-k+1}, \qquad k = 1, \ldots, M, \qquad \text{and} \qquad H(\varphi)^0 = H(\varphi)^n.$$

This scheme is required for the first time step because, as we will see, the weak Lagrange–Galerkin method, although a two-time level scheme, requires two previously known time levels in order to calculate the particle trajectories.

The Lagrange multiplier used in the Eulerian and Lagrangian schemes presented in this paper is obtained by the following method: let $\varphi \mathbf{u}^*$ be the solution of the momentum equations by either the Runge–Kutta or Lagrange–Galerkin scheme. Since we require that the velocity field and the outward pointing normal to the sphere be orthogonal,

$$\varphi \mathbf{u} \cdot \mathbf{r} = 0.$$

Therefore, the constrained momentum equations satisfy the relation

$$\varphi \mathbf{u} = \varphi \mathbf{u}^* + \mu \mathbf{r}$$

and taking the scalar product of $\mathbf{r}$ yields the Lagrange multiplier

$$\mu = -\frac{\varphi \mathbf{u}^* \cdot \mathbf{r}}{a^2},$$

where $a$ is the radius of the earth. This procedure is applied at the end of each time step in order to ensure that the velocity field remains orthogonal to the radial vector thereby constraining the fluid particles to remain on the surface of the sphere throughout the integration. It is unclear *a priori* what effects this procedure has on the formal order of accuracy of the scheme. A stability analysis of the equations with this term is in order; this is the topic of a future research note.

## 4. WEAK LAGRANGE–GALERKIN METHOD

### 4.1. *Spatial Discretization*

The weak form of (2) is

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

and integrating by parts as such

$$\frac{\partial}{\partial t}(\psi \varphi) = \psi \frac{\partial \varphi}{\partial t} + \varphi \frac{\partial \psi}{\partial t} \qquad \text{and} \qquad \nabla \cdot (\psi \varphi \mathbf{u}) = \psi \nabla \cdot (\varphi \mathbf{u}) + (\varphi \mathbf{u}) \cdot \nabla \psi$$

we get

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

The first bracketed term of (4), using Reynold's transport theorem, can be written as

$$\frac{d}{dt}\int_\Omega (\psi\varphi)\,d\Omega = \int_\Omega \left[\frac{\partial}{\partial t}(\psi\varphi) + \nabla\cdot(\psi\varphi\mathbf{u})\right] d\Omega, \tag{5}$$

and the second bracketed term is actually the characteristic equation

$$\frac{d\psi}{dt} \equiv \frac{\partial\psi}{\partial t} + \mathbf{u}\cdot\nabla\psi = 0, \tag{6}$$

where

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

is used to predict the particle trajectories along the characteristics where the basis functions vanish. Substituting (5) and (6) into (4) yields the system

$$\frac{d}{dt}\int_\Omega (\psi\varphi)\,d\Omega = \int_\Omega \psi\mathbf{S}(\varphi)\,d\Omega. \tag{8}$$

The advection operator has disappeared from the equations; however, the correct particle trajectories are accounted for by the trajectory equation (7). In addition, consider that the divergence of the velocity has also disappeared or rather has been absorbed by the Reynold's transport theorem (see [5] for further details).

### 4.2. *Time Discretization*

Integrating (8) in time by the $\theta$ algorithm yields

$$\int_{\Omega^{n+1}} (\psi\varphi)\,d\Omega^{n+1} = \int_{\Omega^n} (\psi\varphi)\,d\Omega^n + \Delta t\left[\theta\int_{\Omega^{n+1}} \psi\mathbf{S}(\varphi)\,d\Omega^{n+1} + (1-\theta)\int_{\Omega^n} \psi\mathbf{S}(\varphi)\,d\Omega^n\right], \tag{9}$$

which represents a two-time-level scheme and gives the explicit rectangle rule, the trapezoid rule, and the implicit rectangle rule for $\theta = 0, \frac{1}{2}, 1$, respectively. In this paper $\theta = \frac{1}{2}$ is used throughout because it yields a second-order-accurate scheme and is unconditionally stable with respect to the advection operator. The other two schemes are both first order: $\theta = 0$ is not very stable while $\theta = 1$ is quite diffusive. Clearly, Eqs. (7) and (9) define a two-time-level scheme requiring the variables at times $n$ and $n+1$. However, in the following section it is shown that in order to solve Eq. (7) accurately requires the velocity field to be known at previous time levels. The accurate solution of the trajectory equation is perhaps the most important part of Lagrangian methods. In the next sections we extend our new method for calculating the particle trajectories described in [5] to the sphere.

*4.2.1. Midpoint rule.* In most Lagrangian methods particle trajectories are computed by the midpoint rule

$$\mathbf{x}(t^{n+1}) - \mathbf{x}(t^n) = \Delta t\, \mathbf{u}\big(\mathbf{x}\big(t^{n+1/2}\big), t^{n+1/2}\big),$$

which requires the following extrapolation of the velocity field,

$$\mathbf{u}\big(t^{n+1/2}\big) = \frac{3}{2}\mathbf{u}(t^n) - \frac{1}{2}\mathbf{u}(t^{n-1}), \tag{10}$$

where the term $\mathbf{u}(t^n)$ refers to the velocity values at time $n$ at each grid point whereas $\mathbf{u}(\mathbf{x}(t^n), t^n)$ refers to the velocity values at time $n$ but along the characteristics. Since we need to know $\mathbf{x}(t^{n+1/2})$, we have to assume that the trajectory from $\mathbf{x}(t^{n+1}) \to \mathbf{x}(t^n)$ is linear, in other words,

$$\mathbf{x}\big(t^{n+1/2}\big) = \frac{\mathbf{x}(t^{n+1}) + \mathbf{x}(t^n)}{2},$$

which then yields the iterative equation

$$\mathbf{x}\big(t^{n+1/2}\big) = \mathbf{x}(t^{n+1}) - \frac{\Delta t}{2}\,\mathbf{u}\big(\mathbf{x}\big(t^{n+1/2}\big), t^{n+1/2}\big). \tag{11}$$

We solve for the midpoint by starting at the arrival point $\mathbf{x}(t^{n+1})$ as the first approximation to $\mathbf{x}(t^{n+1/2})$. Continuing in this iterative fashion results in convergence in less than five iterations.

This scheme is second order accurate in space and time and is used quite extensively in the semi-Lagrangian community. On the sphere, the midpoint rule has to be modified such that the new departure points computed by (11) remain on the surface of the sphere. In other words, after each iteration we must apply the projection

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

where $a$ is the radius of the sphere.

However, the weakness of this scheme is that upon obtaining the correct midpoint trajectories, the departure points are then computed assuming a linear variation in the trajectories as shown in (11).

*4.2.2. Runge–Kutta scheme.* A higher-order approximation can be obtained by applying the Runge–Kutta scheme to the trajectory equation (7) yielding

$$\mathbf{x}(t^k) = \mathbf{x}(t^{n+1}) - \Delta t \beta \mathbf{u}(\mathbf{x}(t^{k-1}), t^{k-1}), \tag{12}$$

where

$$\beta = \frac{1}{M-k+1}, \qquad k = 1, \ldots, M, \qquad \text{and} \qquad t^{k-1} = \begin{cases} t^{n+1} & \text{for } k = 1 \\ t^{n+1-\beta} & \text{for } k > 1 \end{cases}.$$

This requires knowing $\mathbf{u}(\mathbf{x}(t^{n+1-\beta}), t^{n+1-\beta})$, which can be extrapolated from the velocity

field at the previous time levels. We could use the extrapolation given in (10) but this would formally give only second-order approximations for the velocity. Therefore, the most consistent approach is to use an extrapolation of order equal to that of the Runge–Kutta scheme. At the first Lagrange–Galerkin step ($t = 2\Delta t$) we already know the velocity at times $t = \Delta t$ and $t = 0$ (because the first time step is carried out with the Eulerian method). Therefore at this time step we can only use a maximum of a second-order scheme. Therefore we use a second-order extrapolation as well as a second-order Runge–Kutta scheme ($M = 2$). However, at the next time step ($t = 3\Delta t$) we now know the velocity fields at $t = 2\Delta t$, $\Delta t$ and 0, thereby allowing us to use a third-order scheme. Finally, at the third Lagrange–Galerkin step ($t = 4\Delta t$) we know the velocity field at sufficient time levels to allow us to use a fourth-order scheme. The extrapolations of the velocity field at time $n + 1 - \beta$ then can be obtained from the Taylor series expansions, giving

$$\mathbf{u}(t^{n+1-\beta}) = (1 + a + b + c)\mathbf{u}(t^n) + a\mathbf{u}(t^{n-1}) + b\mathbf{u}(t^{n-2}) + c\mathbf{u}(t^{n-3}),$$

where

$$\begin{array}{llll} a = \delta & b = 0 & c = 0 & \text{for } M = 2 \\ a = 2\delta + \delta^2 & b = -\frac{1}{2}\delta - \frac{1}{2}\delta^2 & c = 0 & \text{for } M = 3 \\ a = 3\delta + \frac{5}{2}\delta^2 + \frac{1}{2}\delta^3 & b = -\frac{3}{2}\delta - 2\delta^2 - \frac{1}{2}\delta^3 & c = \frac{1}{3}\delta + \frac{1}{2}\delta^2 + \frac{1}{6}\delta^3 & \text{for } M = 4 \end{array}$$

and

$$\delta = 1 - \beta.$$

However, it should be understood that getting from $\mathbf{u}(t^n)$ to $\mathbf{u}(\mathbf{x}(t^n), t^n)$ requires an interpolation in space because $x(t^n)$ generally will not fall on a grid point. This is where having explicit basis functions is an advantage because we now have an inherent mechanism for interpolating the velocity field at any point $x(t^n)$. In this algorithm, the interpolation is achieved by using the linear finite element basis functions using the values at the vertices of the triangular element containing this point.

This scheme does not require a first guess starting point as in the previous scheme because this is not an iterative scheme but rather is a multi-step scheme. Therefore we begin from the arrival point, as in the previous scheme, and step through toward the departure point. In addition, upon computing the next value from (12) we have to enforce the new point to remain on the sphere. The simple projection

$$\mathbf{x}^k = a\frac{\mathbf{x}^k}{|\mathbf{x}^k|}$$

enforces this constraint. The advantage of this scheme over the midpoint rule is that the Runge–Kutta scheme never assumes a linear trajectory [5]. Because of its higher-order approximation the Runge–Kutta scheme is better equipped to compute more accurate trajectories than the midpoint rule. A comparison between the composite midpoint rule and the Runge–Kutta scheme is given in [5].

4.3. *Element Equations*

Returning to (9) and substituting the conservation variables from (3) gives

$$\int_{\Omega^{n+1}} \psi \begin{bmatrix} \varphi \\ \varphi u \\ \varphi v \\ \varphi w \end{bmatrix}^{n+1} d\Omega^{n+1}$$

$$= \int_{\Omega^n} \psi \begin{bmatrix} \varphi \\ \varphi u \\ \varphi v \\ \varphi w \end{bmatrix}^{n} d\Omega^n + \Delta t\theta \int_{\Omega^n+1} \psi \begin{bmatrix} 0 \\ -\varphi \frac{\partial \varphi}{\partial x} + f\left(\frac{z}{a}\varphi v - \frac{y}{a}\varphi w\right) \\ -\varphi \frac{\partial \varphi}{\partial y} + f\left(\frac{x}{a}\varphi w - \frac{z}{a}\varphi u\right) \\ -\varphi \frac{\partial \varphi}{\partial z} + f\left(\frac{y}{a}\varphi u - \frac{x}{a}\varphi v\right) \end{bmatrix}^{n+1} d\Omega^{n+1}$$

$$+ \Delta t(1-\theta) \int_{\Omega^n} \psi \begin{bmatrix} 0 \\ -\varphi \frac{\partial \varphi}{\partial x} + f\left(\frac{z}{a}\varphi v - \frac{y}{a}\varphi w\right) \\ -\varphi \frac{\partial \varphi}{\partial y} + f\left(\frac{x}{a}\varphi w - \frac{z}{a}\varphi u\right) \\ -\varphi \frac{\partial \varphi}{\partial z} + f\left(\frac{y}{a}\varphi u - \frac{x}{a}\varphi v\right) \end{bmatrix}^{n} d\Omega^n, \tag{13}$$

which are the equations to be solved within each element. The mass (geopotential) and momentum equations can be decoupled as follows: the mass equation is no longer a function of velocity except through the trajectory equation (7),

$$\int_{\Omega^{n+1}} \psi(\varphi)\, d\Omega^{n+1} = \int_{\Omega^n} \psi(\varphi)\, d\Omega^n,$$

while the $u$-momentum,

$$\int_{\Omega^{n+1}} \psi(\varphi u)\, d\Omega^{n+1} - \Delta t\theta \int_{\Omega^{n+1}} \psi\left[ f\left(\frac{z}{a}\varphi v - \frac{y}{a}\varphi w\right)\right] d\Omega^{n+1}$$

$$= \int_{\Omega^n} \psi(\varphi u)\, d\Omega^n + \Delta t(1-\theta) \int_{\Omega^n} \psi\left[ -\varphi \frac{\partial \varphi}{\partial x} + f\left(\frac{z}{a}\varphi v - \frac{y}{a}\varphi w\right)\right] d\Omega^n$$

$$+ \Delta t\theta \int_{\Omega^{n+1}} \psi\left(-\varphi \frac{\partial \varphi}{\partial x}\right) d\Omega^{n+1},$$

$v$-momentum,

$$\int_{\Omega^{n+1}} \psi(\varphi v)\, d\Omega^{n+1} - \Delta t\theta \int_{\Omega^{n+1}} \psi\left[ f\left(\frac{x}{a}\varphi w - \frac{z}{a}\varphi u\right)\right] d\Omega^{n+1}$$

$$= \int_{\Omega^n} \psi(\varphi v)\, d\Omega^n + \Delta t(1-\theta) \int_{\Omega^n} \psi\left[ -\varphi \frac{\partial \varphi}{\partial y} + f\left(\frac{x}{a}\varphi w - \frac{z}{a}\varphi u\right)\right] d\Omega^n$$

$$+ \Delta t\theta \int_{\Omega^{n+1}} \psi\left(-\varphi \frac{\partial \varphi}{\partial y}\right) d\Omega^{n+1},$$

and $w$-momentum,

$$\int_{\Omega^{n+1}} \psi(\varphi w)\, d\Omega^{n+1} - \Delta t\theta \int_{\Omega^{n+1}} \psi \left[ f\left(\frac{y}{a}\varphi u - \frac{x}{a}\varphi v\right)\right] d\Omega^{n+1}$$
$$= \int_{\Omega^n} \psi(\varphi w)\, d\Omega^n + \Delta t(1-\theta) \int_{\Omega^n} \psi \left[ -\varphi\frac{\partial \varphi}{\partial z} + f\left(\frac{y}{a}\varphi u - \frac{x}{a}\varphi v\right)\right] d\Omega^n$$
$$+ \Delta t\theta \int_{\Omega^{n+1}} \psi\left(-\varphi\frac{\partial \varphi}{\partial z}\right) d\Omega^{n+1},$$

are all coupled through the Coriolis terms.

### 4.4. *Global Equations*

Approximating the conservation variables within each element $\Omega$ by the expansion

$$\varphi = \sum_{j=1}^{3} \psi_j \varphi_j \tag{14}$$

and adding all of the contributions from the elements to each of the global grid points results in the following linear system of global equations

$$A_{i,j}\, \varphi_j^{n+1} = b_i^{\varphi} \tag{15}$$

and

$$\begin{bmatrix} A_{i,j} & -D_{i,j} & C_{i,j} \\ D_{i,j} & A_{i,j} & -B_{i,j} \\ -C_{i,j} & B_{i,j} & A_{i,j} \end{bmatrix} \begin{bmatrix} (\varphi u)_j \\ (\varphi v)_j \\ (\varphi w)_j \end{bmatrix}^{n+1} = \begin{bmatrix} b_i^u \\ b_i^v \\ b_i^w \end{bmatrix} \qquad i, j = 1, \ldots, N_p, \tag{16}$$

where $N_p$ denotes the number of grid points in the grid. The elements of the above block matrix are

$$A_{i,j} = \int_{\Omega^{n+1}} \psi_i \psi_j\, d\Omega^{n+1}$$
$$B_{i,j} = \Delta t\theta \int_{\Omega^{n+1}} \psi_i \psi_j \psi_k \psi_l \left(\frac{x_l}{a}\right) f_k\, d\Omega^{n+1}$$
$$C_{i,j} = \Delta t\theta \int_{\Omega^{n+1}} \psi_i \psi_j \psi_k \psi_l \left(\frac{y_l}{a}\right) f_k\, d\Omega^{n+1}$$
$$D_{i,j} = \Delta t\theta \int_{\Omega^{n+1}} \psi_i \psi_j \psi_k \psi_l \left(\frac{z_l}{a}\right) f_k\, d\Omega^{n+1}$$

and $b_i^{\varphi}, b_i^u, b_i^v, b_i^w$ are the right-hand side vectors for the mass and momentum equations, respectively, and are defined as

$$b_i^{\varphi} = \int_{\Omega^n} \psi(\varphi)\, d\Omega^n$$
$$b_i^u = \int_{\Omega^n} \psi(\varphi u)\, d\Omega^n + \Delta t(1-\theta) \int_{\Omega^n} \psi \left[ -\varphi\frac{\partial \varphi}{\partial x} + f\left(\frac{z}{a}\varphi v - \frac{y}{a}\varphi w\right)\right] d\Omega^n$$
$$+ \Delta t\theta \int_{\Omega^{n+1}} \psi\left(-\varphi\frac{\partial \varphi}{\partial x}\right) d\Omega^{n+1}$$

$$b_i^v = \int_{\Omega^n} \psi(\varphi v)\, d\Omega^n + \Delta t(1-\theta) \int_{\Omega^n} \psi \left[ -\varphi \frac{\partial \varphi}{\partial y} + f\left( \frac{x}{a}\varphi w - \frac{z}{a}\varphi u \right) \right] d\Omega^n$$
$$+ \Delta t\theta \int_{\Omega^{n+1}} \psi \left( -\varphi \frac{\partial \varphi}{\partial y} \right) d\Omega^{n+1}$$
$$b_i^w = \int_{\Omega^n} \psi(\varphi w)\, d\Omega^n + \Delta t(1-\theta) \int_{\Omega^n} \psi \left[ -\varphi \frac{\partial \varphi}{\partial z} + f\left( \frac{y}{a}\varphi u - \frac{x}{a}\varphi v \right) \right] d\Omega^n$$
$$+ \Delta t\theta \int_{\Omega^{n+1}} \psi \left( -\varphi \frac{\partial \varphi}{\partial z} \right) d\Omega^{n+1}.$$

The right-hand side values are obtained by integrating on $\Omega^n$, which represents the element at the departure points along the characteristics. In this paper, the integrals are obtained using a quintic quadrature rule (see [5] for further details).

Consider that the matrices $A, B, C$, and $D$ are all symmetric positive definite but the global system is not. In fact, the global system is skewed symmetric and for this type of system there is no one best solver. The global matrix is sparse, however, and iterative methods may be the best methods of solution. The good news is that although we began with a system of nonlinear equations, discretizing the equations by the weak Lagrange–Galerkin method results in a system of linear equations.

Because the system is not symmetric positive definite we cannot utilize many of the best known solvers such as the conjugate gradient method. There are, however, variants of these methods that can be used such as the biconjugate gradient method. Because the system is quite sparse, direct methods may be inefficient. For the moment, we are employing a direct LU decomposition method for simplicity, but we are also exploring ad hoc direct solvers for this type of a skewed-symmetric matrix and multi-grid methods.

Another possibility for solving the momentum equations more efficiently is to simplify the equations in order to make them symmetric positive definite. This approach is introduced in [1] and the strategy is to extrapolate the velocity field (using the values at $n$ and $n-1$) to get an approximation at $n+1$. This now allows moving the $B, C$, and $D$ matrices in (16) to the right-hand side. Each conservation variable can now be solved explicitly; in other words, all four equations are now completely decoupled. However, doing this restricts the size of the time step that can be used but the advantages are that only one left-hand side matrix needs to be stored and inverted. This matrix is the mass matrix which is symmetric positive definite and can be inverted quite easily using straightforward conjugate gradient methods.

## 5. ICOSAHEDRAL GRIDS

One of the difficulties of using icosahedral grids is that we are restricted to only a few possible grids because the number of grid points increases by a factor of 4 with each refinement of the grid. In other words, from the fourth ($n=4$) to fifth ($n=5$) refinement we go from 2562 to 10,242 grid points (see [3, 6]). This is a very limiting constraint when trying to do, say, a grid convergence study on a workstation. A way around this dilemma is to construct the following general icosahedral grid.

We begin with the initial icosahedron having 12 points and 20 equilateral triangular elements. Then we subdivide each triangle of the initial icosahedron by a $p$th-order Lagrange polynomial. Before doing so, however, it is best to map this triangle onto a gnomonic space.

The most unbiased mapping is obtained by mapping about the centroid of the icosahedral triangle. (Because all of the initial triangles are equilateral, the centroid also happens to be the circumcenter of the triangle.) Let $(\lambda_c, \theta_c)$ be the centroid of the triangle we wish to map. Then, the gnomonic mapping is given by

$$\begin{aligned} x &= \frac{a\cos\theta\sin(\lambda-\lambda_c)}{\sin\theta_c\sin\theta+\cos\theta_c\cos\theta\cos(\lambda-\lambda_c)}, \\ y &= \frac{a[\cos\theta_c\sin\theta-\sin\theta_c\cos\theta\cos(\lambda-\lambda_c)]}{\sin\theta_c\sin\theta+\cos\theta_c\cos\theta\cos(\lambda-\lambda_c)}, \end{aligned} \tag{17}$$

which is rather complicated. However, if we first apply a rotation mapping whereby in the new coordinate system the coordinates $(\lambda, \theta)$ are located at $(0, 0)$, then (17) becomes

$$x = a\tan\lambda', \qquad y = a\tan\theta'\sec\lambda', \tag{18}$$

where the rotation mapping is

$$\begin{aligned} \lambda' &= \arctan\left[\frac{\cos\theta\sin(\lambda-\lambda_c)}{\sin\theta_c\sin\theta+\cos\theta_c\cos\theta\cos(\lambda-\lambda_c)}\right], \\ \theta' &= \arcsin[\cos\theta_c\sin\theta-\sin\theta_c\cos\theta\cos(\lambda-\lambda_c)]. \end{aligned} \tag{19}$$

This approach results in the construction of a general icosahedral grid with the properties

$$\begin{aligned} N_p &= 10(p-1)^2+20(p-1)+12 \\ N_e &= 2(N_p-2) \\ N_s &= 3(N_p-2), \end{aligned}$$

where $N_p$, $N_e$, and $N_s$ denote the number of points, triangular elements, and sides comprising the grid, and $p$ is the order of the Lagrange polynomial used to subdivide the 20 initial triangles of the icosahedron. If we substitute $p=2^n$ then we recover the results for the icosahedral grids used in [3, 6, 9]. The results of the new icosahedral grid ($p$) and the old icosahedral grid ($n$) are shown in Table I. For up to 10,242 grid points there are only 5 possible old icosahedral grid configurations whereas there are 32 possible new icosahedral grids.

### 5.1. *Searching Algorithms*

The crux of any Lagrangian scheme is the accurate calculation of the particle trajectories, in other words, the correct solution of the trajectory equation (7). Once the correct trajectories are computed, it then remains to interpolate the conservation variables at the departure points. In [3] a quadtree-type data structure was introduced which took advantage of the subdivision of each triangle into four smaller triangles, as is done in the typical icosahedral grids. However, because we no longer have this quadtree structure, we must devise an alternate searching strategy. The general icosahedral grid, in fact, makes it much easier to search for the departure points.

We begin by constructing an integer pointer array intma_tree $[1:20, 1:p^2]$, which stores the triangular element identification numbers contained within each of the first initial 20

**TABLE I**
**The Number of Grid Points, Elements, and Sides for the Icosahedral Grid**

| $p$ | $n$ | $N_p$ | $N_e$ | $N_s$ |
|---|---|---|---|---|
| 1 | 0 | 12 | 20 | 30 |
| 2 | 1 | 42 | 80 | 120 |
| 4 | 2 | 162 | 320 | 480 |
| 8 | 3 | 642 | 1280 | 1920 |
| 10 | — | 1002 | 2000 | 3000 |
| 15 | — | 2252 | 4500 | 6750 |
| 16 | 4 | 2562 | 5120 | 7680 |
| 20 | — | 4002 | 8002 | 12000 |
| 30 | — | 9002 | 18000 | 27000 |
| 32 | 5 | 10242 | 20480 | 30720 |
| 40 | — | 16002 | 32000 | 48000 |
| 64 | 6 | 40962 | 81920 | 122880 |

*Note.* The generalized (new) icosahedral grid is given by $p$ and the typical (old) icosahedral grid is given by the different refinement values $n$.

triangles of the icosahedron. For a given departure point, we search through the initial triangles of the icosahedron and use the inclusion tests described in [3]. Once we have the $i$th initial triangle which contains the point, we can then simplify the search to that of an ordered list because, after all, the $p^2$ triangles (children) contained within each initial triangle (parent) are equally spaced within their parent element. Therefore, by first mapping the departure point in Cartesian coordinates $(x_d, y_d, z_d)$ to the corresponding local coordinates $(\xi_d, \eta_d)$ of the parent, we can then compute the indices of the child that contains this departure point. (Figure 1 shows a parent triangle for $p = 3$.) In [3] the conditions for testing for an inclusion are described. In that paper the area coordinates are used for this test. Therefore,

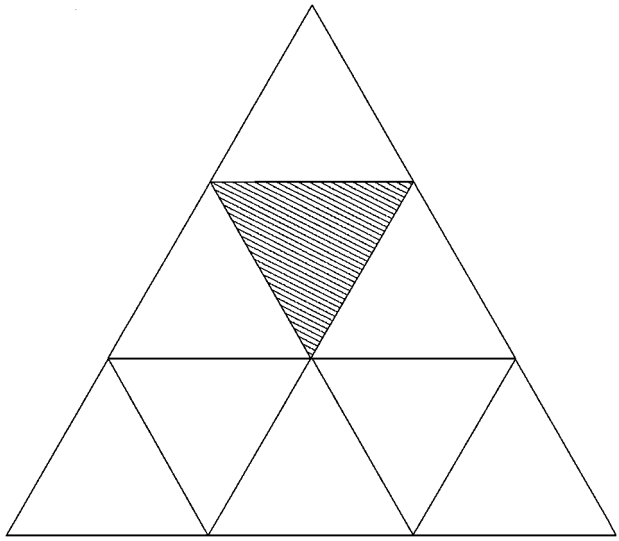

**FIG. 1.** A schematic of one of the initial 20 triangles of the icosahedron for $p = 3$ which yields $p^2$ subtriangles.

since the area coordinates for the departure point are defined by

$$\begin{aligned}\psi_1(x_d, y_d, z_d) &= 1 - \xi_d - \eta_d \\ \psi_2(x_d, y_d, z_d) &= \xi_d \\ \psi_3(x_d, y_d, z_d) &= \eta_d,\end{aligned} \tag{20}$$

we automatically have the mapping from Cartesian to local space. The indices of the child which contains this point are then given by the relations

$$i_d = 1 + \text{int}(p\,\xi_d), \qquad j_d = 1 + \text{int}(p\eta_d),$$

where int is the integer function. However, the local coordinates of $\mathbf{x}_d$ inside the triangular element defined by $(i_d, j_d)$ are not equal to $(\xi_d, \eta_d)$ and so we must map this local coordinate from the parent element to its child $(i_d, j_d)$. This can be accomplished by using the transformations

$$\xi_d = p\,\xi_d - (i_d - 1), \qquad \eta_d = p\,\eta_d - (j_d - 1),$$

which essentially scales the local coordinates from $[0, 1]$ to $[0, p]$. This transformation yields the local coordinates of the departure point in the square $(i_d, j_d)$, as is shown in Fig. 2; however, there are two triangles on this square and we need to determine which one of these two triangles contains the point. In order to determine which triangle contains the departure point, we simply test the value of the function along the hypotenuse

$$\begin{aligned}&\text{if} \quad 1 - \xi_d - \eta_d \geq 0 \qquad \text{then lower triangle} \\ &\text{if} \quad 1 - \xi_d - \eta_d < 0 \qquad \text{then upper triangle}\end{aligned}$$

and rewrite the basis functions according to the result of this test. For example, if the lower triangle (left) contains the point, then the basis functions are given by (20), whereas if the

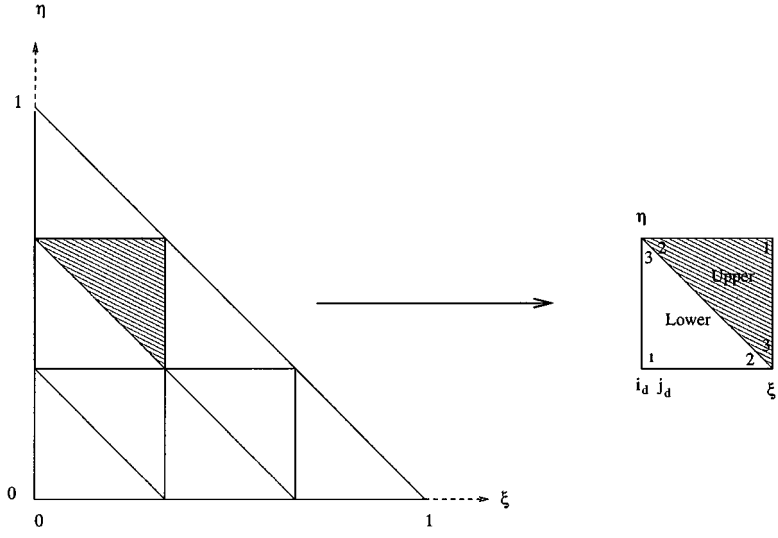


**FIG. 2.** A schematic of the initial triangle of the icosahedron in local coordinate space.

upper triangle (right) contains the point, then the basis functions are given by

$$\begin{aligned}\psi_1(x_d, y_d, z_d) &= \xi_d + \eta_d - 1\\ \psi_2(x_d, y_d, z_d) &= 1 - \xi_d\\ \psi_3(x_d, y_d, z_d) &= 1 - \eta_d,\end{aligned} \tag{21}$$

which amounts to making the change of coordinates $\xi_d = 1 - \xi_d$ and $\eta_d = 1 - \eta_d$. This searching strategy requires only one set of inclusion tests to determine whether a triangle contains the departure point. After the initial triangle (parent) is found, the rest of the searching strategy simplifies to searching an ordered table using a few floating point operations.

## 6. RESULTS

For the numerical experiments, the following terms are used in order to judge the performance of the scheme: the $L_2$ error norms for the mass and momentum,

$$M = \frac{\int_\Omega (\varphi_{\text{exact}} - \varphi)^2 \, d\Omega}{\int_\Omega \varphi_{\text{exact}}^2 \, d\Omega}$$

$$U = \frac{\int_\Omega (u_{\text{exact}} - u)^2 + (v_{\text{exact}} - v)^2 + (w_{\text{exact}} - w)^2 \, d\Omega}{\int_\Omega u_{\text{exact}}^2 + v_{\text{exact}}^2 + w_{\text{exact}}^2 \, d\Omega},$$

the trajectory norm,

$$T = \frac{\int_\Omega \left(\mathbf{x}_d - \mathbf{x}_d^{\text{exact}}\right)^2 d\Omega}{\int_\Omega \left(\mathbf{x}_a - \mathbf{x}_d^{\text{exact}}\right)^2 d\Omega},$$

where a and d denote the arrival and departure points, and the two additional conservation measures

$$M_1 = \frac{\int_\Omega \varphi \, d\Omega}{\int_\Omega \varphi_{\text{exact}} \, d\Omega}$$

and

$$M_2 = \frac{\int_\Omega \varphi(u^2 + v^2 + w^2) + \varphi^2 \, d\Omega}{\int_\Omega \varphi_{\text{exact}} \left(u_{\text{exact}}^2 + v_{\text{exact}}^2 + w_{\text{exact}}^2\right) + \varphi_{\text{exact}}^2 \, d\Omega}.$$

The $L_2$ error norms $M$ and $U$ compare the root mean square percent errors of the numerical and exact solutions, $T$ measures the accuracy of the trajectories, $M_1$ measures the conservation property of the mass, and $M_2$ measures the conservation of the total available energy. The ideal scheme should yield $L_2$ error and trajectory norms of zero, and mass and energy measures of one.

Six test cases are used in order to test the algorithm. Test cases 1, 3, 4, and 5 correspond to test cases 1, 2, 3, and 6 given in [12]. Test case 2 is similar to case 1 but for a time-dependent velocity field and is given in [1], whereas test case 6 is given in [7]. Test cases 1 and 2 involve the mass equation only whereas the remainder of the test cases concern the

full shallow water equations. In addition, cases 1–4 have analytic solutions and are used to determine the accuracy of the Lagrange–Galerkin method quantitatively. Test cases 5 and 6, on the other hand, do not have analytic solutions and are thus only used to determine the accuracy of the scheme qualitatively. All the cases were run using a time step twice as large as that allowed by the second-order Runge–Kutta Eulerian finite element scheme.

### 6.1. *Case 1: Steady-State Advection*

This test case concerns the solid body rotation of a cosine wave. It only tests the mass (geopotential) equation as the velocity field remains unchanged throughout the computation. Results are reported after one full revolution, which corresponds to an integration of 12 days.

By using the mapping from spherical to Cartesian space

$$\begin{aligned} x &= a\cos\theta\cos\lambda \\ y &= a\cos\theta\sin\lambda \\ z &= a\sin\theta, \end{aligned} \tag{22}$$

where

$$\begin{aligned} \lambda &= \arctan\left(\frac{y}{x}\right) \\ \theta &= \arcsin\left(\frac{z}{a}\right) \end{aligned} \tag{23}$$

we can write the initial conditions in terms of Cartesian coordinates. The spherical coordinates $(\lambda, \theta)$ correspond to the longitude and latitude. This results in the following velocity field in Cartesian space

$$\begin{aligned} u &= -u_s\sin\lambda - v_s\sin\theta\cos\lambda \\ v &= +u_s\cos\lambda - v_s\sin\theta\sin\lambda \\ w &= +v_s\cos\theta, \end{aligned}$$

where $u_s$ and $v_s$ are the zonal and meridional velocity components in spherical coordinates.

A convergence study is shown in Fig. 3 for various grids. This figure shows that refining the grid by a factor of two increases the accuracy by a factor of four. In other words, this figure is showing that the method is second-order accurate. Figure 4 shows the $L_2$ norm of the mass with respect to time for the grids $p = 8$, 16, and 32. This figure shows that the second-order accuracy is maintained throughout the 12-day integration. Figure 5 shows the effects of the time step on the solution accuracy for the grid $p = 16$. The trajectory error increases with an increase in time step, but the mass error decreases, reaches a plateau, and then increases. This plot shows that for a given problem, there is an optimal time step. The reason we do not see a decrease in the mass error with decreasing time step is that although the trajectories become more accurate, more time steps also require more interpolations. Because interpolations are used to obtain the conservation variables at the quadrature points, we then expect a decrease in accuracy as the number of interpolations increase. This behavior will remain regardless of the order of the interpolations. In other words, going to a higher-order interpolation will not make this source of error disappear.

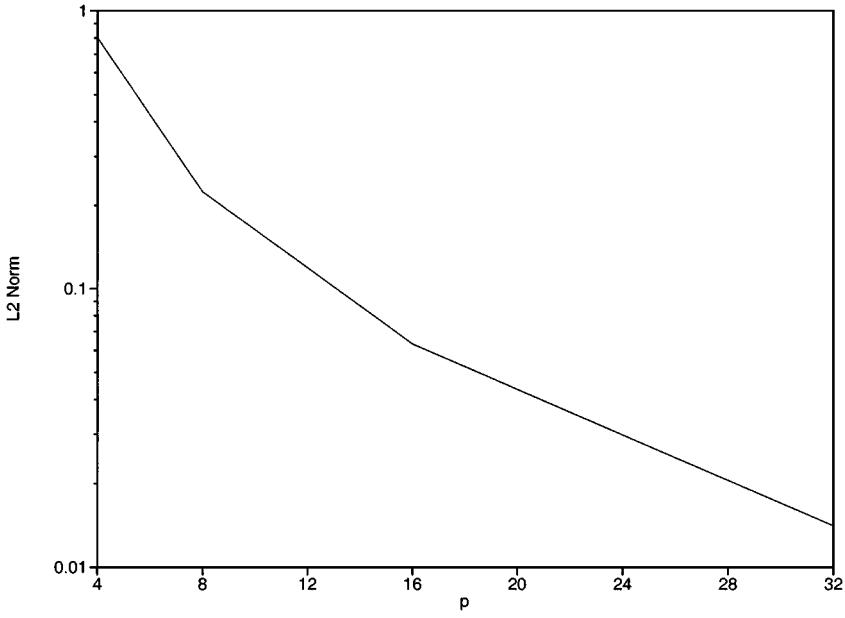


**FIG. 3.** Case 1. Convergence study of the mass error.

### 6.2. *Case 2: Time-Dependent Advection*

This test case is similar to case 1 except that the velocity field is now a function of time and is given in [1] as

$$u = (-u_s \sin\lambda - v_s \sin\theta \cos\lambda)\left(1 + \cos\frac{t\omega}{a}\right)$$

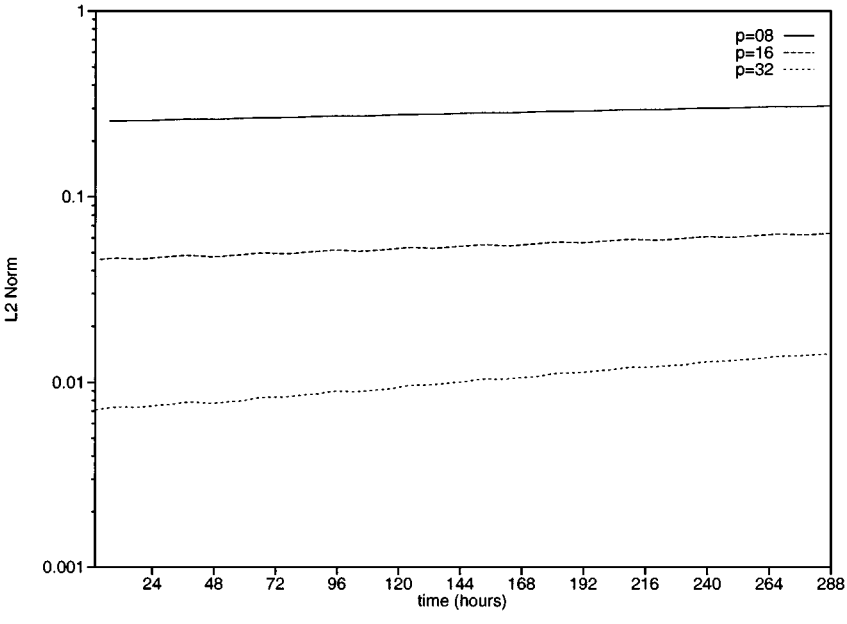


**FIG. 4.** Case 1. Time history study of the mass error for three different grids.

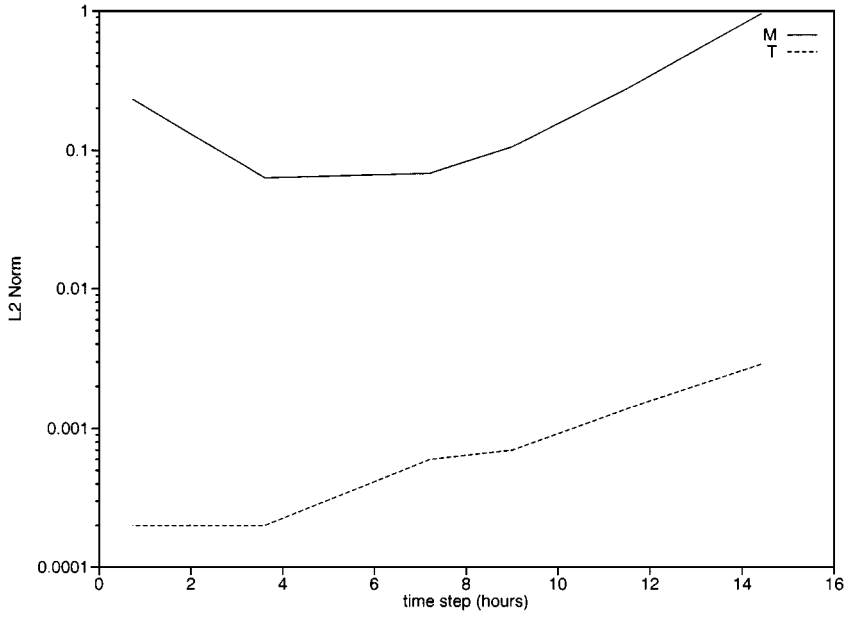


**FIG. 5.** Case 1. Time-step study of the mass (M) and trajectory (T) errors for the grid $p = 16$.

$$v = (+u_s \cos\lambda - v_s \sin\theta \sin\lambda)\left(1 + \cos\frac{t\omega}{a}\right)$$

$$w = (+v_s \cos\theta)\left(1 + \cos\frac{t\omega}{a}\right),$$

where the time is normalized such that it is in the range

$$t \in [0, 1]$$

and $\omega$ is the angular velocity as given in [12] as

$$\omega = \frac{2\pi a}{12 \text{ days}},$$

where $a$ is the radius of the earth. At the beginning and end of the revolution, the velocity is twice the steady-state value (case 1) and zero at a half revolution. This test case is good for determining the accuracy of the trajectory calculations (Eq. (7)) and to determine the sensitivity of the time step on a time-dependent velocity field.

The exact solution of this test case (as well as case 1) is given by

$$\varphi_{\text{exact}}(\mathbf{x}, t + \Delta t) = \varphi(\mathbf{x} - \Delta t\mathbf{u}, t).$$

By applying a rotation transformation, we get the arrival points in the rotated space

$$\lambda'_a = \arctan[\cos\theta_a \sin(\lambda_a - \lambda_\alpha), \cos\theta_a \cos(\lambda_a - \lambda_\alpha)\cos\theta_\alpha + \sin\theta_a \sin\theta_\alpha]$$

$$\theta'_a = \arcsin[\sin\theta_a \cos\theta_\alpha - \cos\theta_a \cos(\lambda_a - \lambda_\alpha)\sin\theta_\alpha],$$

which now only consists of motion about the equator. The departure points in the rotated space are

$$\lambda'_{\mathrm{d}} = \lambda'_{\mathrm{a}} - \left[\frac{\Delta t\omega}{a} + \sin\frac{(t+\Delta t)\omega}{a} - \sin\frac{t\omega}{a}\right]$$
$$\theta'_{\mathrm{d}} = \theta'_{\mathrm{a}},$$

where $\lambda_\alpha = 0$ and $\theta_\alpha = \alpha$. The parameter $\alpha$ represents the axis of rotation with respect to the equator. In other words, $\alpha = 0$ corresponds to rotation about the equator whereas $\alpha = 90$ corresponds to a rotation about the poles. (The value $\alpha = 0$ was used for all of the test cases, as there was no noticeable difference between various values of $\alpha$). Using the inverse transformation, we get

$$\begin{aligned}\lambda_{\mathrm{d}} &= \lambda_\alpha + \arctan[\cos\theta'_{\mathrm{d}}\sin\lambda'_{\mathrm{d}}, \cos\theta'_{\mathrm{d}}\cos\lambda'_{\mathrm{d}}\cos\theta_\alpha - \sin\theta'_{\mathrm{d}}\sin\theta_\alpha]\\ \theta_{\mathrm{d}} &= \arcsin[\cos\theta'_{\mathrm{d}}\cos\lambda'_{\mathrm{d}}\sin\theta_\alpha + \sin\theta'_{\mathrm{d}}\cos\theta_\alpha],\end{aligned} \tag{24}$$

which are the departure points in the original (unrotated) spherical space. The departure points in Cartesian space can now be obtained using the mapping (22).

A convergence study is shown in Fig. 6 for various grids. The results presented for this test case also show that the scheme is yielding second order accuracy. Figure 7 shows the $L_2$ norm of the mass with respect to time for the grids $p = 8$, 16, and 32. The results for this test case are similar to those of case 1. This test case tells us that the particle trajectories are being computed accurately regardless of the time-dependent velocity field. Figure 8 shows the effects of the time step on the solution accuracy for the grid $p = 16$. In order to get accuracies of the same order of accuracy as those obtained for case 1, we had to run time steps half the size of those used in case 1. This is because at the beginning and at the end of the 12 day integration, the velocity field for this case is twice as large as the steady-state

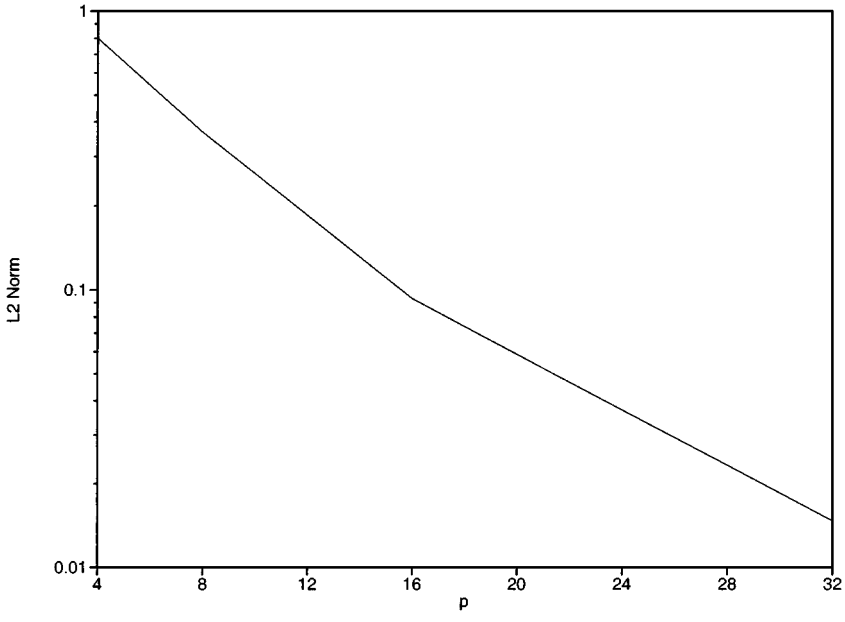


**FIG. 6.** Case 2. Convergence study of the mass error.

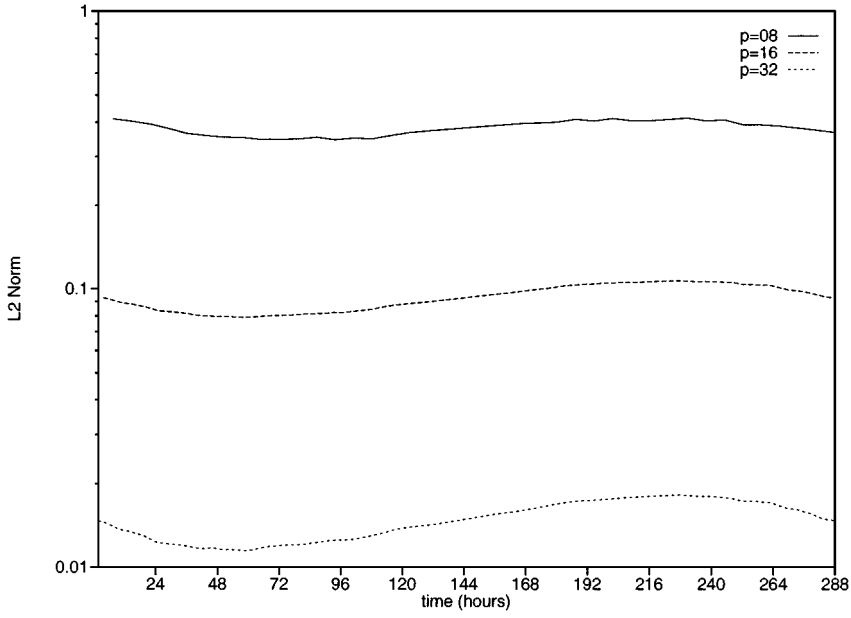


**FIG. 7.** Case 2. Time history study of the mass error for three different grids.

case. Otherwise, the results between cases 1 and 2 are identical. Having confidence in the precision of our particle trajectories, we next turn to the full shallow water equations.

### 6.3. *Case 3: Global Steady-State Nonlinear Zonal Geostrophic Flow*

This case is a steady-state solution to the nonlinear shallow water equations. The equations are geostrophically balanced and remain so for the duration of the 5-day integration. The velocity field remains constant throughout the computation, while the geopotential height

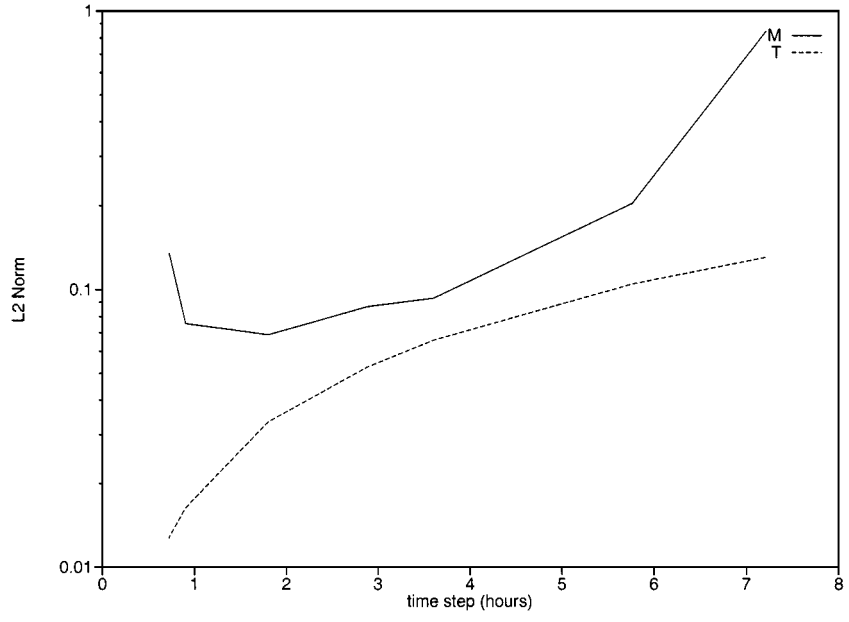


**FIG. 8.** Case 2. Time-step study of the mass (M) and trajectory (T) errors for the grid $p = 16$.

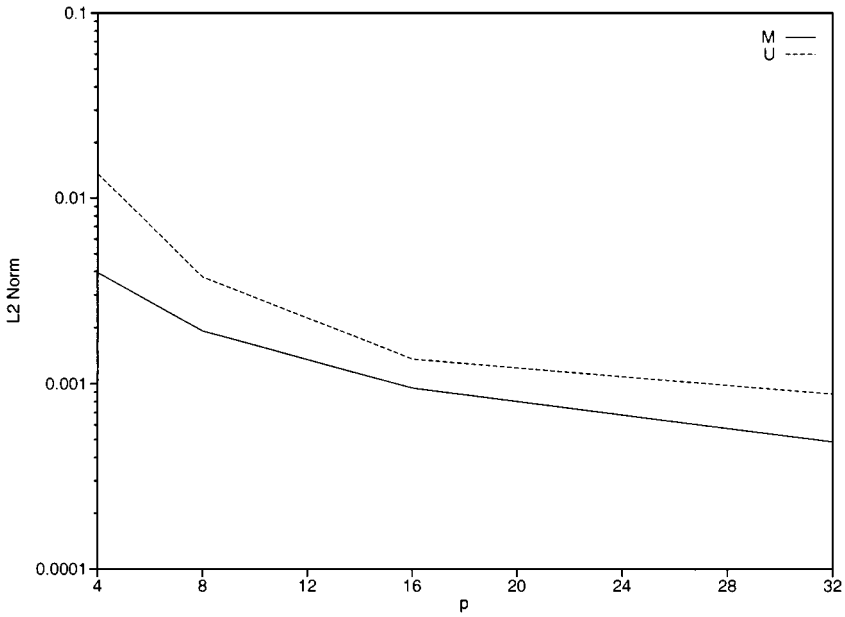


**FIG. 9.** Case 3. Convergence study of the mass (M) and momentum (U) errors.

undergoes a solid body rotation. The velocity field is the same field used in case 1, and the geopotential is given as a constant band. Since the initial geopotential height rotates along its axis of isotropy, the solution of the geopotential height looks the same throughout the time integration. In this sense, it is a steady-state solution to the shallow water equations.

A convergence study for the mass and momentum errors is shown in Fig. 9 for various grids. The errors for the mass and momentum converge at the same rate and are approximately of the same order of magnitude. Figure 10 shows the $L_2$ norm of the mass

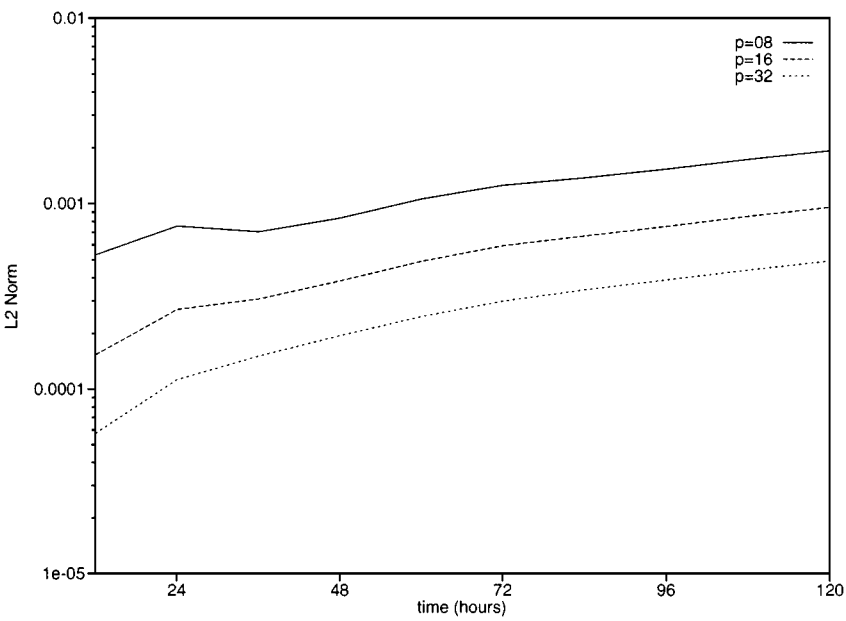


**FIG. 10.** Case 3. Time history study of the mass error for three different grids.

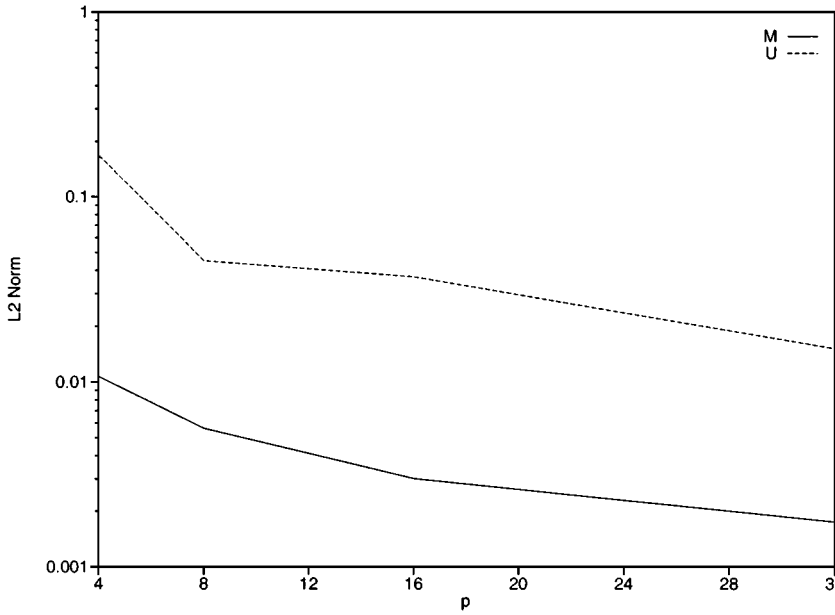


**FIG. 11.** Case 4. Convergence study of the mass (M) and momentum (U) errors.

with respect to time for the grids $p = 8$, 16, and 32. The solution is clearly converging for increased resolution but the rate appears to be closer to first rather than second order.

### 6.4. *Case 4: Steady-State Nonlinear Zonal Geostrophic Flow with Compact Support*

This case is similar to case 3 except that the velocity is zero everywhere except in a very small isolated region. This isolated region, or jet, encapsulates the flow and restricts the geopotential height field to remain within a very confined region. The results reported below are for a 5-day integration.

Figure 11 shows that the mass and momentum converge at roughly the same rate but the mass is more accurate than the momentum by an order of magnitude. Figure 12 shows the $L_2$ norm of the mass with respect to time for the grids $p = 8$, 16, and 32. The $L_2$ norm remains constant throughout the 5-day integration. This case also appears to be converging at a first-order rate both for the mass and momentum.

### 6.5. *Case 5: Rossby–Haurwitz Wave*

Although these waves are not analytic solutions to the shallow water equations, they have become a de facto test case. In a nondivergent barotropic model, the initial geopotential height field (mass) undergoes a solid body rotation in a counterclockwise direction (when viewed from the north pole). The angular velocity is given by

$$\nu = \frac{R(3+R)\omega - 2\Omega}{(1+R)(2+R)},$$

where $R = 4$ is the wave number. Because this solution does not have an analytic solution, we have used the fourth-order Runge–Kutta finite element solution presented in Section 3 with the grid $p = 40$ as the reference solution.

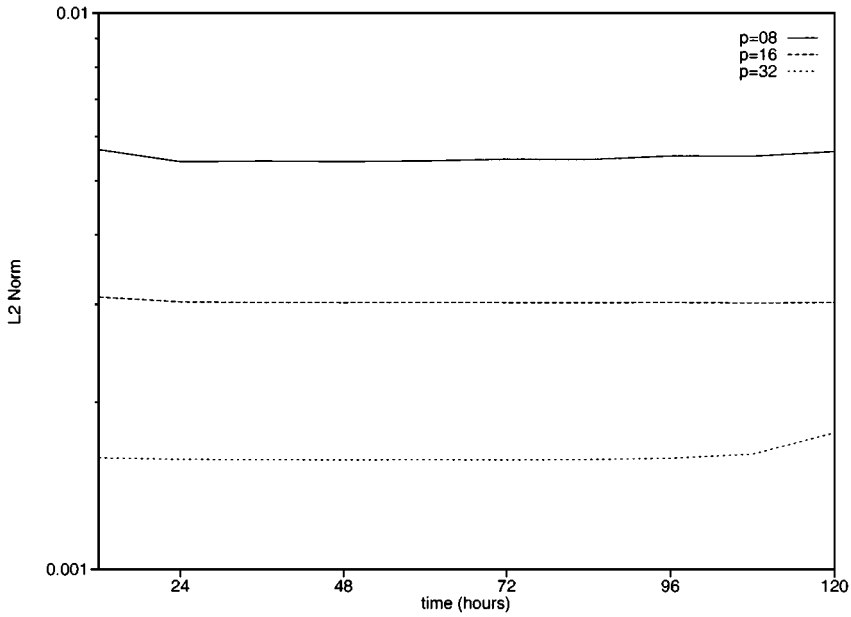


**FIG. 12.** Case 4. Time history study of the mass error for three different grids.

The final contours for the geopotential height (mass) and velocity field (momentum) for the grid $p = 20$ after a 7-day integration are shown in Fig. 13. These contours are drawn from the view point $(\lambda, \theta) = (0, 0)$. In these figures, the velocity components illustrated are actually the spherical velocity components $(u_s, v_s)$. These values are obtained by the inverse mapping of (23).

Figures 14 and 15 show the $L_2$ norms of the mass and momentum with respect to time for the grids $p = 10$, 15, and 20. The errors increase with time at a constant rate; however, the mass and momentum convergence to second order. This is particularly noticeable by comparing the grids $p = 10$ and $p = 20$ at the end of the 7-day integration.

### 6.6. *Case 6: Dancing Hi–Lo Waves*

This test case comes from [7] and, like case 5, is not an analytic solution to the shallow water equations. The initial geopotential height (mass) consists of two large waves with the low wave situated on the left and the high wave on the right, when viewed from $(\lambda, \theta) = (0, 90)$. The waves rotate clockwise in a dance-like fashion, at which point at 5 days of integration, the high wave is now on the left and the low is on the right. The final contours for the geopotential height (mass) and velocity field (momentum) for the grid $p = 30$ are shown in Fig. 16. Once again we used the Eulerian Runge–Kutta scheme with $p = 40$ as the reference solution.

Figures 17 and 18 show the $L_2$ norms of the mass and momentum with respect to time for the grids $p = 10$, 20, and 30. The errors oscillate with respect to time but they do increase at a slow rate. However, as in cases 3 and 4, the errors converge to first order.

### 6.7. *Summary of Results*

The results obtained for cases 1 and 2 represent very impressive levels of accuracy. The results were second-order accurate whether the velocity field was constant or changing

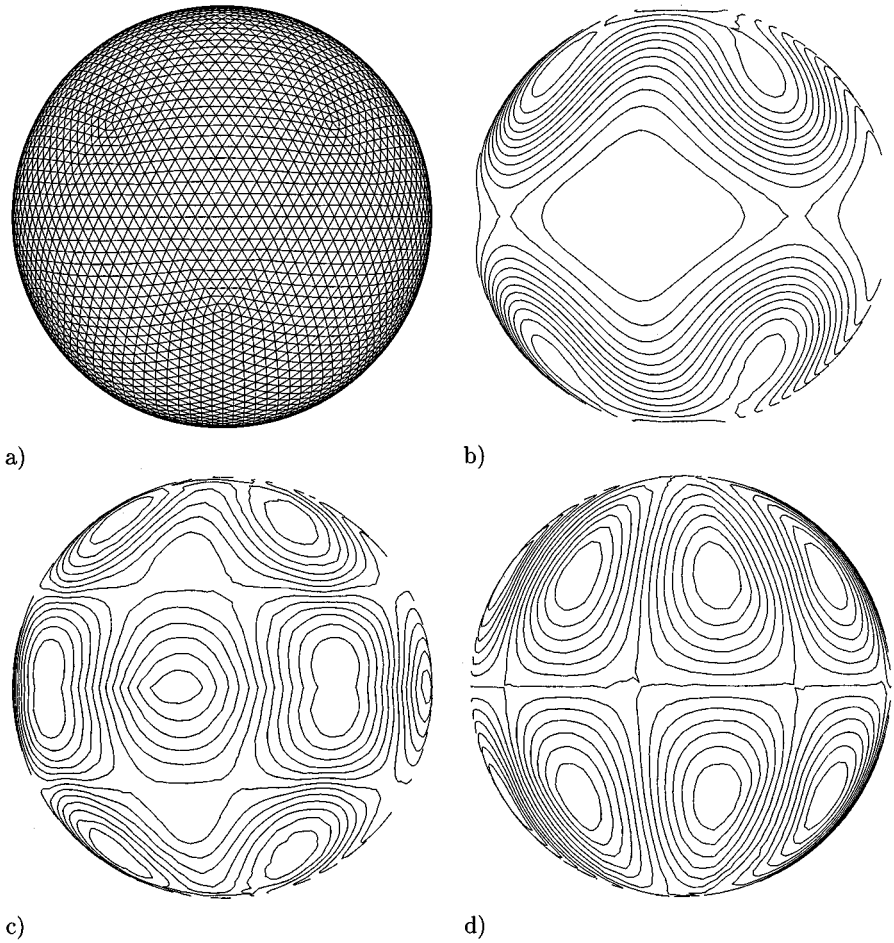


**FIG. 13.** Case 5. The (a) grid, (b) mass, (c) zonal velocity, and (d) meridional velocity on grid $p = 20$ after 7 days.

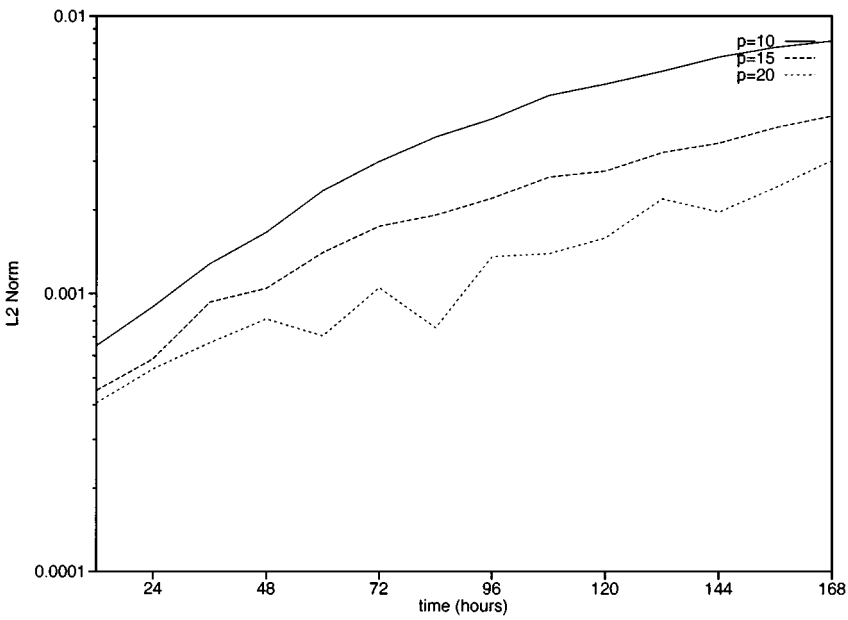


**FIG. 14.** Case 5. Time history study of the mass error for three different grids.

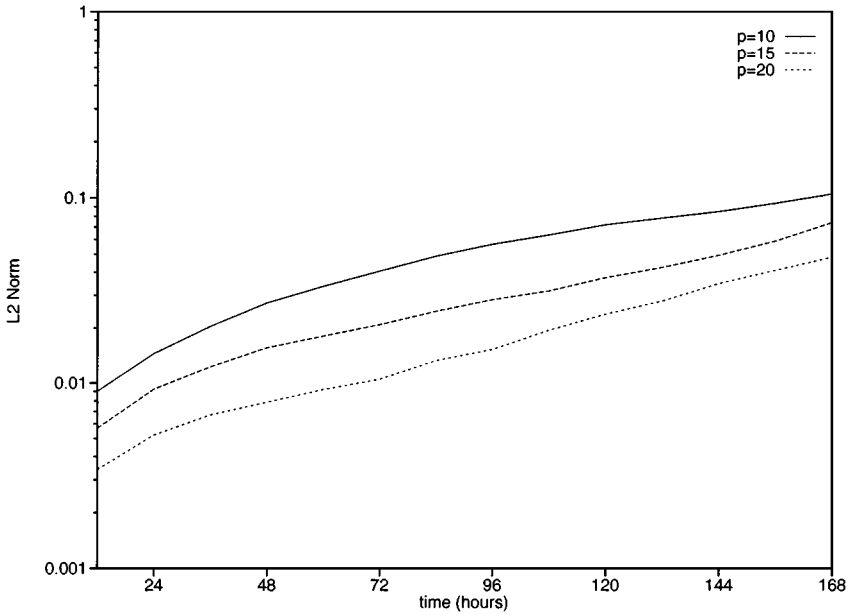


**FIG. 15.** Case 5. Time history study of the momentum error for three different grids.

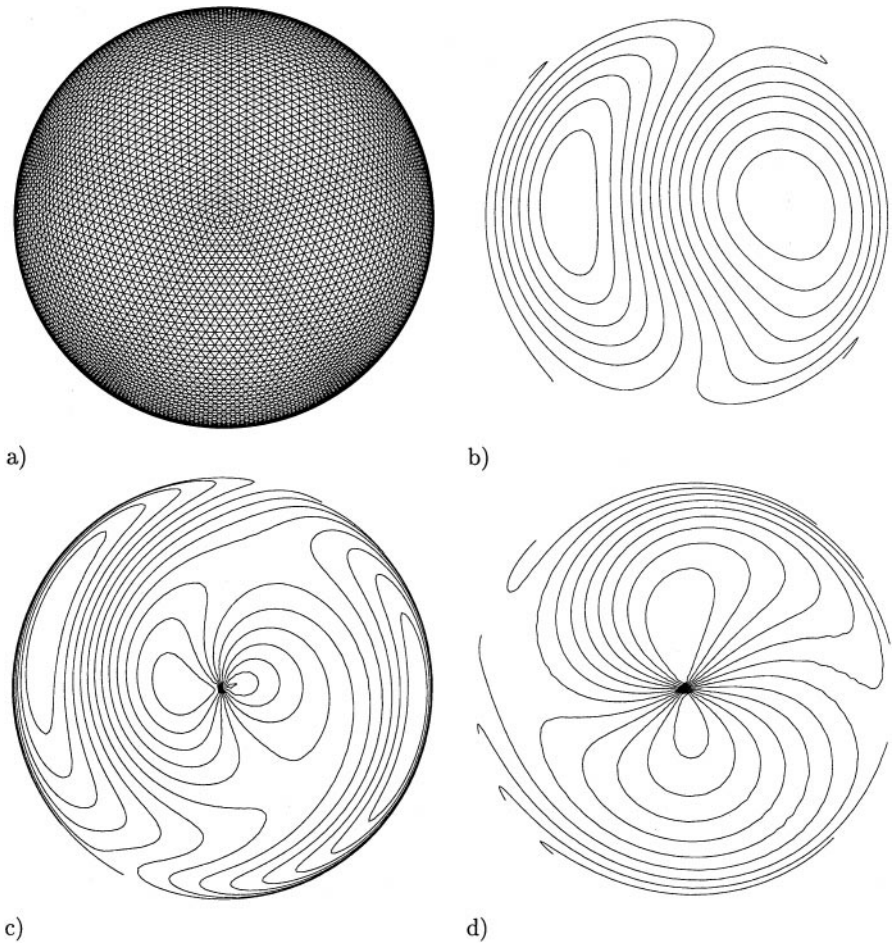


**FIG. 16.** Case 6. The (a) grid, (b) mass, (c) zonal velocity, and (d) meridional velocity on grid $p = 30$ after 5 days.

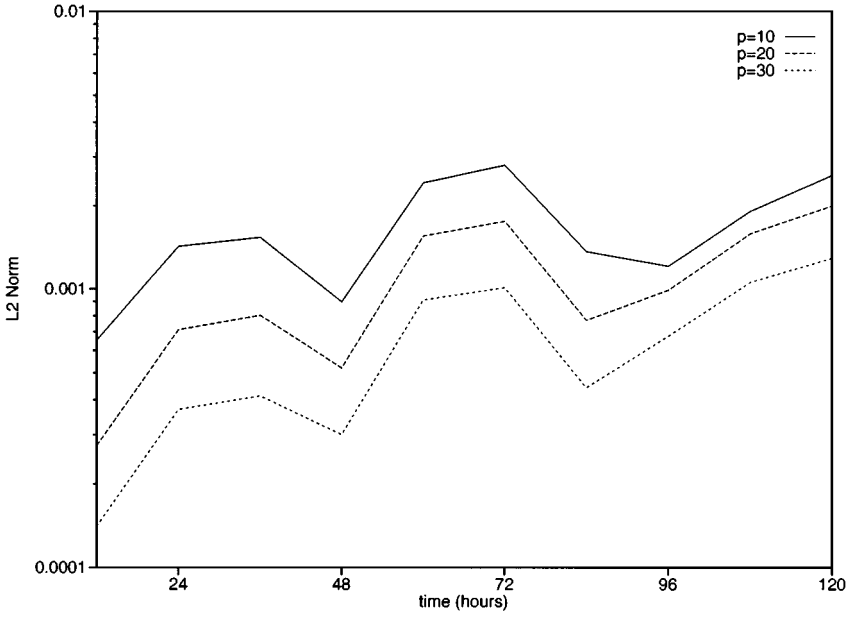


**FIG. 17.** Case 6. Time history study of the mass error for three different grids.

in time. These test cases confirmed that the Runge–Kutta trajectory calculation scheme computes accurate departure points.

Cases 3 and 4 tested the accuracy of the full nonlinear shallow water equations on steady-state problems. For case 3 the mass and momentum errors increased with time while for case 4 they remained constant with time. The results obtained for these two cases were quite good, although the convergence rate was around first order.

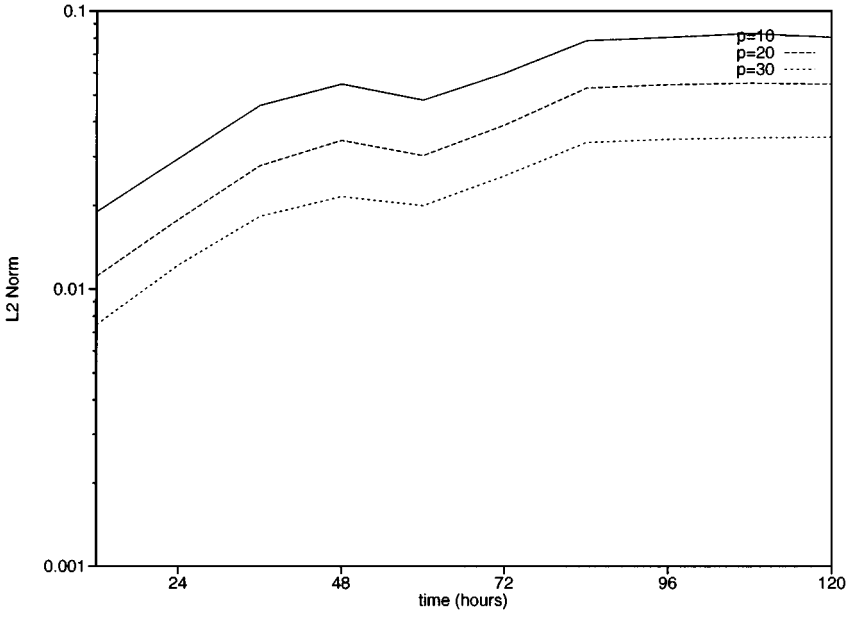


**FIG. 18.** Case 6. Time history study of the momentum error for three different grids.

Cases 5 and 6 tested the stability of the full nonlinear shallow water equations on time-dependent problems. Although these test cases do not offer analytic solutions, we can still judge the behavior of the Lagrange–Galerkin algorithm by comparing the solution to some other algorithm run on a high-resolution grid. For these two test cases, the Eulerian method described in Section 3 was used. The solutions obtained for both of these test cases were also of very good quality; case 5 converged to second order while case 6 converged to first order. Because these test cases do not have analytic solution, the results for these two test cases should only be interpreted qualitatively. Nonetheless, these results compare well against results from the literature. In addition, the algorithm conserved mass and total energy up to about 0.05% for all test cases and for all grids.

Upon considering the behavior of our algorithm on the six test cases it can be concluded that the algorithm worked correctly and quite accurately; however, improvements can always be made. The one obvious deficiency of the scheme is that the solution of the mass equation tended to be far more accurate than that of the momentum equations. Although these two sets of equations are decoupled, they are still physically linked through the particle trajectories. Therefore, in order to get accurate mass values, the momentum equations must be obtained accurately as well. There seems to be a paradox here and part of the explanation for this behavior may have to do with solving the full shallow water equations in a decoupled fashion as is done in the Lagrange–Galerkin method. However, this explanation does not seem sufficient to account for the large discrepancies in accuracy between the mass and momentum.

Recall that in order to constrain the particles to remain on the sphere a Lagrange multiplier was used. It can be only conjectured at this point that the Lagrange multiplier is restricting the convergence rate of the momentum equations. It seems that this extra forcing term adversely affects the Lagrange–Galerkin method. To test whether this term is indeed responsible for the slower rate of convergence, a convergence study of the Eulerian Runge–Kutta method, used as the reference solution in case 5 and 6, was obtained. This method also exhibited the same trend experienced by the Lagrange–Galerkin method, namely that the mass was an order of magnitude more accurate than the momentum, and the method converged to first order.

Although this inherent problem with the Cartesian form of the equations seems insurmountable, it is of little concern because the final goal of the current project is to construct a baroclinic model. These barotropic tests have been applied only to test the majority of the components that comprise the Lagrange–Galerkin scheme. For a baroclinic model, the Lagrange multiplier is no longer needed because the fluid particles are not required to remain on the surface of the sphere. Having fully three-dimensional degrees of freedom not only simplifies the Cartesian equations but should ameliorate the slow convergence exhibited by the momentum equations. In fact, going from the current barotropic model to the baroclinic version will require very few changes: most of the algorithms remain the same but the basis functions rather than being the area coordinates (triangles) will now be the volume coordinates (tetrahedra).

### 6.8. *Computational Cost*

Figures 19 and 20 summarize the computational cost of using the Lagrange–Galerkin method for various grid configurations $p$. These results were obtained using case 3 but they are generally representative of the method. These results were obtained on a DEC

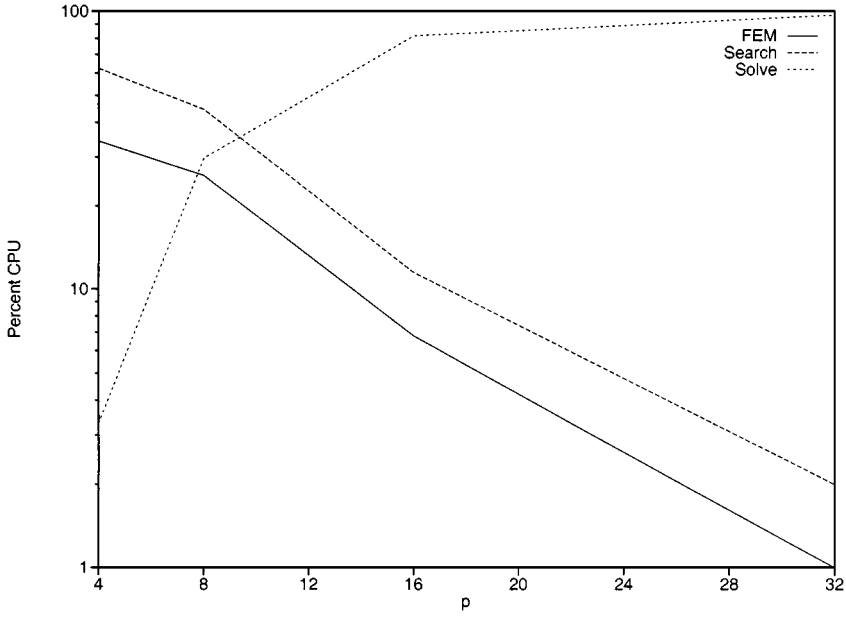


**FIG. 19.** The percentage breakdown of the computational costs incurred by the various operations of the Lagrange–Galerkin method as a function of grid parameter $p$ for case 3 after 5 days.

Alpha 8400 (EV5 CPU chip) running only one processor at a clock speed of 300 MHz. The computational cost is divided into three categories: FEM, search, and solve. FEM denotes the percent involving any operation typically associated with the finite element method. This includes the construction of the left- and right-hand side element equations as well as

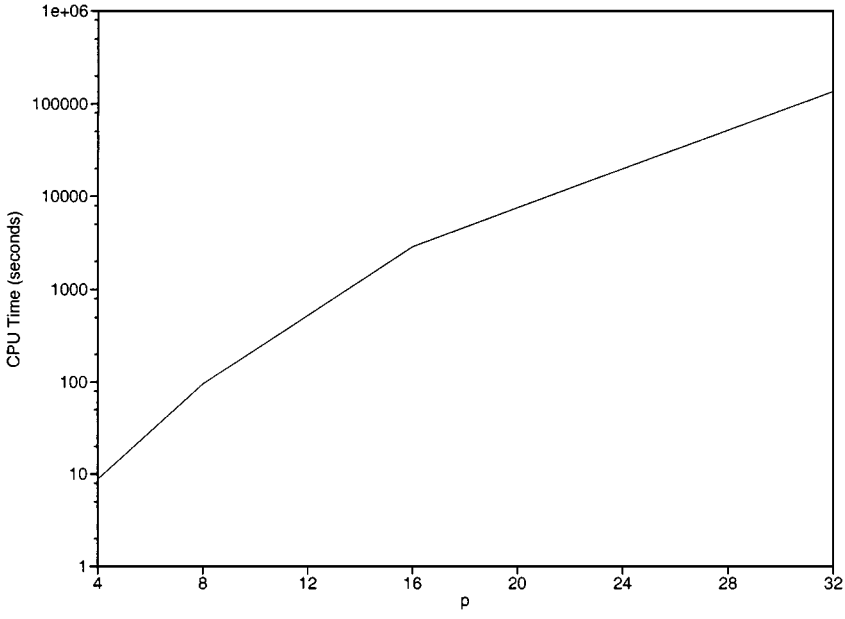


**FIG. 20.** The total computational time incurred by the Lagrange–Galerkin method as a function of grid parameter $p$ for case 3 after 5 days.

the summation of these element contributions which result in the global system of equations. Search involves the percentage of time required for the departure point calculations, searching algorithms, and testing for inclusions. Solve represents the percentage of the total time to invert the mass and momentum matrices. This includes the LU decompositions and the back substitution of these matrices.

Figure 19 shows that for small $p$, the finite element and searching algorithms dominate the CPU time. However, the cost of these operations drops exponentially with increasing $p$ as the solve operation costs increase. In fact, for large $p$ the cost of solving the equations dominates the computational cost completely. The interesting thing about these results is that if the algorithm must be made more efficient, it will only be accomplished either by constructing a more efficient matrix solver or by picking a better one. The good news is that the operations related to the Lagrangian portion of the algorithm, such as the searching of departure points, do not incur significant costs at all. The costs incurred are those which plague any implicit method, that is, the inversion of a large sparse matrix.

Many other possibilities for increasing the efficiency of the current algorithm can be explored: multigrid-type methods have proven to be the most efficient for solving linear equations, the LU solver can be streamlined by using a node renumbering scheme which would then allow the storage of the matrix in a tightly banded form, and finally a parallel implementation of the existing algorithms would remove many of the current bottlenecks. Currently we are exploring multigrid methods and parallel implementations using Open MP and MPI.

## 7. CONCLUSIONS

The results obtained for all six problems showed that the weak Lagrange–Galerkin finite element method offers a viable method for solving the shallow water equations on the sphere. Rather than solving the equations in spherical coordinates, the equations are transformed to Cartesian coordinates, which have two advantages over the equations in spherical space: the equations no longer contain singularities at the poles, and natural (area) coordinates can be constructed for the linear triangles. These area coordinates are the basis functions used to approximate the variables within the elements and are also the interpolation functions used to determine the value of the variables at the departure and quadrature points. Because these functions can be written explicitly, they lend themselves to discretizing the equations simply because the functions, derivatives, and closed form integration rules are all defined. In addition, a new icosahedral grid has been introduced which offers many more possible grid configurations over previous icosahedral grids while possessing a much simpler data structure for searching the departure points. The ideas presented in this paper are easily generalizable toward the construction of a parallel baroclinic model, which is the topic of a future paper.

## APPENDIX

### A.1. *Basis Functions*

The conservation variables belong to the space

$$(\varphi, \varphi u, \varphi v, \varphi w) \in H^1(\Omega)$$

and their test functions, likewise, are defined as

$$\psi_i^{(\varphi,\varphi u,\varphi v,\varphi w)} \in H^1(\Omega);$$

in other words, they belong to the set of square integrable functions whose first derivatives are also square integrable. These functions are, in fact, the linear triangular basis functions and are written as

$$\psi_i = \frac{a_i x + b_i y + c_i z}{\det \Delta_{i,j,k}} \tag{A.1}$$

and have the exact integration rule

$$\int_\Omega \psi_i^\alpha \psi_j^\beta \psi_k^\gamma \, d\Omega = \text{cross } \Delta_{i,j,k} \frac{\alpha!\,\beta!\,\gamma!}{(\alpha+\beta+\gamma+2)!}, \tag{A.2}$$

where

$$\begin{aligned} \text{cross } \Delta_{i,j,k} &= |(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)| = |\mathbf{N}_{i,j,k}|, \\ \det \Delta_{i,j,k} &= \mathbf{x}_i \cdot \mathbf{N}_{i,j,k}, \end{aligned}$$

and

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

The indices $i, j, k = 1, \ldots, 3$ are cycled in the following order:

$$\begin{array}{ccccc} & & k & & \\ & \swarrow & & \nwarrow & . \\ i & & \longrightarrow & & j \end{array}$$

In addition, these basis functions actually describe the natural or area coordinates on a triangle.

## A.2. *Area Coordinates*

If the basis functions $\psi_i$ represented area coordinates on the triangle $\Delta_{1,2,3}$, then these functions could be written as

$$\psi_i(\mathbf{x}_p) = \frac{\text{cross } \Delta_{p,j,k}}{\text{cross } \Delta_{i,j,k}}, \tag{A.3}$$

where $\mathbf{x}_p$ is any point on the triangle $\Delta_{1,2,3}$ and $i, j, k$ are cyclical through the three vertices of the triangle. Consider that the basis functions given by (A.1) can be written as

$$\psi_i(\mathbf{x}_p) = \frac{\det \Delta_{p,j,k}}{\det \Delta_{i,j,k}} \tag{A.4}$$

and in vector form as

$$\psi_i(\mathbf{x}_p) = \frac{\mathbf{x}_p \cdot [(\mathbf{x}_j - \mathbf{x}_p) \times (\mathbf{x}_k - \mathbf{x}_p)]}{\mathbf{x}_i \cdot [(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)]},$$

which can be written more compactly as

$$\psi_i(\mathbf{x}_p) = \frac{\mathbf{x}_p \cdot \mathbf{N}_{p,j,k}}{\mathbf{x}_i \cdot \mathbf{N}_{i,j,k}}. \tag{A.5}$$

The identity

$$\mathbf{x}_i \cdot [\mathbf{x}_j \times \mathbf{x}_k] = \mathbf{x}_i \cdot [(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)]$$

has been used in both the numerator and the denominator. Consider further that if $\mathbf{x}_p$ is on $\Delta_{1,2,3}$, then $\mathbf{N}_{p,j,k}$ and $\mathbf{N}_{i,j,k}$ will not necessarily have the same magnitudes (unless $\mathbf{x}_p = \mathbf{x}_i$) but they will have the same direction. In other words, since $\mathbf{N}_{p,j,k}$ is also normal to $\Delta_{1,2,3}$, we can also define the plane of the triangle $\Delta_{1,2,3}$ as

$$\mathbf{N}_{p,j,k} \cdot (\mathbf{x}_p - \mathbf{x}_i) = 0. \tag{A.6}$$

Substituting (A.6) into (A.5), we get

$$\psi_i(\mathbf{x}_p) = \frac{\mathbf{x}_i \cdot \mathbf{N}_{p,j,k}}{\mathbf{x}_i \cdot \mathbf{N}_{i,j,k}} = \frac{|\mathbf{x}_i||\mathbf{N}_{p,j,k}| \cos\theta_{\mathbf{x}_i:\mathbf{N}_{p,j,k}}}{|\mathbf{x}_i||\mathbf{N}_{i,j,k}| \cos\theta_{\mathbf{x}_i:\mathbf{N}_{i,j,k}}},$$

where $\theta_{\mathbf{x}:\mathbf{y}}$ denotes the angle between the vectors $\mathbf{x}$ and $\mathbf{y}$. In fact, because $\mathbf{N}_{p,j,k}$ and $\mathbf{N}_{i,j,k}$ have the same direction, $\theta_{\mathbf{x}_i:\mathbf{N}_{p,j,k}} = \theta_{\mathbf{x}_i:\mathbf{N}_{i,j,k}}$ and we get

$$\psi_i(\mathbf{x}_p) = \frac{|\mathbf{N}_{p,j,k}|}{|\mathbf{N}_{i,j,k}|} = \frac{\text{cross}\,\Delta_{p,j,k}}{\text{cross}\,\Delta_{i,j,k}}.$$

Therefore, since cross $\Delta_{i,j,k}$ is equal to twice the area of the triangle $\Delta_{i,j,k}$, the basis functions described in (A.1) are, in fact, the area coordinates for a general triangle in 3D Cartesian space.

### A.3. *Derivatives*

The approximation of any function $\varphi$ within an element is given by (14) and its derivative in the $s$ direction as

$$\frac{\partial \varphi^{(e)}}{\partial s} = \sum_{j=1}^{3} \frac{\partial \psi_j}{\partial s} \varphi_j, \tag{A.7}$$

where $s$ can be either $x$, $y$, or $z$. However, on the right-hand side of (13) we see that we need to know the derivatives of $\varphi$ at the grid points themselves for the momentum equations. One possible way of obtaining these derivatives is to consider the following: the derivative within an element can be written as done previously using the function values themselves at the grid points as given in (A.7). However, if the derivatives at the grid points themselves were known, then we could also write the derivatives within an element as

$$\frac{\partial \varphi^{(e)}}{\partial s} = \sum_{j=1}^{3} \psi_j \frac{\partial \varphi_j}{\partial s}, \tag{A.8}$$

and equating (A.7) and (A.8) in a weak sense, we arrive at

$$\int_\Omega \psi_i \psi_j \, d\Omega \frac{\partial \varphi_j}{\partial s} = \int_\Omega \psi_i \frac{\partial \psi_j}{\partial s} \varphi_j \, d\Omega, \tag{A.9}$$

where the element contributions are summed in order to form a global system which then yields the derivatives at the grid points. This derivative matrix is symmetric positive definite and can be inverted easily; however, it can also be diagonalized for the sake of efficiency. This strategy can be used to obtain second-order-accurate derivative approximations and is, in fact, quite similar to a centered finite difference scheme (see [4]). In addition, it does have a mathematical basis, as it is the exact same scheme obtained by attempting to approximate the derivatives by using the chain rule

$$d\varphi = \frac{\partial \varphi}{\partial x} dx + \frac{\partial \varphi}{\partial y} dy + \frac{\partial \varphi}{\partial z} dz,$$

where for each element, we have the contributions to the vertex $i$ from the $j$ and $k$ vertices

$$\varphi_j - \varphi_i = \frac{\partial \varphi_i}{\partial x}(x_j - x_i) + \frac{\partial \varphi_i}{\partial y}(y_j - y_i) + \frac{\partial \varphi_i}{\partial z}(z_j - z_i)$$

and

$$\varphi_k - \varphi_i = \frac{\partial \varphi_i}{\partial x}(x_k - x_i) + \frac{\partial \varphi_i}{\partial y}(y_k - y_i) + \frac{\partial \varphi_i}{\partial z}(z_k - z_i).$$

This $2 \times 3$ system can be written in the matrix form

$$\begin{bmatrix} (x_j - x_i) & (y_j - y_i) & (z_j - z_i) \\ (x_k - x_i) & (y_k - y_i) & (z_k - z_i) \end{bmatrix} \begin{bmatrix} \frac{\partial \varphi_i}{\partial x} \\ \frac{\partial \varphi_i}{\partial y} \\ \frac{\partial \varphi_i}{\partial z} \end{bmatrix} = \begin{bmatrix} \varphi_j - \varphi_i \\ \varphi_k - \varphi_i \end{bmatrix},$$

which is solved for at the element level and an area-weighted average yields the derivatives at the grid points. In other words, equation (A.9) can be thought of as the derivatives obtained by an elemental approximation to the chain rule.

### A.4. *Integration*

All of the finite element integrals can be obtained in closed form using (A.2). This can be easily accomplished for all of the left hand side integrals (those at time $n + 1$). However, for the right-hand side integrals (those at time $n$) this would require finding the intersection regions between these Lagrangian elements (those at time $n$) and the Eulerian elements comprising the grid (see [3, 5] for details). Therefore, for efficiency we have used a quintic quadrature rule for all of the right-hand side integrals.

## ACKNOWLEDGMENTS

I thank my sponsor, the Office of Naval Research, for supporting this work through program PE-0602435N. I also thank the three anonymous reviewers for their comments and suggestions.

## REFERENCES


1. G. Chukapalli, *Weather and Climate Numerical Algorithms: A Unified Approach to an Efficient, Parallel Implementation*, Ph.D. dissertation, University of Toronto (1997).
2. J. Côté, A Lagrange multiplier approach for the metric terms of semi-Lagrangian models on the sphere, *Quart. J. Roy. Meteorological Soc.* **114**, 1347 (1988).
3. F. X. Giraldo, Lagrange–Galerkin methods on spherical geodesic grids, *J. Comput. Phys.* **136**, 197 (1997).
4. F. X. Giraldo, Trajectory calculations for spherical geodesic grids in Cartesian space, *Monthly Weather Rev.* **127**, 1651 (1999).
5. F. X. Giraldo, The Lagrange–Galerkin method for the 2D shallow water equations on adaptive grids, *Internat. J. Numer. Methods Fluids*, in press.
6. 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, *Monthly Weather Rev.* **123**, 1862 (1995).
7. A. McDonald and J. R. Bates, Semi-Lagrangian integration of a gridpoint shallow water model on the sphere, *Monthly Weather Rev.* **117**, 130 (1989).
8. B. Neta, F. X. Giraldo, and I. M. Navon, Analysis of the Turkel–Zwas scheme for the 2D shallow water equations in spherical coordinates, *J. Comput. Phys.* **133**, 102 (1997).
9. A. Priestley, The Taylor–Galerkin method for the shallow-water equations on the sphere, *Monthly Weather Rev.* **120**, 3003 (1992).
10. R. J. Purser, Non-standard grids, in *Proceedings of the European Center for Medium-Range Weather Forecasting Annual Seminar, September 7–11, 1998*.
11. M. Taylor, J. Tribbia, and M. Iskandarani, The spectral element method for the shallow water equations on the sphere, *J. Comput. Phys.* **130**, 92 (1997).
12. 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).