A FULLY IMPLICIT DOMAIN DECOMPOSITION ALGORITHM FOR SHALLOW WATER EQUATIONS ON THE CUBED-SPHERE*

CHAO YANG†, JIANWEN CAO‡, AND XIAO-CHUAN CAI§

Abstract. Popular approaches for solving the shallow water equations (SWE) for climate modeling are explicit and semi-implicit methods, both have certain constraints on the time step size. In this paper, we propose and study a fully implicit method which imposes no limit on the time step size, but requires the solution of a large sparse nonlinear system of equations at every time step. The focus of the paper is a parallel, fully coupled, Newton-Krylov-RAS algorithm with a Jacobian matrix explicitly calculated on a weakly non-matching cubed-sphere mesh. Here RAS is a restricted additive Schwarz method. We show numerically that with such a preconditioned nonlinearly implicit method the time step size is no longer constrained by the CFL condition and report superlinear speedup of the algorithm on machines with thousands of processors, and for problems with smooth and non-smooth solutions.

Key words. Shallow water equations, cubed-sphere, fully implicit method, domain decomposition, Newton-Krylov-Schwarz, parallel scalability

1. Introduction. Computer simulations based on the shallow water equations (SWE) play a very important role in many areas of science and engineering including the study of the earth's climate, the understanding of flood, etc. SWE is difficult to solve because its solution admits waves traveling at vastly different speeds. To obtain high resolution numerical solutions, supercomputers with large number of processors are necessary. When designing algorithms for solving SWE on massively parallel computers, the two key factors are (1) the robustness, which means the convergence is achieved for a wide range of physical parameters, wave speeds, mesh and time step sizes; and (2) the parallel scalability, which means as the number of processors increases the overall computing time decreases proportionally.

Popular approaches for solving the shallow water equations for climate modeling are explicit and semi-implicit methods. In semi-implicit schemes the linear terms responsible for the gravity wave are treated implicitly. These schemes allow a larger time step than explicit methods since the stability no longer depends on the gravity waves. When Eulerian methods are used, these semi-implicit schemes are limited by a Courant-Friedrichs-Lewy (CFL) condition based on the local advection speed. When very fine spacial meshes are used in the simulation, the time step size has to be very small in order to satisfy the CFL condition. On the other hand, when using fully implicit schemes, the CFL requirement can be completely relaxed. The scheme is unconditionally stable and the time step selection is based only on the desired solution accuracy [29]. The price to pay for using a fully implicit method is that a large sparse nonlinear system of equations has to be solved at every time step. Some comparisons between fully implicit method and other time integration schemes for other applications can be found in, *e.g.*, [28, 32].

Each of the three techniques (explicit, semi-implicit, and fully implicit) has some

*The research was supported in part by DOE under DE-FC02-04ER25595, and in part by NSF under grants ACI-0305666, CNS-0420873, CCF-0634894, and CNS-0722023.

†Institute of Software, Chinese Academy of Sciences, Beijing 100190, P. R. China (yang@mail.rdcps.ac.cn)

‡Institute of Software, Chinese Academy of Sciences, Beijing 100190, P. R. China (caojianwen@gmail.com)

§Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (cai@cs.colorado.edu)

advantages and disadvantages and in an ideal simulation framework one may want to switch between the three techniques on the fly depending on if a really small time step is needed in order to capture some critical changes of the solution in time or a large time step is needed in order to study long time behavior of the solution which may not be sensitive to small scale changes of the solution. In certain situations, such as commercial weather forecast, large time steps have to be used so that the simulation can be completed by a given time and the accuracy is a secondary issue. In this paper, we do not address the issue of when to use which one of the techniques and how to switch between them. There are many publications devoted to explicit (*e.g.*, [21]) and semi-implicit (*e.g.*, [8, 42]) approaches for SWE, we focus only on the less well-understood fully implicit method, in particular, we study a parallel, fully coupled, Newton-Krylov-Schwarz algorithm with a Jacobian matrix calculated on a weakly non-matching cubed-sphere mesh for solving such sparse nonlinear system of equations. More precisely speaking, we first apply a third-order implicit time integration scheme, and then, to guarantee the nonlinear consistency, we use a Newton-Krylov-Schwarz algorithm to solve the large sparse nonlinear system of algebraic equations containing all physical variables at every time step. In a Newton-Krylov algorithm, a system is solved by applying outer Newton iterations, whose Jacobian systems are solved with a preconditioned Krylov subspace method. The success of the overall approach depends heavily on the preconditioner. SWE has three components and is mostly hyperbolic. Algebraic preconditioners, such as ILU and AMG, often do not work well for coupled systems that are not strongly elliptic. Multigrid preconditioners work well if the problem is diffusion dominant, and for non-diffusion dominant problem, some type of splitting (such as physics-based splitting, or Schur complement based splitting) may be necessary to single out the diffusion components [3, 39]. We use an overlapping restricted additive Schwarz preconditioner which does not require any splitting of the multi-component system. The restricted version of Schwarz methods requires much less communication than the classical additive Schwarz method, and is therefore more efficient on machines with a large number of processors.

We would like to point out that fully implicit approaches have recently been successfully applied to several classes of important applications in multi-physics simulations [3, 17, 18, 22, 23, 35, 40, 46], but very little work has been done for SWE on the cubed-sphere, as far as we know. Schwarz type preconditioning techniques were used by a number of authors [10, 12, 13, 14, 30, 31, 48] for solving the elliptic (Helmholtz type operators) component of SWE, and excellent scalability results were reported for large scale calculations on computers with many processors. In [37], a Schwarz type domain decomposition method was used, without the Krylov acceleration, for solving the Poisson equation in SWE discretized on a Yin-Yang grid. Our work is the first attempt to use overlapping Schwarz methods for the whole system without splitting out the elliptic-like subsystem. The whole system approach has the potential to allow more physics to be added easily to the system without changing much of the algorithmic and software framework.

For the purpose of comparison, we also implemented an explicit algorithm and we used it to compare the results obtained with the implicit approach. Both implementations are based on the PETSc (Portable Extensible Toolkit for Scientific computation [2]) library thus providing a convenient test bed for investigations of parallel properties of the algorithms. We report scalability studies on fine meshes and on machines with thousands of processors.

The remainder of this paper is organized as follows. In Section 2, we discuss the

model SWE problem and provide some useful definitions. The spatial discretization on the cubed-sphere is covered in Section 3. The details of algorithms are described in Sections 4, and numerical results are reported in Section 5. The paper is concluded in Section 6.

2. Shallow water equations. The shallow water equations on a rotating sphere can be written in the following flux form ([52]):

$$\begin{cases} \dfrac{\partial h}{\partial t} + \nabla \cdot (h\mathbf{v}) = 0, \\ \dfrac{\partial (h\mathbf{v})}{\partial t} + \nabla \cdot (h\mathbf{v}\mathbf{v}) = \Psi_C + \Psi_G, \end{cases} \tag{2.1}$$

where h is the depth of the fluid, $\mathbf{v}$ is the velocity, Ψ_C and Ψ_G are the source terms due to the Coriolis force and the gravity force, respectively. The Coriolis force term can be written as

$$\Psi_C = -fh(\hat{\mathbf{k}} \times \mathbf{v}),$$

where $\hat{\mathbf{k}}$ is the outward unit normal vector on the sphere and $f = 2\omega \sin\theta$ is the Coriolis parameter with the angular velocity ω. Throughout the paper, the spherical coordinates of a point on the sphere is denoted as (λ, θ), $\lambda \in [-\pi, \pi)$, $\theta \in [-\pi/2, \pi/2]$. The source term due to the gravity force can be written as

$$\Psi_G = -gh\nabla(h + h_s),$$

where g is the gravitational constant and h_s is the height of the spherical surface describing a variable bottom topography (*e.g.*, mountains).

When dealing with non-orthogonal curvilinear meshes, such as the cubed-sphere mesh [45], we denote (ξ, η) as the curvilinear coordinates, and rewrite (2.1) as

$$\frac{\partial Q}{\partial t} + \mathcal{A}(Q) = 0, \tag{2.2}$$

with $Q = (q_1, q_2, q_3)^T = (h, hu, hv)^T$ and $\mathcal{A}(Q) = (\mathcal{A}_1(Q), \mathcal{A}_2(Q), \mathcal{A}_3(Q))^T$, where

$$\mathcal{A}_1(Q) = \frac{1}{\Lambda}\left[\frac{\partial}{\partial \xi}(\Lambda q_2) + \frac{\partial}{\partial \eta}(\Lambda q_3)\right], \tag{2.3}$$

$$\begin{aligned} \mathcal{A}_2(Q) =& \frac{1}{\Lambda}\left\{\frac{\partial}{\partial \xi}\left[\Lambda\left(\frac{q_2 q_2}{q_1} + \frac{1}{2}g g^{11} q_1 q_1\right)\right] + \frac{\partial}{\partial \eta}\left[\Lambda\left(\frac{q_2 q_3}{q_1} + \frac{1}{2}g g^{12} q_1 q_1\right)\right]\right\} \\ &+ \left[\Gamma^1_{11}\left(\frac{q_2 q_2}{q_1} + \frac{1}{2}g g^{11} q_1 q_1\right) + 2\Gamma^1_{12}\left(\frac{q_2 q_3}{q_1} + \frac{1}{2}g g^{12} q_1 q_1\right) + \Gamma^1_{22}\left(\frac{q_3 q_3}{q_1} + \frac{1}{2}g g^{22} q_1 q_1\right)\right] \\ &+ f\Lambda\left(g^{12} q_2 - g^{11} q_3\right) + g\left(g^{11}\frac{\partial h_s}{\partial \xi} + g^{12}\frac{\partial h_s}{\partial \eta}\right) q_1, \end{aligned} \tag{2.4}$$

$$\begin{aligned} \mathcal{A}_3(Q) =& \frac{1}{\Lambda}\left\{\frac{\partial}{\partial \xi}\left[\Lambda\left(\frac{q_2 q_3}{q_1} + \frac{1}{2}g g^{12} q_1 q_1\right)\right] + \frac{\partial}{\partial \eta}\left[\Lambda\left(\frac{q_3 q_3}{q_1} + \frac{1}{2}g g^{22} q_1 q_1\right)\right]\right\} \\ &+ \left[\Gamma^2_{11}\left(\frac{q_2 q_2}{q_1} + \frac{1}{2}g g^{11} q_1 q_1\right) + 2\Gamma^2_{12}\left(\frac{q_2 q_3}{q_1} + \frac{1}{2}g g^{12} q_1 q_1\right) + \Gamma^2_{22}\left(\frac{q_3 q_3}{q_1} + \frac{1}{2}g g^{22} q_1 q_1\right)\right] \\ &+ f\Lambda\left(g^{22} q_2 - g^{12} q_3\right) + g\left(g^{12}\frac{\partial h_s}{\partial \xi} + g^{22}\frac{\partial h_s}{\partial \eta}\right) q_1, \end{aligned} \tag{2.5}$$

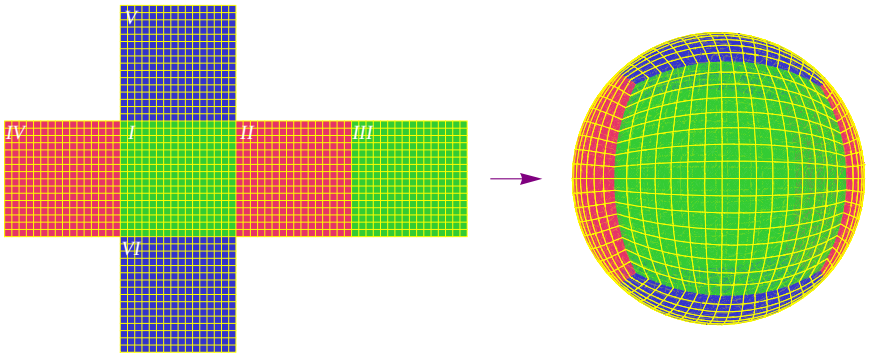

FIG. 2.1. *The cubed-sphere mesh can be obtained by mapping the six faces of an inscribed cube covered with uniform meshes to the sphere surface.*

where g^{ij} is the ij-th component of the inverse metric tensor for the curvilinear coordinates, Λ is the square root of the determinant of the metric tensor and Γ^m_{ij} is the mij-th Christoffel symbol.

The cubed-sphere mesh was originally introduced in [45] and further developed in [41]. Using the cubed-sphere mesh, the spherical surface is divided into six identical patches. Each patch is assigned one local curvilinear coordinates system via a mapping from the surface of an inscribed cube to the sphere, see Fig. 2.1. There are mainly two different mappings to define a cubed-sphere mesh, the original gnomonic mapping [45] and the conformal mapping [38] (see also in [1] for a modified version). Some other mappings can be found in [36] and references therein. In the present study we use the gnomonic cubed-sphere mesh which is highly non-orthogonal but more uniform than the conformal one. The local gnomonic coordinates are defined as angles $(\xi, \eta) \in [-\pi/4, \pi/4]^2$. More details on gnomonic cubed-sphere can be found in [9, 30, 41]. For all six patches, the corresponding inverse metric tensors are equal and can be written as

$$\mathcal{G}^{-1} = \begin{pmatrix} g^{11} & g^{12} \\ g^{12} & g^{22} \end{pmatrix} = \frac{1 + \tan^2\xi + \tan^2\eta}{(a \sec\xi \sec\eta)^2} \begin{pmatrix} \sec^2\eta & \tan\xi\tan\eta \\ \tan\xi\tan\eta & \sec^2\xi \end{pmatrix},$$

and thus $\Lambda = \sqrt{\det \mathcal{G}} = (a \sec\xi \sec\eta)^2 / \sqrt{(1 + \tan^2\xi + \tan^2\eta)^3}$, where a is the radius of the sphere. The Christoffel symbols are

$$\Gamma^1_{11} = \frac{2\tan\xi\tan^2\eta}{1 + \tan^2\xi + \tan^2\eta}, \quad \Gamma^1_{12} = -\frac{\tan\eta\sec^2\eta}{1 + \tan^2\xi + \tan^2\eta}, \quad \Gamma^1_{22} = 0,$$
$$\Gamma^2_{22} = \frac{2\tan^2\xi\tan\eta}{1 + \tan^2\xi + \tan^2\eta}, \quad \Gamma^2_{12} = -\frac{\tan\xi\sec^2\xi}{1 + \tan^2\xi + \tan^2\eta}, \quad \Gamma^2_{11} = 0.$$

3. Discretization of SWE on the cubed-sphere. Special considerations are necessary in order to discretize a differential equation on the cubed-sphere. Several schemes are available for scalar transport equations [10, 26, 30, 36]. For the multi-component SWE, there are also a few schemes such as the finite volume method [9, 43], the spectral element method [13, 49], and the discontinuous Galerkin method [12, 14, 31], among others. Since our interests are mainly on the parallel solvers we decide to use a relatively simple finite volume method based on Osher's scheme [33, 34]. This method is similar to that used in [25], which is specially designed for a reduced latitude-longitude mesh and a stereographic mesh, with a modified κ scheme

to determine the constant states. Most other discretizations can be applied to our parallel solvers.

3.1. Spatial discretization. Suppose each of the six patches of the cubed-sphere is covered by a logically rectangular $N\times N$ mesh, which is equally spaced in the computational domain $\{(\xi,\eta)\in[-\pi/4,\pi/4]^2\}$ with mesh size $\hbar=\pi/2N$. Denote the center point of the (i,j)-th mesh cell on patch k as P_{ij}^k, $i,j=1,\cdots,N$, $k=1,\cdots,6$. We use a fixed time step Δt for the implicit method. At the n-th time step, we denote the values of any function $f(\xi,\eta,k,t)$ at point P_{ij}^k as $f_{ij}^{k,(n)}$. Sometimes we ignore the superscript k or (n) or both of them if no misunderstanding is introduced.

Let the discretization of the nonlinear spatial operator $\mathcal{A}$ in (2.2)-(2.5) at a point P_{ij} be $A_{ij}=((A_1)_{ij},(A_2)_{ij},(A_3)_{ij})^T$ which is defined as

$$(A_1)_{ij}=\frac{1}{\Lambda_{ij}}\left[\delta_\xi(\Lambda\tilde{q}_2)_{ij}+\delta_\eta(\Lambda\tilde{q}_3)_{ij}\right], \tag{3.1}$$

$$\begin{aligned}(A_2)_{ij}=&\frac{1}{\Lambda_{ij}}\left\{\delta_\xi\left[\Lambda(\frac{\tilde{q}_2\tilde{q}_2}{\tilde{q}_1}+\frac{1}{2}gg^{11}\tilde{q}_1\tilde{q}_1)\right]_{ij}+\delta_\eta\left[\Lambda(\frac{\tilde{q}_2\tilde{q}_3}{\tilde{q}_1}+\frac{1}{2}gg^{12}\tilde{q}_1\tilde{q}_1)\right]_{ij}\right\}\\&+\left[\Gamma_{11}^1(\frac{q_2q_2}{q_1}+\frac{1}{2}gg^{11}q_1q_1)+2\Gamma_{12}^1(\frac{q_2q_3}{q_1}+\frac{1}{2}gg^{12}q_1q_1)+\Gamma_{22}^1(\frac{q_3q_3}{q_1}+\frac{1}{2}gg^{22}q_1q_1)\right]_{ij}\\&+\left[f\Lambda\left(g^{12}q_2-g^{11}q_3\right)\right]_{ij}+g\left[(g^{11})_{ij}\delta_\xi(h_s)_{ij}+(g^{12})_{ij}\delta_\eta(h_s)_{ij}\right](q_1)_{ij},\end{aligned} \tag{3.2}$$

$$\begin{aligned}(A_3)_{ij}=&\frac{1}{\Lambda_{ij}}\left\{\delta_\xi\left[\Lambda(\frac{\tilde{q}_2\tilde{q}_3}{\tilde{q}_1}+\frac{1}{2}gg^{12}\tilde{q}_1\tilde{q}_1)\right]_{ij}+\delta_\eta\left[\Lambda(\frac{\tilde{q}_3\tilde{q}_3}{\tilde{q}_1}+\frac{1}{2}gg^{22}\tilde{q}_1\tilde{q}_1)\right]_{ij}\right\}\\&+\left[\Gamma_{11}^2(\frac{q_2q_2}{q_1}+\frac{1}{2}gg^{11}q_1q_1)+2\Gamma_{12}^2(\frac{q_2q_3}{q_1}+\frac{1}{2}gg^{12}q_1q_1)+\Gamma_{22}^2(\frac{q_3q_3}{q_1}+\frac{1}{2}gg^{22}q_1q_1)\right]_{ij}\\&+\left[f\Lambda\left(g^{22}q_2-g^{12}q_3\right)\right]_{ij}+g\left[(g^{12})_{ij}\delta_\xi(h_s)_{ij}+(g^{22})_{ij}\delta_\eta(h_s)_{ij}\right](q_1)_{ij},\end{aligned} \tag{3.3}$$

where $\delta_\xi(\cdot)_{ij}$ and $\delta_\eta(\cdot)_{ij}$ are the standard second-order central difference operators with respect to ξ and η directions respectively

$$\delta_\xi(\cdot)_{ij}=\frac{(\cdot)_{i+\frac{1}{2},j}-(\cdot)_{i-\frac{1}{2},j}}{\hbar},\quad \delta_\eta(\cdot)_{ij}=\frac{(\cdot)_{i,j+\frac{1}{2}}-(\cdot)_{i,j-\frac{1}{2}}}{\hbar}.$$

The cell boundary value $\tilde{Q}=(\tilde{q}_1,\tilde{q}_2,\tilde{q}_3)^T$ is the properly interpolated value of Q and defined on the four cell boundaries as $\tilde{Q}_{i+\frac{1}{2},j},\tilde{Q}_{i-\frac{1}{2},j},\tilde{Q}_{i,j+\frac{1}{2}}$, and $\tilde{Q}_{i,j-\frac{1}{2}}$. On each cell boundary $\tilde{Q}$ is connected to Q via the "left" and "right" constant states Q^L and Q^R. Throughout this paper, we use Osher's method [33, 34] to obtain $\tilde{Q}$. On the left or right cell boundary, $\tilde{Q}$ is defined through

$$\begin{aligned}\tilde{q}_1(Q^L,Q^R)=&\frac{1}{4gg^{11}}\left[\frac{1}{2}\left(\frac{q_2^L}{q_1^L}-\frac{q_2^R}{q_1^R}\right)+\sqrt{gg^{11}q_1^L}+\sqrt{gg^{11}q_1^R}\right]^2,\\\frac{\tilde{q}_2(Q^L,Q^R)}{\tilde{q}_1(Q^L,Q^R)}=&\frac{1}{2}\left(\frac{q_2^L}{q_1^L}+\frac{q_2^R}{q_1^R}\right)+\sqrt{gg^{11}q_1^L}-\sqrt{gg^{11}q_1^R},\\\frac{\tilde{q}_3(Q^L,Q^R)}{\tilde{q}_1(Q^L,Q^R)}=&\begin{cases}\frac{q_3^L}{q_1^L}+\frac{g^{12}}{g^{11}}\left[\frac{1}{2}\left(\frac{q_2^R}{q_1^R}-\frac{q_2^L}{q_1^L}\right)+\sqrt{gg^{11}q_1^L}-\sqrt{gg^{11}q_1^R}\right], & \text{if } \tilde{q}_2\geq 0\\\frac{q_3^R}{q_1^R}+\frac{g^{12}}{g^{11}}\left[\frac{1}{2}\left(\frac{q_2^L}{q_1^L}-\frac{q_2^R}{q_1^R}\right)+\sqrt{gg^{11}q_1^L}-\sqrt{gg^{11}q_1^R}\right], & \text{otherwise.}\end{cases}\end{aligned} \tag{3.4}$$

While on the bottom and top boundaries, the subscripts 2 and 3 in the definition of $\tilde{Q}$ in (3.4) should be swapped in the following way

$$\begin{aligned}
\tilde{q}_1(Q^L,Q^R) =& \frac{1}{4gg^{22}}\left[\frac{1}{2}\left(\frac{q_3^L}{q_1^L}-\frac{q_3^R}{q_1^R}\right)+\sqrt{gg^{22}q_1^L}+\sqrt{gg^{22}q_1^R}\right]^2,\\
\frac{\tilde{q}_3(Q^L,Q^R)}{\tilde{q}_1(Q^L,Q^R)} =& \frac{1}{2}\left(\frac{q_3^L}{q_1^L}+\frac{q_3^R}{q_1^R}\right)+\sqrt{gg^{22}q_1^L}-\sqrt{gg^{22}q_1^R},\\
\frac{\tilde{q}_2(Q^L,Q^R)}{\tilde{q}_1(Q^L,Q^R)} =& \begin{cases}\frac{q_2^L}{q_1^L}+\frac{g^{12}}{g^{22}}\left[\frac{1}{2}\left(\frac{q_3^R}{q_1^R}-\frac{q_3^L}{q_1^L}\right)+\sqrt{gg^{22}q_1^L}-\sqrt{gg^{22}q_1^R}\right], & \text{if } \tilde{q}_3\geq 0\\ \frac{q_2^R}{q_1^R}+\frac{g^{12}}{g^{22}}\left[\frac{1}{2}\left(\frac{q_3^L}{q_1^L}-\frac{q_3^R}{q_1^R}\right)+\sqrt{gg^{22}q_1^L}-\sqrt{gg^{22}q_1^R}\right], & \text{otherwise.}\end{cases}
\end{aligned} \tag{3.5}$$

It should be noticed that (3.4) and (3.5) are defined under the assumptions $|q_2/q_1| < \sqrt{gg^{11}q_1}$ and $|q_3/q_1| < \sqrt{gg^{22}q_1}$, respectively. If these conditions are not satisfied, the definition of $\tilde{Q}$ should be changed accordingly (refer to [25] for more details) to avoid negative $\tilde{q}_1$ which is nonphysical. For the reconstruction of the left and right constant states Q^L and Q^R the following two approaches are implemented in our code.

1. Centered method (second order):

$$\begin{aligned}
Q^L_{i+\frac{1}{2},j} &= Q^R_{i+\frac{1}{2},j} = \tfrac{1}{2}\left(Q_{ij}+Q_{i+\frac{1}{2},j}\right),\\
Q^L_{i-\frac{1}{2},j} &= Q^R_{i-\frac{1}{2},j} = \tfrac{1}{2}\left(Q_{ij}+Q_{i-\frac{1}{2},j}\right),\\
Q^L_{i,j+\frac{1}{2}} &= Q^R_{i,j+\frac{1}{2}} = \tfrac{1}{2}\left(Q_{ij}+Q_{i,j+\frac{1}{2}}\right),\\
Q^L_{i,j-\frac{1}{2}} &= Q^R_{i,j-\frac{1}{2}} = \tfrac{1}{2}\left(Q_{ij}+Q_{i,j-\frac{1}{2}}\right).
\end{aligned} \tag{3.6}$$

2. One-sided method (first order):

$$\begin{aligned}
Q^L_{i+\frac{1}{2},j} &= Q_{ij}, & Q^R_{i+\frac{1}{2},j} &= Q_{i+1,j},\\
Q^L_{i-\frac{1}{2},j} &= Q_{i-1,j}, & Q^R_{i-\frac{1}{2},j} &= Q_{ij},\\
Q^L_{i,j+\frac{1}{2}} &= Q_{ij}, & Q^R_{i,j+\frac{1}{2}} &= Q_{i,j+1},\\
Q^L_{i,j-\frac{1}{2}} &= Q_{i,j-1}, & Q^R_{i,j-\frac{1}{2}} &= Q_{ij}.
\end{aligned} \tag{3.7}$$

For the centered method (3.6), we always have $\tilde{Q} = Q^L = Q^R$ on each cell boundary and this scheme is used when the solution is smooth, such as the steady-state zonal flow problem in Section 5. The one-sided method is utilized to handle possible sharp gradients in the solution, such as the dam-break case to be presented later. In general, one can obtain better accuracy by using more sophisticated methods such as the $\kappa = 1/3$ scheme ([51], see also [20]) together with a TVD limiter.

3.2. Patch interface conditions. Each patch of the cubed-sphere mesh is covered by a logically rectangular mesh and can be treated separately. Values of a function should be correctly passed between patches so that the whole mesh can be coupled together properly. For that purpose, ghost cells extended from the mesh boundaries are used on each patch. One method for passing information is directly copying the solution values from the mesh cells in the neighboring patch to the ghosted cells. This method is cheap but not accurate, especially for the case with a vector-valued field such as the velocity.

Another patch interface condition is based on certain interpolation (see [41]). An useful observation is that the center points of ghost cells lie on the same line as the

center points of the mesh cells in the neighboring patch. So only a one-dimensional interpolation is needed. In the paper, we use a linear interpolation to obtain the ghost values, as illustrated in Fig. 3.1. It should be noticed that for a vector-valued field, one should first transform the values into the spherical coordinates form or the cartesian coordinates form before performing the interpolation and then transform them back into the cubed-sphere coordinates form, because the coordinate bases are not the same from one patch to another on the cubed-sphere.

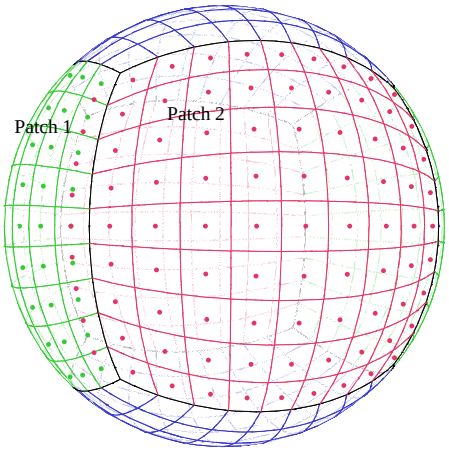

FIG. 3.1. *Patch interface treatment for the cubed-sphere. Patch 1 is a neighbor patch of patch 2. All interior mesh points together with one layer of ghost points located in Patch 2 are drawn in red. Several layers of interior mesh points of Patch 1 is drawn in green. One can see that the ghost points lie on the same line as the nearest layer of mesh points in the neighboring patch.*

Whether for scalar-valued or vector-valued functions, *e.g.*, for $Q = (h, hu, hv)^T$ in SWE, the patch interface interpolation depends only on the cubed-sphere geometry. For example, see Fig. 3.1, if we want to obtain one layer of the ghost values defined at points $\{\bar{P}_j\}_{j=1}^N$ belonging to patch 2 interpolated from the values defined at points $\{P_j\}_{j=1}^N$ in patch 1, the interpolation takes the form

$$\bar{h}_j = \alpha_j h_j + (1-\alpha_j) h_{\tilde{j}} \tag{3.8}$$

$$\begin{pmatrix} \bar{u}_j \\ \bar{v}_j \end{pmatrix} = (J^2)_j^{-1} \left[\alpha_j (J^1)_j \begin{pmatrix} u_j \\ v_j \end{pmatrix} + (1-\alpha_j)(J^1)_{\tilde{j}} \begin{pmatrix} u_{\tilde{j}} \\ v_{\tilde{j}} \end{pmatrix} \right]. \tag{3.9}$$

Here for simplicity, we denote $f(P_j)$ by f_j and $f(\overline{P}_j)$ by $\overline{f}_j$ for any function f. In (3.8)-(3.9), $\tilde{j}$ is the neighboring index of j (along the first layer of ghost points) and can be obtained by

$$\tilde{j} = \begin{cases} j+1, & j < \lfloor \frac{N}{2} \rfloor + 1, \\ j-1, & \text{otherwise}, \end{cases}$$

α_j is the linear interpolation coefficient determined by $\alpha_j = \text{dist}(\bar{P}_j, P_{\tilde{j}})/\text{dist}(P_j, P_{\tilde{j}})$, and J^k serves as the Jacobian that maps a point on patch k from its cubed-sphere coordinates to the corresponding latitude-longitude coordinates. For example, J^1 is

given by

$$J^1 = J^1(\xi^1, \eta^1) = \begin{pmatrix} \frac{\partial \lambda}{\partial \xi^1} & \frac{\partial \lambda}{\partial \eta^1} \\ \frac{\partial \theta}{\partial \xi^1} & \frac{\partial \theta}{\partial \eta^1} \end{pmatrix},$$

where (ξ^1, η^1) is the cubed-sphere coordinates on patch 1. In summary the patch interface interpolation can be written as

$$\overline{\begin{pmatrix} q_1 \\ q_2/q_1 \\ q_3/q_1 \end{pmatrix}}_j = T_j^{(1)} \begin{pmatrix} q_1 \\ q_2/q_1 \\ q_3/q_1 \end{pmatrix}_j + T_j^{(2)} \begin{pmatrix} q_1 \\ q_2/q_1 \\ q_3/q_1 \end{pmatrix}_{\tilde{j}}, \tag{3.10}$$

where

$$T_j^{(1)} = \alpha_j \begin{pmatrix} 1 & 0 \\ 0 & (J^2)_j^{-1}(J^1)_j \end{pmatrix}, \quad T_j^{(2)} = (1-\alpha_j) \begin{pmatrix} 1 & 0 \\ 0 & (J^2)_j^{-1}(J^1)_{\tilde{j}} \end{pmatrix}. \tag{3.11}$$

Since the mesh geometry is fixed, the interpolation coefficients $T_j^{(1)}$ and $T_j^{(2)}$ are computed at the beginning of the calculation and stored for repeated use at later time steps.

4. A parallel fully implicit domain decomposition algorithm. Our temporal discretization is based on the following backward differentiation formulas of orders 1, 2 and 3 (BDF-1,2,3):

$$G^{(m+1)} = \frac{1}{\Delta t}\left(Q^{(m+1)} - Q^{(m)}\right) + A^{(m+1)}, \tag{4.1}$$

$$G^{(m+1)} = \frac{1}{2\Delta t}\left(3Q^{(m+1)} - 4Q^{(m)} + Q^{(m-1)}\right) + A^{(m+1)}, \tag{4.2}$$

$$G^{(m+1)} = \frac{1}{6\Delta t}\left(11Q^{(m+1)} - 18Q^{(m)} + 9Q^{(m-1)} - 2Q^{(m-2)}\right) + A^{(m+1)}, \tag{4.3}$$

with G being the nonlinear residual of the discretized SWE. After enough initial solutions become available, the calculation uses the third order scheme (4.3). At each time step, we need to solve a system of nonlinear algebraic equations, which is obtained by putting the finite difference equations (4.3) in a certain order. For some algorithms, the orderings of the unknowns and the finite difference equations are not important, but for our algorithm the ordering is important. We use a point-wise ordering, in other words, all unknowns associated with a mesh point stay together. More precisely, we define

$$X = ((q_1)_{11}^1, (q_2)_{11}^1, (q_3)_{11}^1, (q_1)_{21}^1, (q_2)_{21}^1, (q_3)_{21}^1, \cdots)^T.$$

The non-zero structure of the Jacobian matrix would look very different if a field-wise ordering is used; *i.e.*, $X = ((q_1)_{11}^1, (q_1)_{21}^1, (q_1)_{31}^1, \cdots, (q_2)_{11}^1, (q_2)_{21}^1, (q_2)_{31}^1, \cdots)^T$. Similarly, we use a point-wise ordering for G as

$$G = ((G_1)_{11}^1, (G_2)_{11}^1, (G_3)_{11}^1, (G_1)_{21}^1, (G_2)_{21}^1, (G_3)_{21}^1, \cdots)^T.$$

The point-wise ordering is important because it helps in keeping the coupling in each 3×3 block, which is essential for our fully coupled implicit method.

The system

$$G(X) = 0$$

is solved with a one-level Newton-Krylov-Schwarz (NKS) [5, 6], which is briefly described here. Let the initial guess X_0 be the solution of the previous time step, and X_n be the current approximate solution, we find the next solution X_{n+1} as

$$X_{n+1} = X_n + \lambda_n S_n, \quad n = 0, 1, ... \tag{4.4}$$

where λ_n is the steplength and S_n is the search direction. The search direction S_n is obtained by solving the Jacobian system approximately using a Krylov subspace iterative method

$$\|M_n^{-1}[G'(X_n)S_n + G(X_n)]\| \leq \max\{\zeta_r \|M_n^{-1}G(X_n)\|, \zeta_a\},$$

where M_n^{-1} is an additive Schwarz type preconditioner to be defined shortly. The accuracy of the Jacobian solver is determined by the two linear tolerances $\zeta_r, \zeta_a \geq 0$. The steplength λ_n is determined by a backtracking linesearch procedure (see, *e.g.*, section 6.3 in [15]). In practice, the algorithm is insensitive to the details of the method used to determine λ_n. The stopping condition for the nonlinear iteration (4.4) is

$$||G(X_{n+1})|| \leq \max\{\varepsilon_r ||G(X_0)||, \varepsilon_a\},$$

where $\varepsilon_r, \varepsilon_a \geq 0$ are nonlinear tolerances.

Newton methods can be implemented with or without the Jacobian matrix. [24] provides a framework for implementing Newton methods without using the explicit form of the Jacobian. In this paper, we do not take the matrix-free approach since the explicit form of the Jacobian is required to construct the preconditioner. Because of the simplicity of our discretization scheme, we are able to calculate the Jacobian matrix explicitly. A few techniques are available when an explicit Jacobian calculation is not possible (or not preferred), such as Automatic Differentiation [19], or multi-colored finite difference methods (MC-FD) [11]. These methods offer some kind of approximations of the Jacobian. Later in the paper, we present a performance comparison of the explicitly calculated Jacobian and the MC-FD calculated Jacobian.

Here we show briefly how the Jacobian matrix is calculated. The Jacobian can be written as $G' = \gamma I + \partial A/\partial Q$, where γ is a constant from the temporal discretizations (4.1)-(4.3). The contribution from the spatial discretization is $\partial A/\partial Q$. Suppose a simple five point spatial discretization is used for SWE. Then we have $A_{ij} = A(Q_{ij}, \tilde{Q}_{i-1/2,j}, \tilde{Q}_{i+1/2,j}, \tilde{Q}_{i,j-1/2}, \tilde{Q}_{i,j+1/2})$ for any given point P_{ij}. The (block) diagonal entry $\partial(A_r)_{ij}/\partial(q_s)_{ij} = \partial A_r(Q_{ij}, \tilde{Q}_{i-1/2,j}, \cdots)/\partial(q_s)_{ij}$ is easy to obtain since $\tilde{Q}$ is explicitly dependent on Q. The calculation of off-diagonal entries can be straightforwardly done using chain rules if the dependency between $\tilde{Q}$, $\bar{Q}$ and Q is clear. For example if $i \neq 1$, then $\tilde{Q}_{i-1/2,j}$ is dependent on Q_{ij} and $Q_{i-1,j}$; thus we have

$$\frac{\partial(A_r)_{ij}}{\partial(q_s)_{i-1,j}} = \sum_{\alpha=1}^{3} \frac{\partial(A_r)_{ij}}{\partial(\tilde{q}_\alpha)_{i-\frac{1}{2},j}} \times \frac{\partial(\tilde{q}_\alpha)_{i-\frac{1}{2},j}}{\partial(q_s)_{i-1,j}}.$$

If $i = 1$, then $\tilde{Q}_{i-1/2,j}$ is dependent on Q_{ij} and $\bar{Q}_{i-1,j}$ while $\bar{Q}_{i-1,j}$ is dependent on $Q^{(1)}$ and $Q^{(2)}$ because of the interpolation from the points in the left neighboring patch; thus we have

$$\frac{\partial (A_r)_{ij}}{\partial q_s^{(l)}} = \sum_{\alpha,\beta=1}^{3} \frac{\partial (A_r)_{ij}}{\partial (\tilde{q}_\alpha)_{i-\frac{1}{2},j}} \times \frac{\partial (\tilde{q}_\alpha)_{i-\frac{1}{2},j}}{\partial (\bar{q}_\beta)_{i-1,j}} \times \frac{\partial (\bar{q}_\beta)_{i-1,j}}{\partial q_s^{(l)}}, \quad l = 1, 2.$$

To formally define the preconditioner M_n^{-1}, we first decompose each of the six patches, Ω^k, $k = 1, \ldots, 6$, of the cubed-sphere into N_p non-overlapping rectangular subdomains, Ω_j^k, $j = 1, \ldots, N_p$. Here N_p is the number of processors. The subdomains in each patch is then mapped onto the processors. The same map is used for all six patches; *i.e.,* each processor has six subdomains. As seen in Fig. 4.1, all subdomains with the same color are assigned to the same processor.

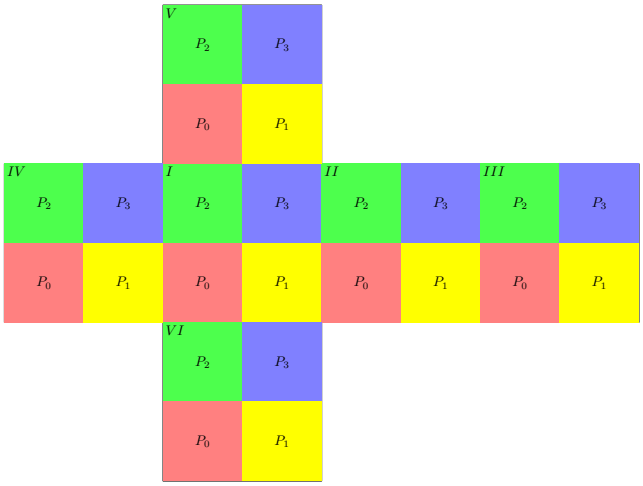

FIG. 4.1. *Map subdomains onto processors. The six patches of the cubed-sphere are indicated by Roman letters. Each patch is divided into subdomains in the same way. All subdomains with the same color go to the same processor. This is an example for 4 processors (24 subdomains), denoted as* $P_0, P_1, P_2,$ *and* P_3.

To obtain an overlapping decomposition of the domain, we extend each subdomain Ω_j^k to a larger subdomain $(\Omega_j^k)'$, as shown in Fig. 4.2. Note that the overlapping pieces may not be on the same processor or neighboring processors. We assume the size of Ω_j^k is $H_\xi \times H_\eta$ and the size of $(\Omega_j^k)'$ is $H'_\xi \times H'_\eta$. The overlapping size δ is defined as $(H'_\xi - H_\xi)/2$ (or $(H'_\eta - H_\eta)/2$).

For each of the overlapping subdomain we define B_j^k as the restriction of G' to the overlapping subdomain $(\Omega_j^k)'$. This is equivalent to taking the derivative of the nonlinear function G over the subdomain with homogeneous Dirichlet boundary conditions. Here M_n^{-1} is chosen as a left restricted additive Schwarz preconditioner (RAS) [7, 50] defined as

$$M_n^{-1} = \sum_{k=1}^{6} \sum_{j=1}^{N_p} (R_{j,0}^k)^T (B_j^k)^{-1} R_{j,\delta}^k. \tag{4.5}$$

Let m be the total number of cells and $(m_j^k)'$ the total number of cells in $(\Omega_j^k)'$. Then, $R_{j,\delta}^k$ is an $(m_j^k)' \times m$ block matrix that is defined as: its 3×3 block element

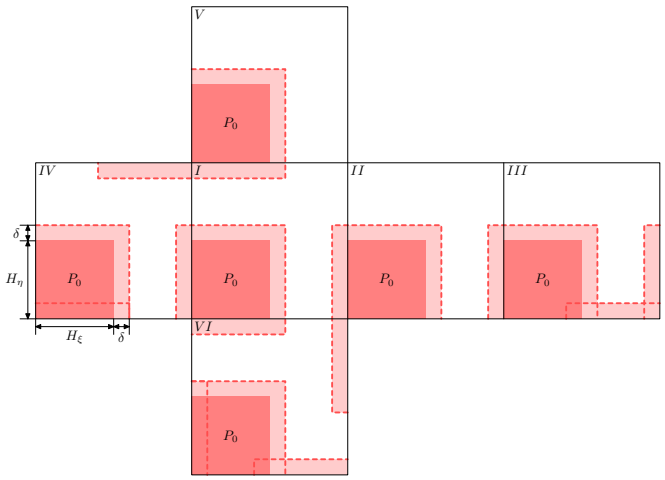

FIG. 4.2. *Decomposition of the domain into overlapping subdomains.*

$(R^k_{j,\delta})_{p_1,p_2}$ is an identity block if the integer indices $1 \le p_1 \le (m^k_j)'$ and $1 \le p_2 \le m$ belong to a cell in $(\Omega^k_j)'$, or a block of zeros otherwise. The $R^k_{j,\delta}$ serves as a restriction matrix because its multiplication by a block $m \times 1$ vector results in a smaller $(m^k_j)' \times 1$ block vector by dropping the components corresponding to cells outside $(\Omega^k_j)'$. Various inexact additive Schwarz preconditioners can be constructed by replacing the matrices B^k_j with convenient and inexpensive to compute matrices, such as those obtained with incomplete factorizations. In this paper we employ the LU factorization. Note that in (4.5) $R^k_{j,0}$ is a restriction to the non-overlapping subdomain.

The optimal convergence theory of RAS is still not available, even for elliptic equations. If the closely related classical additive Schwarz preconditioner is applied to a matrix resulting from the discretization of an elliptical problem, the condition number of the preconditioned system satisfies

$$\kappa \le C(1 + H/\delta)/H^2 \tag{4.6}$$

for the one-level method and $\kappa \le C(1 + H/\delta)$ for the two-level method, where H is of order H_ξ (or H_η) and the constant C is independent of H, δ, and the mesh size [47, 50]. The factor $1/H^2$, which is proportional to the number of subdomains, in the one-level method indicates an increase in the number of iterations with the increase of the number of processors. The two-level method is often preferred in order to remove the dependency of the number of iterations on the number of processors. However, in our case, the Jacobian system from SWE is not elliptic and the above mentioned condition number estimate does not apply. Our numerical experiments suggest that, with the one-level RAS method, the condition number growth is slower than that in the elliptic case. Similar observations were made before for other time dependent problems [4, 53] with the additive Schwarz method.

5. Numerical results. We implemented the algorithms described in the previous sections using PETSc [2]. The six patches of the cubed-sphere are handled by one PETSc DA (Distributed Array) with modified boundary operations to reflect the weakly non-matching patch interface conditions. The numerical tests are carried out on an IBM BlueGene/L with 1024 dual-processor compute nodes (2048 CPUs).

Each node has two IBM PowerPC 440 processors running at 700 MHz and 512 MB of memory. For our 2048-processor tests, the BG/L works under virtual-node mode and the available memory for each processor is reduced to 256 MB.

The stopping conditions for the nonlinear and linear iterative processes are:

- The relative tolerance for linear solver: $\zeta_r \leq 10^{-4}$,
- The absolute tolerance for linear solver: $\zeta_a \leq 10^{-14}$,
- The relative tolerance for nonlinear solver: $\varepsilon_r \leq 10^{-6}$,
- The absolute tolerance for nonlinear solver: $\varepsilon_a \leq 10^{-9}$.

On each subdomain, we use LU factorization as our subdomain solver and set the overlapping size between subdomains to be $\delta = 0, \hbar, 2\hbar$. GMRES(30), restarted at every 30 iterations, is used to solve the Jacobian system at each Newton step. For the problem with an analytical solution, the relative errors are measured using the following measures

$$l_1 = \frac{I(|h - \tilde{h}|)}{I(|\tilde{h}|)}, \quad l_2 = \left(\frac{I\left((h-\tilde{h})^2\right)}{I\left(\tilde{h}^2\right)} \right)^{\frac{1}{2}}, \quad l_\infty = \frac{\max\limits_{i,j,k} |h_{ij}^k - \tilde{h}_{ij}^k|}{\max\limits_{i,j,k} |\tilde{h}_{ij}^k|},$$

where h is the numerical solution, $\tilde{h}$ is the exact solution, and I is the discrete summation over all cell centers of the cubed-sphere mesh defined as

$$I(h) = \sum_{k=1}^{6} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\Lambda_{ij} h_{ij}^k \right),$$

where (i, j) is the mesh cell index on patch k.

We also implemented an explicit method to compare with the fully implicit method. For the shallow water equations (2.3)-(2.5), the CFL number can be obtained via

$$\text{CFL} = \frac{\Delta t}{\hbar} \max\{|u| + \sqrt{g g^{11} h},\, |v| + \sqrt{g g^{22} h}\}.$$

In the computation, we fix the maximum allowable CFL number to be 0.3 and calculate Δt adaptively in each time step to insure the stability of the explicit method. The same spatial discretization methods as described in Section 3 are implemented in the explicit code. We use forward Euler at the beginning of the time integration. And then the following second order Adams formula is used

$$Q^{(m+1)} = Q^{(m)} + \Delta t^{(m+1)} \left(A^{(m)} + \frac{\Delta t^{(m+1)}}{2\Delta t^{(m)}} (A^{(m)} - A^{(m-1)}) \right),$$

where $\Delta t^{(m)} = t^{(m)} - t^{(m-1)}$ is the time step size for step m.

5.1. A steady-state nonlinear zonal geostrophic flow. Our first test case (test two in [52]) describes a zonal geostrophic flow which exactly balances the geopotential of the fluid and the influence coming from the rotation of the sphere (*i.e.*, the Coriolis force). Since this is a test case with an exact solution, we use it to verify the correctness and measure the error of our discretization. We take the characteristic time scale and length scale as $\tau = 86400$s (one day) and $\rho = 6371220$m (the radius of the Earth), respectively, and use the non-dimensional parameters $a = 1.0\rho$, $\omega = 6.300288\tau^{-1}$, $g = 11489.57\rho\tau^{-2}$. Suppose that there is an angle α

between the polar axis and the axis of the solid body rotation, then the Coriolis parameter is

$$f = 2\omega(\cos\alpha\sin\theta - \sin\alpha\cos\lambda\cos\theta).$$

The velocity field is initially (and for all time) set as

$$\begin{aligned} u_\lambda &= u_0(\cos\alpha + \sin\alpha\cos\lambda\tan\theta)/a \\ u_\theta &= -u_0\sin\alpha\sin\lambda/a, \end{aligned}$$

where the reference wind velocity is $u_0 = (\pi/6)\rho\tau^{-1}$ which means that a full rotation around the earth finishes every 12 days (approximately equal to 40 m s^{-1}). It is worth pointing out that our definitions of u and v differ from that of [52] by a factor of $a\cos\theta$ and a, respectively. This is due to the fact that the velocity components in this paper are taken as coordinates components, which are different from the physical components defined in [52]. The exact solution of this test case is a steady state flow

$$gh = gh_0 - \left(a\omega u_0 + \frac{u_0^2}{2}\right)(\cos\alpha\sin\theta - \sin\alpha\cos\lambda\cos\theta)^2,$$

where $gh_0 = 5.4066669\rho^2\tau^{-2}$.

In the tests, the flow orientation angle α is chosen to be $\alpha = \pi/4$, which is challenging for the cubed-sphere. For this example, we use the centered scheme (3.6) for spatial discretization since the solution is smooth. The mesh size for the calculation is $40 \times 40 \times 6$ and the time step size is $\Delta t = 0.05\tau = 4320$s. The implicit code needs 100 time steps to obtain the results for day 5 ($t = 5.0\tau$). Fig. 5.1 shows the contour plots of the h field and the relative error after five days on the six patches of the cubed-sphere. The second order accuracy of the scheme is shown in Table 5.1 using N from 20 to 160.

TABLE 5.1
Order of accuracy of the h field with respect to the mesh size N for the steady-state zonal flow at day 5 with 8 processors, $\Delta t = 0.05\tau$.

N	l_1	Order	l_2	Order	l_∞	Order
20	3.068×10^{-3}	–	3.951×10^{-3}	–	1.584×10^{-2}	–
40	6.478×10^{-4}	2.24	8.278×10^{-4}	2.25	2.481×10^{-3}	2.67
80	1.634×10^{-4}	1.99	2.047×10^{-4}	2.02	5.736×10^{-4}	2.11
160	4.176×10^{-5}	1.97	5.172×10^{-5}	1.98	1.433×10^{-4}	2.00

Since the time step size is no longer constrained by the CFL condition in the fully implicit method, we can use very large time step in the experiments, *e.g.*, $\Delta t = 1.0\tau$ corresponding to a CFL number around 100.6. Table 5.2 gives a comparison of the number of iterations and the total compute time using different time step sizes Δt with 8 processors. We can see that Δt has very little impact on the nonlinear convergence. Although GMRES convergence suffers as Δt becomes larger, the total compute time still decreases.

5.2. A dam-break problem. Our second test case is a dam break problem featuring an unsteady solution with a sharp front moving through the sphere. Similar test cases can be found in [27, 44]. Because of the non-smoothness of the solution, we discretize SWE using the one-sided scheme as discussed in Section 3.1. We discuss

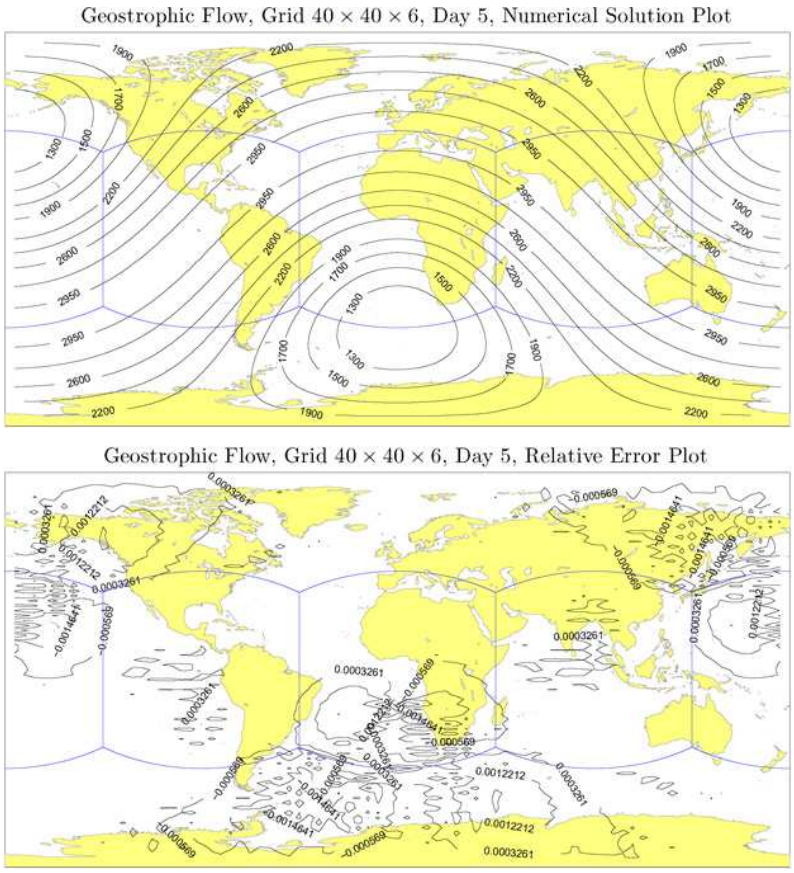

FIG. 5.1. *The contour plot of the numerical solution (top) and the relative error of the h field (bottom) for the geostrophic flow on a* $40 \times 40 \times 6$ *cubed-sphere mesh. The relative error of the h field varies from* -4.1×10^{-3} *to* 3.9×10^{-3}.

TABLE 5.2
Number of iterations and total compute time for the steady-state zonal flow at day 5 *on a* $40 \times 40 \times 6$ *mesh using different time step sizes with 8 processors, overlap* $\delta = 2h$.

Δt	CFL	Steps	Newton (avg.)	GMRES/Newton	Compute time (s)
0.2τ	20.1	25	2.0	11.4	18.49
0.5τ	50.3	10	2.0	24.7	9.67
1.0τ	100.6	5	2.0	62.6	8.08

several issues in this section including the scalability issue with up to 2048 processors, a comparison of the explicitly calculated Jacobian with the MC-FD calculated Jacobian, a comparison with an explicit method and a comparison with an approximate subdomain solver.

We set the radius of the sphere $a = 1.0$ and the gravitational constant $g = 1.0$.

The initial velocity of the flow is zero and the height field is

$$h = \begin{cases} 1.0, & \text{dist}\{(\lambda,\theta),(\lambda^\star,\theta^\star)\} < \pi/5, \\ 0.5, & \text{otherwise}, \end{cases}$$

where $\text{dist}\{\cdot,\cdot\}$ is the great circle distance on the surface of the sphere and $(\lambda^\star,\theta^\star)$ is chosen to be the center point of the first patch.

To show the parallel scalability of the implicit method, we consider two meshes $512 \times 512 \times 6$ and $1024 \times 1024 \times 6$. We use a fixed time step size $\Delta t = 0.2$ and run the code for 10 time steps, although the method converges well with much larger Δt with decreased accuracy. We run the tests using three overlapping factors $\delta = 0, \hbar, 2\hbar$ with different number of processors. Tables 5.3 and 5.4 exhibit the number of linear and nonlinear iterations and the compute time results with respect to the number of processors. It is clear that the number of nonlinear iterations per time step is almost independent of both the number of processors and the overlapping size. But the number of linear iterations grows slowly with the increase of the number of processors, which suggests that the condition number of the preconditioned linear system in the case follows a better estimation than (4.6). From Tables 5.3 and 5.4, we can see that larger overlap can reduce the number of linear iterations. However, larger overlap also requires larger amount of extra memory to store matrix elements corresponding to ghost points from the overlapping regions, leading to larger amount of communication and computation per iteration. Thus, as indicated in Tables 5.3 and 5.4, the best compute time is obtained with overlapping factor $\delta = 0$ with any given number of processors. The case of zero overlap is particularly interesting because, in theory, it corresponds to the smallest possible overlap (half mesh size) and a special condition number estimate is available for elliptic equations [16] and, in practice, it corresponds to the case that the communication cost is exactly zero during the preconditioning step of the algorithm.

From the compute time results listed in Tables 5.3 and 5.4, we see that superlinear speedups are obtained for any tested overlapping factors from 64 processors all the way to 2048 processors, except that the 2048-processor run is slower than the 1024-processor run in Table 5.3, which is because both the available memory and the available L3 cache are reduced by half for the 2048-processor tests. Some tests with $N_p = 128$ and $N_p = 2048$ are not carried out for Tables 5.3 and 5.4, because of the lack of memory.

TABLE 5.3

Test results using different overlapping factors with respect to the number of processors, the dam-break problem, $512 \times 512 \times 6$ *mesh (# of unknowns = 4,718,592), time step size* $\Delta t = 0.2$*, 10 time steps.*

N_p	Newton (avg.)			GMRES/Newton			Compute time (s)		
	$\delta = 0$	$\hbar$	$2\hbar$	$\delta = 0$	$\hbar$	$2\hbar$	$\delta = 0$	$\hbar$	$2\hbar$
64	3.3	3.2	3.2	12.03	10.13	9.30	436.4	461.1	509.7
128	3.3	3.2	3.2	14.07	12.55	12.09	154.4	170.7	202.5
256	3.4	3.2	3.2	16.18	14.85	14.21	67.5	82.5	97.0
512	3.2	3.2	3.2	22.93	21.50	20.52	26.3	39.2	49.2
1024	3.2	3.2	3.2	28.53	25.95	24.11	11.9	23.5	28.6
2048	3.2	3.2	3.2	42.83	38.55	35.97	6.4	25.1	34.3

TABLE 5.4
Test results using different overlapping factors with respect to the number of processors, the dam-break problem, $1024 \times 1024 \times 6$ *mesh (# of unknowns=18,874,368), time step size* $\Delta t = 0.2$*, 10 time steps.*

N_p	Newton (avg.)			GMRES/Newton			Compute time (s)		
	$\delta = 0$	$\hbar$	$2\hbar$	$\delta = 0$	$\hbar$	$2\hbar$	$\delta = 0$	$\hbar$	$2\hbar$
128	3.6	–	–	16.08	–	–	1219.7	–	–
256	3.6	3.5	3.6	17.06	15.49	14.73	507.2	558.7	634.3
512	3.6	3.5	3.5	22.99	21.71	21.08	191.8	232.8	276.2
1024	3.6	3.5	3.5	28.43	26.76	25.50	86.5	127.2	145.3
2048	3.5	–	–	42.24	–	–	42.2	–	–

To compare with the explicitly calculated exact Jacobian as discussed in Section 4, we run the test using the approximate Jacobian calculated by the multi-colored finite difference method (MC-FD) [2, 11]. The results are shown in Table 5.5, where the mesh size is $510 \times 510 \times 6$ instead of $512 \times 512 \times 6$ because of an implementation limitation of the MC-FD software. From Table 5.5, we can see that the number of nonlinear and linear iterations barely suffers from the approximation of the MC-FD calculated Jacobian. However, it is clear that the exact Jacobian is far better than the approximate one in terms of the total compute time and scalability.

TABLE 5.5
Comparisons between using the MC-FD Jacobian and using the exact Jacobian, the dam-break problem, $510 \times 510 \times 6$ *mesh, time step size* $\Delta t = 0.2$*, 10 time steps, overlapping size* $\delta = 0$*.*

N_p	Newton (avg.)		GMRES/Newton		Compute time (s)		Speedup	
	FD	Exact	FD	Exact	FD	Exact	FD	Exact
64	3.4	3.3	12.34	12.31	847.0	433.5	1.00	1.00
128	3.4	3.3	14.14	14.14	366.8	153.2	2.31	2.83
256	3.4	3.4	16.21	16.21	174.4	66.3	4.86	6.54
512	3.2	3.2	22.96	22.96	82.9	26.3	10.22	16.47
1024	3.3	3.2	28.50	28.58	45.3	11.8	18.52	36.80
2048	3.2	3.2	42.86	42.86	25.6	6.4	32.47	67.98

We next compare the fully implicit method with an explicit method. The time step size of the explicit method is adaptively selected so that the CFL number is always fixed to 0.3. As a result, the average time step size is around 0.01 for the overall simulation. The time step size of the implicit method is fixed at 0.2. The numerical solutions of the h field on an $36 \times 36 \times 6$ mesh are given in Fig. 5.2 for $t = 0.2, 0.4, 0.6, 0.8, 1.0$, respectively. The left column is for the explicit method and the right column is for the implicit method. They look similar for the first few snap shots, but some differences are shown in later steps. Because of the much smaller time step sizes used in the explicit method, we believe the explicit results are more accurate.

A compute time comparison is shown in the top figure of Fig. 5.3 for two meshes $512 \times 512 \times 6$ and $1024 \times 1024 \times 6$. The implicit method is always faster for both meshes and for any number of processors. The corresponding speedup curves are given in the bottom figures of Fig. 5.3. Both the explicit method and the implicit method have good speedups, while the implicit method exhibits a superlinear speedup due to

its better cache performance and the explicit method shows a speedup curve slightly below the ideal linear speedup.

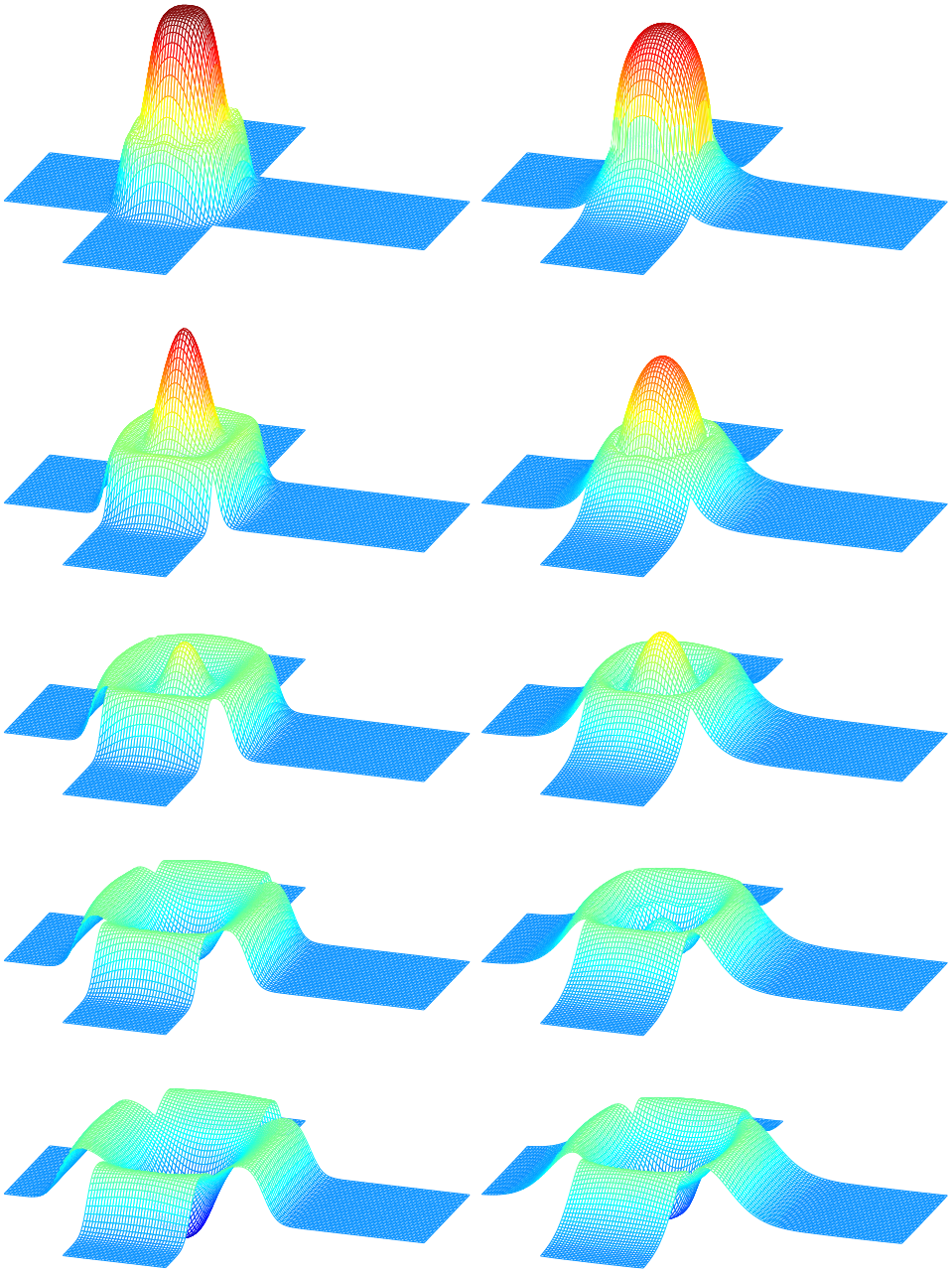

FIG. 5.2. *Numerical solution of the h field for the dam-break problem on* $36 \times 36 \times 6$ *mesh. The figures show the surface plots of the height field on the six patches of the cubed-sphere at time* $t = 0.2, 0.4, 0.6, 0.8, 1.0$*, respectively. The figures on the left column are the results from the explicit method with adaptive time stepping and the right column is from the fully implicit method with a fixed time step.*

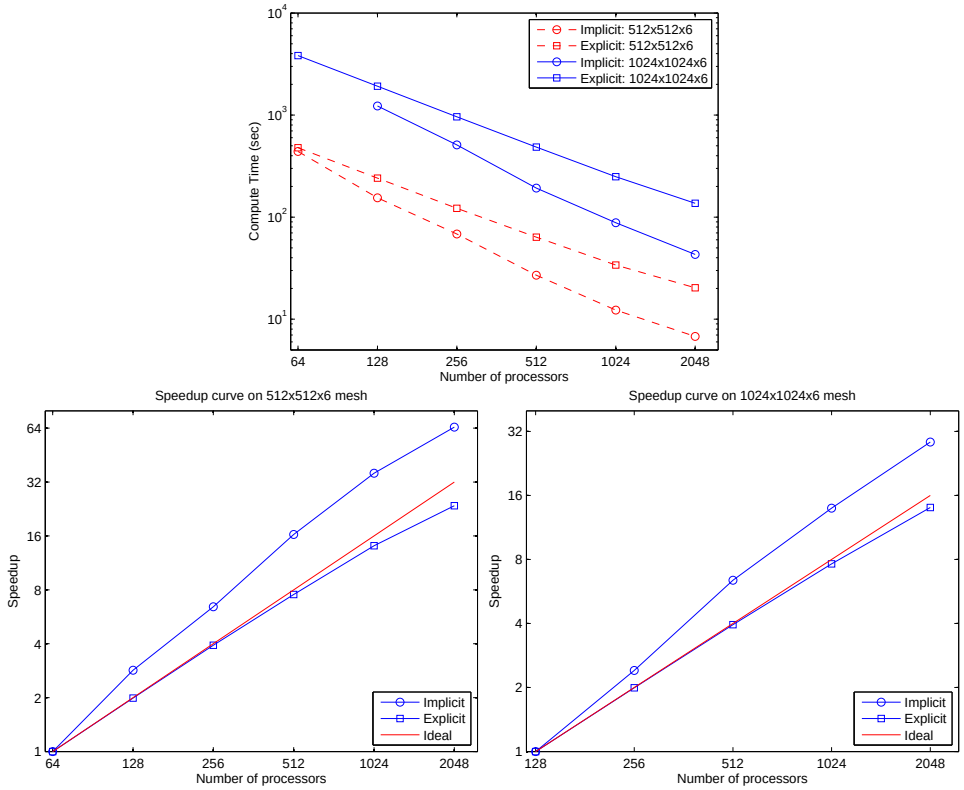

FIG. 5.3. *Compute time and speedup comparison for the dam-break problem on* $512 \times 512 \times 6$ *and* $1024 \times 1024 \times 6$ *meshes with* $64, 128, 256, 512, 1024,$ *and* 2048 *processors. Final simulation time* $t = 2.0$. *Implicit method:* 10 *time steps with a fixed time step size* $\Delta t = 0.2$. *Explicit method:* 2988 *and* 6002 *time steps with a fixed* $CFL = 0.3$ *on* $512 \times 512 \times 6$ *and* $1024 \times 1024 \times 6$ *meshes, respectively.*

5.3. Comments on ILU-based subdomain solver. In this subsection we replace the LU subdomain solver with a point-block ILU(ℓ) solver and compare the results. Here ℓ is the fill-in level for the incomplete LU factorization. The point-block size is 3×3 which is essential since block coupling will be destroyed if using a general point ILU solver. We set the overlapping factors $\delta = 0, \hbar, 2\hbar$ and set the fill-in levels $\ell = 0, 1, 2, 3, 4$. Larger overlap or larger fill-in helps in reducing the total number of linear iterations as the number of processors increases. However, memory complexity and work amount are both increased at every linear iteration.

For the zonal flow problem, ILU subdomain solver works well for small time steps such as $\Delta t = 0.01\tau$. However when we use a larger time step as those listed in Table 5.2, the Jacobian solver fails to converge. For the dam-break problem, as numerical experiments suggest, best compute times can be obtained with fill-in level $\ell = 3$ and overlapping size $\delta = 0$, with any number of processors. In Fig. 5.4, we show compute time results using ILU(ℓ), $\ell = 0, 1, 2, 3, 4$, compared with the results using LU, both with overlapping size $\delta = 0$. From the figure we see that ILU is faster in terms of compute time, but LU shows a better scalability.

6. Some final remarks. In this paper, we developed an overlapping domain decomposition method on the cubed-sphere for the fully implicit and fully coupled

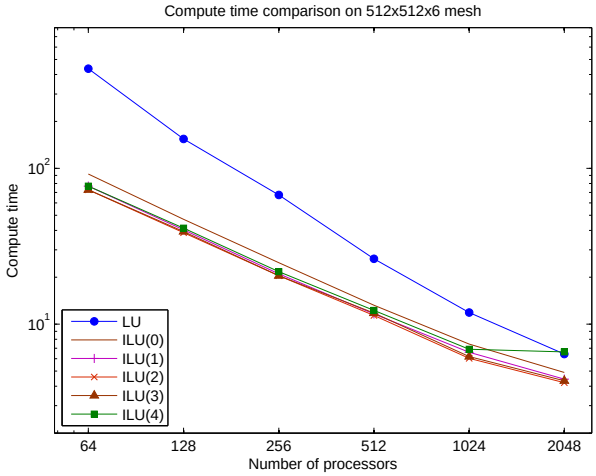

FIG. 5.4. *Compute time comparison of LU and point-block ILU subdomain solvers, dam-break problem on* $512 \times 512 \times 6$ *mesh, running for* 10 *implicit time steps with step size* $\Delta t = 0.2$.

solution of the shallow water equations. The Newton-Krylov-RAS method presented in the paper converges well for coupled nonlinear systems in both test cases, one with a steady-state smooth solution and the other with an unsteady sharp moving front. A comparison of the implicit method with a fixed time step size with an explicit method with adaptive time step sizes was performed, and the explicit solution is slightly more accurate due to the much smaller time step sizes, but the implicit method is faster in terms of the overall compute time. The number of nonlinear iterations is nearly independent of the number of subdomains, and the number of linear iterations grows slowly as the number of subdomains increases. The sub-optimal linear scalability, in terms of number of iterations, does not degenerate the scalability of the algorithm in terms of the actual compute time. In fact, both the explicit method and the implicit method offer excellent fixed problem size speedup, while the implicit method exhibits a superlinear speedup and the explicit method shows a speedup curve slightly below the ideal linear speedup for tests with up to 18 millions unknowns running on an IBM BG/L with up to 2048 processors.

REFERENCES

[1] A. ADCROFT, J.-M. CAMPIN, C. HILL, AND J. MARSHALL, *Implementation of an atmosphere-ocean general circulation model on expanded spherical cube*, Mon. Wea. Rev., 132 (2004), pp. 2845–2863.

[2] S. BALAY, K. BUSCHELMAN, W. D. GROPP, D. KAUSHIK, M. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *PETSc Users Manual*, Argonne National Laboratory, 2007.

[3] P. N. BROWN, D. E. SHUMAKER, AND C. S. WOODWARD, *Fully implicit solution of large-scale non-equilibrium radiation diffusion with high order time integration*, J. Comput. Phys., 204 (2005), pp. 760–783.

[4] X.-C. CAI, *Additive Schwarz algorithms for parabolic convection-diffusion equations,* Numer. Math., 60 (1990), pp. 41–62.

[5] X.-C. CAI, W. D. GROPP, D. E. KEYES, R. G. MELVIN, AND D. P. YOUNG, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[6] X.-C. CAI, W. D. GROPP, D. E. KEYES, AND M. D. TIDRIRI, *Newton-Krylov-Schwarz methods in CFD*, in Proceedings of the International Workshop on the Navier-Stokes Equations, Notes in Numerical Fluid Mechanics, R. Rannacher, eds. Vieweg Verlag, Braunschweig,

1994.
[7] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.
[8] V. Casulli, *Semi-implicit finite difference methods for the two-dimensional shallow water equation*, J. Comput. Phys., 86 (1990), pp. 56–74.
[9] C. Chen and F. Xiao, *Shallow water model on cubed-sphere by multi-moment finite volume method*, J. Comput. Phys., 227 (2008), pp. 5019–5044.
[10] V. Cheruvu, R. D. Nair, and H. M. Tufo, *A spectral finite volume transport scheme on the cubed-sphere*, Appl. Numer. Math., 57 (2007), pp. 1021–1032.
[11] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.
[12] J. M. Dennis, R. D. Nair, H. M Tufo, M. Levy, and T. Voran, *Development of a scalable global discontinuous Galerkin atmospheric model*, Int. J. Comput. Sci. Eng., in press.
[13] J. M. Dennis, A. Fournier, W. F. Spotz, A. St.-Cyr, M. A. Taylor, S. J. Thomas, and H. Tufo, *High resolution mesh convergence properties and parallel efficiency of a spectral element atmospheric dynamical core*, Int. J. High Perf. Comput. Appl., 19 (2005), pp. 225–235.
[14] J. M. Dennis, M. Levy, R. D. Nair, H. M. Tufo, and T. Voran, *Towards an efficient and scalable discontinuous Galerkin atmospheric model*, in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), Workshop 13, Volume 14, 2005.
[15] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.
[16] M. Dryja and O. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comp., 15 (1994), pp. 604–620.
[17] K. J. Evans and D. A. Knoll, *Temporal accuracy of phase change convection simulations using the JFNK-SIMPLE algorithm*, Int. J. Num. Meth. Fluids., 55 (2007), pp. 637–655.
[18] K. J. Evans, D. A. Knoll, and M. Pernice, *Enhanced algorithm efficiency for phase change convection using a multigrid preconditioner with a SIMPLE smoother*, J. Comput. Phys., 223 (2007), pp. 121–126.
[19] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2000.
[20] W. Hundsdorfer, B. Koren, M. van Loon and J. G. Verwer, *A Positive Finite-Difference Advection Scheme*, J. Comput. Phys., 117 (1995), pp. 35–46.
[21] K. Kashiyama, H. Ito, M. Behr, and T. Tezduyar, *Three-step explicit finite element computation of shallow water flows on a massively parallel computer*, Int. J. Num. Meth. Fluids., 21 (1995), pp. 885–900.
[22] D. E. Keyes, D. R. Reynolds, and C. S. Woodward, *Implicit solvers for large-scale nonlinear problems*, J. Phys.: Conf. Series, 46 (2006), pp. 433–442.
[23] D. A. Knoll, L. Chacon, L. G. Margolin, and V. A. Mousseau, *On balanced approximations for time integration of multiple time scale systems*, J. Comput. Phys., 185 (2003), pp. 583–611.
[24] D. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397.
[25] D. Lanser, J. G. Blom, and J. G. Verwer, *Spatial discretization of the shallow water equations in spherical geometry using Osher's scheme*, J. Comput. Phys., 165 (2000), pp. 542–565.
[26] M. N. Levy, R. D. Nair, and H. M. Tufo, *High-order Galerkin methods for scalable global atmospheric models*, Comp. & Geosci., 33 (2007), pp. 1022–1035.
[27] R. Liska and B. Wendroff, *Shallow water conservation laws on a sphere*, International series of numerical mathematics, H. Freistühler and G. Warnecke (eds), 141 (2001), pp. 673–682.
[28] R. B. Lowrie, *A comparison of implicit time integration methods for nonlinear relaxation and diffusion*, J. Comput. Phys., 196 (2004), pp. 566–590.
[29] V. A. Mousseau, D. A. Knoll, and J. M. Reisner, *An implicit nonlinearly consistent method for the two-dimensional shallow-water equations with Coriolis force*, Mon. Wea. Rev., 130 (2002), pp. 2611–2625.
[30] R. D. Nair, S. J. Thomas, and R. D. Loft, *A discontinuous Galerkin transport scheme on the cubed sphere*, Mon. Wea. Rev., 133 (2005), pp. 814–828.
[31] R. D. Nair, S. J. Thomas, and R. D. Loft, *A discontinuous Galerkin global shallow water model*, Mon. Wea. Rev., 133 (2005), pp. 876–888.
[32] C. C. Ober and J. N. Shadid, *Studies on the accuracy of time-integration methods for the*

radiation–diffusion equations, J. Comput. Phys., 195 (2004), pp. 743–772.

[33] S. Osher and F. Solomon, *Upwind difference schemes for hyperbolic systems of conservation laws*, Math. Comp., 38 (1982), pp. 339–374.

[34] S. Osher and S. Chakravarthy, *Upwind schemes and boundary conditions with applications to Euler equations in general geometries*, J. Comput. Phys., 50 (1983), pp. 447–481.

[35] S. Ovtchinnikov, F. Dobrian, X.-C. Cai, and D.E. Keyes, *Additive Schwarz-based fully coupled implicit methods for resistive Hall magnetohydrodynamic problems*, J. Comput. Phys., 225 (2007), pp. 1919–1936.

[36] W. M. Putman and S.-J. Lin, *Finite-volumn transport on various cubed-sphere grids*, J. Comput. Phys., 227 (2007), pp. 55–78.

[37] A. Qaddouri, L. Laayouni, J. Cote, and M. Gander, *Optimized Schwarz methods with an overset grid for the shallow-water equations: Preliminary results*, Applied Numer. Math., 58 (2008), pp. 459–471.

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

[39] J. Reisner, A. Wyszogrodzki, V. Mousseau, and D. Knoll, *An efficient physics-based preconditioner for the fully implicit solution of small-scale thermally driven atmospheric flows*, J. Comput. Phys., 189 (2003), pp. 30–44.

[40] D. R. Reynolds, R. Samtaney, and C. S. Woodward, *A fully implicit numerical method for single-fluid resistive magnetohydrodynamics*, J. Comput. Phys., 219 (2006), pp. 144–162.

[41] C. Ronchi, R. Iacono, and P. Paolucci, *The cubed sphere: A new method for the solution of partial differential equations in spherical geometry*, J. Comput. Phys., 124 (1996), pp. 93–114.

[42] J. Ruge, S. McCormick, and S. Yee, *Multilevel adaptive methods for semi-implicit solution of shallow-water equations on a sphere*, Mon. Wea. Rev., 123 (1995), pp. 2197–2205.

[43] J. A. Rossmanith, *A wave propagation method for hyperbolic systems on the sphere*, J. Comput. Phys., 213 (2006), pp. 629–658.

[44] J. A. Rossmanith, *A Wave Propagation Method with Constrained Transport for Ideal and Shallow Water Magnetohydrodynamics*, PhD thesis, University of Washington, 2002.

[45] R. Sadourny, *Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids*, Mon. Wea. Rev., 100 (1972), pp. 211–224.

[46] J. N. Shadid, R. S. Tuminaro, K. D. Devine, G. L. Hennigan, and P.T. Lin, *Performance of fully coupled domain decomposition preconditioners for finite element transport/reaction simulations*, J. Comput. Phys., 205 (2005), pp. 24–47.

[47] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition. Parallel Multilevel Methods for Eliptic Partial Differential Equations*, Cambridge University Press, New York, 1996.

[48] S. J. Thomas, J. Dennis, H. Tufo, and P. Fischer, *A Schwarz preconditioner for the cubed-sphere*, SIAM J. Sci. Comp., 25 (2003), pp. 442–453.

[49] S. J. Thomas and R. D. Loft, *Semi-implicit spectral element atmospheric model*, J. Sci. Comput., 17 (2002), pp. 339–350.

[50] A. Toselli and O. Widlund, *Domain Decomposition Methods – Algorithms and Theory*, Springer-Verlag, Berlin, 2005.

[51] B. van Leer, *Towards the ultimate conservative difference scheme III: Upstream-centered finite-difference schemes for ideal compressible flow*, J. Comput. Phys., 23 (1977), pp. 263–275.

[52] D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swartztrauber, *A standard test set for numerical approximations to the shallow water equations in spherical geometry*, J. Comput. Phys., 102 (1992), pp. 211–224.

[53] Y. Wu, X.-C. Cai, and D. E. Keyes, *Additive Schwarz methods for hyperbolic equations,* in Proceedings of the 10th Intl. Conf. on Domain Decomposition Methods, J. Mandel, C. Farhat and X.-C. Cai, eds., AMS, 1998, pp. 513–521.