Parallel Multilevel Methods for Implicit Solution of Shallow Water Equations with Nonsmooth Topography on the Cubed-sphere ☆

Chao Yang[a,b], Xiao-Chuan Cai[a]

[a] *Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA*
[b] *Institute of Software, Chinese Academy of Sciences, Beijing 100190, P. R. China*

Abstract

High resolution and scalable parallel algorithms for the shallow water equations on the sphere are very important for modeling the global climate. In this paper, we introduce and study some highly scalable multilevel domain decomposition methods for the fully implicit solution of the nonlinear shallow water equations discretized with a second-order well-balanced finite volume method on the cubed-sphere. With the fully implicit approach, the time step size is no longer limited by the stability condition, and with the multilevel preconditioners, good scalabilities are obtained on computers with a large number of processors. The investigation focuses on the use of semismooth inexact Newton method for the case with nonsmooth topography and the use of two- and three-level overlapping Schwarz methods with different order of discretizations for the preconditioning of the Jacobian systems. We test the proposed algorithm for several benchmark cases and show numerically that this approach converges well with smooth and nonsmooth bottom topography, and scales perfectly in terms of the strong scalability and reasonably well in terms of the weak scalability on machines with thousands and tens of thousands of processors.

Keywords:
Shallow water equations, fully implicit method, Newton-Krylov-Schwarz, multilevel domain decomposition method, cubed-sphere mesh, parallel processing, strong and weak scalability
2000 MSC: 86A10, 65M55, 65Y05, 35L65

☆ This research was supported in part by NSF under DMS-0913089 and DOE under DE-FC-02-06ER25784, and the author CY was also supported in part by NSF China under 10801125 and 863 Program of China under 2010AA012300.
Email addresses: chao.yang@colorado.edu (Chao Yang), cai@cs.colorado.edu (Xiao-Chuan Cai)

1. Introduction

Numerical simulation of the shallow water equations (SWEs) in spherical geometry lies at the root in the development of numerical algorithms and high performance software for the modeling of atmospheric circulation. There are several family of meshes available for numerical computation on the sphere. In this study, we use the cubed-sphere mesh of gnomonic type [31], which is generated by mapping the six faces of an inscribed cube to the sphere surface using the gnomonic projection. The six patches are attached together with proper interface conditions. The SWEs in the local curvilinear coordinates on each patch can be written as follows.

$$\frac{\partial Q}{\partial t}+\frac{1}{\Lambda}\frac{\partial(\Lambda F)}{\partial x}+\frac{1}{\Lambda}\frac{\partial(\Lambda G)}{\partial y}+S=0,\quad (x,y)\in[-\pi/4,\pi/4]^2 \tag{1}$$

with

$$Q=\begin{pmatrix}h\\ hu\\ hv\end{pmatrix},\ F=\begin{pmatrix}hu\\ huu\\ huv\end{pmatrix},\ G=\begin{pmatrix}hv\\ huv\\ hvv\end{pmatrix},\ S=\begin{pmatrix}0\\ S_1\\ S_2\end{pmatrix}, \tag{2}$$

where

$$\begin{aligned}S_1&=\Gamma^1_{11}(huu)+2\Gamma^1_{12}(huv)+f\Lambda\left(g^{12}hu-g^{11}hv\right)+gh\left(g^{11}\frac{\partial Z}{\partial x}+g^{12}\frac{\partial Z}{\partial y}\right),\\ S_2&=2\Gamma^2_{12}(huv)+\Gamma^2_{22}(hvv)+f\Lambda\left(g^{22}hu-g^{12}hv\right)+gh\left(g^{12}\frac{\partial Z}{\partial x}+g^{22}\frac{\partial Z}{\partial y}\right).\end{aligned} \tag{3}$$

Here h is the thickness of the fluid (atmosphere), (u,v) are the contravariant components of the fluid velocity, g is the gravitational constant and f is the Coriolis parameter due to the rotation of the sphere. The bottom topography is b which describes the height of the spherical surface, and the surface level of the fluid is $Z=h+b$. The variable coefficients g^{mn}, Λ and Γ^{ℓ}_{mn} are only dependent on the curvilinear coordinates and their detailed expressions can be found in [48].

The latitude-longitude (lat-lon) mesh, although has been popular with spectral methods in the last several decades, is highly nonuniform and the resulting matrix is quite dense, which is not easy to deal with in large scale parallel simulations. Recently several efforts have been made on using numerical discretizations on composite meshes, such as the icosahedron geodesic mesh [35, 43], the cubed-sphere mesh [34] and the Yin-Yang mesh [19], all consist of several patches that are either connected through interfaces or overlapping in order to cover the whole sphere.

Depending on the speed of the fastest wave that is of interest in the chosen climate model, the simulations can be carried out using either one or a combination of the three classes of methods: explicit, semi-implicit and fully implicit. If the wave is so fast such that the corresponding Courant-Friedrichs-Lewy (CFL) number is less than one, then explicit methods are the choice since they are easy to implement and the algorithms are highly parallelizable. If the fast waves don't exist in the model, or are not of interests, then one can choose to use a semi-implicit method which requires only linear solvers but may have some limitations on the time step size or the fully implicit method whose time step size is limited only by the accuracy requirement. Although fully implicit methods enjoy an advantage that the time step size is no longer constrained by any stability conditions. If not used properly, the large time step size in a fully implicit method may result in large simulation errors, especially when the dynamical timescale is not properly resolved [22]. In [13], the accuracy of a fully implicit method for a spectral element shallow water code on the cubed-sphere was carefully studied for several test cases and compared with existing fully explicit leapfrog and semi-implicit methods. The numerical experiments in [13] suggested that typical fully implicit time step sizes that are 30 to 60 times larger than the gravity wave stability limits and 6 to 20 times larger than the advective scale stability limits are free of unacceptable accuracy loss.

For high fidelity simulations on supercomputers with a large number of processors, the scalabilities of the algorithm with respect to the number of processors are critically important. There are several scalability issues we address in this paper including (a) the machine independent scalability of the algorithm characterized mainly by the number of linear and nonlinear iterations; (b) the strong scalability of the algorithm in terms of the total compute time when the size of the overall mesh is fixed; and (c) the weak scalability of the algorithm in terms of the total compute time when the size of the mesh per processor is fixed. Note that without (a), it is usually not possible to have (b) or (c), but even if (a) holds, (b) and (c) may still not possible since both (b) and (c) depend on the machine and the software implementation of the algorithm, among other factors.

Scalability by itself is not necessarily a sufficient measure of a good parallel algorithm. For example, we show in this paper that both the classical explicit algorithm and our newly introduced fully implicit algorithm are almost perfectly scalable in the strong sense, but the implicit algorithm can be, for example, many times faster in terms of the total compute time. On the other hand, if we consider the weak scalability, the explicit method is not scalable at all in the sense that the compute time increases linearly in the number of processors, but the growth of compute time of the implicit method is only a log

function of the number of processors.

The price to pay for using fully implicit method is that a large nonlinear algebraic system, with an extremely ill-conditioned sparse Jacobian matrix, has to be solved at each time step. To solve the nonlinear system efficiently, we use an inexact Newton method, within which a Krylov subspace method is used to solve the Jacobian system at each inexact Newton step. If a non-scalable algorithm is used for solving the system, as the mesh is refined or the number of processors is increased, the increase of Krylov iterations will directly result in the simulation time increase, which may neutralize the benefits from the unconstrained time step size [12]. One way to keep the simulation time at an acceptable level is to use a preconditioner which is not only inexpensive to apply but also capable of maintaining the number of Krylov iterations to nearly a constant as the mesh is refined or the number of processors is increased. Therefore, the development of an effective and efficient preconditioner is crucial for a scalable fully implicit solver, which is the main goal of this paper.

A one-level domain decomposition method was studied by the authors in [48] for solving the SWEs on the cubed-sphere. It was demonstrated that when the solution is smooth and when a first-order finite volume scheme is used in the spatial discretization, the one-level Schwarz preconditioner works well in terms of the total number of linear iterations. The parallel scalability is quite acceptable when the number of processors is not too large. However, the number of linear iterations grows as the number of processors increases. The situation becomes worse when a higher order discretization is used, because the resulting Jacobian system is much more ill-conditioned and has considerably more non-zero elements. To deal with these much denser linear systems, one-level additive Schwarz preconditioners based on a lower order discretization of the SWEs was studied in [46]. To take advantage of modern supercomputers for large-scale atmospheric simulations, and to exploit flexibilities of large time step sizes and deal with nonsmooth bottom topography, we propose and study two- and three-level overlapping Schwarz preconditioners in this paper. Both strong and weak scalabilities are carefully studied with around $O(10^3 - 10^5)$ processors. This class of multilevel methods is sufficiently robust and we believe they are suitable for even larger processor count, as long as more levels are included in the preconditioner. When a variable bottom topography is involved in the SWEs, the spatial discretization must satisfy a well-balanced property to avoid spurious oscillations near the non-smooth area of the topography. In [46], a well-balanced finite volume method was proposed and studied for the SWEs on a cubed-sphere, together with an explicit Runge-Kutta method. If a fully implicit method is used instead,

the smoothness requirement by the Newton method is sometimes violated, due to the nonsmooth evaluations of nonlinear functions in the spatial discretization and the topographic source terms, and the convergence of the Newton method may be problematic. In this situation, we introduce a semismooth Newton method [27, 28] in which the generalized Jacobian [6] is calculated by freezing some multiple-valued entries at each Newton step.

In the paper we focus only on issues related the fully implicit methods, and have not paid attention to the comparison with semi-implicit methods which are excellent choices in some situations. For example, semi-implicit semi-Lagrangian (SISL) method works quite well for hyperbolic problems such as the SWEs, but it is beyond the scope of this paper to compare the parallel performance of our fully implicit approach with SISL since it requires multiple linear solves per time step and these solves have to be carried out one after another, which may reduce the overall parallelism in large scale calculations.

The remainder of this paper is organized as follows. In Section 2, we introduce a fully implicit finite volume discretization of the SWEs on the cubed-sphere. The scheme is second-order and well-balanced so that nonsmooth topographic source terms can be handled successfully. Some variants of Newton-Krylov-Schwarz methods with adaptive stopping conditions are then discussed in Section 3 for solving the nonlinear systems resulting from the fully implicit time integration. In Section 4, some multilevel Schwarz preconditioning methods, which are the most important part of the scalable approach, are covered in detail. Numerical results for two benchmark cases are provided in Section 5 where several key issues are then discussed, including an accuracy and efficiency comparison with an explicit method, the effects on using different parameters in the preconditioner, the robustness of the method when the time step is large, and the strong and weak scalability of the algorithm when $O(10^3 - 10^5)$ processors are used on two supercomputers. The paper is then concluded in Section 6.

2. Fully implicit well-balanced discretization of the SWEs on the cubed-sphere

Although the cubed-sphere mesh was initially proposed in 1972 [34], the research community paid little attention to it until it was revisited in 1990s [29, 31]. Since then several discretization schemes have been studied for the SWEs on the cubed-sphere, e.g., the finite volume method [5, 26, 32], the spectral element method [39, 40], the discontinuous Galerkin method [9, 23]. In this paper, we introduce a fully implicit, well balanced finite volume scheme, which is a reformulation of an explicit scheme from [46].

Let us denote the six patches of the cubed-sphere as $\mathscr{P}^k$, $k = 1, \cdots, 6$. Suppose $\mathscr{P}^k$ is covered by

a logically rectangular $N \times N$ mesh, which is equally spaced in the computational domain $\{(x,y) \in [-\pi/4, \pi/4]^2\}$ with mesh size $\hbar = \pi/2N$. Patch $\mathscr{P}^k$ is then divided into mesh cells $\mathscr{C}_{ij}^k$ centered at (x_i, y_j), $i, j = 1, \cdots, N$. The approximate solution in cell $\mathscr{C}_{ij}^k$ at time t is defined as

$$Q_{ij}^k \approx \frac{1}{\Lambda_{ij}^k \hbar^2} \int_{y_j-\hbar/2}^{y_j+\hbar/2} \int_{x_i-\hbar/2}^{x_i+\hbar/2} \Lambda(x,y)\, Q(x,y,t)\, dxdy.$$

Here Λ_{ij}^k is evaluated at the cell center. The superscript k is sometimes ignored for convenience.

After discretizing the shallow water system (1) using a cell-centered finite volume method, we obtain the following semi-discrete system:

$$\frac{\partial Q_{ij}}{\partial t} + \frac{1}{\Lambda_{ij}\hbar}\left[(\Lambda F)_{i+1/2,j} - (\Lambda F)_{i-1/2,j}\right] + \frac{1}{\Lambda_{ij}\hbar}\left[(\Lambda G)_{i,j+1/2} - (\Lambda G)_{i,j-1/2}\right] + S_{ij} = 0. \tag{4}$$

Here the numerical fluxes on the four cell boundaries are approximated as

$$(\Lambda F)_{i\pm1/2,j} \approx \frac{1}{\hbar}\int_{y_j-\hbar/2}^{y_j+\hbar/2} \Lambda\left(x_i \pm \hbar/2, y\right) F\left(x_i \pm \hbar/2, y, t\right) dy,$$
$$(\Lambda G)_{i,j\pm1/2} \approx \frac{1}{\hbar}\int_{x_i-\hbar/2}^{x_i+\hbar/2} \Lambda\left(x, y_j \pm \hbar/2\right) G\left(x, y_j \pm \hbar/2, t\right) dx.$$

In this paper we use Osher's Riemann solver [24, 25] to calculate the numerical fluxes, i.e.,

$$(\Lambda F)_{i+1/2,j} = \Lambda_{i+1/2,j} F^{(o)}\left(Q_{i+1/2,j}^-, Q_{i+1/2,j}^+\right) = \Lambda_{i+1/2,j} F\left(Q_{i+1/2,j}^*\right),$$
$$(\Lambda G)_{i,j+1/2} = \Lambda_{i,j+1/2} G^{(o)}\left(Q_{i,j+1/2}^-, Q_{i,j+1/2}^+\right) = \Lambda_{i,j+1/2} G\left(Q_{i,j+1/2}^*\right),$$

where $Q_{i+1/2,j}^{\pm}$ and $Q_{i,j+1/2}^{\pm}$ are the reconstructed states of Q on the local cell boundaries. To deal with the non-smooth bottom topography, we calculate $Q_{i+1/2,j}^*$ with the following scheme ([46]):

$$h_{i+1/2,j}^* = \frac{1}{4g g_{i+1/2,j}^{11}}\left[\frac{1}{2}\left(u_{i+1/2,j}^- - u_{i+1/2,j}^+\right) + \sqrt{g g_{i+1/2,j}^{11}}\left(\sqrt{h_{i+1/2,j}^-} + \sqrt{h_{i+1/2,j}^+}\right)\right]^2,$$
$$u_{i+1/2,j}^* = \frac{1}{2}\left(u_{i+1/2,j}^- + u_{i+1/2,j}^+\right) + \sqrt{g g_{i+1/2,j}^{11}}\left(Z_{i+1/2,j}^- - Z_{i+1/2,j}^+\right)\Big/\left(\sqrt{h_{i+1/2,j}^-} + \sqrt{h_{i+1/2,j}^+}\right),$$
$$v_{i+1/2,j}^* = \begin{cases} v_{i+1/2,j}^- + \left(g_{i+1/2,j}^{12}\Big/ g_{i+1/2,j}^{11}\right)\left(u_{i+1/2,j}^* - u_{i+1/2,j}^-\right), & \text{if } u_{i+1/2,j}^* \geq 0 \\ v_{i+1/2,j}^+ + \left(g_{i+1/2,j}^{12}\Big/ g_{i+1/2,j}^{11}\right)\left(u_{i+1/2,j}^* - u_{i+1/2,j}^+\right), & \text{otherwise,} \end{cases}$$

when $|u| < \sqrt{g g^{11} h}$. The calculation of $(\Lambda G)_{i,j+1/2}$ follows a similar scheme,

$$h_{i,j+1/2}^* = \frac{1}{4g g_{i,j+1/2}^{22}}\left[\frac{1}{2}\left(v_{i,j+1/2}^- - v_{i,j+1/2}^+\right) + \sqrt{g g_{i,j+1/2}^{22}}\left(\sqrt{h_{i,j+1/2}^-} + \sqrt{h_{i,j+1/2}^+}\right)\right]^2,$$
$$v_{i,j+1/2}^* = \frac{1}{2}\left(v_{i,j+1/2}^- + v_{i,j+1/2}^+\right) + \sqrt{g g_{i,j+1/2}^{22}}\left(Z_{i,j+1/2}^- - Z_{i,j+1/2}^+\right)\Big/\left(\sqrt{h_{i,j+1/2}^-} + \sqrt{h_{i,j+1/2}^+}\right),$$
$$u_{i,j+1/2}^* = \begin{cases} u_{i,j+1/2}^- + \left(g_{i,j+1/2}^{22}\Big/ g_{i,j+1/2}^{12}\right)\left(v_{i,j+1/2}^* - v_{i,j+1/2}^-\right), & \text{if } v_{i,j+1/2}^* \geq 0 \\ u_{i,j+1/2}^+ + \left(g_{i,j+1/2}^{22}\Big/ g_{i,j+1/2}^{12}\right)\left(v_{i,j+1/2}^* - v_{i,j+1/2}^+\right), & \text{otherwise,} \end{cases}$$

when $|v| < \sqrt{gg^{22}h}$. In the explicit scheme [46], the values of $v^*_{i+1/2,j}$ and $u^*_{i,j+1/2}$ are taken from the previous time step, and in the fully implicit scheme, the values have to be taken in the current time level. Because of these branching conditions, these equations may not be differentiable.

Suppose the cell-averaged values of Q on cell $\mathscr{C}_{ij}$, $\mathscr{C}_{i\pm1,j}$ and $\mathscr{C}_{i,j\pm1}$ are already known, there are several different ways to reconstruct Q on the cell boundaries of $\mathscr{C}_{ij}$. The simplest one is to use the piecewise constant method by forcing $Q = Q_{ij}$ on $\mathscr{C}_{ij}$, which leads to a first-order method [48]. To achieve higher order of accuracy, we use a second-order reconstruction in this paper, i.e.,

$$Q^{\mp}_{i\pm1/2,j} = Q_{ij} \pm \big(Q_{i+1,j} - Q_{i-1,j}\big)/4\,, \quad Q^{\mp}_{i,j\pm1/2} = Q_{ij} \pm \big(Q_{i,j+1} - Q_{i,j-1}\big)/4\,.$$

To pass information between the boundary of the six patches, ghost cells can be introduced by extending several layers of meshes outward for each patch [31, 32]. On each patch interface, we only use one layer of ghost cells and the numerical fluxes are calculated symmetrically across the interface to insure the numerical conservation of total mass; see [46] for further details.

The following two terms in the SWEs involve the bottom topography,

$$S_{T1} = gh\left(g^{11}\frac{\partial Z}{\partial x} + g^{12}\frac{\partial Z}{\partial y}\right), \quad S_{T2} = gh\left(g^{12}\frac{\partial Z}{\partial x} + g^{22}\frac{\partial Z}{\partial y}\right).$$

The discretization of these terms should be carried out with special care, or spurious oscillations could be introduced [14]. A well-balanced discretization was proposed in [46] for the explicit solution of the SWEs in an equivalent form of (1)-(3), which corresponds to discretizing S_{T1} and S_{T2} as

$$\begin{aligned}
(S_{T1})_{ij} &= \left(gh^*_{i+1/2,j} + gh^*_{i-1/2,j}\right)\left[\left(g^{11}\Lambda\right)_{i+1/2,j} + \left(g^{11}\Lambda\right)_{i-1/2,j}\right]\left(Z^*_{i+1/2,j} - Z^*_{i-1/2,j}\right)\Big/\left(4\Lambda_{ij}\hbar\right) \\
&\quad + \left(gh^*_{i,j+1/2} + gh^*_{i,j-1/2}\right)\left[\left(g^{12}\Lambda\right)_{i,j+1/2} + \left(g^{12}\Lambda\right)_{i,j-1/2}\right]\left(Z^*_{i,j+1/2} - Z^*_{i,j-1/2}\right)\Big/\left(4\Lambda_{ij}\hbar\right), \\
(S_{T2})_{ij} &= \left(gh^*_{i+1/2,j} + gh^*_{i-1/2,j}\right)\left[\left(g^{12}\Lambda\right)_{i+1/2,j} + \left(g^{12}\Lambda\right)_{i-1/2,j}\right]\left(Z^*_{i+1/2,j} - Z^*_{i-1/2,j}\right)\Big/\left(4\Lambda_{ij}\hbar\right) \\
&\quad + \left(gh^*_{i,j+1/2} + gh^*_{i,j-1/2}\right)\left[\left(g^{22}\Lambda\right)_{i,j+1/2} + \left(g^{22}\Lambda\right)_{i,j-1/2}\right]\left(Z^*_{i,j+1/2} - Z^*_{i,j-1/2}\right)\Big/\left(4\Lambda_{ij}\hbar\right),
\end{aligned}$$

where

$$\begin{aligned}
Z^*_{i+1/2,j} &= \frac{1}{4gg^{11}_{i+1/2,j}}\left[\frac{1}{2}\left(u^-_{i+1/2,j} - u^+_{i+1/2,j}\right) + \sqrt{gg^{11}_{i+1/2,j}}\left(\sqrt{Z^-_{i+1/2,j}} + \sqrt{Z^+_{i+1/2,j}}\right)\right]^2, \\
Z^*_{i,j+1/2} &= \frac{1}{4gg^{22}_{i,j+1/2}}\left[\frac{1}{2}\left(u^-_{i,j+1/2} - u^+_{i,j+1/2}\right) + \sqrt{gg^{22}_{i,j+1/2}}\left(\sqrt{Z^-_{i,j+1/2}} + \sqrt{Z^+_{i,j+1/2}}\right)\right]^2.
\end{aligned}$$

Given a semi-discrete system

$$\frac{\partial Q_{ij}}{\partial t} + \mathscr{L}(Q_{ij}) = 0, \tag{5}$$

the second-order backward differentiation formula (BDF-2)

$$\frac{1}{2\Delta t}\left(3Q_{ij}^{(m)} - 4Q_{ij}^{(m-1)} + Q_{ij}^{(m-2)}\right) + \mathscr{L}(Q_{ij}^{(m)}) = 0 \tag{6}$$

is employed for the time integration. Here $Q^{(m)}$ denotes the value of Q at the m-th time step with a fixed time step size Δt. Only at the first time step, a first-order backward Euler (BDF-1) method is used. A major advantage of the fully implicit method is that the time step size Δt is no longer constrained by the CFL condition, which is often required by explicit or semi-implicit techniques.

For comparison purpose, we also implement an explicit second-order Strong Stability Preserving Runge-Kutta (SSP RK-2) method

$$\begin{aligned} \overline{Q}^{(m)} &= Q^{(m-1)} - \Delta t \mathscr{L}\left(Q^{(m-1)}\right), \\ Q^{(m)} &= (1/2)\,(Q^{(m-1)} + \overline{Q}^{(m)}) - (\Delta t/2)\,\mathscr{L}(\overline{Q}^{(m)}). \end{aligned} \tag{7}$$

The time step size is adaptively controlled so that the corresponding CFL number is fixed to 0.5. Here the explicit CFL number is calculated via $\max\left\{|u| + \sqrt{gg^{11}h}, |v| + \sqrt{gg^{22}h}\right\}\left(\Delta t/\hbar\right)$.

3. Some variants of Newton-Krylov-Schwarz methods

In the fully implicit method, a system of nonlinear algebraic equations

$$\mathscr{F}(X) = 0 \tag{8}$$

has to be constructed and solved at each time step. We use a Newton-Krylov-Schwarz (NKS) [2, 3] type method to solve (8). To turn the set of equations defined on a mesh into an algebraic system, both the unknowns and the equations need to be sorted in a certain order. We use a point-wise (field-coupling) ordering instead of a component-wise (field-splitting) ordering in our implementation. This ordering helps improving not only in the cache performance but also the parallel efficiency in load and communication balance. Interested readers should see a Gordon-Bell Prize winning application in [16] for example.

3.1. Inexact Newton method with adaptive stopping conditions

To solve (8) at time step m, we first let the initial guess $X_0 = X^{(m-1)}$ be the solution of the previous time step, then the next approximate solution X_{n+1} is obtained by

$$X_{n+1} = X_n + \lambda_n S_n, \quad n = 0, 1, \ldots \tag{9}$$

Here λ_n is the steplength determined by a linesearch procedure [8] and S_n is the Newton correction vector. To calculate S_n, the Jacobian system

$$J_n S_n = -\mathscr{F}(X_n) \tag{10}$$

is constructed and solved for each Newton iteration. The Jacobian matrix $J_n = \mathscr{F}'(X_n)$ is calculated at the current approximate solution X_n. In inexact Newton method, the linear system (10) does not need to be solved exactly. In practice, we use a restarted GMRES to approximately solve the right-preconditioned system

$$J_n M^{-1}(MS_n) = -\mathscr{F}(X_n), \tag{11}$$

until the linear residual $r_n = J_n S_n + \mathscr{F}(X_n)$ satisfies

$$\|r_n\| \leq \eta \|\mathscr{F}(X_n)\|. \tag{12}$$

A flexible version of GMRES [33] has to be used if the preconditioner changes during the GMRES iterations. This happens when the coarse problems in a multilevel preconditioner are solved iteratively. The accuracy (relative tolerance) of the Jacobian solver is determined by the nonlinear forcing term η.

To achieve a uniform residual error for different time steps, we use the following adaptive stopping conditions for the Newton iteration:

$$\|\mathscr{F}(X_{n+1})\| \leq \min\left\{\hat{\varepsilon}_a, \max\left\{\check{\varepsilon}_a, \varepsilon_r \|\mathscr{F}(X_0)\|\right\}\right\}. \tag{13}$$

Here the relative tolerance ε_r and the safeguard $\hat{\varepsilon}_a$ are both fixed for all time steps, and the absolute tolerance $\check{\varepsilon}_a$ is chosen as $\check{\varepsilon}_a^{(0)} \in [0, \hat{\varepsilon}_a)$ at the first time step and then adaptively determined by

$$\check{\varepsilon}_a^{(m)} = \max\left\{\check{\varepsilon}_a^{(m-1)}, \|\mathscr{F}(X^{(m-1)})\|\right\}.$$

Remark 3.1. *The optimal choices of the parameters that control the stopping conditions and the restart of GMRES in the NKS solver are problem-dependent. In the numerical tests presented in this paper, FGMRES restarts at every* 30 *iterations. A fixed nonlinear forcing term* $\eta = 10^{-3}$ *is used to control the relative accuracy of the linear solver, the nonlinear tolerances for the Newton iteration are chosen as* $\varepsilon_r = 1.0 \times 10^{-7}$, $\hat{\varepsilon}_a = 1.0 \times 10^{-7}$ *and* $\check{\varepsilon}_a^{(0)} = 1.0 \times 10^{-8}$. *There are other more flexible choices for the nonlinear forcing term [11] that can be used sometimes to obtain more efficient or more robust solutions, but we don't play with these tricks in this paper.*

3.2. Approximate formulations of the Jacobian matrix for nonsmooth functions

There are two discrete operators $\mathscr{F}$ and J in the Newton's equation (10). In a traditional Newton method, $J(X)$ is simply an algebraic derivative of $\mathscr{F}(X)$ under the assumption that $\mathscr{F}(X)$ is everywhere differentiable. However, when a finite volume method is applied to hyperbolic conservation laws, $\mathscr{F}(X)$ usually becomes non-differentiable, at some points, due to the conditional statements in the Riemann solver or the flux/slope limiter. If non-smooth bottom topography is included, then the topographic source term in the SWEs could be discontinuous thus makes $J(X)$ difficult to calculate. Denote $\mathscr{F} = (f_1, f_2, ..., f_d)^T$ and $X = (x_1, x_2, ..., x_d)^T$. To obtain an entry of $\mathscr{F}'(X)$, say $\frac{\partial f_m}{\partial x_n}$, we need to use the chain rule. Without loss of generalities, we assume $\mathscr{F}(X)$ is locally Lipschitz continuous and $f_m(x_n) = p(q(r(x_n)))$. Here one or more of the intermediate functions p, q, r are piecewisely defined as, for instance

$$q(r) = \begin{cases} q_1(r), & \text{if} \quad r \geq r_0, \\ q_2(r), & \text{otherwise.} \end{cases}$$

Although it is possible that $\frac{\partial q}{\partial r}(r_0)$ does not exist, we still can formally freeze it to be $\frac{\partial q}{\partial r}(r_0) = \frac{\partial q_1}{\partial r}(r_0)$. Then by the chain rule we obtain

$$\frac{\partial f_m}{\partial x_n} = \frac{\partial p}{\partial q}\frac{\partial q}{\partial r}\frac{\partial r}{\partial x_n}.$$

The method is based on the definition of the generalized Jacobian in the sense of Clarke [6] and its modification of Qi [27]. Local and global convergence studies on some nonsmooth or semismooth Newton methods can be found in, e.g., [17, 27, 28].

It is easy to see that when $\mathscr{F}$ arises from the discretization of a nonlinear partial differential equation, the accuracy of the numerical solution of the continuous problem is determined solely by the order of discretization employed in $\mathscr{F}$. While the convergence rate of Newton method depends on the well-posedness and the accuracy of the Jacobian matrix, it is independent of the preconditioner M^{-1} used in (11), if we do not consider the impact of the preconditioner on the stopping conditions. In the paper, as mentioned before, $\mathscr{F}$ is obtained by a second-order well-balanced discretization of the SWEs. We refer to the generalized Jacobian of $\mathscr{F}$ as the second-order Jacobian J, and call the first-order Jacobian, denoted as $\tilde{J}$, as the Jacobian corresponding to an $\tilde{\mathscr{F}}$ obtained from a first-order discretization of the SWEs on the same mesh. In practice, the second-order Jacobian J, which is consistent to the nonlinear function $\mathscr{F}$, is used throughout the Newton iterations (10). But the preconditioner M^{-1} in (11), served as a "cheap" approximation of J^{-1}, can be constructed from either J or $\tilde{J}$. Here both Jacobians are analytically calculated by hand.

There are several techniques available to calculate the Jacobian with less programming effort, such the multi-colored finite difference (MCFD) [7] method and the automatic differentiation (AD) [15] method. If a finite difference method is used, the numerical differentiation procedure needs to evaluate $\mathscr{F}$ at certain points repeatedly so that the Jacobian can be approximately calculated. In AD (or more precisely, symbolic differentiation), the Jacobian is generated based on the fact that any computer program calculating $\mathscr{F}$ can be decomposed into a sequence of elementary assignments and thus can be differentiated by the computer using the chain rule. Therefore, both AD and MCFD are in fact more time-consuming and sometimes less accurate compared to the hand-coded method. Some comparison results between using hand-coded method and using MCFD method for the SWEs can be found in [48].

In NKS, the second-order Jacobian is only needed in the form of matrix-vector multiplications, which can be done approximately in a matrix-free manner [20], i.e.,

$$J(X)Y \approx [\mathscr{F}(X+\varepsilon Y) - \mathscr{F}(X)]/\varepsilon\,, \tag{14}$$

where $\varepsilon > 0$ is small. However at least one nonlinear function evaluation is needed for every matrix-vector multiplication, which is costly. On the other hand, once an analytic Jacobian is generated, its matrix-vector multiplication can be done much more economically. The main drawback of the latter method is that extra memory is needed to explicitly store the Jacobian matrix. The Jacobian-free method is potentially more favorable for three-dimensional atmospheric problems since the spatial discretization is much more complicated and the analytic Jacobian matrix may be too expensive to form.

3.3. Several multilevel Schwarz preconditioners with mixed-order discretizations

We decompose all six patches of the cubed-sphere respectively into $p \times p$ non-overlapping subdomains. Each subdomain is then mapped onto one processor. Thus $6p^2$ is the number of processors and also the total number of subdomains. To obtain an overlapping decomposition of the domain, we extend each subdomain Ω_j to a larger subdomain Ω'_j, $j = 1, \cdots, 6p^2$, as shown in Fig 1. Assume the sizes of Ω_j and Ω'_j are respectively $H_x \times H_y$ and $H'_x \times H'_y$. Then the overlapping size is $\delta = (H'_x - H_x)/2 = (H'_y - H_y)/2$. Since the size of each patch is $\pi/2 \times \pi/2$, the logical length of subdomain Ω_j is then $H_x = H_y = \pi/(2p)$.

For each overlapping subdomain we define B_j as the restriction of the second-order Jacobian J to the overlapping subdomain Ω'_j. Then we can define a one-level restricted additive Schwarz (RAS, [4, 42]) preconditioner

$$M_{one}^{-1} = \sum_{j=1}^{6p^2} (R_j^0)^T B_j^{-1} R_j^\delta. \tag{15}$$

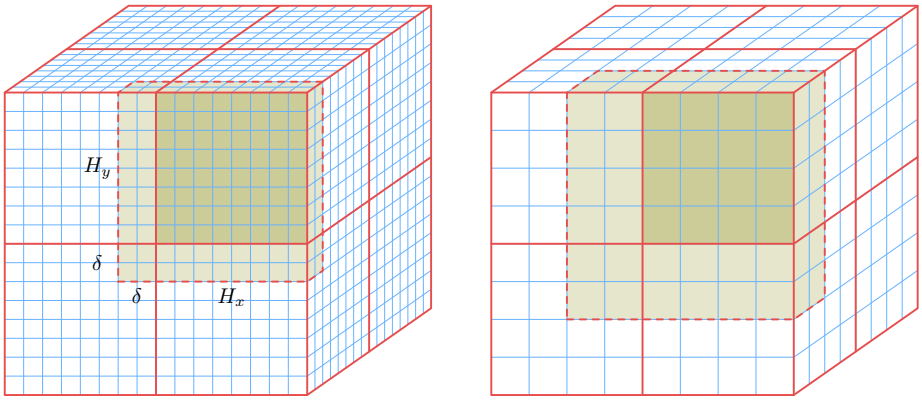

Figure 1: Domain decompositions of the cubed-sphere with overlaps. The red solid lines indicate the partition of the domain into $24 = 6 \times 2^2$ non-overlapping subdomains of size $H_x \times H_y$, the dotted lines show the extended boundary of an overlapping subdomain. Left figure: fine mesh with mesh size $\hbar$ and overlap $\delta = 2\hbar$. Right figure: coarse mesh of size $2\hbar$ and overlap $\delta = 2(2\hbar) = 4\hbar$.

In the following context, by RAS(δ) we mean the RAS preconditioner with overlapping factor δ. Let m be the total number of cells and m'_j the total number of cells in Ω'_j. Then, R_j^δ is an $m'_j \times m$ block matrix that is defined as: its 3×3 block element $(R_j^\delta)_{p_1,p_2}$ is an identity block if the integer indices $1 \le p_1 \le m'_j$ and $1 \le p_2 \le m$ belong to a cell in Ω'_j, or a block of zeros otherwise. The matrix R_j^δ serves as a restriction operator because its multiplication by a block $m \times 1$ vector results in a smaller $m'_j \times 1$ block vector by dropping the components corresponding to cells outside Ω'_j. In particular, R_j^0 in (15) is a restriction to the non-overlapping subdomain. RAS(0) reduces to a block-Jacobi preconditioner.

When a first-order spatial discretization is used, the one-level RAS preconditioner (15) was found to be robust [48]. Observations were made in [47] that the one-level RAS preconditioner constructed based on the first-order scheme is still efficient when the spatial discretization is second-order, even better than the preconditioner built directly from the second-order scheme in some cases. Denote $\tilde{B}_j$ as the restriction of the first-order Jacobian $\tilde{J}$ to the overlapping subdomain Ω'_j. The new one-level RAS(δ) preconditioner is thus defined by:

$$\tilde{M}_{one}^{-1} = \sum_{j=1}^{6p^2} (R_j^0)^T \tilde{B}_j^{-1} R_j^\delta. \tag{16}$$

Sparse LU or incomplete LU (ILU) factorizations can be used to obtain the inverse or an approximate inverse of the subdomain Jacobian. Our experiments show that the regular pointwise incomplete factorizations don't work well for these highly ill-conditioned Jacobian matrices. We use some point-block versions of ILU that keeps the coupling between all physical components of each mesh point. This is

essential for the success of our fully coupled solver.

To improve the scalability of the one-level RAS preconditioner (16), especially when a large number of processors is used, we employ a hybrid preconditioner (see, [21]) by composing the one-level additive Schwarz preconditioner B_f with a coarse-level preconditioner B_c in a multiplicative manner

$$M_{two}^{-1} = \underset{J_f}{\text{hybrid}}\left(B_c, B_f\right) = B_c + B_f - B_f J_f B_c, \tag{17}$$

where $B_c = \mathscr{I}_c^f J_c^{-1} \mathscr{I}_f^c$, and $\mathscr{I}_f^c$ and $\mathscr{I}_c^f$ are restriction and prolongation operators mapping between vectors defined on fine level and coarse level. On the coarse level, a smaller linear system associated with the Jacobian matrix J_c is then solved for each application of the two-level preconditioner (17) by using GMRES with a relative tolerance η_c. The coarse level preconditioner can be either one-level (16) or two-level (17).

More precisely speaking, in the hybrid two-level preconditioner, we first apply a coarse mesh preconditioning

$$w = \left(\mathscr{I}_c^f J_c^{-1} \mathscr{I}_f^c\right) x, \tag{18}$$

and then correct the coarse solution by adding (without overlap) the fine level solution from each overlapping subdomain

$$y = w + \left(\sum_{j=1}^{6p^2} (R_j^0)^T \tilde{B}_j^{-1} R_j^\delta\right)(x - J_f w). \tag{19}$$

We note that in the traditional multiplicative Schwarz or the V-cycle multigrid approach, the operation (19) is also applied before the coarse mesh preconditioning (18), but our experiments suggest that there is no benefit to include the second swipe of the one-level preconditioning for the implicit solution of the SWEs. We also remark that a similar performance can be observed if we switch the order of (18) and (19). However, the pure additive version of the two-level approach performs considerably worse than the hybrid methods.

The best choices for some of the options in the multilevel RAS preconditioner defined recursively in (17) are problem-dependent. In the current study, we use a three-level version with a coarse-to-fine mesh ratio 1 : 2 in each direction. On the finest level ($N \times N \times 6$), the preconditioner is

$$M_N^{-1} = \underset{J_N}{\text{hybrid}}\left(\mathscr{I}_{N/2}^N J_{N/2}^{-1} \mathscr{I}_N^{N/2}, \sum_{j=1}^{6p^2} \left[((R_N)_j^0)^T (\tilde{B}_N)_j^{-1} (R_N)_j^{2\hbar}\right]\right), \tag{20}$$

where the subdomain solver of $(\tilde{B}_N)_j^{-1}$ is ILU(2) factorization of the first-order Jacobian on this level. Here a linear average restriction operator $\mathscr{I}_N^{N/2}$ and a piecewise constant interpolation operator $\mathscr{I}_{N/2}^N$ are used due to their simplicities. Then the Jacobian system of $J_{N/2}$ on the second level is solved by an inner GMRES, preconditioned by one of the following methods.

(1) A one-level RAS($\delta_{N/2}$) preconditioner on the coarse level

$$M_{N/2}^{-1} = \sum_{j=1}^{6p^2} \left[(R_{N/2})_j^0)^T (\tilde{B}_{N/2})_j^{-1} (R_{N/2})_j^{\delta_{N/2}} \right]. \tag{21}$$

This results in a two-level method.

(2) Another two-level RAS($\delta_{N/2}$) preconditioner on the coarse level

$$M_{N/2}^{-1} = \underset{J_{N/2}}{\text{hybrid}} \left(\mathscr{I}_{N/4}^{N/2} M_{N/4}^{-1} \mathscr{I}_{N/2}^{N/4}, \sum_{j=1}^{6p^2} \left[(R_{N/2})_j^0)^T (\tilde{B}_{N/2})_j^{-1} (R_{N/2})_j^{\delta_{N/2}} \right] \right), \tag{22}$$

resulting in a three-level method. Here on the third level, instead of solving the Jacobian system iteratively, we apply the RAS($\delta_{N/4}$) preconditioner

$$M_{N/4}^{-1} = \sum_{j=1}^{6p^2} \left[(R_{N/4})_j^0)^T (\tilde{B}_{N/4})_j^{-1} (R_{N/4})_j^{\delta_{N/4}} \right]. \tag{23}$$

Choosing the right subdomain solver at each level is very important for the overall performance of the preconditioner on a specific computer with the given amount of memory and cache. A large number of numerical experiments is often necessary to identify the right selection. For example, for the SWEs on an IBM BG/L, we use ILU(2) subdomain solvers for (22) and LU for (21), (23).

For a hyperbolic system like the SWEs, there is very little theoretical work on the convergence of domain decomposition methods, see for instance [10, 45]. If a classical additive Schwarz preconditioner is applied to solve an elliptical problem, the condition number of the preconditioned system satisfies

$$\kappa \leq C(1 + H/\delta)/H^2 \tag{24}$$

and

$$\kappa \leq C(1 + H/\delta) \tag{25}$$

for one-level and two-level methods respectively [36, 42]. Here H is the subdomain size and C is independent of H, δ and $\hbar$. Suppose each processor is assigned with one subdomain, then the number of processors is in proportion to $1/H^2$. If the discretized system is solved using a Krylov-type method with

a fixed relative tolerance, the required iteration number is approximately in proportion to $1/H$ when a one-level method is used and this dependency can be removed by using coarse levels. However, these condition number estimates do not apply to the SWEs. Some numerical results in [48] suggest that the condition number growth of the Jacobian system preconditioned by the one-level RAS method for the SWEs is less severe than for the elliptic case. The strong and weak scaling tests in this paper will provide more understanding of the one-level and the multilevel methods for the SWEs.

4. Numerical results

The NKS algorithms described in the previous sections are implemented with PETSc [1]. Each of the six patches corresponds to one MPI group, which is handled by a PETSc Distributed Array (DA) object for parallel implementation. The DA object in PETSc, which contains the parallel data layout and communication information, is intended for use with a logically rectangular mesh to optimize the communication of vector operations among distributed processors. To take advantage of DA in PETSc, the six patches of the cubed-sphere can either be put on top of each other as a “thick" DA with $3 \times 6 = 18$ degrees of freedom per mesh point [48] or side by side with six DAs and only 3 degrees of freedom per mesh point for each patch. No significant difference is observed between the parallel performance of the two strategies. In the current study we choose to use the latter method; the six patches are coupled with each other via interface conditions on patch boundaries. The numerical tests are carried out on an IBM BlueGene/L with 4096 dual-processor compute nodes. Each node contains 512 MB of memory shared by two Power PC 440 processors, and there is a dual Floating Point Unit (FPU) on each processor. In our tests, the BG/L works in the virtual-node mode so that both processors within each node are available for computing. The second FPU is deactivated by using compiling option `-qarch=440`, thus the theoretical peak performance of each processor is 1.4 GFLOPS.

4.1. Accuracy of the fully implicit solver

In this subsection, we study the accuracy and the conservative properties of the newly introduced scheme and compare them with an explicit scheme for two benchmark problems.

4.1.1. Rossby-Haurwitz wave

The four-wave Rossby-Haurwitz problem is the 6th test case in the Williamson benchmark set [44]. It serves as a good tool for middle-term test, although it is not an analytic solution of the SWEs. The

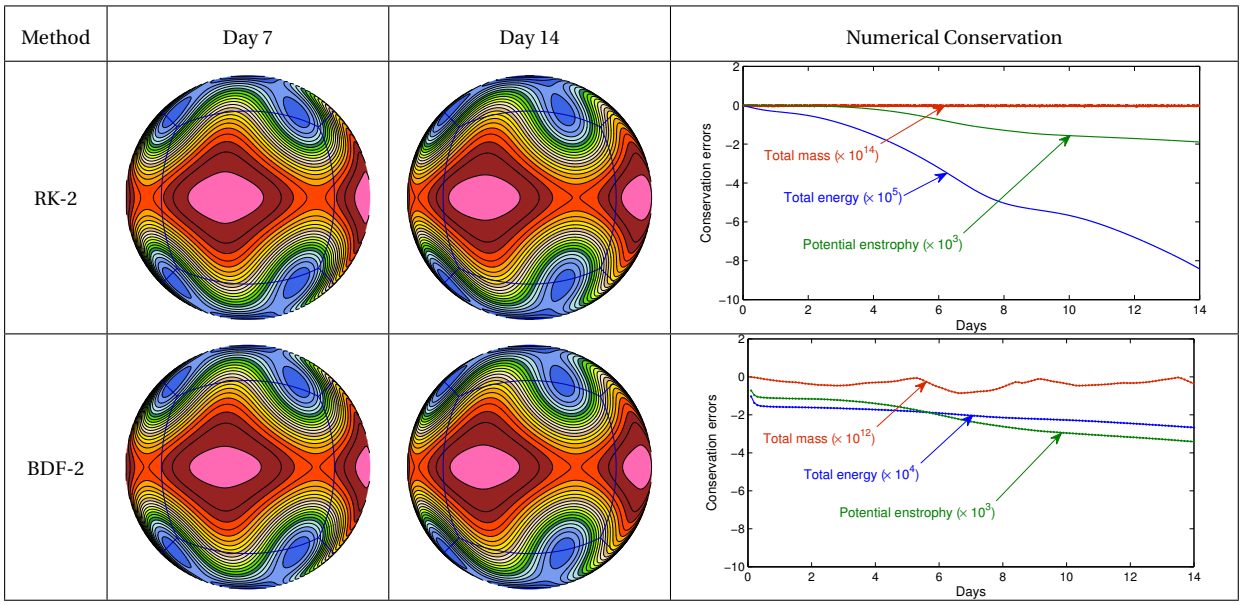

Figure 2: Height field contours of the Rossby-Haurwitz problem at day 7, 14 and the numerical conservation histories of the total mass, total energy, and potential enstrophy. The calculations are done on a $128 \times 128 \times 6$ mesh with 96 processors. The results using SSP RK-2 with CFL = 0.5 are shown in the top row and the results using BDF-2 with $\Delta t = 0.1$days (CFL ≈ 55) are in the bottom row. The contour levels are from 8100m to 10500m with an interval of 100m. The innermost lines near to the equator are at 10500m.

detailed initial conditions for the Rossby-Haurwitz wave can be found in [44]. Numerical solutions obtained using the explicit SSP RK-2 method and the implicit BDF-2 method are provided in the upper and lower rows of Fig 2 respectively. The numerical results are consistent with the reference solutions in [18] and the implicit results agree well with the explicit results.

Numerical conservation is crucial in the simulation of the SWEs. The normalized conservation error at time t is measured as $\delta(\cdot) = [I(\cdot,t) - I(\cdot,0)]/I(\cdot,0)$, where I is the discrete integral operator

$$I(M) = \sum_{k=1}^{6} \sum_{i,j=1}^{N} \left(\Lambda_{ij} M_{ij}^{k} \right). \tag{26}$$

The integral invariants of interest to us are the total mass $\delta(h)$, the total energy $\delta(E)$ and the potential enstrophy $\delta(\xi)$, where

$$E = \frac{h}{2}\mathbf{v}\cdot\mathbf{v} + \frac{g}{2}(Z^2 - b^2) = \frac{\Lambda^2 h}{2}(g^{11}v^2 + g^{22}u^2 - 2g^{12}uv) + \frac{g}{2}(Z^2 - b^2),$$
$$\xi = \frac{1}{2\Lambda h}\left\{ \frac{\partial}{\partial x}[\Lambda^2(g^{11}v - g^{12}u)] - \frac{\partial}{\partial y}[\Lambda^2(g^{22}u - g^{12}v)] + \Lambda f \right\}^2.$$

Conservation evolutions of the three integral invariants can be found in the right panel of Fig 2, which shows that the numerical conservation of the total mass is to the machine precision when the explicit

time integration is used, and around 10^{-12} when the fully implicit method is used instead. The slight loss of mass conservation comes mainly from the inexact solution of the nonlinear system arising from each implicit time step. The accumulation of mass loss is not observed in Fig 2, which is crucial for long-term simulations. The result is quantitatively, in terms of the conservation error, better than those reported in [32] where the finite volume scheme is not well-balanced, and comparable to those presented in [38] where a SISL method is applied. The numerical conservations of the other two integral invariants are also satisfactory.

4.1.2. Isolated mountain

Test case 5 in [44] describes a zonal flow over an isolated mountain. The case is obtained by a modification of a steady-state geostrophic flow by adding a compactly supported, conical mountain to the domain. In this test the flow is purely zonal ($\alpha = 0$) with initial parameters $h_0 = 5960$m and $u_0 = 20\text{m}\cdot\text{s}^{-1}$. The mountain is centered at $(\lambda_c, \theta_c) = (-\pi/2, \pi/6)$ with height $b = b_0(1 - r/r_0)$, where $r_0 = 2000$m, $r_0 = \pi/9$ and $r = \min\{r_0, \sqrt{(\lambda-\lambda_c)^2 + (\theta-\theta_c)^2}\}$. Note that the mountain height is not continuously differentiable at the center and on the boundary of the cone thus the topography term has discontinuous coefficients. This discontinuity in the topography term could result in spurious oscillations [14] if the numerical scheme is not well-balanced.

Results using the explicit SSP RK-2 method and the implicit BDF-2 method are shown in the left and the right columns of Fig 3 respectively. As no analytical solution is available for this test, we compare the results with a reference solution obtained using a spectral method with high resolution in [18]. We find that no spurious oscillations are observed in both the explicit and implicit solutions and the numerical conservation histories are comparable to those in the first test case. The explicit solution is in agreement with the reference solution, and the implicit solution reproduces most details except for some small discrepancies near the equator (e.g., the 5950m contour line on day 15 is missing). In Fig 4 we provide a result on day 15 using the same implicit method but with a time step that is comparable to the CFL limit. The result in Fig 4 agrees perfectly with the explicit result, indicating that the discrepancies in the implicit solution in Fig 3 are due to the much larger time step size.

4.2. Impact of the discretization order of J, and choices of solver options

Larger overlap between subdomains or larger fill-in of ILU factorization of subdomain matrices certainly helps reduce the total number of linear iterations as the number of processors increases. However,

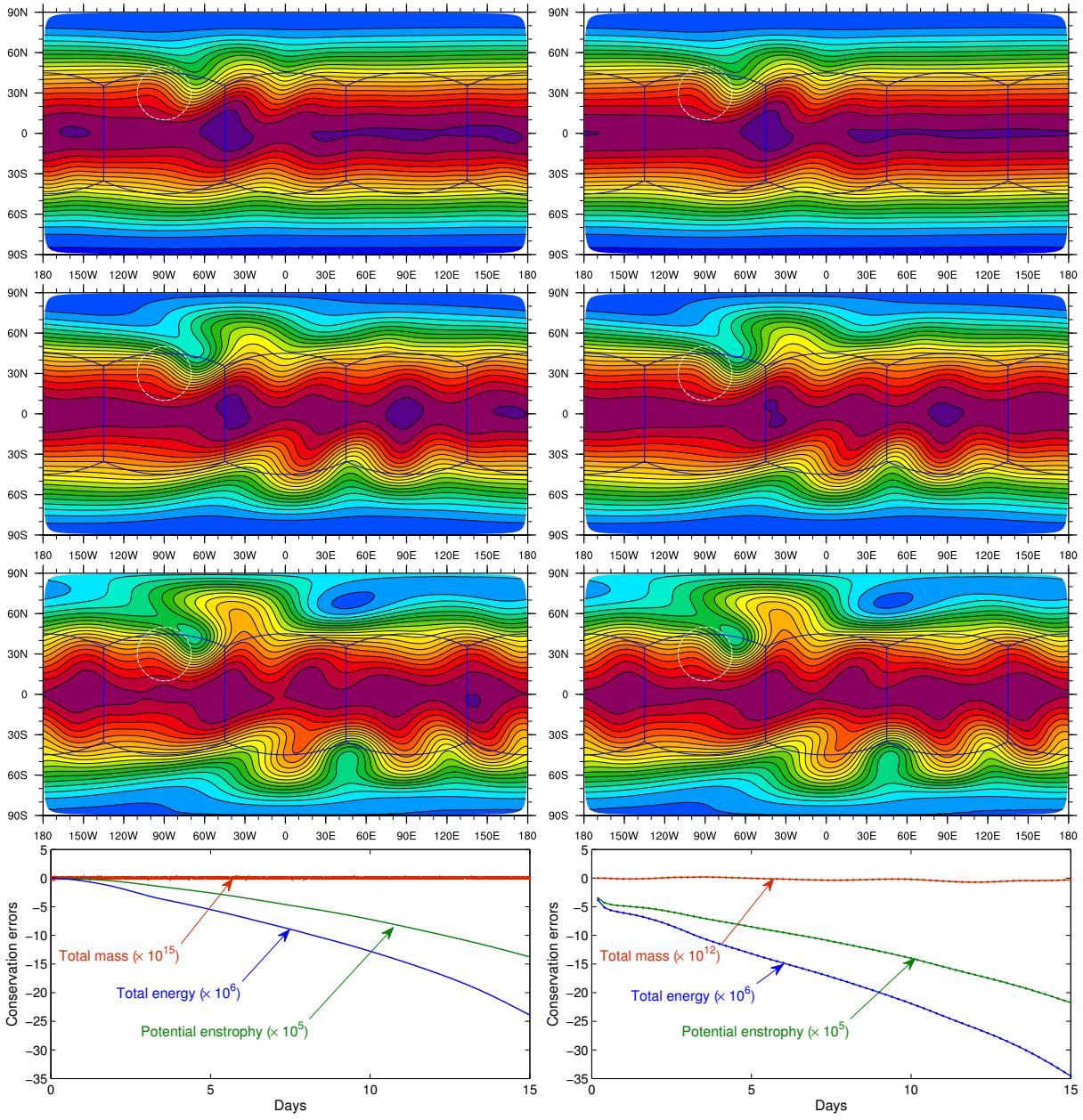

Figure 3: Height field contours of the isolated mountain problem at day 5, 10, 15 and the numerical conservation histories of the total mass, total energy, and potential enstrophy. The calculations are carried out on a $128\times128\times6$ mesh with 96 processors. The results using SSP RK-2 with CFL = 0.5 are listed in the left column and the results using BDF-2 with $\Delta t = 0.2$days (CFL ≈ 79) are in the right column. The contour levels are from 5000m to 5950m with an interval of 50m. The innermost lines near to the equator are at 5950m.

the memory complexity and the amount of floating-point operations are both increased at every linear iteration. When a first-order spatial discretization is used for the SWEs, our previous tests in [48] indi-

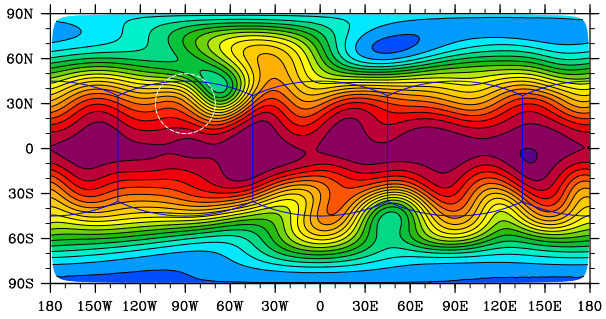

Figure 4: Height field contours of the isolated mountain problem at day 15. The calculations are carried out on a $128 \times 128 \times 6$ mesh with 96 processors by using BDF-2 with $\Delta t = 3.6$minutes (CFL ≈ 1). The contour levels are from 5000m to 5950m with an interval of 50m. The innermost lines near to the equator are at 5950m.

cate that zero overlap and complete LU factorization are the optimal choice in terms of the total compute time. The situation becomes totally different when we switch to a second-order spatial discretization. For the Rossby-Haurwitz problem, our recent work in [47] suggests: (1) Poor results are observed if the one-level RAS preconditioner is constructed directly from the second-order Jacobian matrices; (2) Overlap $\delta = 2\hbar$ and ILU(3) or ILU(2) applied to the subdomain matrices from the first-order Jacobians give better results in terms of the total compute time.

In the following tests, both LU and ILU(2) subdomain solvers are considered and the overlap size is fixed to $\delta = 2\hbar$ which is found to be optimal in most cases. The second-order Jacobian matrices are generated analytically and the performance is compared with a Jacobian-free method. Some results using second-order Jacobian matrix to construct the RAS preconditioner as in (15) are also provided.

We experiment with the isolated mountain problem using a fixed $576 \times 576 \times 6$ mesh (5.97 million unknowns) and a fixed time step size $\Delta t = 0.2$days. It is worth pointing out that although the time step size in the fully implicit method is no longer constrained by any stability conditions, it still can not be arbitrarily large due to dynamical timescale limits [13, 22]. We test the one-level fully implicit solver with various parameters. Performance results on the average number of GMRES iterations per Newton step and the total compute time are given in Fig 5 from which we have the following observations.

(1) We see much better performance when the RAS preconditioner is constructed from a first-order discretization of the Jacobian (22) than from a second-order discretization that is the same as in the nonlinear function evaluation. The trade-off between exact and inexact subdomain solver is interesting. Although the number of iterations nearly doubles when we switch from LU to ILU(2), the total compute time is actually shorter. However, the speed advantage of ILU(2) becomes less obvious when the number

of processors becomes large (i.e., the size of the subdomain matrix per processor is much smaller.)

(2) Compared with explicitly generating the second-order Jacobian, the Jacobian-free method provides sufficiently accurate approximation to the Jacobian-vector multiplication thus gives similar GMRES iteration counts, yet the compute time is longer. Although using the second-order Jacobian in the construction of the RAS preconditioner (15) provides better iteration counts, the compute speed is much slower because the subdomain matrices are a lot denser. Besides, the number of iterations increases very rapidly as the number of processors increases. We have also tried to use ILU as the subdomain solver in (15), but find that GMRES fails to converge in most situations.

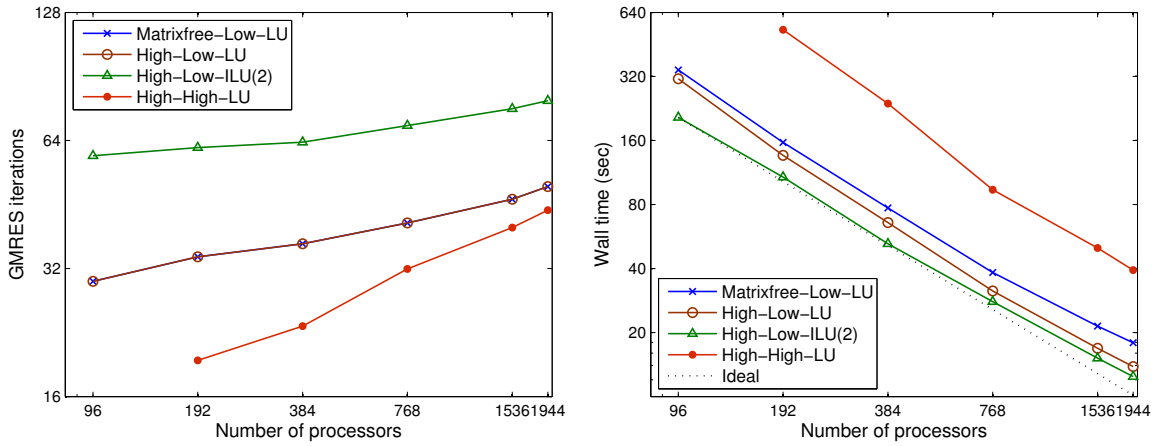

Figure 5: Performance comparisons with various solver options for the isolated mountain problem on a fixed $576\times576\times6$ mesh with time step size $\Delta t = 0.2$days. Left panel: the averaged number of GMRES iterations per Newton step. Right panel: the total compute time per time step. Here `High-Low-LU` or `High-Low-ILU(2)` means using hand-coded method to generate both the second-order and first-order Jacobian matrices and construct the RAS preconditioner based on the first-order Jacobian with LU or ILU(2) as subdomain solvers, `Matrixfree` means using the Matrix-free method and `High-High` means the RAS preconditioner is also based on the second-order Jacobian.

4.3. Performance and robustness

We focus on the isolated mountain problem in this subsection. The performance of the fully implicit solver is tested and compared by using the abbreviations listed in Table 1. Both the second-order and the first-order Jacobians are evaluated analytically using the method introduced in Section 3.2 and the latter is used to construct the RAS preconditioner.

First we study the strong scalability of the algorithms by using a fixed mesh $1152\times1152\times6$ (23.89 million unknowns) and also a fixed time step size $\Delta t = 0.2$days. As the number of processor (np) increases,

Table 1: Preconditioners used in the NKS solver. For multilevel preconditioners, the respective values of each column are given from the finest level to the coarsest level . The $\star$ stands for the directly application of preconditioner without iteration.

Short name	#Levels	η	δ	Jacobian	Subdomain solver
1-ILU	one-level	0.001	$2\hbar$	analytic	ILU(2)
1-LU	one-level	0.001	$2\hbar$	analytic	LU
2-level	two-level	0.001, 0.1	$2\hbar$, 0	analytic	ILU(2), LU
3-level	three-level	0.001, 0.1, $\star$	$2\hbar$, $4\hbar$, 0	analytic	ILU(2), ILU(2), LU

Table 2: Strong scaling results on the isolated mountain problem, fixed $1152 \times 1152 \times 6$ mesh, $\Delta t = 0.2$days, the simulation is stopped at day two. The averaged number of inner iterations on the second level is shown in brackets.

np	Newton/Δt				GMRES/Newton/Δt				Wall-clock time (sec)			
	1-ILU	1-LU	2-level	3-level	1-ILU	1-LU	2-level	3-level	1-ILU	1-LU	2-level	3-level
384	3.0	3.0	3.0	3.0	116.8	46.5	15.1 (9.7)	15.5 (6.8)	347.7	378.7	252.8	217.9
768	3.0	3.0	3.0	3.0	123.5	51.6	15.1 (10.9)	15.4 (7.2)	178.8	171.7	128.7	113.6
1536	3.0	3.0	3.0	3.0	127.4	57.3	14.7 (12.2)	14.9 (7.5)	89.7	86.5	62.4	57.7
3072	3.0	3.0	3.0	3.0	146.0	71.1	14.6 (13.9)	14.6 (8.1)	50.7	46.1	31.2	32.1
3456	3.0	3.0	3.0	3.0	142.5	70.8	14.3 (14.3)	14.4 (8.3)	45.9	42.7	29.2	27.0
6144	3.0	3.0	3.0	3.0	161.0	88.8	14.1 (16.4)	14.2 (9.0)	28.0	27.3	17.9	17.5
7776	3.0	3.0	3.0	3.0	172.4	98.4	14.1 (17.5)	14.2 (9.5)	23.6	23.5	15.1	15.3

the total compute time should be reduced proportionally in the ideal situation. Results on Newton and GMRES iterations as well as compute times using the fully implicit solvers are provided in Table 2. The table clearly indicates that the number of Newton iterations remains to be independent of np, and the number of GMRES iterations depends on the preconditioner employed in the solver. For the one-level solver, the number of GMRES iterations suffers as np increases, but the increase of iteration numbers is much smaller than in the case of elliptic equations. Compared with ILU(2), LU subdomain solver helps in reducing the number of GMRES iterations and outperforms ILU(2) in the case when np is large. With the multilevel preconditioners, the number of GMRES iterations is kept to a low level and even slightly decreases as np increases.

For further comparisons and analysis, we present some results in terms of the total compute time and the sustained FLOPS in Fig 6. The results using the explicit SSP RK-2 method are also provided for comparison. We observe that, in terms of the total compute time, the two-level method is about

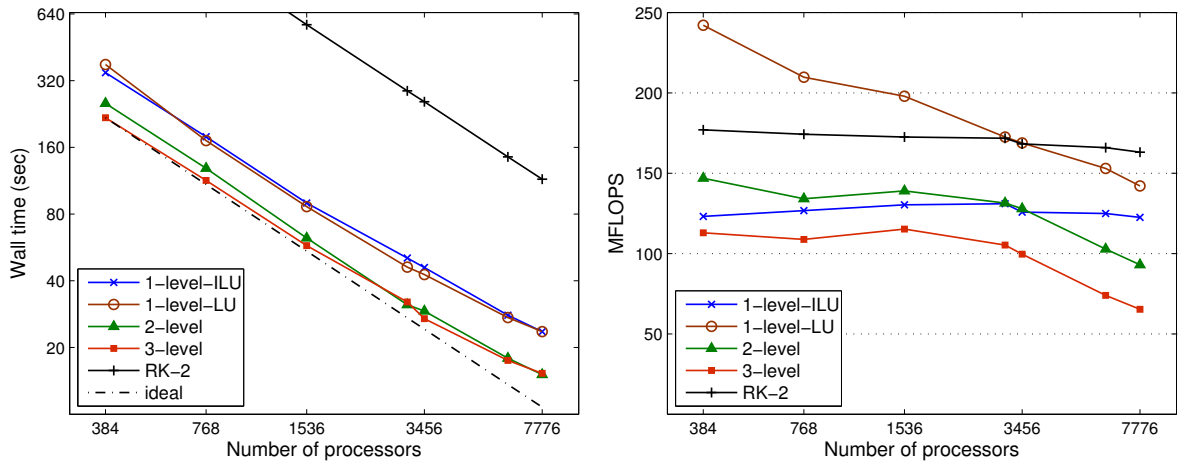

Figure 6: Plots on the total compute time (left) and sustained MFLOPS per processor (right) for the strong scaling tests in Table 2. The results using an explicit RK-2 method are also provided. The left panel is drawn in the log-log scale so that the ideal speedup curve appears as a straight line.

33 – 55% faster than the one-level, and the three-level method is another 5 – 15% faster if np is not too large. The explicit method is about 34 times slower than the three-level method, although the scalability is nearly ideal. For the fully implicit solver, the efficiency measured in the total compute time remains above 85% as np increases to 3456, but slightly decrease when np becomes larger because the amount of computations on each processor is too small. As np tends to 7776, the scalability efficiency of the fully implicit solver with different numbers of levels are respectively 72.8%, 79.2%, 82.7% and 70.3%. In terms of the FLOPS rate, the fully implicit solver is comparable to the explicit solver. The decrease of the FLOPS when $np > 3456$ is due to the insufficient work load per processor.

It is worth pointing out that neither scalability nor FLOPS tells the whole story. For example, the explicit method scales perfectly but is in fact far slower than the fully implicit method; the three-level method has the lowest FLOPS rate yet is the fastest among all methods. If the Jacobian-free method is used instead of generating the second-order Jacobian by hand, the FLOPS rate can be improved by about 50%, but the total compute time becomes longer as shown in Section 4.2.

In strong scaling tests, because the problem size is fixed, np can neither be too small (insufficient memory for the problem) nor too large (insufficient work load per processor). Therefore we further test our algorithm in terms of the weak scalability in which we refine the mesh as np increases so that the same number of unknowns per processor is maintained. We start with a small $48 \times 48 \times 6$ mesh for $np = 6$ and end up with a large $1728 \times 1728 \times 6$ mesh (53.75 million unknowns) for $np = 7776$. The time step size

is fixed to $\Delta t = 0.2$days for all tests. We show the weak scaling results on the number of iterations and the total compute times of the fully implicit solver in Table 3. The total compute time and the sustained FLOPS are then plotted in Fig 6 which also includes comparisons with the explicit SSP RK-2 method.

Table 3: Weak scaling results on the isolated mountain problem with a fixed 48×48 mesh per processor, $\Delta t = 0.2$days, the simulation ends at day two. The averaged number of inner iterations on the second level is shown in brackets. The * means divergence of GMRES.

np	Newton/Δt				GMRES/Newton/Δt				Wall time (sec)			
	1-ILU	1-LU	2-level	3-level	1-ILU	1-LU	2-level	3-level	1-ILU	1-LU	2-level	3-level
6	3.0	3.0	3.0	3.0	11.7	10.0	7.7 (3.6)	7.7 (3.0)	9.5	15.3	13.2	13.8
24	3.0	3.0	3.0	3.0	17.0	13.8	8.6 (4.6)	8.8 (3.4)	11.2	16.9	14.5	15.1
96	3.0	3.0	3.0	3.0	26.7	20.4	10.0 (5.9)	10.0 (4.1)	14.3	20.3	16.6	17.0
384	3.0	3.0	3.0	3.0	47.7	31.9	11.9 (7.8)	12.1 (5.0)	20.4	25.6	20.1	20.1
864	3.0	3.0	3.0	3.0	68.9	41.3	12.9 (9.6)	13.2 (5.8)	25.1	29.4	21.7	22.5
1536	3.0	3.0	3.0	3.0	91.5	50.5	13.6 (11.2)	13.7 (6.7)	30.7	33.1	23.7	24.6
1944	3.0	3.0	3.0	3.0	103.4	55.3	13.8 (12.0)	13.9 (7.0)	33.9	35.0	24.7	25.4
2904	3.0	3.0	3.0	3.0	129.0	65.5	14.1 (13.6)	14.2 (7.9)	41.1	39.9	27.2	26.6
4056	3.0	3.0	3.0	3.0	156.3	76.3	14.4 (15.1)	14.6 (8.7)	48.4	44.7	29.4	27.5
6144	3.0	3.0	3.0	3.0	207.8	93.5	14.8 (17.1)	15.0 (9.8)	61.5	52.3	31.7	29.1
7776	*	3.0	3.0	3.0	*	105.6	15.1 (18.5)	15.2 (10.6)	*	57.7	34.3	31.1

As the mesh resolution doubles in each direction, the required number of time steps in the explicit method grows proportionally due to the CFL limit. So even in the ideal situation, the explicit method is highly non-scalable in terms of the weak scalability. For the fully implicit solver, as shown in Table 3, although the mesh size does not affect the number of Newton iterations, the number of GMRES iterations suffers because the condition number of the preconditioned Jacobian becomes worse as the mesh is refined. For the one-level method, the number of GMRES iterations grows rapidly with np, though slower than in the elliptic case, but faster than in strong scaling tests. In the one-level solvers, LU subdomain solver is more stable than ILU(2) and the latter sometimes results in GMRES failure when np is large. Adding coarse levels does reduce the number of GMRES iterations but unlike that in the strong scaling tests, the dependency on np is not removed. We believe that there is an extra term like $\log(1/\hbar)$ exists if there holds a theoretical condition number estimate for the SWEs analogous to (25) for elliptic problems.

All the methods tested in the weak scaling study maintain nearly constant FLOPS rates. However,

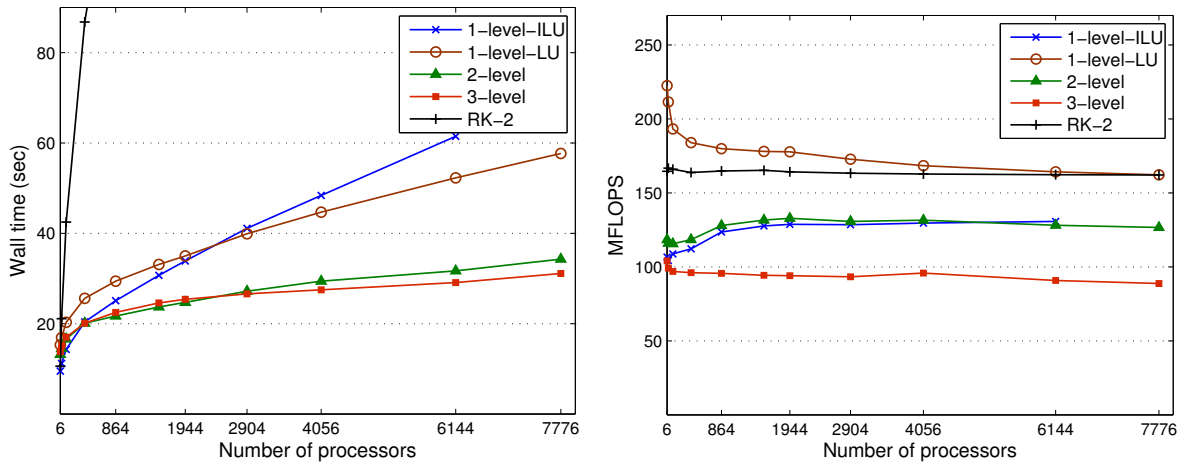

Figure 7: Plots on the total compute time (left) and sustained MFLOPS per processor (right) for the weak scaling tests in Table 3. The results using the SSP RK-2 method are also provided.

in terms of the total compute time, none of them stays near the perfect scaling curve. For $np = 7776$, the two-level method is about 68% faster than the one-level method and the three-level method offers another 17% speedup. If the explicit RK-2 method is used instead, the compute time increases from 10.6s ($np = 6$) to 395.1s ($np = 7776$), which is over 10 to 30 times slower than the three-level method. The compute time curve in Fig 7 of the three-level method is flatter compared to other methods. For the three-level method, the compute time only increases by 125%, as np increases from 6 to 7776 (1296 times larger).

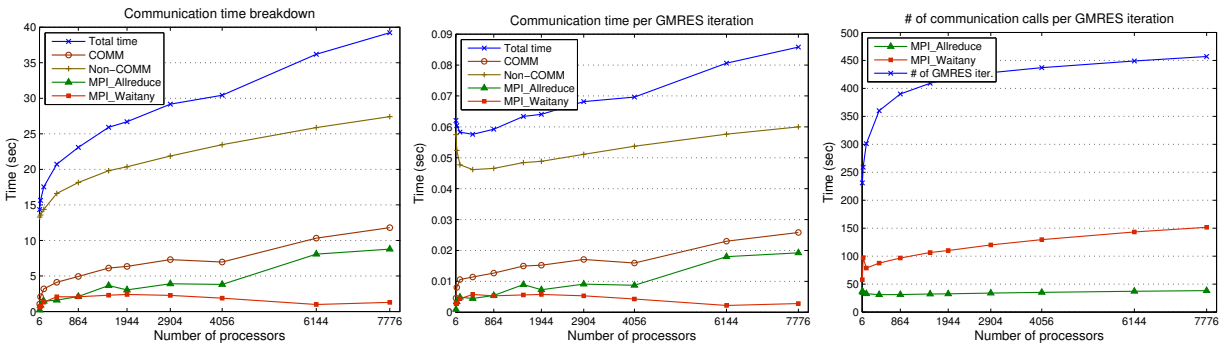

Figure 8: Communication statistics of the three-level solver. Left panel: communication time breakdown. Middle panel: communication time per GMRES iteration. Right panel: number of communication calls per GMRES iteration.

The communication statistics of the three-level method in the preceding weak-scaling test are given in Fig 8, where we can see that the communication time increases slowly as the number of processors

becomes larger and most of the communication time takes place in `MPI_Allreduce` and `MPI_Waitany`. In terms of the wall clock time, the `MPI_Waitany` part is scalable which suggests that the work load on different processors is well balanced. However, as shown in the second panel, the time spent by `MPI_Allreduce` is not scalable even after averaged by the total number of GMRES iterations, although we observe from the third panel that the number of `MPI_Allreduce` calls per GMRES iteration is ideal. In other words, as the number of processors increases the time spent by each `MPI_Allreduce` call becomes larger and thus degrades the overall scalability of the solver.

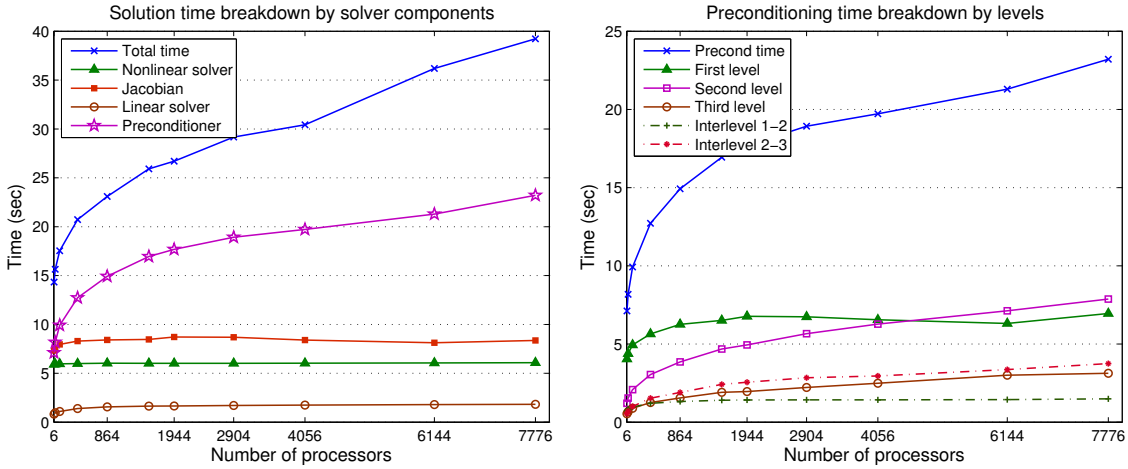

Figure 9: Left panel: Solution time breakdown by solver components. Right panel: Preconditioning time breakdown by levels.

The solution times of the three-level method divided by solver components are provided in the left panel of Fig 9. It clearly indicates that all components are scalable except for the preconditioner which is in fact the dominant part of the solver. The compute time spent on each level of the preconditioner is provided in the right panel of Fig 9. We observe that the scalability on the finest level is acceptable and the time spent on the coarsest level is small, thus the major non-scalable part lies within the second level.

Next, to show the robustness of the fully implicit solver and the unconditional stability of the time integration scheme, we investigate the three-level method for solving the same problem with different time step sizes. The results in Table 4 clearly indicate that the combination of Newton, FGMRES, three-level RAS, and point-block ILU works well for even very large value of Δt.

Table 4: Robustness of the three-level method for various values of Δt, the isolated mountain problem with 1944 processors, the simulation is stopped at day one. The averaged number of inner iterations on the coarse level is shown in brackets.

		576 × 576 × 6 Grid			1152 × 1152 × 6 Grid		
Δt	Steps	Newton/Δt	GMRES/Newton/Δt	Wall time (sec)	Newton/Δt	GMRES/Newton/Δt	Wall time (sec)
0.01	100	3.0	6.00 (2.77)	63.24	3.0	7.12 (3.01)	239.90
0.05	20	3.0	8.82 (4.08)	15.68	3.0	10.90 (4.31)	60.26
0.10	10	3.0	10.80 (4.71)	9.04	3.0	12.63 (5.58)	34.84
0.20	5	3.0	12.80 (6.58)	5.65	3.0	14.60 (7.67)	21.32
0.50	2	3.5	14.08 (11.7)	3.81	3.5	15.13 (14.0)	13.95
1.00	1	4.0	18.00 (21.6)	3.99	4.0	17.75 (26.3)	13.86

4.4. Performance with larger processor count and multi-threaded subdomain solve

Finally, we provide some preliminary studies of the algorithm on a new supercomputer, Tianhe-1A, equipped with 14,336 six-core Xeon X5670 CPUs. In the tests we disable all GPUs and assign one MPI process to each six-core CPU, corresponding to one subdomain. Multi-threading is used within each MPI process to fully utilize the six cores in each subdomain solve. The isolated mountain problem is selected in the tests with a fixed time step size of $\Delta t = 0.2$days for the BDF-2 scheme. The three-level method depicted in Table 1 is employed.

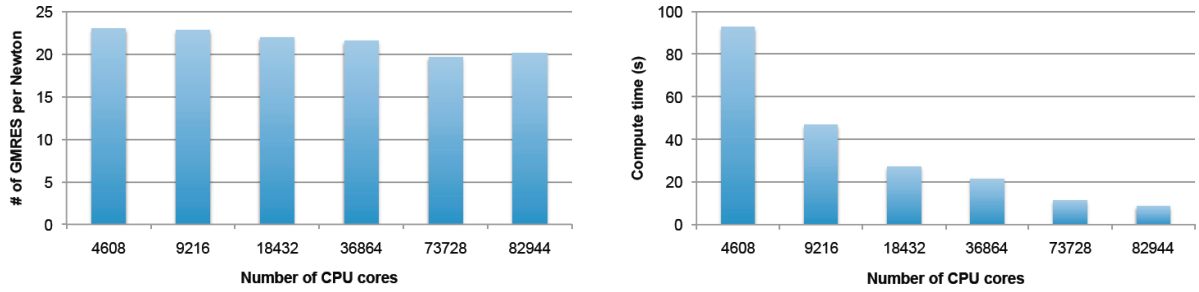

Figure 10: Strong scaling results of the three-level method on Tianhe-1A for the isolated mountain problem. The averaged number of linear iterations per Newton step is shown in the left panel and the total compute time is in the right.

In the strong scaling test, we use a fixed $6144 \times 6144 \times 6$ mesh (0.68 billion unknowns) and the results are shown in Fig 10. It can be seen that as the number of CPU cores increases the averaged number of linear iterations slightly decreases, analogous to the results in Table 2. The total compute time is saved by more than 90% as the number of CPU cores increases from 4608 to 82944, indicating that the overall parallel efficiency is about 60%.

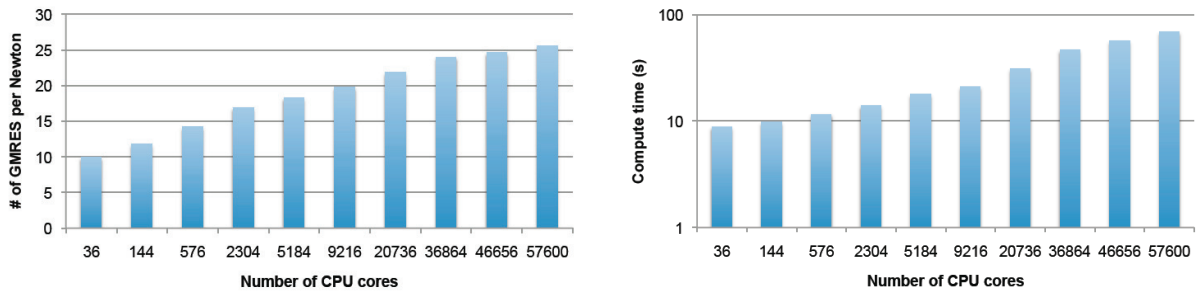

Figure 11: Weak scaling results of the three-level method on the Tianhe-1A, isolated mountain problem. The averaged number of linear iterations per Newton step is shown in the left panel and the total compute time is in the right.

In the weak scaling test, we start with a small $256 \times 256 \times 6$ mesh with 36 CPU cores and end up with a large $10240 \times 10240 \times 6$ mesh (1.89 billion unknowns) using up to 57600 CPU cores. The weak scaling results using the fully implicit method are shown in Fig 11, from which we can see that the averaged number of linear iterations becomes about 2.57 times larger and the total compute time increases by about 6.3 times as the number of CPU cores goes from 36 to 57600. We tend to believe that better weak scaling results might be obtainable on Tianhe-1A if more levels are included in the preconditioner.

5. Concluding remarks

Some multilevel domain decomposition based fully implicit approaches for the shallow water equations on the cubed-sphere were introduced and studied in this paper. The technique features a second-order finite volume discretization which is capable of treating nonsmooth topographies. An inexact Newton method with adaptive stopping conditions is employed to solve the nonlinear systems of equations arising from the fully implicit BDF-2 time integration. At each nonlinear iteration we use the flexible GMRES method to solve the linear Jacobian system. The Jacobian matrices are generated analytically with a semismooth technique that freezes certain entries of the matrix associated with some of nonsmooth components of the problem. Such a modification of Jacobian enables Newton to converge rapidly even when some of the nonlinear functions in the system are not differentiable. The scalability of the approach depends almost entirely on the design of the linear preconditioner. After experimenting with many different overlapping Schwarz type preconditioners, we found one class of hybrid two- and three-level restricted Schwarz methods based on a low order discretization of the SWEs that works particularly well for the problems under investigation. We found that the use of a fully coupled ordering of the unknown/equations and a point-block version of ILU are extremely beneficial. Finally, we

tested the proposed algorithms and software on some cases with nonsmooth topographies and achieved nearly perfect strong scalability and reasonably good weak scalability on a IBM BlueGene/L machine with about 8,000 processors, and a new supercomputer, Tianhe-1A, with more than 80,000 processors.

We believe that this family of approaches, when additional levels are added multiplicatively to the preconditioner, is suitable for larger problems and on machines with much larger number of processors. We plan to extend the approach to other climate models, such as the nonhydrostatic model based on the compressible Euler or Navier-Stokes equations [30, 37, 41].

References

[1] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc Users Manual, Argonne National Laboratory, 2010.

[2] X.-C. Cai, W.D. Gropp, D.E. Keyes, R.G. Melvin, D.P. Young, Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation, SIAM J. Sci. Comput. 19 (1998) 246–265.

[3] X.-C. Cai, W.D. Gropp, D.E. Keyes, M.D. Tidriri, Newton-Krylov-Schwarz methods in CFD, in: R. Rannacher (Ed.), Notes in Numerical Fluid Mechanics: Proceedings of the Intl. Workshop on the Navier-Stokes Equations, Vieweg Verlag, Braunschweig, 1994, pp. 123–135.

[4] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 21 (1999) 792–797.

[5] C. Chen, F. Xiao, Shallow water model on cubed-sphere by multi-moment finite volume method, J. Comput. Phys. 227 (2008) 5019–5044.

[6] F.H. Clarke, Optimization and Nonsmooth Analysis, Wiley, New York, NY, 1983.

[7] T.F. Coleman, J.J. Moré, Estimation of sparse Jacobian matrices and graph coloring problems, SIAM J. Numer. Anal. 20 (1983) 187–209.

[8] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM, Philadelphia, 1996.

[9] J.M. Dennis, M. Levy, R.D. Nair, H.M. Tufo, 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), IEEE Computer Society, Washington, DC, USA, 2005, p. 257.1.

[10] V. Dolean, S. Lanteri, F. Nataf, Convergence analysis of additive Schwarz for the Euler equations, Appl. Numer. Math. 49 (2004) 153–186.

[11] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1996) 1064–8275.

[12] K.J. Evans, D.W. Rouson, A.G. Salinger, M.A. Taylor, W. Weijer, J.B. White, Iii, A scalable and adaptable solution framework within components of the community climate system model, in: Computational Science - ICCS 2009: The 9th International Conference on Computational Science, part II, vol. 5545 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 332–341.

[13] K.J. Evans, M.A. Taylor, J.B. Drake, Accuracy analysis of a spectral element atmospheric model using a fully implicit solution framework, Mon. Wea. Rev. 138 (2010) 3333–3341.

[14] F.X. Giraldo, J.S. Hesthaven, T. Warburton, Nodal high-order discontinuous Galerkin methods for spherical shallow water equations, J. Comput. Phys. 181 (2002) 499–525.

[15] A. Griewank, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, SIAM, Philadelphia, 2000.

[16] W.D. Gropp, D.K. Kaushik, D.E. Keyes, B.F. Smith, High-performacne parallel implicit CFD, Parallel Computing 27 (2001) 337–362.

[17] S.P. Han, J.S. Pang, N. Rangaraj, Globally convergent Newton methods for nonsmooth equations, Math. Oper. Res. 17 (1992) 586–607.

[18] R. Jakob-Chien, J.J. Hack, D.L. Williamson, Spectral transform solutions to the shallow water test set, J. Comput. Phys. 119 (1995) 164–187.

[19] A. Kageyama, T. Sato, Yin-Yang grid: An overset grid in spherical geometry, Geochem. Geophys. Geosyst. 5 (2004).

[20] D. Knoll, D.E. Keyes, Jacobian-free Newton-Krylov methods: A survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.

[21] J. Mandel, Hybrid domain decomposition with unstructured subdomains, in: A. Quarteroni, Y.A. Kuznetsov, J. Périaux, O.B. Widlund (Eds.), Domain Decomposition Methods in Science and Engineering: The 6th Intl. Conf. on Domain Decomposition, vol. 157 of Contemporary Mathematics, AMS, 1994, pp. 103–112.

[22] V.A. Mousseau, D.A. Knoll, J.M. Reisner, An implicit nonlinearly consistent method for the two-dimensional shallow-water equations with Coriolis force, Mon. Wea. Rev. 130 (2002) 2611–2625.

[23] R.D. Nair, S.J. Thomas, R.D. Loft, A discontinuous Galerkin global shallow water model, Mon. Wea. Rev. 133 (2005) 876–888.

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

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

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

[27] L. Qi, Convergence analysis of some algorithms for solving nonsmooth equations, Math. Oper. Res. 18 (1993) 227–244.

[28] L. Qi, J. Sun, A nonsmooth version of Newton's method, Math. Program. 58 (1993) 353–367.

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

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

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

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

[33] J. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (1993) 461–469.

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

[35] R. Sadourny, A. Arakawa, Y. Mintz, Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere, Mon. Wea. Rev. 96 (1968) 351–356.

[36] B. Smith, P. Bjørstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, New York, 1996.

[37] A. St-Cyr, D. Neckels, A fully implicit Jacobian-free high-order discontinuous Galerkin mesoscale flow solver, in: Computational Science - ICCS 2009: The 9th International Conference on Computational Science, part II, vol. 5545 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 243–252.

[38] A. St-Cyr, S.J. Thomas, Nonlinear operator integration factor splitting for the shallow water equations, Appl. Numer. Math. 52 (2005) 429–448.

[39] M. Taylor, J. Tribbia, M. Iskandarani, The spectral element method for the shallow water equations on the sphere, J. Comput. Phys. 130 (1997) 92–108.

[40] S.J. Thomas, R.D. Loft, Semi-implicit spectral element atmospheric model, J. Sci. Comput. 17 (2002) 229–350.

[41] H. Tomita, M. Satoh, A new dynamical framework of nonhydrostatic global model using the icosahedral grid, Fluid Dynamics Research 34 (2004) 357.

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

[43] D.L. Williamson, Integration of the barotropic vorticity equation on a spherical geodesic grid, Tellus 20 (1968) 642–653.

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

[45] Y. Wu, X.-C. Cai, D.E. Keyes, Additive Schwarz methods for hyperbolic equations, in: J. Mandel, C. Farhat, X.-C. Cai (Eds.), Domain Decomposition Methods X: The 10th Intl. Conf. on Domain Decomposition Methods, vol. 218 of Contemporary Mathematics, AMS, 1998, pp. 513–521.

[46] C. Yang, X.-C. Cai, A parallel well-balanced finite volume method for shallow water equations with topography on the cubed-sphere, J. Comput. Appl. Math. (2010). to appear.

[47] C. Yang, X.-C. Cai, Newton-Krylov-Schwarz method for a spherical shallow water model, in: Y. Huang, R. Kornhuber, O. Widlund, J. Xu (Eds.), Domain Decomposition Methods in Science and Engineering XIX: The 19th Intl. Conf. on Domain Decomposition Methods, vol. 78 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 149–155.

[48] C. Yang, J. Cao, X.-C. Cai, A fully implicit domain decomposition algorithm for shallow water equations on the cubed-sphere, SIAM J. Sci. Comput. 32 (2010) 418–438.