Journal of Computational Physics 230 (2011) 6778–6799

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

High-performance high-resolution semi-Lagrangian tracer transport on a sphere

J.B. White III [a,*], J.J. Dongarra [b]

[a] Climate and Global Dynamics Division, National Center for Atmospheric Research, Boulder, CO 80305, United States
[b] Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, United States

ARTICLE INFO

Article history:
Received 27 January 2011
Received in revised form 10 May 2011
Accepted 11 May 2011
Available online 27 May 2011

Keywords:
Tracer transport
Spherical geometry
Cubed sphere
Semi-Lagrangian
High resolution
High-performance computing

ABSTRACT

Current climate models have a limited ability to increase spatial resolution because numerical stability requires the time step to decrease. We describe a semi-Lagrangian method for tracer transport that is stable for arbitrary Courant numbers, and we test a parallel implementation discretized on the cubed sphere. The method includes a fixer that conserves mass and constrains tracers to a physical range of values. The method shows third-order convergence and maintains nonlinear tracer correlations to second order. It shows optimal accuracy at Courant numbers of 10–20, more than an order of magnitude higher than explicit methods. We present parallel performance in terms of strong scaling, weak scaling, and spatial scaling (where the time step stays constant while the resolution increases). For a 0.2° test with 100 tracers, the implementation scales efficiently to 10,000 MPI tasks.

1. Introduction

The complexity and resolution of global climate models has increased greatly over the past half a century, and climate scientists have experienced adequate throughput from these models because of commensurate increases in the capability of high-performance computers. Climate models currently rely on explicit and semi-implicit time integrators, where the time step of the dynamics must shrink to maintain numerical stability as resolution increases. Decreasing time steps have not been a dominant issue in climate simulation because resolutions have been relatively coarse and high-performance computers have increased exponentially in performance [1].

The decrease in time step with increasing resolution is emerging as a challenging barrier to scientific progress, however, for two main reasons. The first reason concerns the evolution of climate models; increasing resolution is now forcing the time step needed for stability of the dynamics to be significantly smaller than the time step needed to resolve the dominant physical processes [2]. The second reason is technological; until recently, the exponential increase in the number of transistors on an integrated circuit, Moore's Law, has translated into an exponential increase in the performance of a single thread of execution on a computer. Though the transistor density of computer chips continues to grow exponentially, the performance of a single thread has stalled. Instead computer chips are growing in parallelism, with more processor cores, longer vector instructions, and multi-threading [3]. Greater parallelism allows increases in spatial resolution while maintaining throughput, but it does not mitigate reductions in time step.

* Corresponding author. Tel.: +1 303 497 1344.
E-mail address: trey@ucar.edu (J.B. White).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.05.008

Increases in resolution typically benefit a physical model in two ways, by increasing the accuracy of numerical approximations and by resolving finer-scale features. Climate models may currently have greater need for the latter, to resolve topography and physical features such as clouds, while the time scales of the dynamics of interest remain well above the time step required for numerical stability. This need for spatial detail, combined with the growing availability of parallelism in high-performance computers while single-thread performance stagnates, motivates the development of methods that allow increases in spatial resolution without requiring equivalent decreases in time step. Increasing resolution without decreasing the time step at all may provide little benefit in accuracy, but some current methods can force the time step down for numerical stability well beyond the level needed for the desired accuracy. We explore a method that allows arbitrary time steps and is most accurate at relatively high Courant numbers [4].

Previous generations of climate models used methods that could allow long time steps, such as semi-Lagrangian (or arbitrary Lagrangian–Eulerian) spectral methods [5]. Ironically, models have since moved to explicit methods partly in an attempt to improve scalability on parallel computers [6,7]. Parallel implementations of semi-Lagrangian spectral methods do exist [8], but the computational cost of the spectral transform that the methods rely on grows super-linearly with the number of grid points, a growing issue at very high resolution.

As an initial step towards improving full climate models, we investigate two-dimensional tracer transport on the sphere. Tracer transport is an important part of atmospheric models, and it grows in importance as the complexity of physical and chemical processes increases in the model. Greater physical fidelity in atmospheric chemistry, the carbon cycle, the sulfur cycle, *etc.* leads to increases in the number of tracers transported by the dynamics [9].

Global climate models are moving from longitude–latitude horizontal grids to other spherical grids with more-homogenous grid spacing, no singularities, and better suitability for parallel computers. In particular, a growing number of major climate models now support the cubed–sphere grid [5,10,11], and a variety of tracer-transport methods target the cubed sphere, including explicit methods using finite-volume [6], spectral-element [7], and discontinuous-Galerkin spatial discretizations [12], along with semi-Lagrangian methods using finite-volume discretizations [13,14].

Our target is a method with a linear increase in total computational cost as the number of grid points increases, where the method can efficiently spread that cost across parallel tasks to allow resolution increases at nearly constant throughput. Important elements of this method include a semi-Lagrangian formulation, which is numerically stable for large Courant numbers, and the cubed–sphere grid, which allows efficient decomposition into parallel tasks and localized computation of Lagrangian-parcel trajectories.

Desirable features of tracer-transport methods for climate modeling include highly accurate mass conservation and shape preservation. Climate models often run for a century or more of simulated time, which drives the conservation requirement for long-lived tracers. Within this context, "shape preservation" refers to the avoidance of overshoots and undershoots near discontinuous features. We distinguish two negative consequences of overshoots and undershoots, where the first is the inaccuracy associated with the numerically generated waves that do not represent physical flows. A secondary consequence is that undershoots can generate small negative values for non-negative physical fields, and these negative values can cause failures ("crashes") in physical parameterizations [15]. We investigate "range preservation", which targets this secondary problem without fully addressing the first. Conservation and shape or range preservation can be challenging to implement, and different strategies for maintaining them drive much of the variety and ongoing innovation in tracer-transport methods.

Our semi-Lagrangian method conserves mass to the same order of accuracy as the computed state. Other methods, particularly finite-volume methods, conserve mass to near machine accuracy. Adding shape preservation to such methods, however, can decrease accuracy [13] or can add computational cost that increases with the Courant number because of the resulting increase in the number of overlap areas requiring integration [14]. Instead of full shape preservation, we attack only range preservation. Lin and Rood explore this strategy, for example, with their positive-definite locally conservative tracer-transport method in [16].

Methods that do not locally conserve mass to machine accuracy often employ "fixers" that scale values based on global integrals such that the integrals remain constant to within machine accuracy. Our method uses a fixer that maintains machine accuracy in global mass conservation, constrains the solution to the physical range of values, and minimally affects the accuracy of the solution. The fixer adds a parallel-communication volume that increases linearly with the number of tracers and is independent of the time step. It has a liability, however, that is common to all global fixers in a parallel context: it requires global reductions. Parallel computers can have very efficient implementations of reductions, but the cost of reductions grows logarithmically with the number of processors, or at best remains nearly constant, so reductions eventually limit parallel performance [17].

Section 2 describes our tracer-transport method, and Section 3 describes our mass fixer. The Workshop on Transport Schemes on the Sphere, held at the National Center for Atmospheric Research March 30–31, 2011, prompted the creation of a new suite of test cases that probes the accuracy, conservation, shape preservation, and mixing characteristics of tracer-transport methods [18]. Section 4 provides the implementation details of these new tests for our method, along with our results and comparisons with the existing results in [18–20]. Section 5 describes our parallel implementation, and Section 6 provides measurements and analyses of parallel performance, including so-called "strong" and "weak" scaling, along with "spatial" scaling, a variety of weak scaling that keeps the time step constant. Finally, Section 7 gives conclusions and future prospects.

2. Transport method

We write the tracer transport equation in advective form as follows:

$$\frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi = \frac{D\phi}{Dt} = 0, \tag{1}$$

where ϕ is the tracer concentration per unit mass. For the surface of a sphere $\mathcal{S}$, $\mathbf{v}$ is the two-dimensional horizontal wind vector, and ∇ is the two-dimensional gradient operator defined on $\mathcal{S}$. D/Dt is the Lagrangian total derivative,

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla. \tag{2}$$

We discretize the surface of the sphere using a cubed sphere [10]. Table 1 gives the angular face coordinates (α, β) in terms of Cartesian coordinates (x,y,z) for each of the six cube faces forming the sphere. The discrete face coordinates are equi-angular with range $[-\pi/4, \pi/4]$. The discretized tracer-concentration value ϕ_{ijk} has position

$$\alpha_i = -\frac{\pi}{4} + i\delta, \tag{3}$$

$$\beta_j = -\frac{\pi}{4} + j\delta, \tag{4}$$

on face k, where the angular spacing is

$$\delta = \frac{\pi}{2n} \tag{5}$$

for $i, j = 0, \ldots, n$. Fig. 1 shows the grid for $n = 30$. Redundant points exist along face edges, so for example $\phi_{001} = \phi_{n03} = \phi_{004}$.

To integrate Eq. (1) forward a single time step, the method first computes the past location of each point on the grid, its "departure point". In other words, for each grid point, the method computes the location at the previous time step of the infinitesimal parcel that arrives at that grid point at the current time. Consider $\phi = \phi(\alpha(t), \beta(t), t)$ along a Lagrangian path on a particular face of the cubed sphere. Expand the Lagrangian departure point in a Taylor series around small time step Δt.

Table 1
Angular face coordinates of the cubed sphere in terms of Cartesian coordinates. The face numbers are arbitrary; the "Direction" column gives the Cartesian unit vector perpendicular to the center of the face, pointing towards that face from the center of the cubed sphere.

Face	Direction	α	β
1	$-\mathbf{k}$	$\arctan(-x/z)$	$\arctan(y/z)$
2	$\mathbf{i}$	$\arctan(y/x)$	$\arctan(z/x)$
3	$\mathbf{j}$	$\arctan(-x/y)$	$\arctan(z/y)$
4	$-\mathbf{i}$	$\arctan(y/x)$	$\arctan(-z/x)$
5	$-\mathbf{j}$	$\arctan(-x/y)$	$\arctan(-z/y)$
6	$\mathbf{k}$	$\arctan(x/z)$	$\arctan(y/z)$

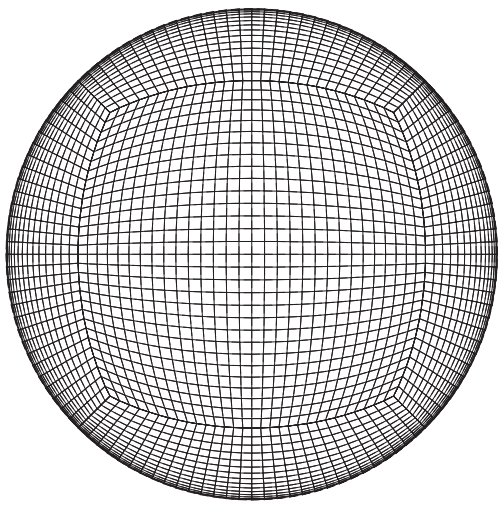


Fig. 1. Cubed-sphere grid for $n = 30$ in Eq. (5).

$$\phi(\alpha(t),\beta(t),t) = \phi(\alpha(t-\Delta t),\beta(t-\Delta t),t-\Delta t), \tag{6}$$

where

$$\alpha(t-\Delta t) = \alpha(t) - \left.\frac{d\alpha}{dt}\right|_t \Delta t + \frac{1}{2}\left.\frac{d^2\alpha}{dt^2}\right|_t \Delta t^2 - \frac{1}{6}\left.\frac{d^3\alpha}{dt^3}\right|_t \Delta t^3 + O(\Delta t^4), \tag{7}$$

and similarly for $\beta(t-\Delta t)$.

The first time derivatives of the face coordinates are just the wind velocities.

$$\left.\frac{d\alpha}{dt}\right|_t = u(\alpha(t),\beta(t),t) \tag{8}$$

$$\left.\frac{d\beta}{dt}\right|_t = v(\alpha(t),\beta(t),t) \tag{9}$$

In terms of the current location, velocities, and derivatives of the velocities, the location at time $t-\Delta t$ (the departure point) is the following.

$$\begin{aligned}\alpha(t-\Delta t) = \alpha(t) - u\Delta t + \frac{1}{2}(uu_\alpha + u_t + u_\beta v)\Delta t^2 - \frac{1}{6}(uu_{\alpha t} + u_{\beta t}v + u_\alpha(uu_\alpha + u_t + u_\beta v) + v(uu_{\alpha\beta} + u_{\beta t} + u_{\beta\beta}v)\\ + u(uu_{\alpha\alpha} + u_{\alpha t} + u_{\alpha\beta}v) + u_\beta(uv_\alpha + vv_\beta + v_t) + u_{tt})\Delta t^3 + O(\Delta t^4),\end{aligned} \tag{10}$$

$$\begin{aligned}\beta(t-\Delta t) = \beta(t) - v\Delta t + \frac{1}{2}(uv_\alpha + vv_\beta + v_t)\Delta t^2 - \frac{1}{6}((uu_\alpha + u_t + u_\beta v)v_\alpha + v_\beta(uv_\alpha + vv_\beta + v_t) + uv_t\\ + u(uv_{\alpha\alpha} + vv_{\alpha\beta} + v_{\alpha t}) + vv_{\beta t} + v(vv_{\beta\beta} + uv_{\alpha\beta} + v_{\beta t}) + v_{tt})\Delta t^3 + O(\Delta t^4).\end{aligned} \tag{11}$$

Subscripts on velocities here indicate partial derivatives, and all velocities and velocity derivatives are at $(\alpha(t),\beta(t),t)$, the arrival location on the fixed cubed-sphere grid at the arrival time.

The test cases in Section 4 all provide analytic functions for the velocities, allowing exact computation of all space and time derivatives to arbitrary accuracy [19]. Full climate models, however, must integrate the evolution of velocities and compute their derivatives numerically. To more closely approximate this target situation, our method uses numerical spatial derivatives.

In (α,β) coordinates, each face of the cubed sphere is a regular grid. A nine-point stencil provides spatial derivatives up to second order with $O(\delta^3)$ accuracy. Most of the domain uses centered differencing, but points at edges and corners of each face use one-sided derivatives. For example, the numerical derivatives for arbitrary field w at the corner grid point (0,0) for arbitrary face k are the following, where the k index is elided.

$$(w_\alpha)_{00} \approx \frac{1}{2\delta}(4w_{10} - 3w_{00} - w_{20}), \tag{12}$$

$$(w_\beta)_{00} \approx \frac{1}{2\delta}(4w_{01} - 3w_{00} - w_{02}), \tag{13}$$

$$(w_{\alpha\alpha})_{00} \approx \frac{1}{\delta^2}(w_{00} - 2w_{10} + w_{20}), \tag{14}$$

$$(w_{\beta\beta})_{00} \approx \frac{1}{\delta^2}(w_{00} - 2w_{01} + w_{02}), \tag{15}$$

$$(w_{\alpha\beta})_{00} \approx \frac{1}{4\delta^2}(9w_{00} - 12w_{10} + 3w_{20} - 12w_{01} + 16w_{11} - 4w_{21} + 3w_{02} - 4w_{12} + w_{22}). \tag{16}$$

The method thus computes the necessary numerical spatial derivatives of u, u_t, v, and v_t for Eqs. (10) and (11) using analytic values of the time derivatives. A "real" application typically would not have analytic values for the wind velocities but would instead use partial-differential equations to solve for the time evolution of the velocities. Our method could form the velocity time derivatives using values at multiple time steps or could convert the velocity time derivatives to space derivatives by repeatedly applying the governing partial-differential equations.

Note that the departure points may have face-coordinate values outside the range $[-\pi/4,\pi/4]$ and so may lie on a different face of the cubed sphere. Therefore the method converts each point to Cartesian coordinates using the functions in Table 2.

Because of the redundant points on the face edges of the cubed-sphere grid, the method computes two departure points for each grid point along a cube edge, and three for each corner. It averages these values in Cartesian coordinates to determine a single departure point for each grid point. This averaging mitigates the numerical instability that might otherwise be introduced by the one-sided numerical derivatives because the average values are approximately centered.

From the Cartesian coordinates, the method determines the face for each departure point, k', and the coordinates on that face, (α',β'), using the functions in Table 1. Now it has all the information needed to interpolate the values of ϕ at the departure points. These interpolated values are the new values of ϕ on the fixed cubed-sphere grid.

Table 2
Functions for converting from face coordinates to Cartesian coordinates, where $s = 1/\sqrt{1+\tan^2\alpha+\tan^2\beta}$.

Face	x	y	z
1	$s\tan\alpha$	$-s\tan\beta$	$-s$
2	s	$s\tan\alpha$	$s\tan\beta$
3	$-s\tan\alpha$	s	$s\tan\beta$
4	$-s$	$-s\tan\alpha$	$s\tan\beta$
5	$s\tan\alpha$	$-s$	$s\tan\beta$
6	$s\tan\alpha$	$s\tan\beta$	s

Our implementation has two options for interpolation, third-order interpolation using a nine-point stencil or fourth-order using a sixteen-point stencil. For third-order interpolation, the method first finds the closest non-edge grid point $(\alpha_{i'}, \beta_{j'})$ to each departure point (α', β'):

$$i' = \max\left(1, \min\left(n-1, \operatorname{nint}\left(\frac{\alpha' + \pi/4}{\delta}\right)\right)\right), \tag{17}$$

$$j' = \max\left(1, \min\left(n-1, \operatorname{nint}\left(\frac{\beta' + \pi/4}{\delta}\right)\right)\right), \tag{18}$$

where "nint" is the Fortran function that rounds its floating-point argument to the nearest integer.

Written in terms of differences $\delta_\alpha = (\alpha' - \alpha_{i'})/\delta$ and $\delta_\beta = (\beta' - \beta_{j'})/\delta$, the interpolated value ϕ' is

$$\begin{aligned}\phi' = \Big((\delta_\alpha-1)\delta_\alpha(\delta_\beta-1)\delta_\beta\phi_{i'-1j'-1k'} - 2(\delta_\alpha-1)\delta_\alpha\left(\delta_\beta^2-1\right)\phi_{i'-1j'k'} + (\delta_\alpha-1)\delta_\alpha\delta_\beta(\delta_\beta+1)\phi_{i'-1j'+1k'}\\ -2(\delta_\alpha^2-1)(\delta_\beta-1)\delta_\beta\phi_{i'j'-1k'} + 4\left(\delta_\alpha^2-1\right)\left(\delta_\beta^2-1\right)\phi_{i'j'k'} - 2\left(\delta_\alpha^2-1\right)\delta_\beta(\delta_\beta+1)\phi_{i'j'+1k'}\\ +\delta_\alpha(\delta_\alpha+1)(\delta_\beta-1)\delta_\beta\phi_{i'+1j'-1k'} - 2\delta_\alpha(\delta_\alpha+1)\left(\delta_\beta^2-1\right)\phi_{i'+1j'k'} + \delta_\alpha(\delta_\alpha+1)\delta_\beta(\delta_\beta+1)\phi_{i'+1j'+1k'}\Big)/4.\end{aligned} \tag{19}$$

Fourth-order interpolation uses modified versions of Eqs. (17) and (18) to account for the larger stencil. Instead of finding the nearest point, they find the indices of the lowest corner of the appropriate stencil. As with the third-order case, points near cube edges use stencils shifted away from the edges, such that the stencils go up to the edges but not beyond them.

$$i' = \max\left(0, \min\left(n-3, \operatorname{nint}\left(\frac{\alpha' + \pi/4}{\delta}\right) - 1\right)\right), \tag{20}$$

$$j' = \max\left(0, \min\left(n-3, \operatorname{nint}\left(\frac{\beta' + \pi/4}{\delta}\right) - 1\right)\right). \tag{21}$$

Again in terms of δ_α and δ_β,

$$\begin{aligned}\phi' = ((\delta_\alpha-3)(\delta_\alpha-2)(\delta_\alpha-1)(\delta_\beta-3)(\delta_\beta-2)(\delta_\beta-1)\phi_{i'j'k'} - 3(\delta_\alpha-3)(\delta_\alpha-2)(\delta_\alpha-1)(\delta_\beta-3)(\delta_\beta-2)\delta_\beta\phi_{i'j'+1k'}\\ +3(\delta_\alpha-3)(\delta_\alpha-2)(\delta_\alpha-1)(\delta_\beta-3)(\delta_\beta-1)\delta_\beta\phi_{i'j'+2k'} - (\delta_\alpha-3)(\delta_\alpha-2)(\delta_\alpha-1)(\delta_\beta-2)(\delta_\beta-1)\delta_\beta\phi_{i'j'+3k'}\\ -3(\delta_\alpha-3)(\delta_\alpha-2)\delta_\alpha(\delta_\beta-3)(\delta_\beta-2)(\delta_\beta-1)\phi_{i'+1j'k'} + 9(\delta_\alpha-3)(\delta_\alpha-2)\delta_\alpha(\delta_\beta-3)(\delta_\beta-2)\delta_\beta\phi_{i'+1j'+1k'}\\ -9(\delta_\alpha-3)(\delta_\alpha-2)\delta_\alpha(\delta_\beta-3)(\delta_\beta-1)\delta_\beta\phi_{i'+1j'+2k'} + 3(\delta_\alpha-3)(\delta_\alpha-2)\delta_\alpha(\delta_\beta-2)(\delta_\beta-1)\delta_\beta\phi_{i'+1j'+3k'}\\ +3(\delta_\alpha-3)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-3)(\delta_\beta-2)(\delta_\beta-1)\phi_{i'+2j'k'} - 9(\delta_\alpha-3)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-3)(\delta_\beta-2)\delta_\beta\phi_{i'+2j'+1k'}\\ +9(\delta_\alpha-3)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-3)(\delta_\beta-1)\delta_\beta\phi_{i'+2j'+2k'} - 3(\delta_\alpha-3)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-2)(\delta_\beta-1)\delta_\beta\phi_{i'+2j'+3k'}\\ -(\delta_\alpha-2)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-3)(\delta_\beta-2)(\delta_\beta-1)\phi_{i'+3j'k'} + 3(\delta_\alpha-2)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-3)(\delta_\beta-2)\delta_\beta\phi_{i'+3j'+1k'}\\ -3(\delta_\alpha-2)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-3)(\delta_\beta-1)\delta_\beta\phi_{i'+3j'+2k'} + (\delta_\alpha-2)(\delta_\alpha-1)\delta_\alpha(\delta_\beta-2)(\delta_\beta-1)\delta_\beta\phi_{i'+3j'+3k'})/36.\end{aligned} \tag{22}$$

Depending on the Courant number, the computation of the departure points is $O(\Delta t^4)$ or $O(\delta^3\Delta t)$, because of the $O(\delta^3)$ numerical derivatives. The interpolation is either $O(\delta^3)$ (Eq. (19)) or $O(\delta^4)$ (Eq. (22)). Integration for a fixed length of time requires $O(\Delta t^{-1})$ time steps. If the Courant number is $O(1)$, convergence for a fixed length of time should be $O(\Delta t^2)$ with third-order interpolation and $O(\Delta t^3)$ with fourth-order interpolation. If the Courant number is large, convergence should be $O(\Delta t^3)$ regardless.

3. Mass fixer

With non-divergent winds, Eq. (1) conserves the global integral of ϕ exactly, but our numerical method should conserve the global integral of ϕ only to the order of accuracy of ϕ itself. (As described in [19], the physically relevant conserved quantity is actually $\rho\phi$, where ρ is the fluid density, but ρ is a constant for this test case.) For either divergent or non-divergent

winds, Eq. (1) maintains the range of ϕ [19], while our numerical method should only maintain the range to within the order of accuracy of ϕ.

Typical global mass fixers, such as those used by the spectral dynamical cores in the Community Atmosphere Model, scale a field by a constant factor that is the ratio of the desired global integral and the current global integral before correction [15]. The following combined mass fixer and limiter uses integrals of the field and the square of the field to both conserve mass to within machine precision and constrain the range of ϕ to only physical values, while maintaining the same order of accuracy. A particularly useful constraint that this fixer can apply is at the lower bound to enforce non-negativity.

Let $[h_{\min}, h_{\max}]$ be the range of possible physical values of ϕ for all simulated time. Given the new values ϕ' computed with Eq. (19) or (22), the limiter step first truncates the range of ϕ' to the physical range.

$$\phi'' = \max(h_{\min}, \min(h_{\max}, \phi')) \tag{23}$$

The mass fixer then scales the values of ϕ'' so that the global integral equals that of ϕ_0 while the range stays within $[h_{\min}, h_{\max}]$.

$$\phi''' = \frac{\phi''(I_2 - h_{\min}\mu - I_1\phi'' + \mu\phi'') + h_{\max}(h_{\min}(\mu + A\phi'' - I_1) - \mu\phi'')}{Ah_{\max}h_{\min} - (h_{\max} + h_{\min})I_1 + I_2}, \tag{24}$$

where

$$A = \iint_{\mathcal{S}} dA, \tag{25}$$

$$\mu = \iint_{\mathcal{S}} \phi dA = \iint_{\mathcal{S}} \phi_0 dA, \tag{26}$$

$$I_1 = \iint_{\mathcal{S}} \phi'' dA, \tag{27}$$

$$I_2 = \iint_{\mathcal{S}} \phi''^2 dA, \tag{28}$$

and ϕ''' is the final result for the tracer concentration at the new time step. Here $\mathcal{S}$ is the surface of the sphere, and A is its area. The fixer computes A and μ only once, but it computes I_1 and I_2 at each time step. We derive Eq. (24) by starting with a quadratic function of ϕ'' and requiring that $\phi''' = h_{\min}$ when $\phi'' = h_{\min}$, $\phi''' = h_{\max}$ when $\phi'' = h_{\max}$, and the integral of ϕ''' equals μ.

Our implementation uses sixth-order numerical quadrature to perform the integrals; it computes the weights once and simply sums the weights times the integrand values on the fixed grid for each integral. The sixth-order weights come from seven-point one-dimensional stencils in each dimension of the face coordinates, with modified weights at the ends to account for numbers of points in a given dimension not divisible by six. The two-dimensional weights are products of two one-dimensional weights and a metric term to convert from face-coordinate area to physical area: $s^3(1 + \tan^2\alpha)(1 + \tan^2\beta)$, where s is as in Table 2. The mass fixer and limiter should change values of ϕ' only to within the order of accuracy, so ϕ''' should have the same order of accuracy as ϕ'.

Eq. (1) does not conserve the global integral of ϕ when the winds are divergent because ρ does not remain constant, but the equation does maintain a constant range of ϕ [19]. Thus the test cases in Section 4 with divergent winds use only the limiter step, Eq. (23). A full climate model would compute the winds and fluid density before transporting tracers, so it could conserve mass using Eq. (24) with integrals modified to include ρ, while still constraining the range of ϕ with Eq. (23). Thus our mass fixer is not applicable to the test cases with divergent winds, but it would be applicable to a full climate model with divergent winds.

4. Numerical tests

A new test suite for two-dimensional tracer transport on a sphere appears in [18], based on the deformational-flow test cases that appear in [19,20]. The tests include convergence with smooth and quasi-smooth initial conditions, preservation of the shape of discontinuous initial conditions, and preservation of nonlinear relationships between two tracers. We augment these tests with convergence tests at high resolution with large Courant numbers—tests of accuracy designed to compliment our performance tests in Section 6. The subsections below describe the implementations and results of all these tests for our semi-Lagrangian method, with comparisons to the results in [18,19]. Our results are among the first for the full suite of tests in [18].

4.1. Convergence

Two types of initial conditions test convergence, Gaussian hills, which are infinitely differentiable, and cosine bells, which have discontinuous second derivatives. In Cartesian coordinates that are implicitly constrained to the surface of the unit sphere, the initial two Gaussian hills, with indices 1 and 2, are

Table 3
Cartesian and spherical coordinates for the initial conditions of the test cases.

Coordinate	Non-divergent winds	Divergent winds
x_1	$-\sqrt{3}/2$	$-1/\sqrt{2}$
y_1	$1/2$	$1/\sqrt{2}$
z_1	0	0
λ_1	$5\pi/6$	$3\pi/4$
θ_1	0	0
x_2	$-\sqrt{3}/2$	$-1/\sqrt{2}$
y_2	$-1/2$	$-1/\sqrt{2}$
z_2	0	0
λ_2	$-5\pi/6$	$-3\pi/4$
θ_2	0	0

$$\phi_0 \equiv \phi(x,y,z,t=0) = h_1 + h_2, \tag{29}$$

where

$$h_i(x,y,z) = h_0 e^{-b_0((x-x_i)^2+(y-y_i)^2+(z-z_i)^2)}, \tag{30}$$

$h_0 = 0.95$, $b_0 = 5$, and the values of the coordinates (x_i,y_i,z_i) are in Table 3. The limiter constrains the range of ϕ using $h_{\min} = 0$ and $h_{\max} = 1$; ϕ_0 never reaches these exact values, so the bounds are not tight in this case.

The initial cosine bells take the following form, in terms of Cartesian coordinates constrained to the surface of the unit sphere.

$$r_i(x,y,z) = \arccos(xx_i + yy_i + zz_i), \tag{31}$$

$$h_i(x,y,z) = \begin{cases} h_{\max}(1+\cos(\pi r_i/r))/2 & \text{if } r_i < r, \\ 0 & \text{otherwise}, \end{cases} \tag{32}$$

$$\phi_0 = b + c(h_1 + h_2). \tag{33}$$

Here $b = 0.1$, $c = 0.9$, $r = 1/2$, and the limiter constrains the values of ϕ using $h_{\min} = 0.1$ and $h_{\max} = 1$, which are tight constraints in this case.

The test suite [18] includes non-divergent and divergent wind fields. In each case, the winds both deform the tracer field and translate it one rotation around the sphere, such that the tracer field takes its initial form at time $T = 5$. In spherical coordinates, the non-divergent wind field in non-dimensional units is

$$u_{(\lambda)} = \kappa \sin^2\lambda' \sin 2\theta \cos\omega t + 2\omega\cos\theta, \tag{34}$$
$$v_{(\theta)} = \kappa \sin 2\lambda' \cos\theta \cos\omega t, \tag{35}$$

where $\kappa = 2$, $\omega = \pi/T$, and $\lambda' = \lambda - 2\omega t$. This corresponds to Case 4 in [19].

The divergent wind field in non-dimensional units is

$$u_{(\lambda)} = -\kappa \sin^2\lambda'/2 \sin 2\theta \cos^2\theta \cos\omega t + 2\omega\cos\theta, \tag{36}$$
$$v_{(\theta)} = \frac{\kappa}{2} \sin\lambda' \cos^3\theta \cos\omega t. \tag{37}$$

This corresponds to Case 3 in [19] with the addition of solid body rotation.

Our method uses the wind fields in face-coordinate velocities, (u, v) from Eqs. (8) and (9), where the fields vary in form for each face of the cubed sphere. To derive the wind fields in face coordinates, we first convert Eqs. (34)–(37) to Cartesian coordinates. For example, the non-divergent wind fields in Cartesian coordinates constrained to the surface of the sphere are the following.

$$u_{(x)} = \kappa z(\cos\omega t + \cos 3\omega t)(x\sin 2\omega t - y\cos 2\omega t) - \omega y, \tag{38}$$
$$v_{(y)} = 4\kappa z\cos^2\omega t \sin\omega t(x\sin 2\omega 2 - y\cos 2\omega t) + \omega x, \tag{39}$$
$$w_{(z)} = \kappa\cos\omega t(2xy\cos 4\omega t + (y^2 - x^2)\sin 4\omega t). \tag{40}$$

We convert these to face coordinates using the functions in Table 2 and corresponding velocity vectors. For example, the velocity vector for Face 1 (the $-\mathbf{k}$ face) is the following in terms of face coordinates and Cartesian unit vectors, where s is as in Table 2.

$$\mathbf{v} = \frac{1}{s}\left(\frac{u}{1+\tan^2\alpha}\mathbf{i} - \frac{v}{1+\tan^2\beta}\mathbf{j} + \left(\frac{u\tan\alpha}{1+\tan^2\alpha} + \frac{v\tan\beta}{1+\tan^2\beta}\right)\mathbf{k}\right) \tag{41}$$

For the non-divergent winds, the velocity components in face coordinates for Face 1 are then

$$u = \frac{\kappa s \cos \omega t}{1+\tan^2 \alpha}(\tan\alpha(\tan^2\beta - \tan^2\alpha - 1)\sin 4\omega t - (1+2\tan^2\alpha)\tan\beta\cos 4\omega t - \tan\beta) + \frac{2\omega\tan\beta}{1+\tan^2\alpha}, \tag{42}$$

$$v = \frac{\kappa s \cos \omega t}{1+\tan^2 \beta}(\tan\beta(\tan^2\beta - \tan^2\alpha + 1)\sin 4\omega t - (1+2\tan^2\beta)\tan\alpha\cos 4\omega t + \tan\alpha) - \frac{2\omega\tan\alpha}{1+\tan^2\beta}. \tag{43}$$

We derive the velocity components for the other faces similarly.

4.1.1. Non-divergent winds

Fig. 2(a) and (b) show the initial conditions for the convergence tests with non-divergent winds. (Fig. 2(c) and (d) relate to Sections 4.2 and 4.3.) Because the state at time $t = T$ should match the initial conditions, the normalized l_1, l_2, and l_∞ norms comparing the initial (ϕ_0) and final (ϕ_T) states show the error in the method, where

$$l_1 = \frac{\iint_S |\phi_0 - \phi_T| dA}{\iint_S |\phi_0| dA}, \tag{44}$$

$$l_2 = \sqrt{\frac{\iint_S (\phi_0 - \phi_T)^2 dA}{\iint_S \phi_0^2 dA}}, \tag{45}$$

$$l_\infty = \frac{\max_S |\phi_0 - \phi_T|}{\max_S |\phi_0|}. \tag{46}$$

Fig. 3 shows convergence of these norms using the resolutions and time steps in Table 4. The maximum Courant number for each of the runs is 10.4, and the number of time steps for each resolution gives the minimum or nearly minimum errors for that resolution.

The convergence of the Gaussian-hills initial conditions follows the third-order line for fourth-order interpolation and the second-order line for third-order interpolation, as expected. The lines with and without the mass fixer are indistinguishable.

The convergence of the cosine-bells initial conditions follows the second-order line regardless of interpolation order, which is expected because of the discontinuous second derivative at the edge of each cosine bell. The fourth-order interpolation has slightly higher error at low resolution and slightly lower at high resolution, particularly for the l_∞ error. The mass fixer does not change the order of accuracy, but it does increase the l_∞ error very slightly.

Fig. 3 is comparable to Fig. 4 in [18], which shows results for the CSLAM method [13]. The best CSLAM results use twice as many time steps and compute the backward trajectory for each grid point using analytic values of all the velocity derivatives at multiple trajectory points. The convergence of our method with fourth-order interpolation is similar to that of unfiltered CSLAM, but our method can have higher absolute error, up to three times higher for the Gaussian hills. The shape-preserving filter used by CSLAM can increase the error, however, such that our method with a fixer has equivalent l_2 error and smaller l_∞ error for the Gaussian hills.

Results for the cosine-bells test at 1.5° resolution appear in [19] for unfiltered CSLAM and for an unfiltered explicit discontinuous-Galerkin (DG) method. Table 5 compares our results with those, including the range errors defined as follows.

$$\phi_{\max} = \frac{\max_S \phi_T - \max_S \phi_0}{\max_S \phi_0 - \min_S \phi_0}, \tag{47}$$

$$\phi_{\min} = \frac{\min_S \phi_T - \min_S \phi_0}{\max_S \phi_0 - \min_S \phi_0} \tag{48}$$

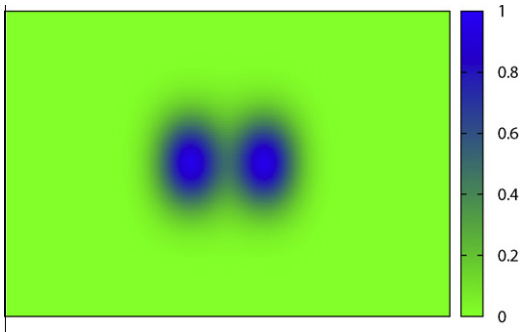
Our errors are lower than those for DG and just slightly higher than for CSLAM, particularly with third-order interpolation, while our time step is twice as long as that for CSLAM and 40 times as long as for DG. Our mass fixer increases most errors very slightly, but it eliminates the $\phi_{\min}$ error to within machine precision.

A least-squares linear regression [18] provides numerical convergence rates $\mathcal{K}_1, \mathcal{K}_2$, and $\mathcal{K}_\infty$ for the respective error norms. Table 6 confirms the visual results in Fig. 3: fourth-order interpolation approaches third-order convergence for the Gaussian hills, while third-order interpolation maintain second-order convergence, and the cosine bells show second-order convergence with little dependence on interpolation order.

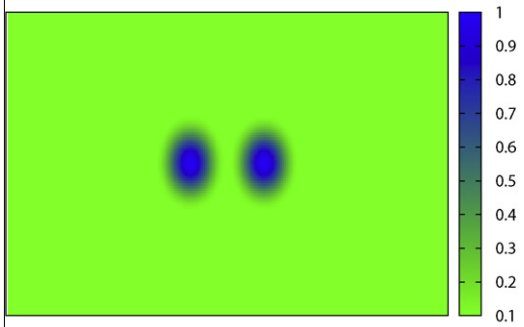
While the mass fixer conserves the global integral of ϕ to machine precision (about 10^{-16}), Fig. 4 shows the error in this quantity without the fixer. Our "un-fixed" method conserves mass to third order with either third- or fourth-order interpolation, for both Gaussian hills and cosine bells.

4.1.2. Divergent winds

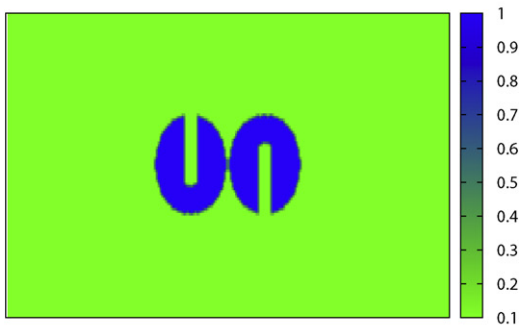
Convergence results for the divergent winds are very similar to those for the non-divergent winds. Convergence rates appear in Table 7. Because the divergent winds do not maintain a constant density, and the test case does not provide an analytic density field, the results include the limiter from Eq. (23) but not the fixer from Eq. (24). Though Lauritzen and Skamarock [18] recommend the divergent-wind tests, it does not provide results for comparison.



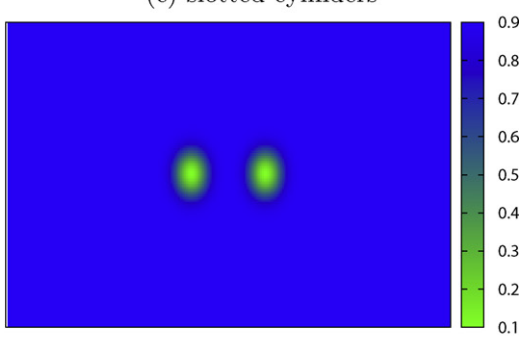
(a) Gaussian hills



(b) cosine bells



(c) slotted cylinders



(d) correlated cosine bells

Fig. 2. Contour plots of initial conditions for the non-divergent winds at 1.5° resolution, interpolated from the cubed-sphere grid onto a longitude–latitude grid.

4.1.3. Effective resolution

"Effective resolution" is defined in [18] as the resolution for which l_2 is approximately 0.033 for the cosine-bells initial conditions with no limiters or fixers. Table 8 shows effective resolutions for our method, defined conservatively as the maximum resolutions where $l_2 \leqslant 0.033$. For this test case, the lower-order interpolation performs better (coarser effective

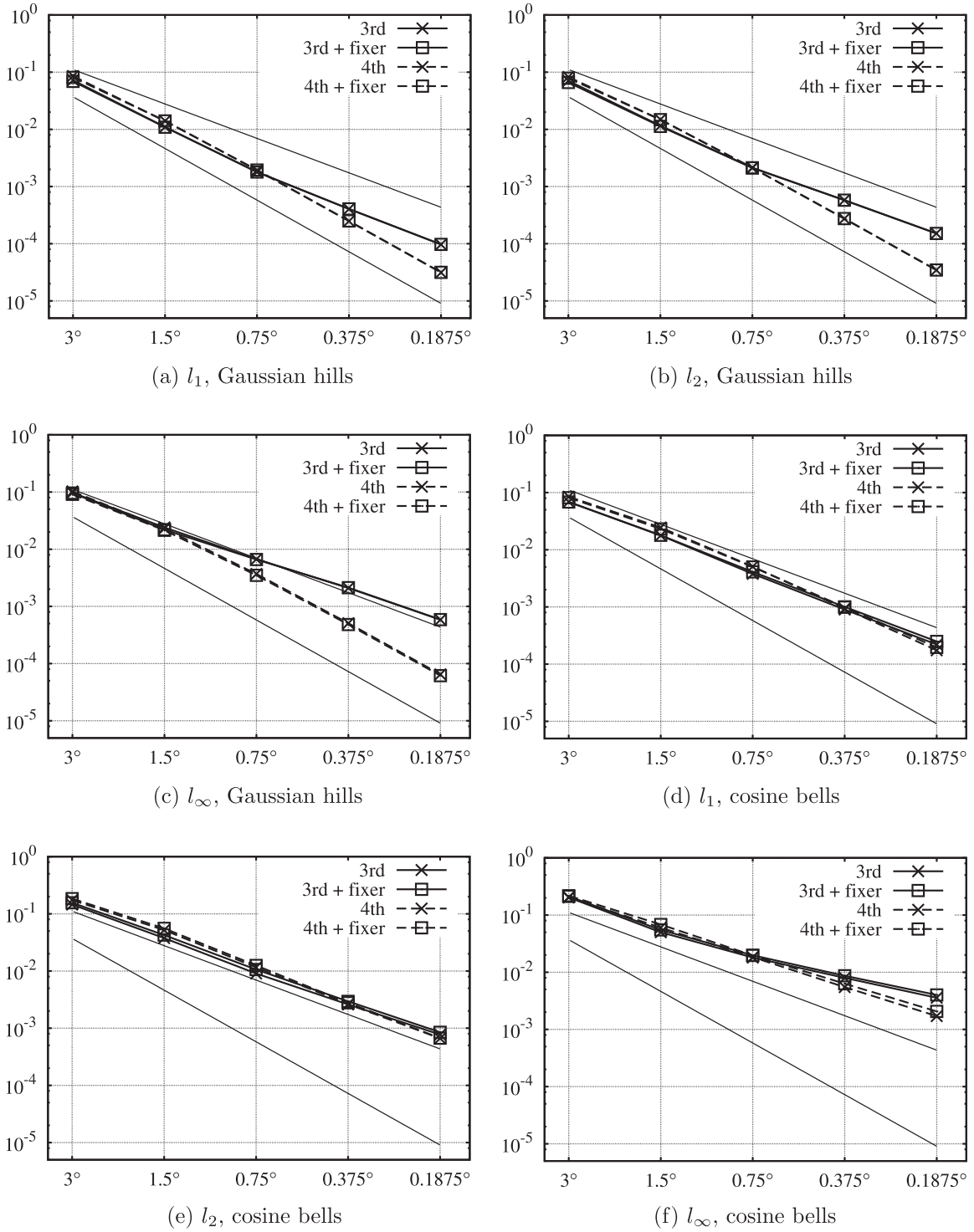


Fig. 3. Normalized error norms at time $T = 5$ for a range of spatial resolutions with non-divergent winds. Time steps decrease with increasing resolution, as shown in Table 4. Solid lines show results with third-order interpolation, and dashed lines with fourth-order. Points marked with "x" are for the original method, and points marked with a box include the mass fixer. The upper thin line shows a slope of second-order convergence, and the lower thin line shows that of third-order convergence.

resolution), and the divergent winds are less challenging than the non-divergent. Lauritzen and Skamarock [18] report that the effective resolution of CSLAM for the non-divergent winds using 120 time steps is 1.5°. This is slightly better than for our method, but ours uses just over half as many time steps.

Table 4
Values used for the convergence plots in Figs. 3–5, where n is as in Eq. (5). Each experiment runs out to time $T = 5$.

Resolution	Points (n)	Number of time steps	Δt
3°	30	30	0.166667
1.5°	60	60	0.083333
0.75°	120	120	0.041667
0.375°	240	240	0.020833
0.1857°	480	480	0.010417

Table 5
A comparison of error norms at $T = 5$ for the cosine-bells test at 1.5° resolution. The $\phi_{\max}$ and $\phi_{\min}$ values show the deviation of the solution from the physical range. "3rd" represents our method with third-order interpolation, and "4th" represents ours with fourth-order.

Method	Time steps	l_1	l_2	l_∞	$\phi_{\max}$	$\phi_{\min}$
DG	2400	0.0330	0.0562	0.1047	−0.0678	−0.0846
CSLAM	120	0.0158	0.0328	0.0473	−0.0068	−0.0214
3rd	60	0.0177	0.0377	0.0500	−0.0166	−0.0301
3rd+fixer	60	0.0179	0.0425	0.0550	−0.0180	-10^{-16}
4th	60	0.0245	0.0512	0.0601	−0.0364	−0.0224
4th+fixer	60	0.0231	0.0556	0.0684	−0.0392	-10^{-16}

Table 6
Convergence rates for our tests with non-divergent winds. "Order" is the order of accuracy of the interpolation. "Fixer" shows whether the mass fixer is active.

Initial conditions	Order	Fixer	$\mathcal{K}_1$	$\mathcal{K}_2$	$\mathcal{K}_\infty$
Gaussian hills	3rd	No	2.4	2.2	1.8
		Yes	2.4	2.2	1.8
	4th	No	2.9	2.8	2.7
		Yes	2.8	2.8	2.7
cosine bells	3rd	No	2.1	1.9	1.4
		Yes	2.0	1.9	1.4
	4th	No	2.3	2.0	1.7
		Yes	2.2	2.0	1.7

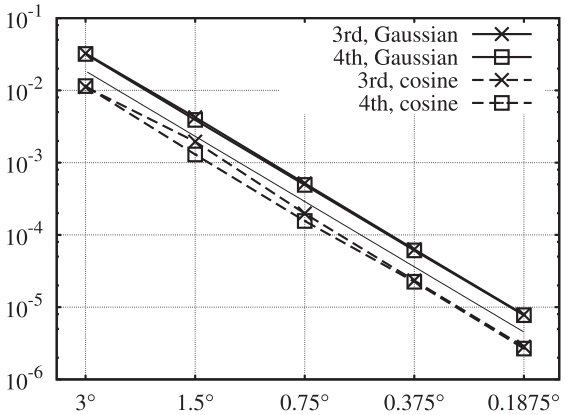


Fig. 4. Absolute value of the error in the global integral of ϕ for a range of spatial resolutions with non-divergent winds. The number of time steps increases with resolution, as shown in Table 4. Solid lines represent the Gaussian-hills initial conditions, and dashed lines represent the cosine-bells initial conditions. Points marked with "x" use third-order interpolation, and points marked with a box use fourth-order. The thin line shows a slope of third-order convergence.

4.2. Shape preservation

"Rough", or discontinuous, initial conditions test the shape-preservation properties of a method. The test here uses two slotted cylinders, as shown in Fig. 2(c). A combination of Cartesian and spherical coordinates most easily defines the slotted cylinders.

Table 7
Convergence rates for our tests with divergent winds. "Order" is the order of accuracy of the interpolation. "Limiter" shows whether the limiter is active.

Initial conditions	Order	Limiter	$\mathcal{K}_1$	$\mathcal{K}_2$	$\mathcal{K}_\infty$
Gaussian hills	3rd	No	2.4	2.3	2.1
		Yes	2.4	2.3	2.1
	4th	No	2.9	2.8	2.8
		Yes	2.9	2.8	2.8
cosine bells	3rd	No	2.2	2.0	1.6
		Yes	2.2	2.0	1.6
	4th	No	2.4	2.1	1.8
		Yes	2.5	2.2	1.7

Table 8
Effective resolution, as indicated by "Resolution", where "Divergent" indicates whether the winds are divergent or non-divergent, "Order" indicates the order of accuracy of the interpolation, and n is as in Eq. (5).

Divergent	Order	Resolution	Points (n)	Time Steps	l_2
No	3	1.4°	64	64	0.0327
	4	1.2°	75	75	0.0325
Yes	3	1.7°	52	52	0.0329
	4	1.6°	57	57	0.0321

$$r_1 = \arccos(xx_1 + yy_1 + zz_1), \tag{49}$$
$$r_2 = \arccos(xx_2 + yy_2 + zz_2), \tag{50}$$

$$\phi_0 = \begin{cases} c & \text{if } r_1 \leqslant r \quad \text{and } |\lambda - \lambda_1| \geqslant r/6, \\ c & \text{if } r_1 \leqslant r \quad \text{and } |\lambda - \lambda_1| < r/6 \text{ and } \theta - \theta_1 < -5r/12, \\ c & \text{if } r_2 \leqslant r \quad \text{and } |\lambda - \lambda_2| \geqslant r/6, \\ c & \text{if } r_2 \leqslant r \quad \text{and } |\lambda - \lambda_2| < r/6 \text{ and } \theta - \theta_2 > 5r/12, \\ b & \text{otherwise.} \end{cases} \tag{51}$$

Here $b = 0.1$, $c = 1$, $r = 1/2$, and (x_i, y_i, z_i) and (λ_i, θ_i) come from Table 3. The mass fixer uses $h_{\min} = b$ and $h_{\max} = c$ in Eqs. (23) and (24).

Fig. 5 shows contour plots of the results of this test with non-divergent winds at 1.5° resolution for $t = T/2$ and $t = T$. We do not include plots for divergent winds, which show less error structure and smaller errors. Table 9 shows error results at 1.5° and 0.75° resolutions for both divergent and non-divergent winds.

For the non-divergent winds, the mass fixer successfully reduces errors in the maximum value, minimum value, and global integral down to machine precision without significantly affecting the other norms. Eq. (24) is inactive for the divergent winds, but the limiter (Eq. (23)) successfully eliminates the error in the maximum and minimum values without significantly affecting the error norms.

Our fixer, which eliminates overshoots outside the physically prescribed range, dramatically reduces all the oscillations in this test case because the discontinuous values lie at the limits of the range. The fixer does not eliminate oscillations in general, so, if one of the cylinders were, for example, half as high as the other, the shorter cylinder would generate oscillations at its top as in the case without a fixer.

Results for CSLAM at 1.5° using 120 time steps (twice our number) for the non-divergent test case appear in [18]. The error norms are almost identical to our results. For the unfiltered case, CSLAM gets $\phi_{\min} = -0.19$ and $\phi_{\max} = 0.15$, similar but slightly larger than our results. With its shape-preserving filter, CSLAM gets $\phi_{\min} = 0.0$ and $\phi_{\max} = -4.34 \times 10^{-3}$, while our limiter gets 0.0 for both.

4.3. Nonlinear relationship

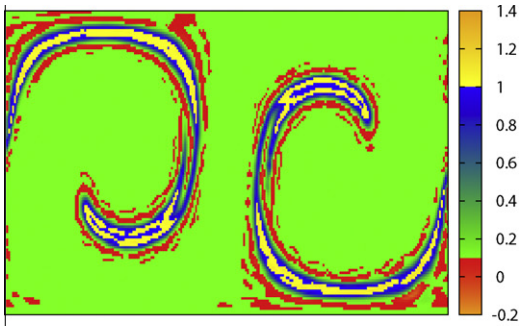
Lauritzen and Thuburn [20] define a new class of tests that explores the mixing characteristics of transport methods using nonlinear relationships between tracers. Correlated cosine bells, ϕ_0 and ϕ_0^*, provide the initial conditions.

$$\phi_0^* = \psi(\phi_0), \tag{52}$$

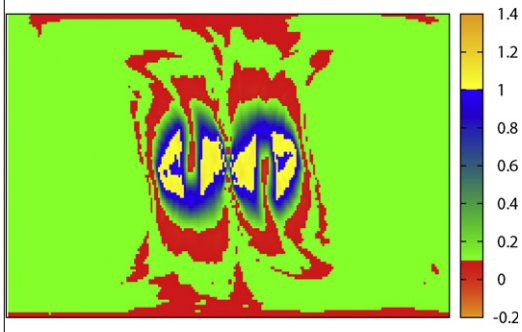
where ϕ_0 is from Eq. (32), and

$$\psi(\chi) = a\chi^2 + b, \tag{53}$$

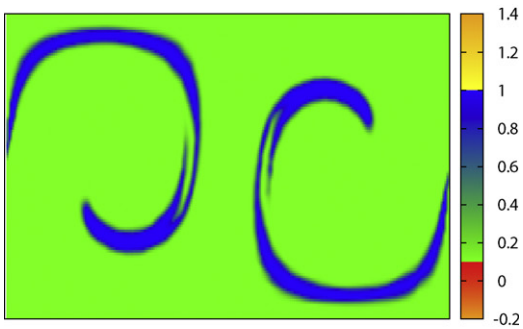
where $a = -0.8$ and $b = 0.9$.



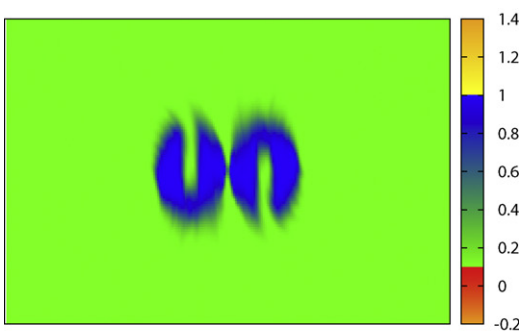
(a) $t = T/2$, no fixer



(b) $t = T$, no fixer



(c) $t = T/2$, fixer



(d) $t = T$, fixer

Fig. 5. Contour plots for slotted cylinders with non-divergent winds and 1.5° resolution (see Table 4), interpolated from the cubed-sphere grid onto a longitude–latitude grid. The green-to-blue gradient shows the range of values of the initial conditions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 9
Results for the slotted-cylinder test cases. "Div" indicates whether the winds are divergent. "Res" indicates the resolution from Table 4. "Lim" shows whether the mass fixer is active for the non-divergent winds and whether the limiter is active for the divergent winds. $\phi_{\min}$ and $\phi_{\max}$ are errors in the global minimum and maximum values, and "Mass" gives the error in the global mass integral.

Div	Res	Lim	l_1	l_2	l_∞	$\phi_{\min}$	$\phi_{\max}$	Mass
No	1.5°	No	0.136	0.243	0.781	0.125	0.137	-0.011
		Yes	0.143	0.262	0.812	10^{-16}	10^{-15}	10^{-16}
	0.75°	No	0.084	0.186	0.803	0.125	0.149	0.005
		Yes	0.089	0.204	0.753	10^{-15}	10^{-16}	10^{-15}
Yes	1.5°	No	0.140	0.250	0.801	0.162	0.163	-0.008
		Yes	0.153	0.274	0.761	0.0	0.0	-0.004
	0.75°	No	0.090	0.196	0.802	0.186	0.186	0.001
		Yes	0.096	0.214	0.765	0.0	0.0	0.005

A plot of ϕ versus ϕ^* should follow a constant curve that bends downward, as in Fig. 6. [20] defines three types of deviation from this curve, identifiable by their mixing characteristics.

1. "Real mixing" causes points to move to the concave side of the curve.

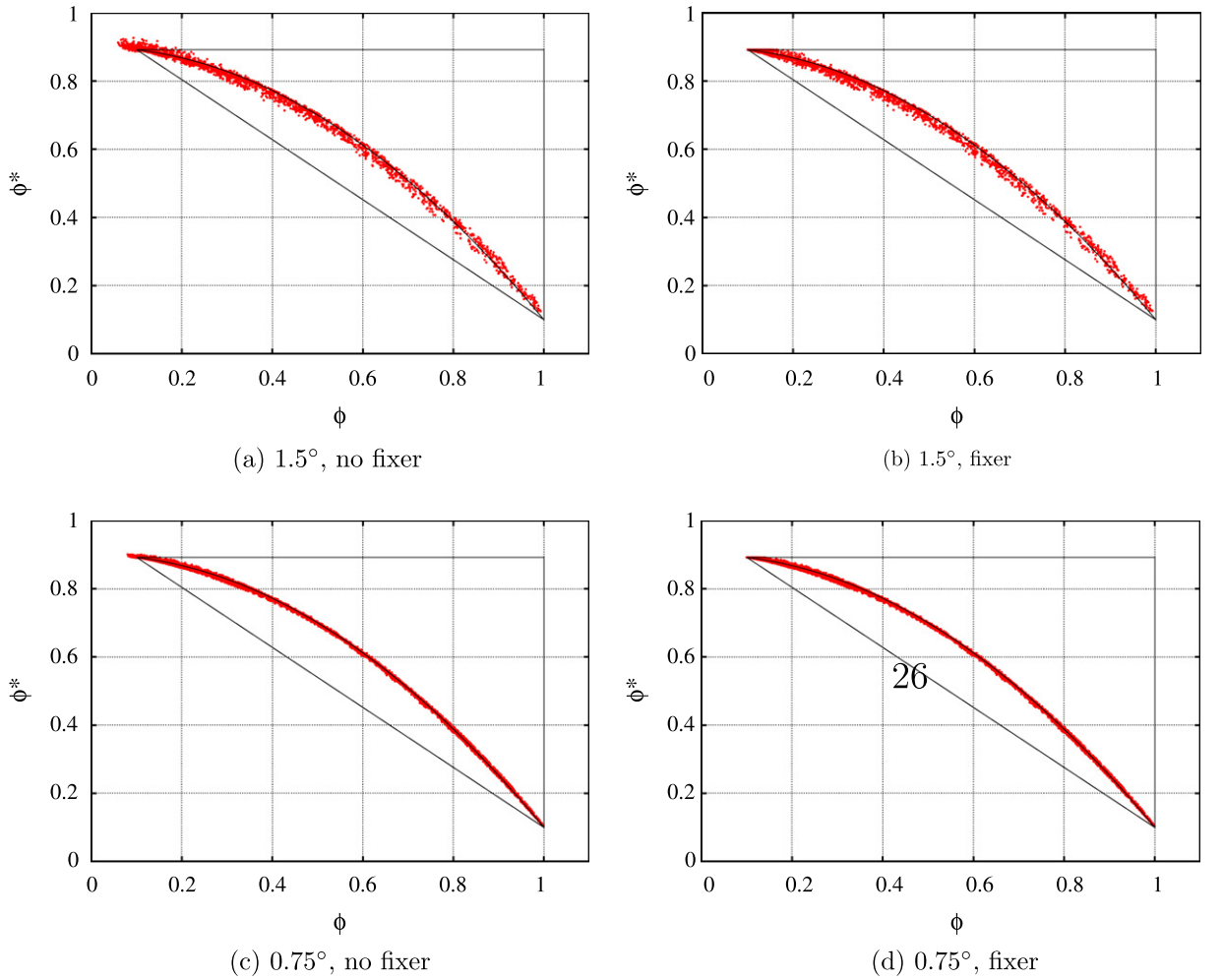


Fig. 6. Scatter plots at $t = T/2$ for nonlinearly correlated (Eq. (52)) cosine-bells initial conditions. Points below the curve within the triangle represent real mixing, points above the curve within the triangle represent range-preserving unmixing, and points outside the triangle represent overshooting. All results use third-order interpolation.

Table 10
Diagnostics for real mixing (l_r), range-preserving unmixing (l_u), and overshooting (l_o) at $t = T/2$ with non-divergent winds. "Resolution" indicates the resolution from Table 4, where the CSLAM results use twice as many time steps. "3rd" indicates our method with third-order interpolation, and "+ fixer" indicates that the mass fixer is active. CSLAM results come from [20], and "+ filter" indicates that the shape-preserving filter is active.

Resolution	Method	l_r	l_u	l_o
1.5°	3rd	7.00×10^{-4}	1.57×10^{-4}	6.36×10^{-4}
	CSLAM	7.55×10^{-4}	1.58×10^{-4}	3.79×10^{-4}
	3rd+fixer	7.78×10^{-4}	1.06×10^{-4}	1.82×10^{-5}
	CSLAM+filter	6.28×10^{-4}	6.73×10^{-5}	0.0
0.75°	3rd	2.01×10^{-4}	8.65×10^{-5}	1.73×10^{-4}
	CSLAM	1.40×10^{-4}	2.99×10^{-5}	3.43×10^{-5}
	3rd+fixer	2.12×10^{-4}	6.94×10^{-5}	2.66×10^{-15}
	CSLAM+filter	1.05×10^{-4}	2.57×10^{-5}	0.0

Table 11
Diagnostics for real mixing (l_r), range-preserving unmixing (l_u), and overshooting (l_o) at $t = T/2$ for our method with divergent winds. "Resolution" indicates the resolution from Table 4. "Limiter" shows whether the limiter is active. All results use third-order interpolation.

Resolution	Limiter	l_r	l_u	l_o
1.5°	No	5.80×10^{-6}	2.08×10^{-6}	1.59×10^{-5}
	Yes	4.18×10^{-6}	2.05×10^{-6}	4.23×10^{-7}
0.75°	No	9.17×10^{-7}	8.10×10^{-7}	2.80×10^{-6}
	Yes	2.94×10^{-7}	8.34×10^{-7}	1.28×10^{-9}

2. "Range-preserving unmixing" causes point to move to the convex side of the curve, while staying within the range of the initial conditions.
3. "Overshooting" causes points to move outside the range of the initial conditions.

Fig. 6 has scatter plots of ϕ versus ϕ^* that illustrate the mixing induced by our method for non-divergent winds. These plots are comparable to those in Fig. 6 of [20], and we can discern no important differences. We do not include plots for divergent winds because of their similarity to our Fig. 6.

The error measure for each type of deviation—l_r for "real", l_u for "unmixing", and l_o for "overshooting"—sums the distance to the correct curve of all points lying in that region, and it divides by the total number of points of all types [20].

Table 10 compares the values of these diagnostics for our method and CSLAM with non-divergent winds, and Table 11 shows values for our method with divergent winds. No results for divergent winds appear for CSLAM in [20]. Without fixers or filters, our method gives results comparable to CSLAM at 1.5°. The CSLAM results improve slightly more with resolution, and the CSLAM filter improves results slightly more than our mass fixer. In general, our results for divergent winds have significantly lower errors than for non-divergent.

4.4. High resolution

A primary motivation for our method is to allow high spatial resolution without requiring small time steps. Two convergence tests not included in [18] or [20] illustrate the accuracy of the method at high spatial resolution and compliment our performance tests in Section 6. Both tests use the cosine-bells initial conditions with non-divergent winds. Results are similar for Gaussian hills and for divergent winds.

Fig. 7 shows the results of the first test, where spatial resolution increases while the time step remains constant. Because of the large time step, the error quickly stops decreasing. The time error dominates, so the order of the spatial interpolation matters little. The mass fixer has little effect on the error, slightly decreasing the l_∞ error. The major benefit of the semi-Lagrangian method is that it remains stable as the Courant number increases, reaching a value of 166 at 0.1875° resolution. This is an extreme test of numerical stability, with Courant numbers well above what a climate model is likely to use. The primary intent is to explore performance characteristics out to extreme regimes, as shown in Section 6.

Fig. 8 shows the results of the second test, which varies the number of time steps while maintaining a constant, high spatial resolution of 0.1875°. For large time steps (small numbers of steps), the time error dominates and decreases at third order. For small time steps (large numbers of steps), the interpolation error dominates and grows because of the increasing number of interpolations. The l_1 and l_2 errors reach a minimum at 240 time steps (Courant number of 20.8),

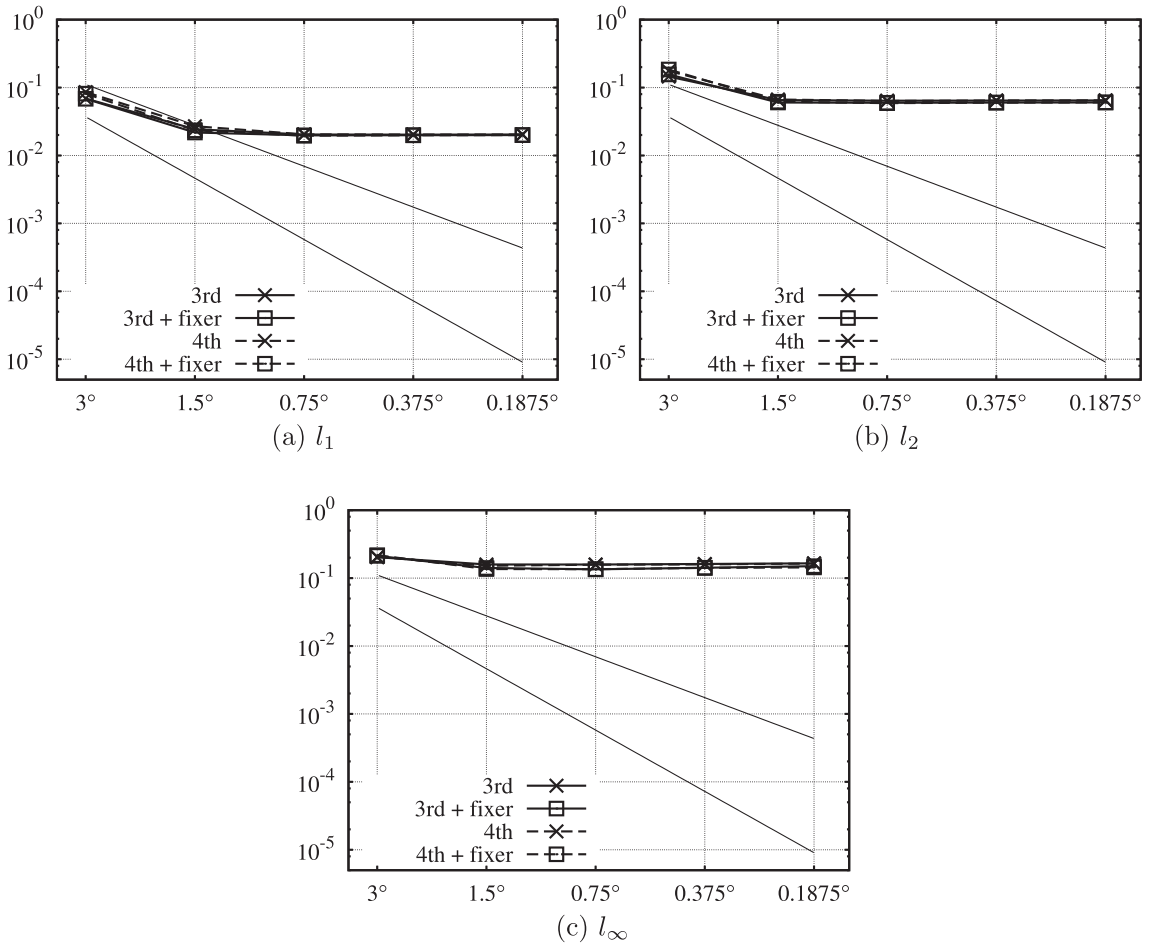


Fig. 7. Error norms for the cosine-bells test case with non-divergent winds over a range of resolutions. The number of time steps is constant at 30. Line and point styles are as in Fig. 3.

while the minimum l_∞ error depends on the order of the interpolation. Third-order interpolation minimizes l_∞ at 120 time steps (Courant number of 40.5), while fourth-order interpolation minimizes l_∞ at 240 time steps, with a slightly lower minimum.

Whereas explicit methods usually require Courant numbers of less than one, our semi-Lagrangian method favors Courant numbers over an order of magnitude higher.

5. Parallel implementation

Our parallel implementation distributes the cubed-sphere grid among tasks that communicate using the Message-Passing Interface (MPI) [21]. To simplify the determination of neighboring tasks and the home task for any grid point, the distribution places the following constraints on the number of tasks, N.

- N must be divisible by six (the number of faces of the cubed sphere).
- $N/6$ must be a perfect square.
- The number of grid points along an edge of the cubed sphere, $n + 1$, must be divisible by $\sqrt{N/6}$.

For example, a grid of $n + 1 = 30$ ($n = 29$) allows task counts of 6, 24, 54, 150, 216, 600, 1350, or 5400.

Full climate models transport more than one tracer, so our implementation allows the transport of an arbitrary number of identical tracers to better approximate the performance characteristics of full models.

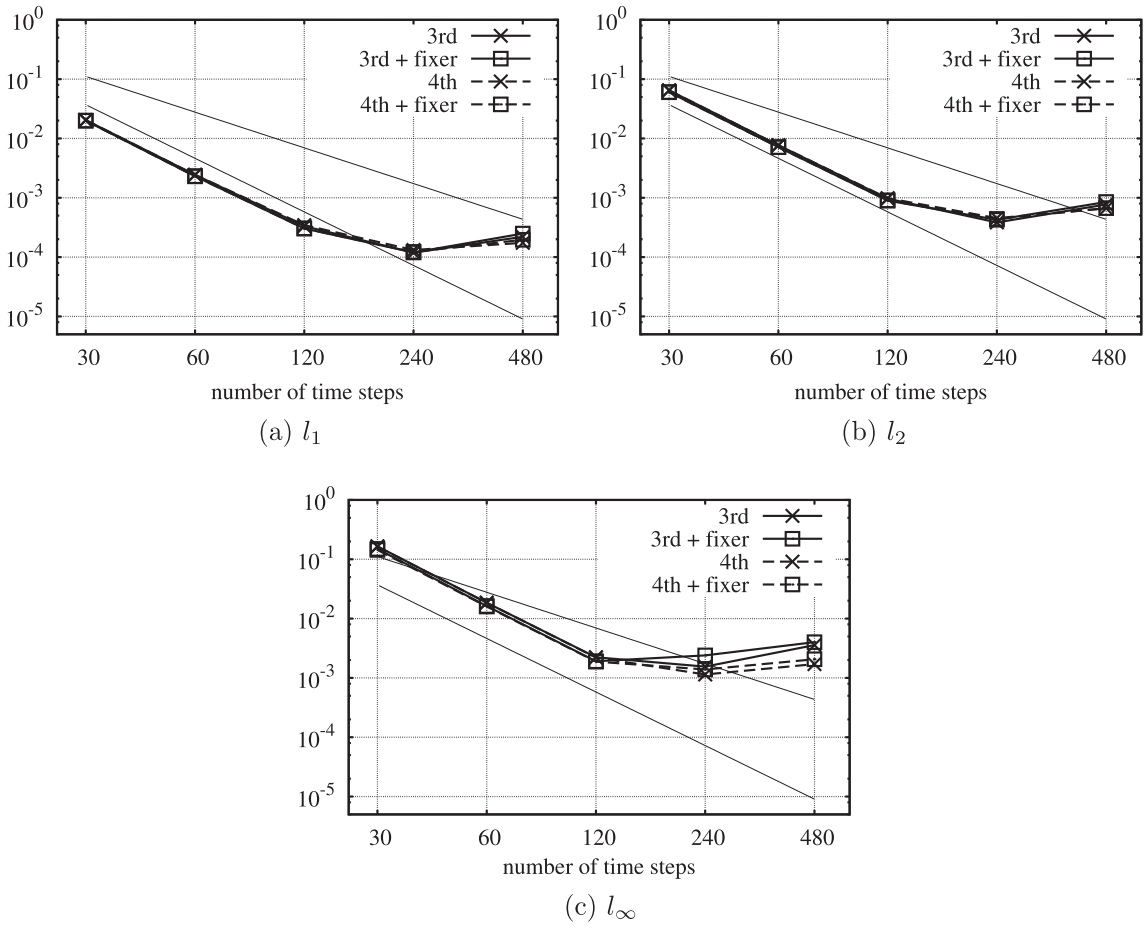


Fig. 8. Error norms for the cosine-bells test case with non-divergent winds over a range of time steps. The spatial resolution is constant at 0.1875°. Line and point styles are as in Fig. 3.

Each time step performs up to four stages of parallel communication, in the following order.

1. Each task exchanges "halos" with its eight neighbors, where the halos contain the values of u, u_t, v, and v_t needed for the nine-point stencils of the spatial derivatives used in Eqs. (10) and (11). No communication occurs between faces because of the one-sided derivatives on face edges, so edge tasks communicate with only five neighbors, and corner tasks communicate with only three. For all tasks, the volume of communication is independent of the number of tracers. If $N = 6$, with one task per face, no communication occurs at this stage.
2. After the method computes all the departure points in Cartesian coordinates, it must average the coordinate values along the edges. Each edge task exchanges edge values with the neighboring task on the adjoining cube face. Edge tasks communicate with a single neighbor, and corner tasks communicate with two neighbors. The volume of communication is independent of the number of tracers. If $N = 6$, with one task per face, each task communicates with four neighbors.
3. The communication needed to interpolate the tracer concentrations is dynamic. The departure points change with each time step, and the tasks that a particular task communicates with can change. Algorithms 1 and 2 provide a detailed description of this phase. In addition to the number of tasks and the size of the grid, the volume of communication depends on the values of the wind fields and the size of the time step. Parts of this phase increase linearly with the number of tracers. The phase also requires a global synchronization that grows logarithmically in cost with the number of tasks.
4. When the mass fixer is active, it must perform a global reduction to compute the integrals in Eq. (24). MPI libraries typically have efficient implementations of such global reductions, but they can increase logarithmically in cost with the number of tasks [22]. Each tracer requires two integrated values, so the volume of communication grows linearly with the number of tracers.

Algorithm 1: Parallel interpolation for each task

```
for each local grid point do
  compute face coordinates and nearest grid point for departure point
end for
empty task list and message lists
for each local grid point do
  if task for departure point is in task list then
    add grid point to message list for that task
  else
    add new task to task list and grid point to its message list
  end if
end for
for each task in task list do
  fill message buffer with departure points to be interpolated
end for
issue non-blocking receives for synchronization with parent and children
for each task in task list do
  issue non-blocking receive of interpolated values
  issue non-blocking send of desired departure points
end for
for each task in task list do
  repeat
    check for incoming requests (Algorithm 2)
  until send of departure points to other task completes
  wait for receipt of interpolated values from that task
end for
copy new values from received messages
repeat
  check for incoming requests (Algorithm 2)
until synchronization messages arrive from children
send synchronization message to parent
repeat
  check for incoming requests (Algorithm 2)
until synchronization message arrives from parent
send synchronization messages to children
```

Algorithm 2: Check for incoming requests

```
loop
  probe for incoming requests
  if no requests waiting to be received then
    return
  end if
  receive list of departure points from requesting task
  for each departure point in received list do
    compute nearest local grid point
    interpolate values for all tracers at departure point
  end for
  send list of interpolated values to requesting task
end loop
```

Timers measure the performance of the time-stepping loop, and counters track the volume of sent messages and the number of neighbors that are sent interpolation requests. Our implementation outputs

- the total runtime for the time-stepping loop,

- the number of tracers times the number of time steps per second (a measure of throughput),
- the average number of bytes sent per task per time step (which does not include communication volume from the global reductions), and
- the average number of neighbors per task that were sent interpolation requests.

6. Performance tests

This section provides performance results of our parallel implementation run on the Hopper-II computer hosted by the National Energy Research Scientific Computing Center (NERSC). Table 12 gives details of the computer and software. All the tests use the cosine-bells initial conditions with non-divergent winds and third-order interpolation.

Tests of parallel performance often come in two varieties, "strong scaling" and "weak scaling". Strong-scaling tests measure the increasing performance of a fixed-size problem as the number of parallel tasks increases. Fig. 9 shows the performance of a strong-scaling test at 0.2° resolution ($n = 479$) with 480 time steps.

Because the problem size remains constant, the amount of computational work per task shrinks until the latency costs of communication eventually dominate. The test scales linearly up to about 1000 tasks for one tracer and up to almost 10,000 tasks for 100 tracers. Beyond 10,000 tasks the performance declines. Increasing the resolution or the amount of computational work per tracer, as in a full climate model, would likely improve scaling to higher task counts. Throughput improves as the number of tracers increases, likely as a result of increasing computational efficiency. The performance impact of the mass fixer is negligible for one tracer, but it grows with tracer count and reduces performance by about 30% for 100 tracers.

In contrast to strong-scaling, weak-scaling tests change the size of the problem as the number of tasks increases such that the number of grid points per task remains constant. Fig. 10 shows the throughput and runtime for a weak-scaling test, where each task has a domain of 30×30 grid points per tracer. The throughput lines show only small deviations from horizontal, indicating good scalability. The relative cost of the fixer grows with the number of tracers, up to 25–30% for 100 tracers.

Fig. 10(b) demonstrates the issue with traditional weak scaling through increasing resolution. Because the number of time steps increases linearly with resolution, perfect weak scalability for a two-dimensional problem leads to an increase in total runtime proportional to the square root of the number of tasks.

The time stability of the semi-Lagrangian method allows a third type of scaling, "spatial scaling", where the spatial resolution increases with the number of tasks, but the time step stays constant. Perfect spatial scaling has constant performance and constant total runtime for a fixed number of tracers.

Table 12
Computer and software details for parallel performance tests on NERSC Hopper II.

Computer	Cray XE6
Compute nodes	6392
Memory per node	32 GB DDR 1333 MHz
Processors per node	2 × AMD 12-core Opteron 6172 at 2.1 GHz
Interconnect	Cray Gemini 3D Torus
Compiler	PGI Fortran 10.9.0
MPI	Cray MPT 5.1.3

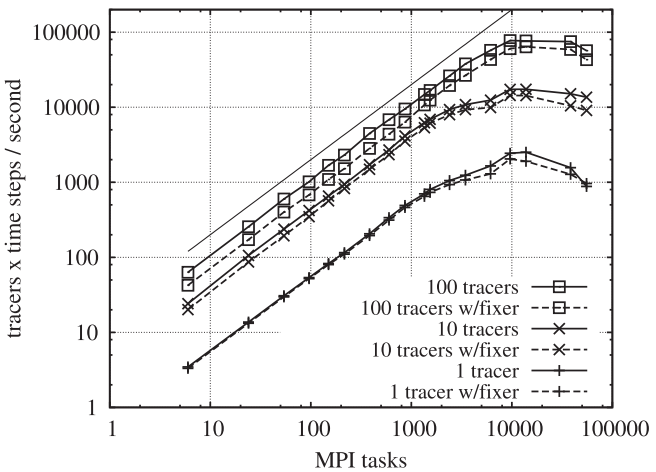


Fig. 9. Parallel throughput of the strong-scaling test. The test uses 0.2° resolution ($n = 479$) with 480 time steps. Dashed lines indicate "w/fixer", where the mass fixer is active. The thin solid line shows linear performance increase.

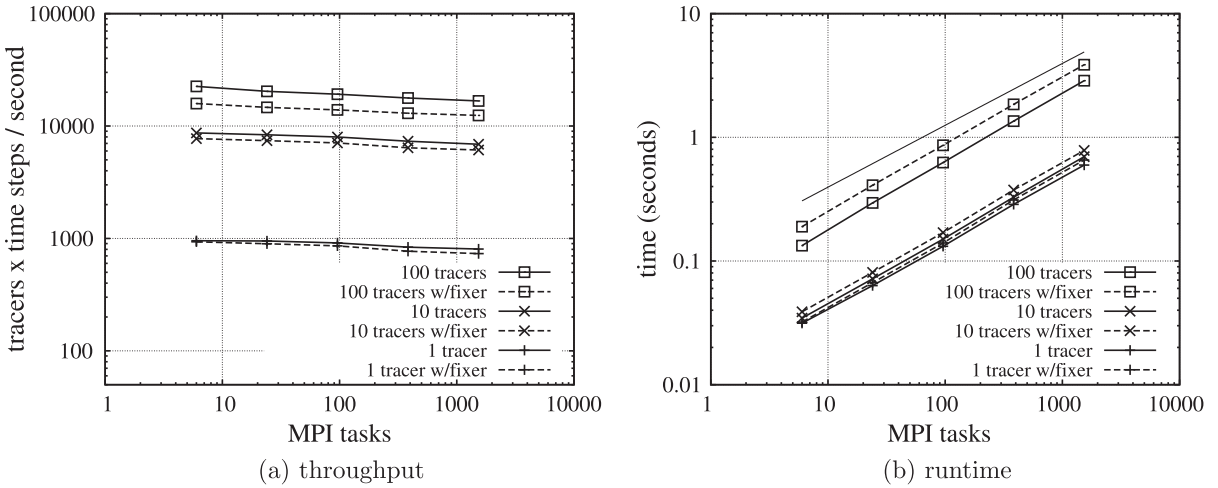


Fig. 10. Weak-scaling test. Each MPI task has a domain of 30×30 grid points per tracer. The number of time steps increases with the square root of the number of tasks (linearly with resolution). Dashed lines indicate "w/fixer", where the mass fixer is active. The thin solid line in Fig. 10(b) shows a runtime increase proportional to the square root of the number of tasks.

Fig. 11 shows the throughput and runtime of a spatial-scaling test, where each task has a domain of 30×30 grid points per tracer, and the number of time steps stays constant at 30. This performance test corresponds to the convergence test shown in Fig. 7. The throughput does not scale as well as for the weak-scaling test, and the runtime goes up by a factor of as much as 2.7 from 6 to 1536 tasks (for 100 tracers with no fixer). Compare this, however, to the factor of 21.6 for the equivalent weak-scaling case.

Fig. 12 shows the cost increase in the interpolation phase, which explains the differences in scalability between the weak- and spatial-scaling tests. Fig. 12(a) shows that the communication volume per task of the spatial-scaling test increases significantly more than that of the weak-scaling test, and Fig. 12(b) shows some of the reason why. For the weak-scaling test, the average number of neighbors each task queries during the interpolation phase shrinks slightly with increasing task count. For the strong-scaling test, however, the number of neighbors increases significantly.

For spatial scaling, the time step stays constant as the resolution increases, so each departure point gets farther and farther away relative to the grid spacing. The likelihood that a departure point is not in the local domain gets higher until, at some resolution, all the departure points are remote to that task. The departure points may also spread farther from each other relative to the grid spacing, since the increasing Courant number amplifies any variability in the wind fields.

For the case of weak scaling, however, the time step stays constant relative to the grid spacing, so the departure points do not get farther away. As the resolution increases, the wind fields have less variability across the local domain, so the spread of departure points decreases. Thus the average number of neighbors decreases.

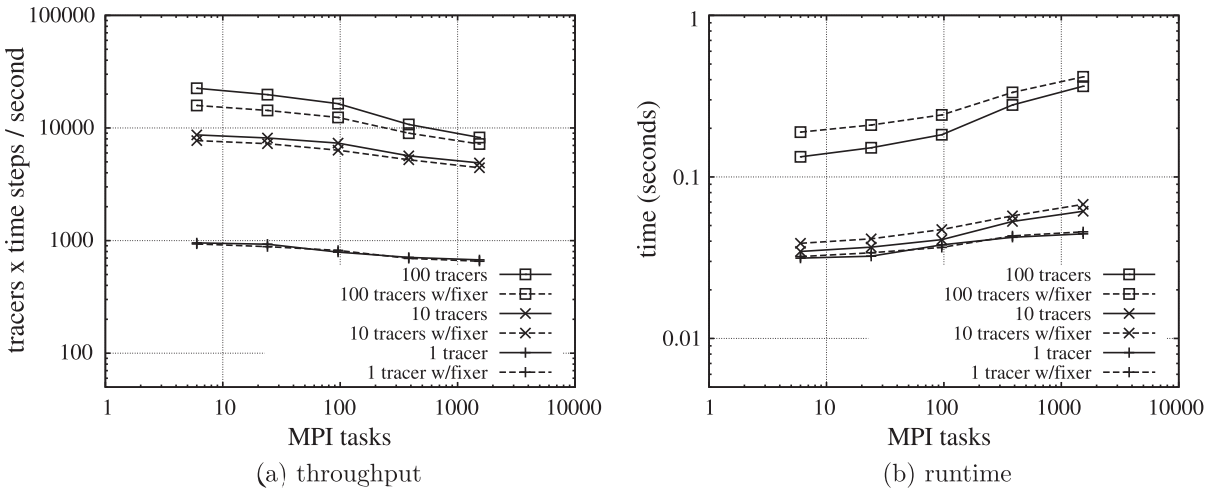


Fig. 11. Spatial-scaling test. Each MPI task has a domain of 30×30 grid points per tracer. The number of time steps stays constant at 30. Dashed lines indicate "w/fixer", where the mass fixer is active.

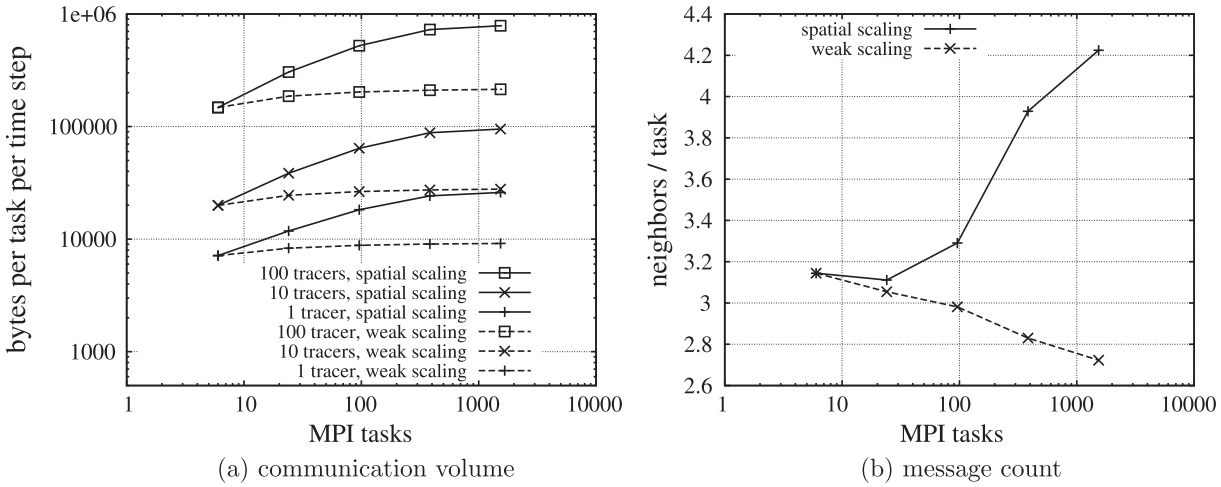


Fig. 12. Communication costs of interpolation for the spatial-scaling and weak-scaling tests with increasing task counts. The communication volume is the average number of bytes sent by each task during each time step, not including global reductions. The message count is the average number of neighbors per time step that a task sends interpolations requests to.

7. Conclusions

We presented a semi-Lagrangian method for two-dimensional tracer transport on a sphere. Our method is stable for arbitrary Courant numbers, grows linearly in computational complexity with the number of grid points, and has computational complexity independent of the Courant number. The method uses a cubed-sphere discretization, backward projection of the Lagrangian departure points that is $O(\Delta t^4)$ per time step, and interpolation that is $O(\delta^3)$ or $O(\delta^4)$, where Δt is the time step and δ is the grid spacing. The resulting method has accuracy approaching $O(\Delta t^3)$ for simulations of a fixed time length with smooth initial conditions or with large Courant numbers. As is, the method conserves mass to $O(\Delta t^3)$.

We also presented a mass fixer that conserves mass to machine precision for non-divergent wind fields, while limiting tracer values to a prescribed range, also to within machine precision. The fixer maintains the same accuracy as the unaltered method. The fixer requires non-divergent winds for the tests used here because the tests do not define a density field, and the method does not solve for density. For tests with divergent winds, we apply only the limiter. In a full climate simulation, the method for integrating wind dynamics would likely provide a density field, so the fixer would be applicable even if the winds were divergent.

We presented results for the new suite of numerical tests defined in [18] and based on [19,20]. Tests with a constant Courant number of 10.4 show convergence approaching the expected third order for Gaussian hills with fourth-order interpolation and convergence of second order for cosine bells. The effective resolution of our method, defined as the resolution that provides l_2 error of about 0.033 for cosine bells, is about 1.3° for the non-divergent case. A test with discontinuous initial conditions, slotted cylinders, shows the effectiveness of the fixer at conserving mass while eliminating values outside the physical range. A test of nonlinearly correlated cosine bells shows that the method has mixing errors of the same order as the standard error norms, and that the fixer significantly reduces the overshooting error, l_o. Additional tests not included in [18] demonstrate the numerical stability of the method at high Courant numbers, above 150. The method favors relatively large Courant numbers for best accuracy, about 10–20, an order of magnitude higher than the stability limits of explicit methods.

Finally we presented a parallel implementation of the method with a dynamic algorithm for parallel interpolation of departure points. The algorithm allows arbitrary winds and time steps by computing the necessary neighbors for communication at each time step. Performance tests on NERSC Hopper II show scaling past 1000 parallel tasks for a 0.2° strong-scaling problem with one tracer, and up to 10,000 tasks for 100 tracers. Weak-scaling tests show near-perfect parallel efficiency and improving computational efficiency with increasing numbers of tracers, but they also show increasing relative cost of the fixer with increasing numbers of tracers. The numerical stability of the method allows tests of spatial scaling, where the time step and physical dimensions stay constant while the resolution and task count increase. These tests measure the increases in communication costs that come from increasingly remote and dispersed departure points.

Our semi-Lagrangian method and parallel implementation show promise for future climate models to mitigate the throughput limitations of high-performance computers. The method achieves its best accuracy at relatively high Courant numbers, thus reducing the required number of time steps. And it allows still-higher Courant numbers, so resolution may increase to resolve physical features of interest without forcing increases in the number of time steps. This capability, common to semi-Lagrangian methods for tracer transport, joins with parallel scalability to enhance throughput for high resolution on high-performance computers.

In future work, we anticipate investigating scalable semi-Lagrangian methods for the more-challenging dynamics problem of the winds and mass density that drive tracer transport. Semi-Lagrangian methods may enable arbitrary time steps for that problem by efficiently preconditioning implicit methods.

Acknowledgements

We sincerely thank the anonymous reviewers for their helpful suggestions and corrections. Portions of this work were funded by the Office of Biological and Environmental Research in the US Department of Energy. This research used resources of NERSC at Lawrence Berkeley National Laboratory and of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, both of which are supported by the Office of Science of the US Department of Energy.

References

[1] W.M. Washington, L. Buja, A. Craig, The computational future for climate and earth system models: on the path to petaflop and beyond, Philosophical Transactions of the Royal Society A—Mathematical, Physical, and Engineering Sciences 367 (1890) (2009) 833–846.
[2] W. Washington, D. Bader, B. Collins, J. Drake, M. Taylor, B. Kirtman, D. Williams, D. Middleton, A. Bamzai, L. Chatterjee, Scientific Grand Challenges: Challenges in Climate-Change Science and the Role of Computing at the Extreme Scale, US Department of Energy, 2008.
[3] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R.S. Williams, K. Yellick, Exascale computing study: technology challenges in achieving exascale systems, Technical report, DARPA Information Processing Techniques Office, September 2008.
[4] R. Courant, K. Friedrichs, H. Lewy, On the partial difference equations of mathematical physics, IBM Journal (1967) 215–234.
[5] D.L. Williamson, The evolution of dynamical cores for global atmospheric models, Journal of the Meteorological Society of Japan 85B (2007) 241–269.
[6] W.M. Putman, S.-J. Lin, Finite-volume transport on various cubed-sphere grids, Journal of Computational Physics 227 (1) (2007) 55–78.
[7] M. Taylor, J. Tribbia, M. Iskandarani, The spectral-element method for the shallow-water equations on the sphere, Journal of Computational Physics 130 (1) (1997) 92–108.
[8] J. Drake, I. Foster, J. Michalakes, B. Toonen, P. Worley, Design and performance of a scalable parallel community climate model, Parallel Computing 21 (10) (1995) 1571–1591.
[9] O. Morgenstern, M.A. Giorgetta, K. Shibata, V. Eyring, D.W. Waugh, T.G. Shepherd, H. Akiyoshi, J. Austin, A.J.G. Baumgaertner, S. Bekki, P. Braesicke, C. Brühl, M.P. Chipperfield, D. Cugnet, M. Dameris, S. Dhomse, S.M. Frith, H. Garny, A. Gettleman, S.C. Hardiman, M.I. Hegglin, P. Jöckel, D.E. Kinnison, J.-F. Lamarque, E. Mancini, E. Manzini, M. Marchand, M. Michou, T. Nakamura, J.E. Nielsen, D. Olivié, G. Pitari, D.A. Plummer, E. Rozanov, J.F. Scinocca, D. Smale, H. Teyssèdre, M. Toohey, W. Tian, Y. Yamashita, Review of the formulation of present-generation stratospheric chemistry-climate models and associated external forcings, Journal of Geophysical Research—Atmospheres 115 (D00M02) (2010) 1–18.
[10] C. Ronchi, R. Iacono, P.S. Paolucci, The cubed sphere: a new method for the solution of partial differential equations in spherical geometry, Journal of Computational Physics 124 (1) (1996) 93–114.
[11] M.A. Taylor, A. Fournier, A compatible and conservative spectral-element method on unstructured grids, Journal of Computational Physics 229 (17) (2010) 5879–5895.
[12] R.D. Nair, S.J. Thomas, R.D. Loft, A discontinuous-Galerkin transport scheme on the cubed sphere, Monthly Weather Review 133 (4) (2005) 814–828.
[13] P.H. Lauritzen, R.D. Nair, P.A. Ullrich, A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid, Journal of Computational Physics 229 (5) (2010) 1401–1424.
[14] L.M. Harris, P.H. Lauritzen, R. Mittal, A flux-form version of the conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid, Journal of Computational Physics 230 (4) (2011) 1215–1237.
[15] C. Jablonowski, D.L. Willamson, The pros and cons of diffusion, filters, and fixers in atmospheric general circulation models, in: P.H. Lauritzen, C. Jablonowski, M.A. Taylor, R.D. Nair (Eds.), Numerical Techniques for Global Atmospheric Models, Lecture Notes in Computational Science and Engineering, vol. 80, Springer, 2011, pp. 381–494 (Chapter 13).
[16] S.-J. Lin, R.B. Rood, Multidimensional flux-form semi-lagrangian transport schemes, Monthly Weather Review 124 (9) (1996) 2046–2070.
[17] J.B. Drake, P.W. Jones, M. Vertenstein, J.B. White III, P.H. Worley, Software design for petascale climate science, in: D.A. Bader (Ed.), Petascale Computing: Algorithms and Applications, Chapman & Hall, CRC, 2008, pp. 125–146 (Chapter 7).
[18] P.H. Lauritzen, W.C. Skamarock, Test-case suite for 2D passive tracer transport: a proposal for the NCAR transport workshop, March 2011.
[19] R.D. Nair, P.H. Lauritzen, A class of deformational-flow test cases for linear transport problems on the sphere, Journal of Computational Physics 229 (23) (2010) 8868–8887.
[20] P.H. Lauritzen, J. Thuburn, Evaluating advection/transport schemes using interrelated tracers, scatter plots and numerical mixing diagnostics, Quarterly Journal of the Royal Meteorological Society, submitted for publication. Available from: <http://www.cgd.ucar.edu/cms/pel/papers/LT2011QJR.pdf>.
[21] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, 2.1 ed., University of Tennessee, 2008.
[22] J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G.E. Fagg, E. Gabriel, J.J. Dongarra, Performance analysis of MPI collective operations, Cluster Computing—The Journal of Networks, Software Tools, and Applications 10 (2) (2007) 127–143.