Responses to Referee 1

Thank you for your comments on the computational performance of WAVETRISK. Please find our detailed responses below. Significant changes are indicated in blue in the revised manuscript.

Comment

*1) Since the computational cost can be assumed to be proportional to the number of levels, it should be reported as cpu/(columns*level/core) rather than cpu/(columns/core). This would allow direct comparison between benchmarks using a different number of levels.*

Response

Figure 4 (right) has been changed to use this measure (and we now use active nodes rather than active degrees of freedom, i.e. nodes and edges).

Comment

2) Table 2 is useful but should also include the cost i.e. an additional column with something like
*Rossby wave 0.86*160/1.94e5 = 0.7 ms /27 0.026 ms*
*Baroclinic instability 0.40*40/1.64e5 = 0.1 ms /27 0.0037 ms*
*Baroclinic instability 1.13*40/4.07e5 = 0.1 ms /27 0.0037 ms*
*Held-Suarez 1 degree 0.27*40/1.11e5 = 0.1 ms /19 0.005 ms*
*Held?Suarez (1/4) 0.29*320/4.54e5 = 0.2 ms /19 0.01 ms*

Response

We have added this information to Table 2 (and use active number of nodes, rather than active number of degrees of freedom).

However, we would like to emphasize that this measure does not take into account speed up due to adaptivity, and hence does not really measure the true computational efficiency of the adaptive code. It is useful only to estimate how well the code has been optimized and compare its overhead to non-adaptive codes. We explicitly addressed overhead (and optimisation) in Section 5.1, where we compared the performance of WAVETRISK with DYNAMICO, which uses the same basic algorithm (although its computational implementation is quite different). We estimate the overhead as about 50% in the non-adaptive case and about 4 times in the adaptive case.

Although an optimized and computationally efficient code is obviously important, we developed WAVETRISK to demonstrate the possibilities of dynamical adaptivity to reduce computational cost, increase accuracy and allow extremely high local resolutions compared with a similar non-adaptive model (DYNAMICO, in this case). Figure 11 shows how adaptivity can accelerate codes at high resolutions: compression ratios of 200 and higher are achievable.

Comment

3) The data in the Table above does not seem consistent with the data reported in Figure 4, right, with values around 1 ms, rather than 0.1 ms here. This must be checked/clarified. Similarly, why is the Rossby wave case so "expensive"?

Response

We have modified Figure 4 (right) by dividing by the number of vertical levels. The times are slower than reported for Held-Suarez in Table 2 because we adapted on the trend (instead of variables), the code was compiled using gfortran (instead of ifort) and we used RK45ssp time integration instead of RK4 (which requires an additional trend evaluation). Please note that when we did the scaling study our intent was to measure parallel speed up, not absolute speed, which is why we did not use precisely the same set-up as the later test case runs at lower resolution. We explain this difference in the caption to Figure 4.

There was a typo in the cpu/dt value (it should have been 0.4 s). The caption to Table 2 now explains that the Rossby wave case (besides adapting on the tendency, which is more expensive) uses a 4x4 patch size, rather than 8x8, in order to run on 160 cores with Jmin=5. The trade-offs involved in the choice of patch size and number of domains at the coarsest scale (and hence the maximum number of cores) are explained in section 5.1.

Comment

4) p. 25 "the high resolution case is run on 640 cores", yet Table 2 says 320 cores. Which is it?

Response

It is 320 cores (640 cores was the maximum used for the adaptive parallel scaling runs using the Held-Suarez test case). This typo has been corrected.

Comment

5) To avoid any ambiguity "milliseconds" should be spelled out somewhere.

Response

This is spelled out in the caption to Figure 4.

Comment

6) The most favourable case of 0.0037 ms (per timestep per degree of freedom) does not compare well at all with other implementations of comparable numerical schemes. For example for the hydrostatic, sigma-coordinate, ROMS code, this cost was reported to be 2e-5 ms (almost 200 times faster!) and this back in 2005/2007... (I didn't look for more recent numbers) see e.g.

Zuo, Yue, Xingfu Wu, and Valerie Taylor. "Performance Analysis and Optimization of the Regional Ocean Model System on TeraGrid." TeraGrid'07 Conference.

Wang, Ping, et al. "Parallel computation of the regional ocean modeling system." The International Journal of High Performance Computing Applications 19.4 (2005): 375-385.

Even though ROMS has been highly optimised, one would expect even naive implementations of explicit shallow-water solvers to have absolute performance of order at least 1e-4 ms (per timestep per degree of freedom, /dtdf).

I sincerely hope that this is due to a trivial error in the numbers above. If this is not, then this must be discussed in detail (i.e. possible explanations and solutions should be proposed) and the conclusions on the usefulness/applicability of the method/implementation should be toned down (a

Cores	Performance
256	0.1141 ms
128	0.0410 ms
64	0.0180 ms
32	0.0074 ms
16	0.0049 ms

Table 1: ROMS performance figures from Lupo et al 2017.

lot). Note that the numbers given above for ROMS are indicative only (more recent numbers would most probably be even less favourable) and the authors should look for performance data for other (atmospheric) solvers.

Response

The performance figures you cite from these papers **clearly cannot be correct**, and are more than two orders of magnitude faster than ROMS's actual performance. Guillaume Roullet (LOPS, University of Brest) who is a user and developer of ROMS, has confirmed that, depending on the configuration, a cost of a few microseconds is normal for ROMS, and less than 1e-3 ms is out of reach on current supercomputers. In addition, the figures you cite would imply that ROMS is up to 800 times faster than CAM3 (using results from Jablonowski and Williamson 2006 cited in the manuscript)!

Before making such extreme claims it would have been prudent to double check them against other results.

Contrary to your claim, "more recent numbers" confirm that the figures from the two papers you cite are clearly incorrect. The more recent paper Lupo et al (Enhancing Regional Ocean Modeling Simulation Performance with the Xeon Phi Architecture, Ocean 2017, IEEE) "presents a detailed exploration of the acceleration of the Regional Ocean Model System (ROMS) software with the latest Intel Xeon Phi x200 architectures". It reports well-documented results (for the upwelling test case), comparable to WAVETRISK and atmospheric codes, shown in Table 1. It is not credible that the 2007 version of ROMS running on slower machines would be more at least 240 times faster than the 2017 version running on the latest processors!

Note that, because it is a regional model, ROMS can use a regular Cartesian grid, which is more efficient than the irregular grids (such as WAVETRISK's icosahedral grid) that are required on the sphere.

Jablonowski and Williamson (QJR Meteorol Soc 2006, 132, pp 2943–2975) present performance results for three different atmosphere models: NCAR CAM3 (finite volume), NCAR CAM3 (spectral Eulerian) and the GME icosahedral model from the German Weather office (DWD) applied to the baroclinic instability test case we consider in the paper. These performance results are from the same period as the two papers you cite, and should therefore be directly comparable in terms of code development and computer performance. The equivalent performance measures are summarized below (32 Power PC cores) are shown in Table 2.

WAVETRISK performs similarly to these highly optimized operational atmosphere models,

Model	Performance
GME	0.0057 ms
CAM3 FV	0.0039 ms
CAM3 Eulerian	0.0157 ms
WAVETRISK	0.0173 ms (160 cores)

Table 2: Performance figures for the baroclinic wave test from (Jablonowski and Williamson 2006).

even without taking into account the speed-up due to adaptivity. Because WAVETRISK achieves a compression ratio of 5.2 for this test case, its true "adaptive" performance is 0.0035 ms. Greater compression ratios (i.e. at higher resolutions with more scales) improve WAVETRISK's performance. Note that an exact comparison requires running the codes on the same machine and the same numbers of cores.

Recall that we actually did such an apples-to-apples comparison of WAVETRISK with (non-adaptive) DYNAMICO, and found a minimum overhead of approximately 50% in the non-adaptive case and an overhead of about four times in the adaptive case (see Section 5.1). Therefore, WAVETRISK should be faster than DYNAMICO if the compression ratio is greater than 4. This overhead is likely similar compared with other non-adaptive "research" codes.

Finally, Heinzeller et al (Geosci. Model Dev., 9, 77–110, 2016) present recent results for the MPAS climate model. The results are slower than the above results: 0.039 ms (120 km grid on 160 cores) to 0.16 ms (3 km grid on 32768 cores). The slower speed is not surprising since MPAS is a full Earth systems model (atmosphere/ocean/land-ice). But even so, it not possible that MPAS running on the fastest current computers would be up to 1000 times slower than ROMS running on two 15 year old machines over a transcontinental ethernet connection!

Comment

p.23 "Nevertheless, figure 4 (left) shows reasonably linear speed up to at least 640 cores."

"The more sensitive measure of strong scaling in figure 4 (right) is further from a constant (with a maximum variation of 3 times), although there is not a definite trend with increasing numbers of cores."

The scaling for the adaptive case in figure 4, left or right, is definitely not linear. Figure 4-right is not "a more sensitive measure" than figure 4-left (it is exactly the same data), it just leaves less room for wishful interpretation. It is obvious that the two sets of points on figure 4-left have different slopes. The non-adaptive case indeed shows linear scaling (from 8 cores up), as also confirmed more clearly on figure 4-right, while the adaptive case shows a roughly power-law scaling with an exponent clearly lower than 1 (maybe even 1/2? it would be good to indicate this), as more clearly shown on figure 4-right.

Response

A linear fit to the log-log plots shows a scaling of 0.78 and we now indicate this in the caption to Figure 4 and in the text.

We have changed "more sensitive" to "more precise".

Comment

p.28 "We demonstrate excellent strong parallel scaling up to at least 2560 cores in the perfectly balanced case and up to at least 640 cores in an unbalanced case." The second part of this sentence needs to be modified to reflect the previous comment.

Response

The text has been revised.

WAVETRISK-1.0: an adaptive wavelet hydrostatic dynamical core

Nicholas K.-R. Kevlahan[1] and Thomas Dubos[2]

[1]Department of Mathematics and Statistics, McMaster University, Hamilton, Canada
[2]Laboratoire de Météorologie Dynamique, École Polytechnique, Palaiseau, France

Correspondence: N. Kevlahan (kevlahan@mcmaster.ca)

Abstract. This paper presents the new adaptive dynamical core `wavetrisk`. The fundamental features of the wavelet-based adaptivity were developed for the shallow water equation on the β-plane and extended to the icosahedral grid on the sphere in previous work by the authors. The three-dimensional dynamical core solves the compressible hydrostatic multilayer rotating shallow water equations on a multiscale dynamically adapted grid. The equations are discretized using a Lagrangian vertical coordinate version of the `dynamico` model. The horizontal computational grid is adapted at each time step to ensure a user-specified relative error in either the tendencies or the solution. The Lagrangian vertical grid is remapped using an adaptive Lagrangian-Eulerian (ALE) algorithm onto the initial hybrid σ pressure-based coordinates as necessary. The resulting grid is adapted horizontally, but uniform over all vertical layers. Thus, the three-dimensional grid is a set of columns of varying sizes. The code is parallelized by domain decomposition using `mpi` and the variables are stored in a hybrid data structure of dyadic quad trees and patches. A low storage explicit fourth order Runge–Kutta scheme is used for time integration. Validation results are presented for three standard dynamical core test cases: mountain-induced Rossby wave train, baroclinic instability of a jet stream and the Held and Suarez simplified general circulation model. The results confirm good strong parallel scaling and demonstrate that `wavetrisk` can achieve grid compression ratios of several hundred times compared with an equivalent static grid model.

Copyright statement.

1 Introduction

Atmospheric flows are intrinsically non-stationary and multiscale. They are characterized by length scales varying from millimetres to thousands of kilometres and time scales from seconds to decades. Effective climate modelling requires long runs (typically decades) that resolve the dynamically significant scales of motion. However, the smallest significant scales of motion are highly intermittent in space and non-stationary in time. Especially at higher resolutions, the portion of the flow with active small scales is relatively small. Attempting to resolve these important small scales with a uniform grid is necessarily inefficient. Either we need to choose a very fine grid to resolve the smallest scales of interest, which limits simulation times, or we use a coarser grid that filters out these small scales, potentially losing important dynamics. In addition, a uniform resolution does not allow effective control of numerical error. Some regions will be under-resolved and others will be over-resolved. A static com-

putational grid also uses computational resources inefficiently by spending cpu time computing dynamically inactive regions. Ideally, the numerical error should scale predictably with the number of computational elements (e.g. nodes) which should, in turn, be proportional to cpu time. This paper introduces a new wavelet-based adaptive dynamical core, `wavetrisk`, that attempts to achieve these goals.

Behrens (2009) provides a good overview of early developments in adaptive atmospheric modelling, including early work by Skamarock et al. (1989) and previous work using moving grids to model tropical cyclones (Harrison, 1973; Jones, 1977). Since this early work, adaptive mesh refinement (AMR) has found many applications, e.g. fluid dynamics (Popinet and Rickard, 2007) and astrophysics (Mignone et al., 2012; Bryan et al., 2014). In such applications, the main added value of AMR is its ability to accurately capture events that are highly localized in space and time at a relatively low computational cost. Jablonowski et al. (2009) evaluated a block-structured AMR method for scalar two-dimensional transport on the sphere. Refinement and coarsening levels are constrained so that there is a uniform 2:1 mesh ratio at all fine grid/coarse grid interfaces In a series of test cases they find that the additional resolution helps preserve the shape and amplitude of the transported tracer while saving computing resources in comparison to uniform-grid model runs. Kopera and Giraldo (2014) evaluated the performance of AMR in a discontinuous Galerkin based IMEX model on a planar two-dimensional grid. They found that AMR could provide up to a 15 times speed-up with minimal overhead. Ferguson et al. (2016) analyzed the performance of an AMR model for the shallow water equations on the cubed sphere. Their model is built using the general purpose Chombo-AMR toolbox of finite difference and finite volume methods for the solution of partial differential equations on block-structured adaptively refined rectangular grids. They test the model for one or two levels of refinement with refinement ratios of $\times 2$, $\times 4$ and $\times 8$ between each level. In comparison, `wavetrisk` currently uses a fixed refinement ratio of $\times 2$, but has been tested for up to six levels of refinement. Ferguson et al. (2016) conclude that AMR can be effective provided that a sufficiently fine coarsest grid is selected.

Beyond its ability to accurately capture events that are quite localized in space and time at a low cost, several additional properties are required for AMR to be an attractive option for atmospheric applications. Firstly, AMR must demonstrate its ability to accurately simulate complex three-dimensional flows, where a large number of important features such as cyclones and waves continuously appear, move and disappear. `Wavetrisk` has already demonstrated this ability for (two-dimensional) shallow-water flows. Furthermore, especially for applications to climate modelling, the robustness of the method over long time scales and its ability to capture accurate statistics should be shown. Finally, an efficient parallel implementation must be developed in order to compete with state-of-the art operational models. Skamarock and Klemp (1993) presented pioneering initial results from area-limited two- and three-dimensional AMR codes. Popinet et al. (2012) reported results for a three-dimensional AMR code on the cubed sphere at a conference, and Ferguson (2018)'s thesis also describes the development of a global three-dimensional AMR code (based on the CHOMBO liberary). However, although these groups made significant progress in developing global three-dimensional AMR, to the best of our knowledge, no AMR for complex three-dimensional atmospheric flows has been fully documented and analyzed in a peer-reviewed publication.

A key element to the robustness and accuracy of any AMR method is its refinement criterion, which decides when and where to coarsen or refine the computational grid. Ferguson et al. (2016) evaluate several *ad hoc* criteria and identify appropriate ones for their numerical experiments. However they do not find any "clear strategy for establishing the best general refinement

criteria." In contrast, `wavetrisk` uses objective and clearly defined refinement criteria which control the multiscale relative error of the solution or of its tendencies as measured directly by the wavelet coefficients. Although this strategy is objective and clearly defined, since it is based solely on the normalized local approximation error of the solution or tendencies, it is not sufficient to control more qualitative measures of error, such as the location of a vortex. In particular, this approach would need to be augmented when subgrid parameterizations are included, such as for cloud formation or turbulence. Other refinement indicators would be appropriate when considering p-refinement (Naddei et al., 2019).

In the following section we review briefly the basic properties and computational features of the wavelet-based dynamical adaptivity laid out in Dubos and Kevlahan (2013) and Aechtner et al. (2015). Section 3 summarizes the discrete equations solved by the model and presents the ALE vertical coordinate. Section 4 applies the principle of wavelet-based adaptivity to the three-dimensional model. In section 5, implementation details are given and the adaptive solver is evaluated using three test cases of increasing complexity. Section 6 concludes.

2 The adaptive wavelet method

2.1 Wavelet adaptivity on the plane

The foundations for `wavetrisk` were set out in Dubos and Kevlahan (2013) for the shallow water equations on the β plane. We used the TRiSK discretization scheme on the hexagonal–triangular C-grid proposed by Ringler et al. (2010) because of its excellent mimetic properties. This scheme is second order in space in the L^2 norm. The principal goal of our approach was that the adaptivity should be an overlay on the flux-based discretization of any order. Mimetic properties (e.g. mass conservation) should be preserved by the adaptivity and the discretizations should be unchanged. The building blocks of the method are one-scale operators (in this case, the TRiSK discretization) and two-scale operators between a fine scale $j+1$ and a coarse scale j. To conserve the mimetic properties of the TRiSK scheme the restriction operators from scale $j+1$ to scale j and the discrete differential operators (div, grad, curl) must satisfy the following commutation properties

$$R_\mu^j \circ \mathrm{div}^{j+1} = \mathrm{div}^j \circ R_F^j \quad \textit{conserves mass}, \tag{1}$$

$$\mathrm{curl}^j \circ R_{\mathbf{u}}^j = R_\zeta^j \circ \mathrm{curl}^{j+1} \quad \textit{conserves circulation}, \tag{2}$$

$$\mathrm{grad}^j \circ R_B^j = R_{\mathbf{u}}^j \circ \mathrm{grad}^{j+1} \quad \textit{no spurious vorticity}, \tag{3}$$

where R_μ is mass density (or height) restriction, R_F is the flux restriction, R_u is the velocity restriction, R_ζ is the circulation (vorticity) restriction, and R_B is the Bernoulli function restriction. The third commutation relation ensures that a flow with uniform potential vorticity remains uniform under the advection by an arbitrary velocity field (i.e. vorticity is advected like a tracer).

The C-grid is a staggered grid where vorticity is located on triangles (the primal grid), and mass is located on the hexagons formed from the bisectors of the triangle edges (the dual grid). Velocity is located on the edges of triangles, which are also the perpendicular bisectors of the hexagonal edges. (Note that we have chosen the opposite notation to Ringler et al. (2010) and Dubos et al. (2015) since in the multiscale case the triangle grid is generated first by repeated bisection from the icosahedron

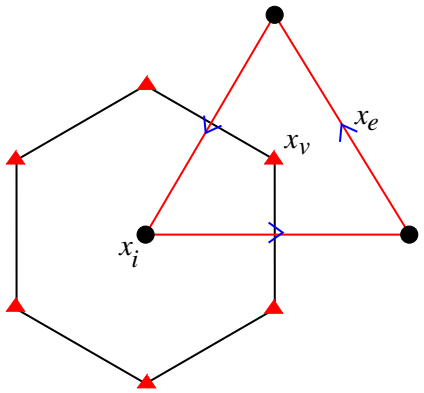

Figure 1. The regular C-grid on the plane. Vorticity is located on the primal grid of triangles at points x_v and mass located on the dual grid of hexagons at nodes x_i. The velocities/fluxes are located on edges x_e. A multiscale hierarchy of nested refined C-grids is generated by bisecting the triangle edges.

and the hexagon grid is generated as the dual grid of the triangles.) The regular C-grid is shown in figure 1. Starting with a coarsest grid, a multiscale hierarchy of primal grids is constructed by bisection of the triangle edges. The dual grid of hexagons is constructed from the perpendicular edge bisectors of the primal grid. On the plane, both the hexagons and triangles are nested and regular. We will see below that this is not the case for the sphere.

This hierarchy of computational grids leads to a so-called wavelet multiresolution analysis (MRA), i.e. the nested sequence of approximation subspaces used to construct the second-generation biorthogonal wavelets (Sweldens, 1998) that are the basis of the grid adaptivity algorithm. The MRA provides a sequence of smooth approximations of a function $f(x)$ on each grid level j and an associated sequence of "details" which give the differences between the approximation at a fine level $j+1$ and a coarse level j. The details are effectively the interpolation errors between scales $j+1$ and j. The smooth approximation at scale j has a basis of *scaling functions* $\{\phi_k^j(x)\}$, while the details between scales $j+1$ and j has a basis of *wavelets* $\{\psi_k^j(x)\}$.

If the wavelet coefficient at a particular position k and scale j is sufficiently small, i.e. $|\psi_k^j(x)| < \varepsilon$, then the value of $f(x)$ is well-approximated by the function $f_{\geq}(x)$ interpolated from neighbouring scaling function values at the same scale. If the wavelet coefficient is large, the value of the wavelet coefficient (i.e. the details) must be retained and added to the interpolated value to obtain an sufficiently accurate approximation. Neglecting the small wavelet coefficients (and associated grid points) generates a multiscale hierarchy of adapted grids. The commutation relations (2–3) ensure that the mimetic properties of the discretization are also satisfied on the adapted grid.

One can prove that this nonlinear wavelet filtering provides error control,

$$||f(x) - f_{\geq}(x)||_\infty = O(\varepsilon), \tag{4}$$

$$\mathcal{N} = O(\varepsilon^{-1/2N}), \tag{5}$$

$$||f(x) - f_{\geq}(x)||_\infty = O(\mathcal{N}^{-2N}), \tag{6}$$

where $\mathcal{N}$ is the number of grid points retained on the adapted grid and N is the order of interpolation. Thus, we can also estimate how many grid points (i.e. computational elements) are required to obtain an approximation with a desired error ε. Setting ε therefore determines the numerical error of the approximation (the tolerance) and also determines the number of grid points in the adapted grid. In principle, it is not necessary to set a maximum resolution scale J since it is determined automatically by ε. Note that an approximation $f^j_{\geq}(x)$ at each scale j is provided by the set of scaling functions $\{\phi^j_k\}_{\geq}$ and their coefficients.

In previous work we have shown that the cpu time τ_n per time step is directly proportional to the number of active grid points $\mathcal{N}$. Combined with the error estimates (6), this shows that adaptive wavelet method is efficient in the sense mentioned in the introduction: the cpu time scales with the specified error and the error is controlled regardless of the structure of the flow. This is not true for non-adaptive methods where there is no explicit link between the actual numerical error and the number of grid points. At best, the approximation error of the discretization on a particular static grid gives only an upper bound on the actual error. Even this upper bound may not be satisfied if the *a priori* estimate of the smallest structures (e.g. strongest gradients) that determined the chosen grid resolution did not properly take into account intermittency and rare events. For example, Yakhot and Sreenivasan (2005) argue that temporal and spatial intermittency mean that turbulent flows require far higher resolution than that found using the usual Kolmogorov scale based estimate, $\Delta x \sim \mathrm{Re}^{-3/4}$.

The above adaptive algorithm controls the error of the solution at each time step, but it does not allow for dynamics. The solution can change over one time step from t^n to $t^n + \Delta t$ by translating at the same scale, coarsening (gradient weakens) or refining (gradient strengthens). If the CFL criterion is one, i.e. $\Delta t < \Delta x_{\min}/||u||_\infty$, the solution can translate by a maximum of one grid point at the same scale over one time step. If the nonlinearities in the governing equations are quadratic, the active scale can increase by a maximum of a factor of two from j to $j+1$ over one time step. To allow for these changes, an "adjacent zone" is added to the set of active wavelet coefficients $\{\psi^j_k\}_{\geq}$ that includes its nearest neighbours in both position and scale (Liandrat and Tchamitchian, 1990). The adaptive grid must also satisfy the perfect reconstruction criterion: there must be sufficient scaling functions (grid points) to construct the wavelets. There must also be sufficient grid points present at each scale j to construct the required TRiSK differential operators (by interpolation, if their associated wavelets are inactive). Once the new adapted grid has been constructed the prognostic variables are interpolated onto the new grid by performing an inverse wavelet transform. After the solution is advanced, the wavelets on the union of the adapted grid and adjacent zone are again filtered using the threshold ε to obtain a new set of active wavelets at time $t^n + \Delta t$.

Because we use a staggered grid, the adaptive wavelet algorithm described above differs fundamentally from previous adaptive wavelet collocation methods (e.g. Mehra and Kevlahan, 2008; Kevlahan and Vasilyev, 2005; Roussel and Schneider, 2010; Schneider and Vasilyev, 2010). Since mass and velocity are located at different points, we must construct two distinct wavelet transforms: a scalar-valued wavelet transform for mass density μ and a vector-valued wavelet transform for velocity u. To control the errors in the tendencies the corresponding thresholds ε_μ and ε_u must be properly scaled. The details of these scalings in the inertia–gravity wave and geostrophic regimes are given in Dubos and Kevlahan (2013).

Finally, note that ideally the time step should also adapt to the local grid scale. In other words, the solution at each scale should be advanced on the time step Δt^j appropriate for that scale. Although this scale-dependent time stepping is optimal,

in practice it does not provide much advantage unless only a small portion of the total active grid points is at the finest scale, which is not usually the case. Domingues et al. (2008) developed a second-order Runge–Kutta scale-dependent time stepping and McCorquodale et al. (2015) extended scale-dependent time stepping to arbitrary order. For simplicity, we have decided not to implement scale-dependent time stepping in `wavetrisk`, although it could be added in the future.

The ability of the adaptive wavelet method to control the errors of the tendencies was verified in Dubos and Kevlahan (2013) and the computational performance and parallel efficiency of the method on the sphere were confirmed in Aechtner et al. (2015). Since the three-dimensional hydrostatic code uses the multilayer shallow equations and horizontal adaptivity only, these properties are inherited by `wavetrisk`.

The prototype `matlab` serial implementation of the adaptive algorithm in the plane is relatively straightforward because the computational grid is uniform and we did not parallelize the algorithm. In the following section we review the extension of the algorithm to the sphere.

2.2 Wavelet adaptivity on the sphere

We have reviewed the essential elements of the adaptive wavelet method in the previous section, however to be practically useful for a dynamical core the method must be extended to the sphere and parallelized. The extension to the sphere presents numerous technical challenges. First, the grid is non-uniform, each computational element is geometrically unique and the icosahedral grid on the sphere has 12 pentagonal dual grid cells (a sphere cannot be tiled uniformly). Secondly, the repeated bisection of the primal triangular grid used to generate the multiscale grid structure produces a grid that is increasingly distorted near the edges of the original icosahedron. Finally, the dual hexagonal grid is no longer nested between successive scales because the triangles used to generate it are not equilateral. This makes it complicated to construct the flux restriction from fine to coarse scales because we must keep track of small overlapping regions. In the following, we focus on the most important modifications necessary to deal with the non-uniform geometry and to parallelize the code in the following.

The wavelet transform on the non-adaptive primal (triangular) icosahedral grid is shown in figure 2. Table 1 gives the properties of the multiscale grids for each level of refinement J.

The basic principles of the adaptive transforms are the same as on the plane, however we must take account of the fact that the geometry of the grid is irregular. The TRiSK discretization Ringler et al. (2010) is second-order accurate for equilateral triangles, but drops to first-order accurate when the triangles are far from equilateral. This problem is not restricted to adaptive methods, but is an issue for all methods on fine icosahedral grids. To deal with this problem we first generate a coarsest grid (e.g. $J_{\min} = 5$ levels of bisection) and then optimize its geometry to ensure that the bisection of primal and dual edges is as close as possible. This optimized primal grid is then bisected as needed to generate the multiscale grids required for the wavelet transform. The default method is to read in optimized grids provided by Heikes et al. (2013). As an alternative, we can also use the grid optimization proposed by Xu (2006), although it produces less optimal grids.

A more fundamental challenge particular to this adaptive flux-based method on staggered grids is that the dual hexagonal grids on two successive scales are no longer nested as they are on the plane. This means that we must keep track of the various configurations of small overlapping hexagons when computing the flux restriction (see section 4.3 and figure 4 of Aechtner

J	N	$\overline{\Delta x}$ (deg)	$\overline{\Delta x}$ (km)
0	12	65.9	7 317
1	42	34.1	3 790
2	162	17.2	1 913
3	642	8.6	960
4	2 562	4.3	480
5	10 242	2.2	240
6	40 962	1.1	120
7	163 842	0.54	60
8	655 362	0.27	30
9	2 621 442	0.14	15
10	10 485 762	0.068	7.5

Table 1. Hierarchy of multiscale primal (triangle) grids derived by edge bisection from the icosahedron at scale $J = 0$. $N = 10 \times 4^J + 2$ is the number of computational elements (lozenges), $\overline{\Delta x} = (4/\sqrt{3}4\pi^2 a^2/20/4^J)^{1/2}$ is the average triangle edge length on the Earth. Each computational element is made up of one node (for scalars), three edges (for velocities) and two triangles (circulation). Therefore the total number of data elements is $5N$ per vertical level. Note that there are two exceptional points to deal with the poles when the icosahedron is unfolded into ten lozenges. Typically the coarsest (optimized) level is $4 \leq J \leq 9$ with three to five levels of refinement.

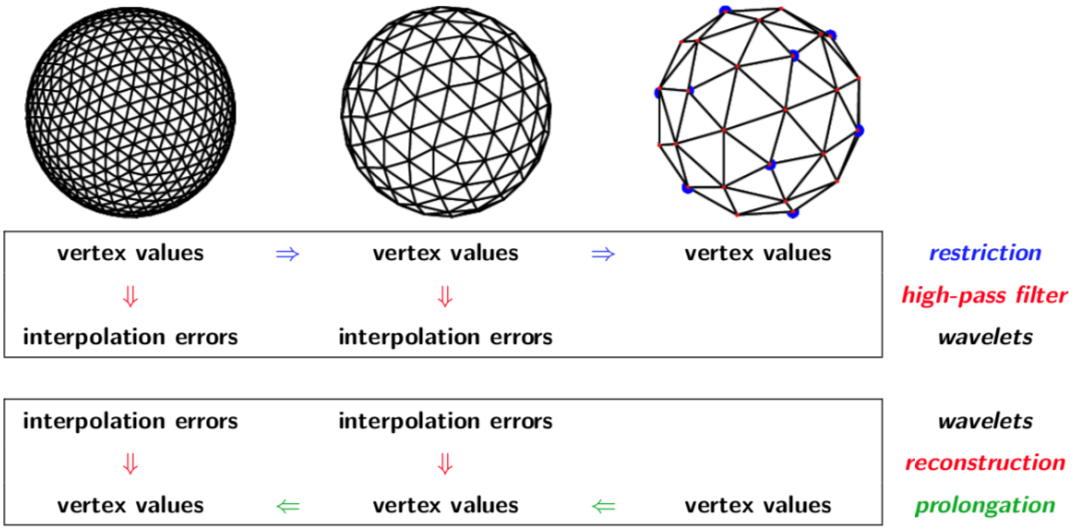

Figure 2. Wavelet transform on a non-adaptive icosahedral grid with three scales.

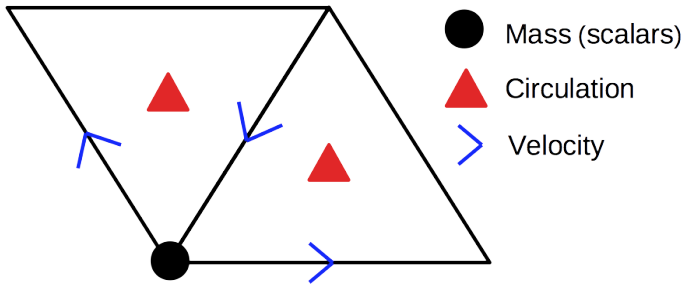

Figure 3. Lozenge: basic computational element containing one node (for mass and other scalars), three edges (for velocities) and two triangles (for circulation). Separate wavelet transforms are provided for the nodes (scalar-valued) and edges (vector-valued). The adaptive grid consists of the the significant nodes and edges, together with nearest neighbours in position and scale necessary for dynamics.

et al. (2015) for more details). Note that the primal grid of triangles remains nested on the sphere, which means that the restrictions of velocity, Bernoulli and circulation are straightforward.

2.3 Parallelization and data structure

The parallelization and data structure must take account both the icosahedral geometry and topology of the spherical discretization and the fact that the grid is adaptive and multiscale. The C-grid is stored as a regular data structure by grouping one node (mass and other scalars), two triangles (circulations) and three edges (velocities) into one computational element, a “lozenge" as shown in figure 3. The icosahedron is composed of 20 triangles grouped into 10 lozenges. Therefore, a grid resulting from refining an icosahedron can be divided into 10 sub-grids each of which can be stored and accessed in a regular fashion. Note that at the edges of the lozenges the two adjacent regular grids of the original icosahedron are rotated with respect to each other. This is dealt with by surrounding the 10 lozenge sub-domains by ghost/halo cells. The halo cells are also used for parallel communication during boundary updates between cores.

The most natural way to store two-dimensional data with a dyadic multiscale structure is to use a quad tree, where each branching represents a new finer scale. However, in an adaptive method some branches are pruned and accessing neighbours would require wasteful traversing of the tree structure. We decided to use a hybrid data structure where each quad tree terminates in a patch. The patches are small 4×4 or 8×8 regular grids. This structure reduces the number of levels in the quad tree and makes it more computationally efficient to find neighbours. A similar hybrid approach was used by Behrens (2009) and Hejazialhosseini et al. (2010). Using patches increases memory slightly because inactive elements are stored. It is also possible to try to optimize the patch size for best computational performance for a particular problem, although 8×8 appears optimal in most cases. Note that the patches do not affect the results of the computation, they just improve computational efficiency.

The domain decomposition used for parallelizing the computation is based on distributing the lozenge sub-domains on the coarsest level $J_{\min}$ (and their associated children) to different cores. Each core can compute several sub-domains and having several small sub-domains per core can improve cache efficiency. Note that there are $10 \times 4^{J_{\min}-(P+1)}$ sub-domains at the

coarsest level with a patch size of $2^P \times 2^P$. For example, if $J_{\text{min}} = 7$ and $P = 2$ the coarsest scale contains $N_D = 2560$ sub-domains. Because the number of cores $N_{\text{core}} \leq N_D$ large numbers of cores are usable only by large J_{min}.

In an adaptive simulation each sub-domain typically has a different number of active computational elements, and thus generates different loads for communication and computation. This load imbalance between cores can seriously degrade parallel performance. To remedy this we use a simple rebalancing algorithm to redistribute sub-domains amongst the cores to produce a more balanced load. This rebalancing is done at each checkpoint save. The checkpoint saves the current load for each coarse sub-domain, which is simply the number of active columns in that sub-domain. This load information is then read when the code restarts. The rebalancing algorithm starts by trying to sequentially distribute the N_D coarse sub-domains over the available ranks to satisfy a user-specified maximum load imbalance (i.e. (maximum load - average load)/average load - 1, currently set to 0.1) while ensuring that every rank has at least one domain and every domain is assigned to a rank. When a rank is "full" (i.e. the load is greater than the average load) the algorithm starts assigning sub-domains to the next rank. If not all sub-domains can been assigned to a rank, the acceptable maximum acceptable load imbalance is increased by a factor of 2 and a new attempt is made to distribute the sub-domains to the available ranks. This maximum acceptable load imbalance is increased iteratively until all domains have been assigned to a rank.

Every sub-domain is extended to hold as many ghost/halo cells as necessary for the various required operators. The values at the halo cells are communicated as needed. Intra-core communication is done by copying and inter-core communication is done using `mpi`. During grid adaption new patches are added and removed as required and grid connectivity between domains is updated (via `mpi` as necessary). Critical communications are carried out locally point-to-point rather than using global communication where possible. Where possible communication is non-blocking so that the computations can continue while communication is taking place in the background.

This parallelization algorithm is reasonably efficient for at least several hundred or a few thousand cores. Dubos and Kevlahan (2013) found that good weak parallel efficiency is possible with as few as 1300 computational elements per core in adaptive runs. The three-dimensional code has better parallel efficiency because the column structure of the data produces a higher computational load for each active grid element. We present some representative strong and weak parallel scaling results in section 5.1.

3 Hydrostatic dynamical equations and ALE vertical coordinates

As mentioned in the introduction, we use the `dynamico` discretization of the three-dimensional hydrostatic multilayer shallow water equations Dubos et al. (2015) in compressible form. The discrete dynamical equations are derived from the discrete Hamiltonian, which allows the construction of energy or potential enstrophy conserving equations. The prognostic variables are m_{ik} (mass), Θ_{ik} (mass-weighted potential temperature) and v_{ek} (velocity), where k labels a full vertical level, l an interface (half-level) between full vertical levels, i an hexagonal or pentagonal cell, v a triangle and e an edge. In terms of the

Hamiltonian, their evolution equations are :

$$\frac{\partial m_{ik}}{\partial t} + \delta_i \frac{\partial H}{\partial v_{ek}} = 0, \qquad \frac{\partial \Theta_{ik}}{\partial t} + \delta_i \left(\theta^*_{ek} \frac{\partial H}{\partial v_{ek}} \right) = 0, \tag{7}$$

$$\frac{\partial v_{ek}}{\partial t} + \left(\frac{f_v + \delta_v v_k}{\overline{m_{ik}}^v} \frac{\partial H}{\partial v_{ek}} \right)^\perp + \delta_e \frac{\partial H}{\partial m_{ik}} + \theta^*_{ek} \delta_e \frac{\partial H}{\partial \Theta_{ik}} = 0, \tag{8}$$

potential vorticity Bernoulli Exner (9)

where δ_i, δ_e and δ_v are discrete divergence, gradient and curl operators yielding values at cells, edges and triangles respectively, f_v is the Coriolis parameter, and θ^*_{ek}, $\overline{m_{ik}}^v$ are values of θ and m reconstructed at edges and triangles, respectively, by appropriate averaging. Indices than can be inferred may be omitted, as in $\delta_v v_k \equiv \delta_v v_{ek}$. Evaluating the Hamiltonian terms and discretizing, we obtain the inviscid discrete dynamical equations

$$\frac{\partial \mu_{ik}}{\partial t} + \delta_i U_k = 0, \tag{10}$$

$$\frac{\partial \Theta_{ik}}{\partial t} + \delta_i (\theta^*_{ek} U_k) = 0, \tag{11}$$

$$\frac{\partial v_{ek}}{\partial t} + \delta_e B_k + \theta^*_{ek} \delta_e \pi_k + (q_k U_k)^\perp_e = 0, \tag{12}$$

where $\mu_{ik} = \rho_{ik} \Delta z_{ik}$ and we have assumed Lagrangian vertical coordinates (so the vertical mass fluxes are not present). Potential temperature $\theta_{ik} = \Theta_{ik}/\mu_{ik}$ and $\theta_{ek} = \overline{\theta_{ik}}^e$. μ_{ik} is a pseudo mass density (equivalent to $\rho_{ik} \Delta z_{ik}$), U_{ek} is the horizontal mass flux, B_{ik} is the Bernoulli function, $\pi_{ik} = \pi(\alpha_{ik}, \theta_{ik})$ is the Exner function, α_{ik} is the specific volume, and q_{vk} is the potential vorticity. In the compressible case we consider here, the Bernoulli function is given by

$$B_{ik} = K_{ik} + \overline{\Phi_{il}}^k, \tag{13}$$

where K_{ik} is the discrete kinetic energy computed from u^2_{ek} using appropriate averaging, and Φ_{il} is the geopotential at vertical layer interfaces l. The discrete operators δ_i (divergence with result at a node), δ_e (gradient with result at an edge) and $(\cdot)^\perp$ (perpendicular flux) are defined as in Ringler et al. (2010) and the shallow water equations version of `wavetrisk`.

Each evaluation of the trend requires integrating vertically down to find the surface pressure and then integrating up to find the Exner function and geopotential (and hence the Bernoulli function).

The choice of Lagrangian vertical coordinates (rather than mass-based) is simple, computationally efficient and especially well suited to ocean modelling since it virtually eliminates numerical vertical diapycnal diffusion, unlike a $z-$coordinate. This will become important when we develop the ocean version of `wavetrisk` (see Kevlahan et al., 2015). The vertical coordinates are pressure-based and may either be evenly spaced or hybrid (σ). As the flow develops the vertical levels expand and contract which can lead to loss of accuracy, or even negative mass (if a layer collapses to zero thickness). To avoid this problem, we remap the vertical coordinates back to the original coordinates either every time step or or every ten time steps or so. The remapping takes about 7 % of cpu time if done every time step.

Remapping Lagrangian vertical coordinates is a common technique in ocean modelling (e.g. Petersen et al., 2015), where the target grid is updated at each time step to optimize numerical accuracy and stability (e.g. the target grid may be based on approximately isopycnal or isentropic coordinates) and is referred to as an “arbitrary Lagrangian–Eulerian" (ALE) coordinate

system. Optimizing the target grid in `wavetrisk` would add a sort of vertical $r-$adaptivity in addition to the wavelet-based horizontal $h-$adaptivity, where the number of vertical levels would remain constant but their locations would be chosen optimally. For example, Kavcic and Thuburn (2018) choose the target grid levels at each time step to keep the vertical levels close to isentropic. This requires solving a small elliptic optimization problem at each grid level. In principle, it is also possible to include vertical $h-$adaptivity by locally de-activating certain vertical layers if the vertical interpolation errors are small. We have tested `wavetrisk` with a variety of piecewise constant, piecewise linear, piecewise parabolic and piecewise quartic remapping schemes, modified from packages supplied by Shchepetkin (2001) and Engwirda and Kelley (2016, https://github.com/dengwirda/PPR). The user can select no limiter, a monotone limiter or a sWENO limiter. Any of these remapping algorithm can be selected simply by changing a parameter.

Mass density μ, potential temperature θ and velocities u, v, w are remapped onto the original hybrid pressure coordinates. This remapping scheme conserves mass, potential temperature and divergence and appears to perform well. We also tested Lin (2004)'s scheme, which remaps momentum and total energy, but it proved to be less stable.

We found that simple piecewise constant remapping gives qualitatively incorrect results for zonally averaged statistics in the Held and Suarez (1994) test case, but that it is sufficient for the mountain-induced Rossby wave and baroclinic instability test cases. Piecewise linear and piecewise parabolic remapping give qualitatively accurate results. The target grid could be optimized at each remap, as in ocean models and as explored by Kavcic and Thuburn (2018) for the `endgame` GCM. However, the current procedure gives good results in the test cases we have examined.

The dynamical equations are advanced in time using the fourth-order four-stage low storage Runge–Kutta routine used in `dynamico` (Dubos et al., 2015). Various strong stability preserving schemes (e.g. the third-order three-stage scheme RK33ssp, RK45ssp) are also available as options (Spiteri and Ruuth, 2002).

In many cases the code runs stably without any additional diffusion or filtering of small scales. However, in rare cases the code crashes due to numerical instability. In order to improve stability a regular or second order hyper-diffusion term is added to the dynamical equations for the prognostic variables to damp the largest wavenumbers (both the divergent and vortical modes of the momentum equation are damped). Using the discrete operator notation of Dubos et al. (2015) the diffusion terms for scalars, divergence and vorticity are, respectively

$$D_\phi = K_\phi \delta_i \left[\frac{l_e}{d_e} \delta_e \left(\frac{\phi}{A_i} \right) \right], \tag{14}$$

$$D_\delta = K_\delta \delta_e \left[\frac{1}{A_i} \delta_i \left(\frac{l_e}{l_d} v_e \right) \right], \tag{15}$$

$$D_\omega = K_\omega \delta_e \left[\frac{1}{A_v} \delta_v \left(v_e \right) \right], \tag{16}$$

where d_e is a triangle edge length (primal grid), l_e is a hexagon edge length (dual grid), A_i is a hexagon area, A_v is a triangle area. These discretizations correspond to the continuous differential operators $\boldsymbol{\nabla} \cdot \boldsymbol{\nabla}(\phi)$, $\boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot \mathbf{u})$ and $\boldsymbol{\nabla} \times (\boldsymbol{\nabla} \times \mathbf{u})$ respectively.

A general p-th order hyperdiffusion operator is defined as an iterated Laplacian operator Δ^p corresponding to D^p_ϕ, D^p_δ, D^p_ω. We choose either $p = 1$ (regular diffusion) or $p = 2$ hyperdiffusion. Diffusion may be applied at each time step, or every

$N_{\text{diff}} > 1$ time steps (where N_{diff} is limited by viscous stability in time). The diffusion coefficient K is chosen to give the same amount of damping over a time step,

$$K = \frac{\overline{\Delta x}^{2p}}{\Delta t} C N_{\text{diff}}, \tag{17}$$

where $\overline{\Delta x} = (4\pi a^2/(10\,4^J + 2))^{1/2}$ is the average grid scale and C is an empirical constant chosen to ensure stability, which may be different for different variables.

Note that in addition to ensuring stability, adding diffusion can improve the efficiency of the adaptivity by damping out small fluctuations that might otherwise produce some local grid refinement. This effect is especially significant when adapting on the wavelets of the tendencies. We present results both with and without explicit diffusion.

4 Adaptivity

To extend the adaptive algorithm for the two dimensional shallow water equations reviewed in section 2.1 to the three-dimensional case we simply apply the two-dimensional algorithm to each vertical layer in turn and then define the adapted grid to be the union of the adapted grids over all vertical layers $k = 1, \ldots, N$.

After the time step and vertical re-gridding have been completed, the wavelet transforms of mass density μ, mass-weighted potential temperature Θ and velocity u are calculated for each vertical level. Then, grid points are labeled active if their associated wavelet coefficient has a magnitude greater than or equal to the appropriate relative error threshold. Nearest neighbours are added to the adjacent zone in position and scale. Note that a node is labelled active if any of its associated scalar wavelets (i.e. mass wavelet or mass-weighted potential temperature wavelet) is significant in *any* vertical layer. An edge is labelled active if its associated vector wavelet (i.e. velocity wavelet) is active in *any* vertical layer. Grid points are added as necessary to satisfy the perfect reconstruction criterion (so the grid points necessary to compute the wavelet coefficients are present). Finally, grid points required for the TRiSK operators are labelled. An inverse wavelet transform interpolates the solution conservatively onto the new adapted grid. The adaptivity algorithm is summarized in algorithm 1.

The adaptive algorithm produces an adapted grid consisting of vertical columns of varying horizontal size. In practise, a maximum scale J is usually set based on available computational resources and user requirements. This “column adaptivity" approach is not optimal for vertically tilted structures, but it provides much better load balancing and is far simpler than dealing with fully three-dimensional adaptivity. In addition, we can take advantage of an ALE formulation for the vertical coordinate which is often more flexible and accurate than a z-coordinate system. We show in the results section that column adaptivity provides accurate results and good grid compression ratios. Note that the vertical grid is remapped (if necessary) before adapting the horizontal grid.

The normalizations for the absolute tolerances $\varepsilon_\mu = \varepsilon||\mu||_\infty$, $\varepsilon_\Theta = \varepsilon||\Theta||_\infty$ and $\varepsilon_u = \varepsilon||u||_\infty$ determining the grid adaptation may be estimated *a priori* by dimensional analysis and knowledge of the flow being simulated, or they are determined dynamically by calculating the appropriate norms separately for each vertical layer at each time step.

Algorithm 1 Adaptive grid algorithm. Executed after the time step and vertical grid remapping is complete. Note that wavelets are calculated and filtered at all vertical levels so that the final adapted grid is the union of adapted grids over all vertical levels.

After time step and vertical grid remapping have been completed
if adapt on trend **then**
  Compute trend
  Compute trend wavelets
end if
for k = 1**to** N **do** {loop over all vertical levels}
  Update relative error thresholds ε_μ, ε_Θ, ε_u by computing appropriate norms at each vertical level
end for
Label all active nodes and edges at coarsest scale $j = J_{\min}$ as **adjacent zone**
Label all nodes and edges for scales $j > J_{\min+1}$ as **inactive**
for $k = 1$ **to** N **do** {check all vertical levels}
  for $j = J - 1$ **to** $J_{\min}$ **do** {check wavelets at all scales}
    label as **active** all nodes with an associated wavelet coefficients $\geq \varepsilon_\mu$ OR $\geq \varepsilon_\Theta$
    label as **active** all edges with an associated wavelet coefficients $\geq \varepsilon_u$
  end for
end for
Add **nearest neighbours** of active grid points at **coarser scale** to **adjacent zone**
Add **nearest neighbours** of active grid points at **same scale** to **adjacent zone**
Add/remove **patches** as required for neighbours at finer scales
for $j = J - 1$ **to** $J_{\min}$ **do**
  add neighbours of active grid points at **finer scales**
end for
Perfect reconstruction criterion: add grid points required to compute active and adjacent zone wavelets
Label grid points required by **TRiSK operators** to compute trend at active and adjacent zone wavelets
Set all **wavelets** not active or in the adjacent zone to zero
Inverse wavelet transform of solution onto new adapted grid

The choice of normalization for the absolute thresholds ε_μ, ε_Θ and ε_u is a crucial and sensitive part of the algorithm since they guarantee the relative accuracy and efficiency of the method. In the shallow water case we determined the tolerances for mass and velocity based on dimensional analysis of the tendencies in the inertia-gravity and quasi-geostrophic regimes. This sort of dimensional analysis is not feasible for the three-dimensional equations so we have developed two strategies to ensure uniform control of the relative error in the tendencies of the prognostic variables in each vertical layer.

As we mentioned earlier, in the first approach these tolerances are set by dimensional analysis using a knowledge of the appropriate scales for the test problem under consider. This gives reasonable results for problems with fairly stationary evolution, but can lead to less accurate results if the dynamics change significantly during the course of the simulation. It also generally means that the tolerances set to the same value for all vertical levels.

In the second approach, the relevant norms are calculated dynamically at each time step and at each vertical level to ensure they are consistent with the actual state of the flow. This approach is more generally applicable and does not require the user to have any *a priori* knowledge of the solution.

In addition to determining how to calculate the normalization for the thresholds, we also need to decide which wavelets we are filtering. In the shallow water case we always directly filtered the wavelets of the solution, which measure the interpolation errors of the variables themselves between two grid levels at a given time step. Although this does not directly control the tendency error, analysis of a linearized problem suggests that it should provide some control of the tendency error provided the solution is smooth.

Consider a set of n coupled linear ordinary differential equations

$$\frac{du}{dt} = A(u), u(0) = u_0, \tag{18}$$

where A is an $(n \times n)$ constant coefficient matrix. Applying an Euler scheme gives the time step $u^{n+1} = u^n + \Delta t\, A u^n$. The error (wavelet coefficient) $\tilde{u}^n$ satisfies the same equation, so we have

$$\begin{aligned} \tilde{u}^{n+1} &= (I + \Delta t A)\tilde{u}^n, \\ \frac{||\tilde{u}^{n+1}||}{||u^n||} &\leq (1 + \Delta t ||A||)\frac{||\tilde{u}^n||}{||u^n||}, \end{aligned}$$

where I is the identity matrix and we have used the triangle inequality. Now, if we assume that $||\tilde{u}^n||/||u^n|| \leq \varepsilon$ by wavelet filtering of u^n, we have the following bound on the relative error

$$\frac{||\tilde{u}^{n+1}||}{||u^n||} \leq \varepsilon(1 + \Delta t ||A||). \tag{19}$$

If A is symmetric then $||A||_2 = \rho(A)$ where $\rho(A)$ is the spectral radius of A (largest magnitude eigenvalue of A, $|\lambda|_{\max}$). Thus, we have that

$$\frac{||\tilde{u}^{n+1}||_2}{||u^n||_2} \leq \varepsilon(1 + \Delta t\, \rho(A)). \tag{20}$$

However, a necessary condition for stability of the Euler method is that

$$\Delta t\, \rho(A) \leq 2. \tag{21}$$

Which gives

$$\frac{||\tilde{u}^{n+1}||_2}{||u^n||_2} \leq 3\varepsilon. \tag{22}$$

We have therefore shown that wavelet filtering of the solution itself provides control of the relative error of the solution over one time step for symmetric discretizations. In general, for non-symmetric A we have

$$\frac{||\tilde{u}^{n+1}||_2}{||u^n||_2} \leq \varepsilon(1 + \Delta t\, \sigma(A)). \tag{23}$$

where $\sigma(A)$ is the largest singular value of A. However, there exists an $\epsilon > 0$ such that $||A|| \leq \rho(A) + \epsilon$ for any matrix norm $||\cdot||$. Thus, we can expect similar relation to (22) to hold for more non-symmetric discretizations. Although we have only considered the linear constant coefficient case, the results suggest that dynamic wavelet filtering of the solution, together with dynamic calculation of the normalization of the thresholds, should control the relative error of the solution over one time step.

Now, consider filtering on the wavelets of the tendencies T themselves at the previous time step so that $||\tilde{T}^n||/||T^n|| \leq \varepsilon$ where $\tilde{T}$ is now the error (wavelet coefficient) of the tendency. To first order in Δt we have

$$\tilde{T}^{n+1} = \tilde{T}^n + \Delta t\, (J_T(\tilde{u}^n, t^n)\tilde{T}^n + \frac{\partial T}{\partial t}(\tilde{u}^n, t^n)), \tag{24}$$

where $J_T(\tilde{u}^n, t^n)$ is the Jacobian matrix of the tendency evaluated at the previous time step. In the case of an Euler method applied to a linear constant coefficient system of ordinary differential equations as we considered above, we have $T = Au$ and the errors in the tendency $\tilde{T}$ (wavelets of the tendency) satisfy

$$\begin{aligned}
\tilde{T}^{n+1} &= (I + A)\tilde{T}^n, \\
||\tilde{T}^{n+1}|| &\leq ||\tilde{T}^n|| + ||A\tilde{T}^n||, \\
||\tilde{T}^{n+1}|| &\leq ||\tilde{T}^n|| + ||A||\,||\tilde{T}^n||,
\end{aligned}$$

where we have used the triangle and Schwartz inequalities. Normalizing by $||T^n||$ and recalling that wavelet filtering ensures that $||\tilde{T}^n||/||T^n|| \leq \varepsilon$ gives the result

$$\frac{||\tilde{T}^{n+1}||}{||T^n||} \leq (1 + ||A||)\,\varepsilon. \tag{25}$$

Again, if A is symmetric, we have $||A||_2 = \rho(A)$ and using $\rho(A) < 1/\Delta t$ for the Euler method we find

$$\frac{||\tilde{T}^{n+1}||}{||T^n||} \leq (1 + \frac{1}{\Delta t})\varepsilon \leq C\varepsilon. \tag{26}$$

Inequality (26) shows that filtering the wavelets of the tendencies $\tilde{T}^n$ from the previous time step effectively controls the relative tendency error in the next time step up to a constant factor depending on the discretized system. Note that filtering the wavelets of the tendencies is slightly more expensive since it requires an additional evaluation of the tendencies (or an additional inverse wavelet transform).

In the following section we apply wavetrisk to solve three standard test problems for hydrostatic dynamical cores. The emphasis is on evaluating the adaptivity in the three-dimensional hydrostatic version, rather the basic accuracy of the method since the underlying discretization is the same as dynamico and we have already assessed the basic features of the error control in previous work on the shallow water equations. We directly compare the options of nonlinear filtering the wavelets of the solution and wavelet filtering the wavelets of the tendencies. Filtering the tendencies is more precise but more sensitive to the choice of threshold.

5 Validation test case results

5.1 Parallel and computational efficiency

The adaptive wavetrisk code has significant computational overhead compared to dynamico, which solves the same discretized equations. This overhead is required to deal with the local geometry of the grid (which is not pre-computed), the multiscale grid structure and the parallel communication on the hybrid data structure. This overhead increases with the number of refinement levels and decreases for larger patch sizes. A lower bound on the overhead can be estimated by directly comparing wall clock time for wavetrisk and dynamico since both codes are based on the same TRiSK discretization. The codes solved dynamical core model intercomparison project (DCMIP) 2008 test case 8 (see section 5.2) on the uniform grid $J = 6$ (1° degree) with 27 Lagrangian vertical levels and 8×8 patches for wavetrisk. The codes were run on 160 cores. This limited test suggests that wavetrisk is approximately 50% slower than dynamico per active grid point. The actual overhead depends on the number of refinement levels and the patch size (larger patch sizes are more computational efficient but limit the number of cores and increase memory overhead). Note that dynamico usually runs with mass-based vertical coordinates, which adds an additional 15% to its cpu time.

We have also directly estimated the overhead due to multiscale adaptivity for wavetrisk by comparing single scale runs (with the largest possible patch sizes for the given resolution for best performance) and multiple scale runs with minimum resolution $J = 4$ and maximum resolutions $J = 6, 7, 8, 9$ on 40 cores with minimum scale $J = 4$. The tolerance was set to zero to give an upper bound on the cost of the multiscale runs. The code solved the Held and Suarez (1994) test case. We found that the overhead due to the multiscale adaptivity is about three times. We therefore conclude that wavetrisk is about 4 times slower per active grid point than dynamico, and therefore a grid compression factor greater than 4 is required for wavetrisk to be faster than dynamico. Remapping accounts for about 7% of the cost, so remapping every 10 time steps decreases the overhead difference to about 3.7 times. Using RK33ssp instead of RK4 further reduces the overhead to about 2.8 times. Note that, unlike dynamico, we have not made a serious effort to optimize the performance of wavetrisk and it is certainly possible to reduce this overhead significantly. Nevertheless, we will show that compression ratios of up to 1000 times are achievable using the adaptive code and it is certainly much faster than the equivalent non-adaptive code for high resolution intermittent problems.

In order to estimate strong parallel scaling performance we ran two Held and Suarez (1994) general circulation experiments. The first run was non-adaptive at horizontal resolution $1/4^\circ$ with 30 vertical levels for ten time steps. This case is balanced and

figure 4 (left) shows that it has linear speed up from about 8 to at least 2560 cores (the lack of linear scaling for four or few cores is due to the intrinsic overhead of parallel computations). Although speed-up is a commonly used measure of parallel scaling, a more sensitive measure is cpu time per time step divided by the number of vertical columns per core. Perfect scaling is then a constant, and its value gives an indication of absolute performance. Figure 4 (right) shows that this measure is indeed approximately constant from 4 to 2560 cores.

The second trial was fully adaptive using trend filtering, with a minimum horizontal resolution of $1/2^\circ$ and three levels of refinement to $1/8^\circ$ and 18 vertical levels computed with a relative error tolerance $\varepsilon = 0.08$. This simulation was first run for 500 h to allow the climate dynamics to develop. Figure 5 shows the solution and adaptive grid at t=500 h. The grid compression ratio is a relatively modest 4.5. The simulation was then restarted and run for another 4 h (about 300 time steps) to estimate strong parallel scaling and timing. The computational load was rebalanced on restart, but there was no further rebalancing. The load imbalance (ratio of highest to average load) varied from 3 to 8 during this simulation. This run therefore represents a very unbalanced case. Nevertheless, figure 4 (left) shows close to linear speed up (power law exponent of 0.78) to at least 640 cores. The more precise measure of strong scaling in figure 4 (right) is further from a constant (with a maximum variation of 3 times), although there is not a definite trend with increasing numbers of cores. Since weak scaling performance is a better indication of parallel performance than the strong scaling, these results suggest weak scaling should be good to much larger numbers of cores. In fact, the shallow water code on which `wavetrisk` is based showed 70% weak scaling efficiency for as few as 1300 computational elements per core Aechtner et al. (2015). The current three-dimensional code has better parallel performance since the sub-domains are distributed to the cores as complete vertical columns, which mean each core has a larger load.

In summary, `wavetrisk` has a minimum overhead of about 50% compared with `dynamico` associated with the management of the adaptive grids and local geometry. This overhead increases with the number of refinement levels and decreases with patch size. However, the cost per active node is independent of the grid compression ratio. We also show that strong parallel performance is good for $O(10^3)$ cores. This suggests that `wavetrisk` requires a grid compression ratio of at least two to be more computationally efficient than an equivalent non-adaptive code. As we show in the following sections, grid compression ratios of 4 to 200 or more are achievable depending on the flow (see figure 11 for example). In general, higher grid compression ratios are achievable at higher resolutions and in more turbulent (intermittent) cases.

Table 2 summarizes absolute computational performance for the test cases considered in the following sections.

5.2 DCMIP 2008 test case 5: mountain-induced Rossby wave train

The first validation we consider is the mountain-induced Rossby wave train used in the dynamical core model inter-comparison project (DCMIP) in 2008 (Jablonowski et al., 2008). This test case is relatively smooth and does not produce much small scale structure. It is, however, a good test of the ability of the adaptive algorithm to track developing wave instabilities. It is similar to the case described in Tomita and Sato (2004), but with a hydrostatic surface pressure. The simulation starts from smooth, balanced isothermal initial conditions and a Rossby wave train instability is generated by an isolated Gaussian mountain over the course of the first 15 days.

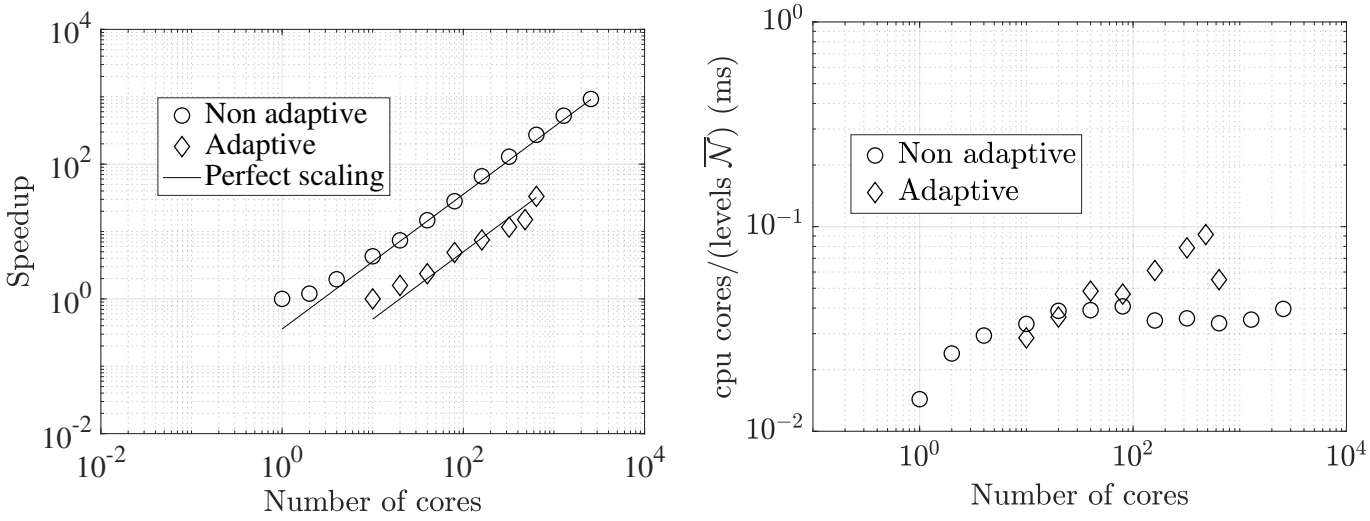

Figure 4. Strong scaling of `wavetrisk` on the Compute Canada machine `niagara` for a simulation of the Held and Suarez (1994) general circulation experiment for a perfectly balanced (non-adaptive) run at a resolution of $J = 8$ ($1/4^\circ$) and for a strongly unbalanced (dynamically adaptive) run at a maximum resolution of $J = 9$ ($1/8^\circ$) resolution with trend based error tolerance $\varepsilon = 0.08$. Left: speed-up compared with perfect (linear) scaling. The non-adaptive case has perfect linear scaling for more than 8 cores while the adaptive case has power law scaling of approximately 0.78. Right: absolute strong scaling performance in milliseconds (ms) (wall-clock time per time step multiplied by the number of cores, i.e. cpu hours per time step, divided by the average number of active nodes over all vertical levels and all scales). $\overline{\mathcal{N}}$ is the average number of active nodes over all scales. Note that the absolute times shown in the right figure are slower than equivalent times reported in Table 2 because we used a RK45ssp (with one additional trend evaluation) and code was compiled with `gfortran` rather than `ifort` for the scaling runs.

Test case	adaptivity	cores	$\overline{\Delta t}$	cpu / day	cpu $/\Delta t$	$\overline{\mathcal{N}} \times 10^4$	compression	total cost
Rossby wave	tendency	160	170 s	203 s	0.40 s	4.85	4.43	0.0508 ms
Baroclinic instability	variables	40	240 s	310 s	0.40 s	4.10	5.22	0.0152 ms
Baroclinic instability	tendency	40	237 s	411 s	1.13 s	10.2	2.11	0.0173 ms
Held–Suarez (1°)	variables	40	287 s	81 s	0.27 s	2.78	1.93	0.0216 ms
Held–Suarez ($1/4^\circ$)	variables	320	66.5 s	375 s	0.29 s	11.4	7.58	0.0456 ms

Table 2. Summary of actual computational performance for each of the test cases considered here. All runs were dones on the Compute Canada machine `niagara` and the values shown are averages over the whole simulation. cpu is wall-clock time, $\overline{\mathcal{N}}$ is the average number of active nodes (over all vertical levels and all scales), total cost is wall-clock time per time step times cores (i.e. cpu hours per time step) per active node per vertical level. (Note that total cost does *not* take into account the speed up due to parallelism or adaptivity: it measures cpu hours per active node.) The Rossby wave run is more expensive because it uses a smaller patch size (4×4 rather than 8×8) in order to run on 160 cores with $J_{\min} = 5$. Please see the discussion at the beginning of section 5.1 for an explanation of the trade-offs involved in patch size versus number of domains.

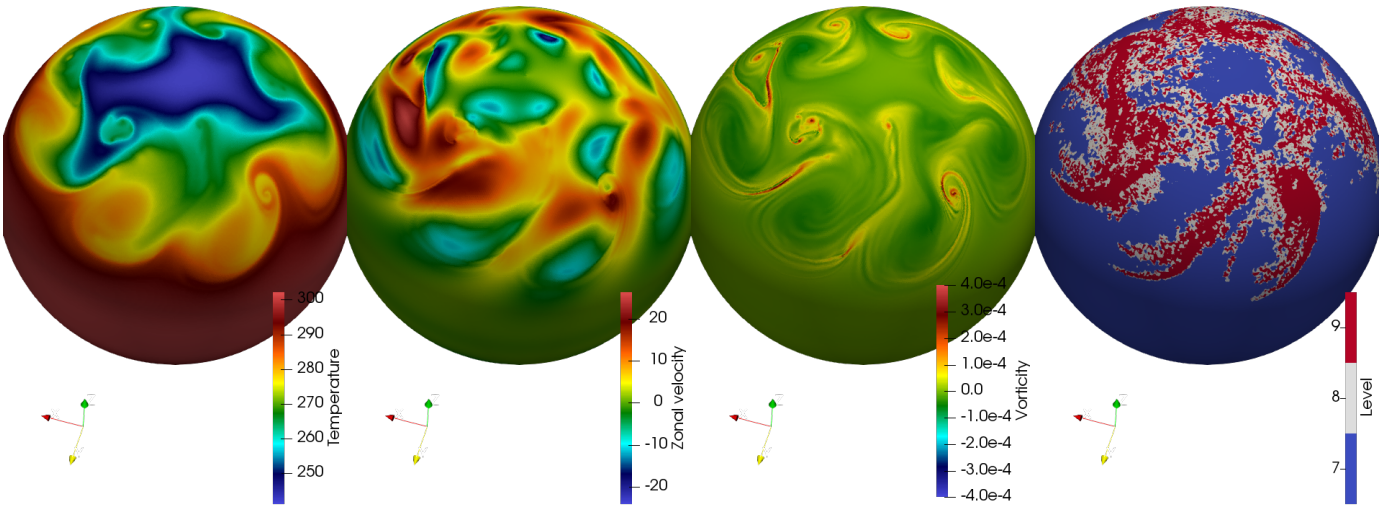

Figure 5. Adaptive Held and Suarez (1994) general circulation experiment at t=500 h with maximum resolution $1/8^\circ$ used for the strong parallel scaling results shown in figure 4. Results are shown at the vertical level corresponding to 850 hPa. The rightmost figure is the adaptive grid with a grid compression ratio of about 4.5.

The coarsest scale in the simulation is $J = 5$ (2°) and the finest scale is $J = 7$ $(1/2^\circ)$ with 27 vertical hybrid σ pressure levels. The vertical grid is remapped when the thickness of a vertical level has dropped to 30% of its initial value. There is no diffusion. The grid is adapted on the tendency wavelets. We run the simulation for a total of 30 days. The results are shown at day 25 on sphere in figure 6.

Figure 7 shows equidistant cylindrical rectangular latitude–longitude projections of the results of the adaptive simulation at 700 hPa at day 25. The adaptive data is first interpolated to a uniform $J = 7$ $(1/2^\circ)$ grid, then interpolated to the desired pressure level and finally projected onto the plane. The results are in good qualitative agreement with those shown in the DCMIP 2008 report (Jablonowski et al., 2008). There is some difference in the weaker structures, which is inevitable given that the adaptive simulation necessarily resolves the more intense structures more highly than the weaker ones.

The mountain induced Rossby wave train is a good first validation of the `wavetrisk`. However, it is relatively smooth, it does not test the ability of the code to deal with intense instabilities and has no “physics" (e.g. cooling, or Rayleigh drag). The following baroclinic instability test case is much more intense and generates small scale vorticity filaments.

5.3 DCMIP 2012 test case 4: baroclinic instability of jet stream

Jablonowski and Williamson (2006) proposed a deterministic test case for dry dynamical cores of atmospheric general-circulation models that simulates the evolution of a baroclinic wave in the northern hemisphere. Perturbation of an analytic

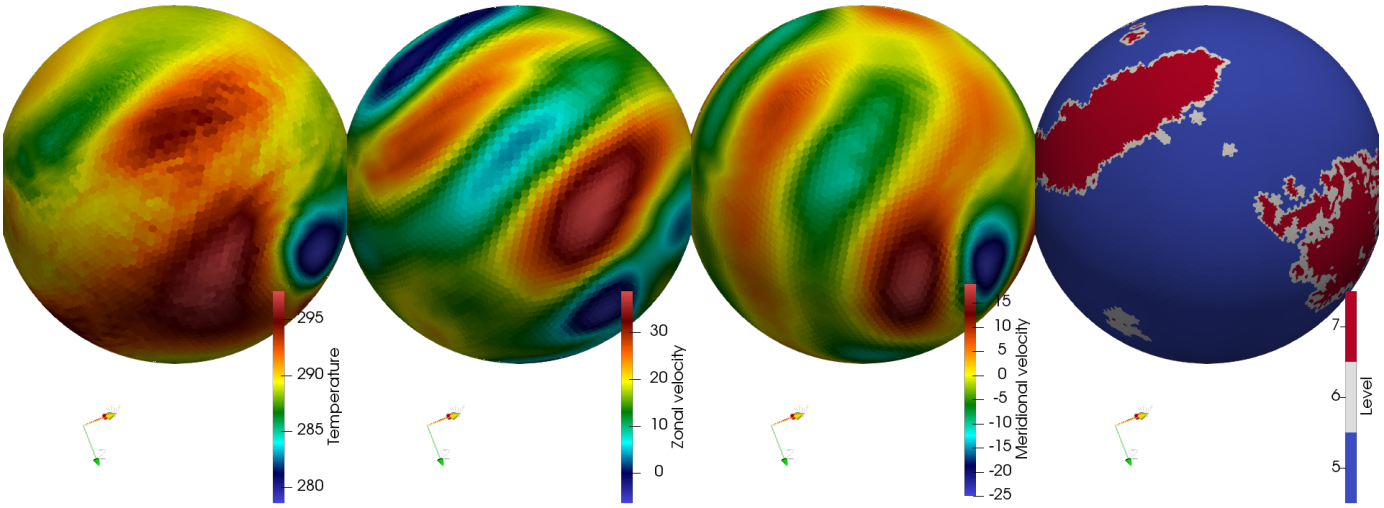

Figure 6. Results of the adaptive simulation of the mountain induced Rossby wave case shown at 700 hPa at day 25.

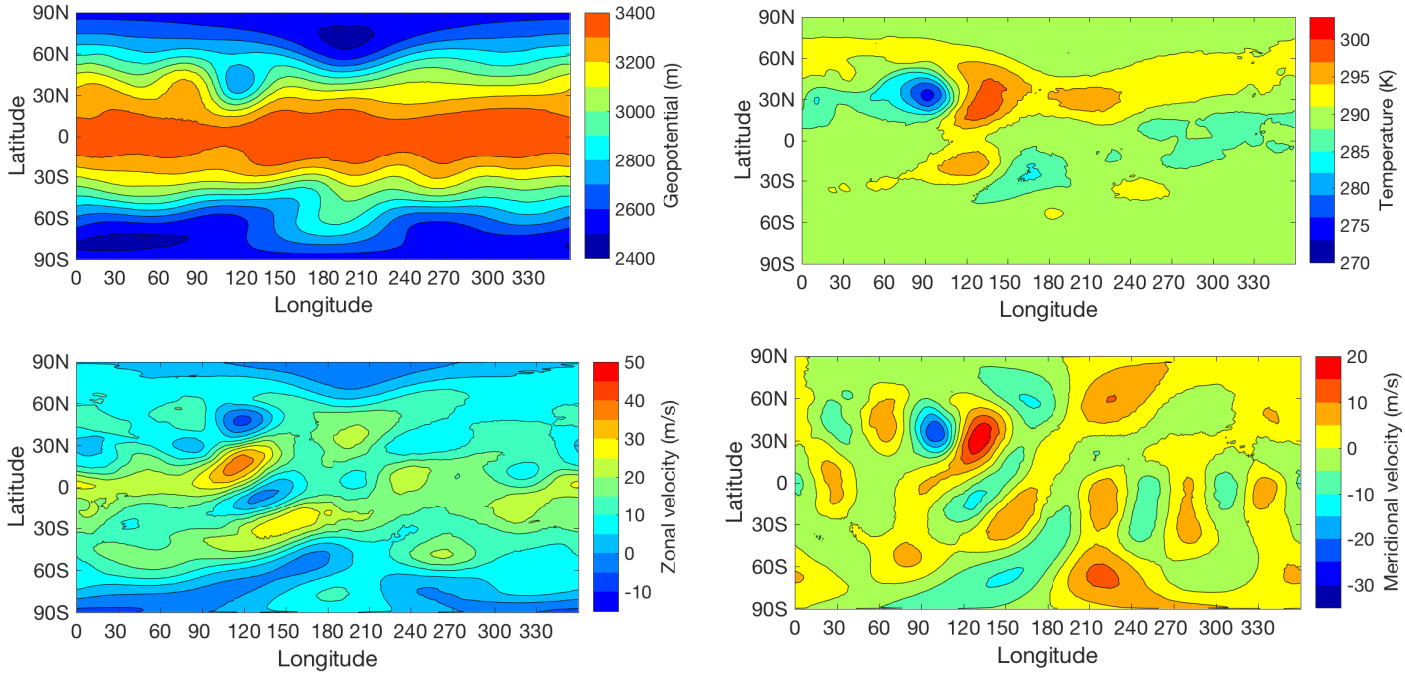

Figure 7. Latitude–longitude projections at 700 hPa of the adaptive simulation of the mountain induced Rossby wave test case at day 25.

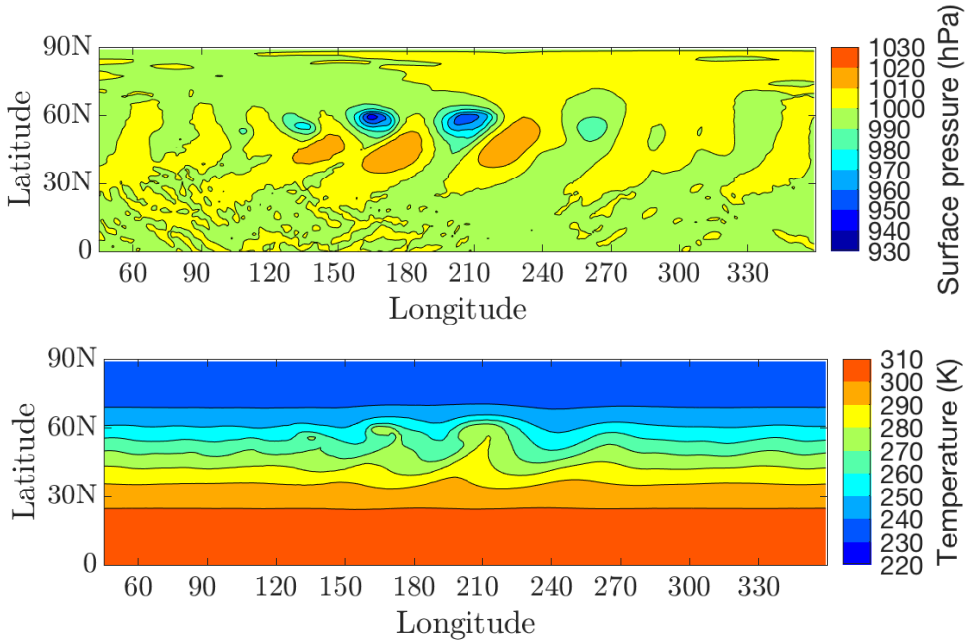

Figure 8. Latitude–longitude projections of surface pressure and temperature at vertical level 4 (about 867 hPa) of the adaptive simulation of the baroclinic instability test case at day 9.

steady state solution triggers a baroclinic instability that generates a series of vortices. These vortices grow and interact, eventually producing a two-dimensional turbulence-like vorticity field of intense filaments and vortex cores. The rapid development of the vortical instability and its subsequent evolution is a challenging case for an adaptive method. This baroclinic instability was Test Case 4 of DCMIP 2012.

With a sufficiently low relative tolerance (e.g. $\varepsilon = 0.03$ when adapting on the solution) `wavetrisk` successfully captures the explosive cyclogenesis around day 8 and subsequent breaking of the wave train around day 12 to produce a series of intense filamentary vortices (see 10). This calculation uses quadratic piecewise parabolic remapping at each time step Engwirda and Kelley (2016, https://github.com/dengwirda/PPR).

Figure 8 shows the surface pressure and temperature at day 9, which agree reasonably well with the equivalent results in figure 6 from Jablonowski and Williamson (2006). Note that we do not expect exact agreement since the adaptive simulation does not resolve all regions uniformly. Beyond day 12 the vortices interact to produce turbulence-like flow in both hemispheres. These results therefore validate the ability of the model to capture suddenly developing instabilities and track their evolution to a turbulence-like state.

Next we compare the characteristics of the solution-filtered and trend-filtered variants of `wavetrisk`. In both cases the maximum scale is $J = 7$ $(1/2^\circ)$ and we use 27 vertical hybrid σ pressure levels as specified in Jablonowski and Williamson (2006). There is no explicit diffusion added to stabilize the dynamics or to damp out grid scale oscillations and both simulations using RK45ssp time integration (Spiteri and Ruuth, 2002). It is important to note that, although this is a deterministic test

case when simulated non-adaptively, adaptivity necessarily neglects some less dynamically important structures and so the details flows with different tolerances eventually diverge once they become turbulent (after roughly 12 days). This is, however, an optimal test case for adaptive methods since at early times only a small portion of the flow is active which allows high compression ratios. In fact, until the instability develops a very coarse grid and large time step is sufficient.

Figure 9 compares the grid compression ratios for the baroclinic instability when adapting on the solution and on the trend. The tolerances were set to achieve similar compression ratios at day 9 and were set based on dimensional analysis (i.e. they are fixed in time). Both options use similar numbers of grid points until about day 12 when the flow becomes more turbulent. Once the flow is turbulent adapting on the trend uses significantly more grid points. Although the number of grid points used by both options is similar until day 12, the methods distribute the same number of grid points quite differently.

Figure 10 compares the vorticity fields and active grids at days 9 and 20. At day 9 adapting on the solution distributes the available grid points to track all the developing vortices. In contrast, adapting on the trend concentrates all grid points on the two strongest vortices. In addition, the peak vorticity when adapting on the solution, $5.2 \times 10^{-4}\ \mathrm{s}^{-1}$, is higher than when adapting on the trend, $1.9 \times 10^{-4}\ \mathrm{s}^{-1}$. Once the flow is turbulent, at day 20, adapting on the trend uses about 2.8 times more grid points than adapting on the solution.

Adapting on the solution appears to be more efficient since adapting on the trend is sensitive to grid scale noise in the trend (the solution is smoother than the trend). Adapting on the trend uses fewer overall grid points when the tolerances are rescaled dynamically (i.e. when the trend norms are re-computed at each time step), but the resulting adapted grid is quite sensitive to local fluctuations in the trend. This effect is much less pronounced when diffusion is added. Based on this and other examples we have investigated, adapting on the solution appears to be preferable when there is little or no diffusion added to damp out grid scale noise.

It is interesting to investigate how the grid compression ratios change as we increase the maximum allowed resolution. The main advantage of adaptive methods is for intermittent problems that require very high local resolution not attainable using static grid methods. This requires that the grid compression ratio increases significantly with the maximum allowed resolution. Figure 11 compares grid compression ratios for four different maximum resolutions ranging from $J = 7$ $(1/2^\circ)$ to $J = 10$ $(1/16^\circ)$ from day 8 to day 9 at the onset of the instability. The simulations have no diffusion and are adapted on the trends. Unsurprisingly, the compression is very high at the onset of the instability at day 8, over 1300 at $J = 10$. At day 9 once the initial vortices have developed the compression ranges from about 20 at $J = 7$ to about 200 at $J = 10$. Note that we have found that we can also use a higher relative error threshold at higher resolutions which further increases the compression.

This scaling test suggests that the adaptive method should be especially advantageous at high maximum resolutions for vorticity dominated flows. Although in this flow the instability is highly localized, Aechtner et al. (2015) found compression ratios of about 20 for a homogeneous isotropic shallow water turbulence case on the sphere with scales $J = 5, \ldots, 10$.

5.4 Held and Suarez general circulation experiment

Held and Suarez (1994) is a classic test case for dynamical cores of atmospheric general circulation models. It uses simplified "physics" (i.e. radiation and friction/drag models) that nevertheless produce realistic general circulation over relatively short

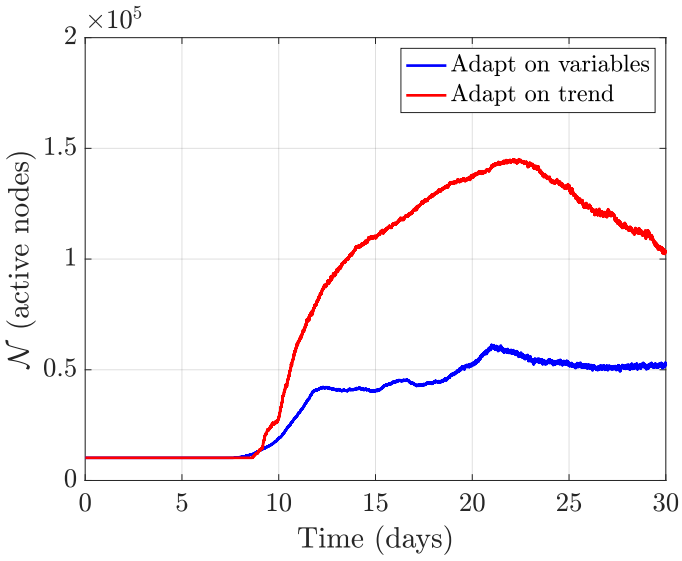

Figure 9. Comparison of the number of active nodes $\mathcal{N}$ for the baroclinic instability when adapting on the trend or adapting on the solution. The relative error threshold was set to $\varepsilon = 0.04$ when adapting on the solution and $\varepsilon = 2.6$ when adapting on the trend. In both cases the active scales are $j = 5, 6, 7$ and there is no additional diffusion. The tolerances were set to achieve similar compression ratios at 9 days. The total number of available grid points is 8.6×10^5, and so the minimum grid compression ratio is 3.6 when adapting on the solution and 1.5 when adapting on the trend. The average compression ratio is 5.2 when adapting on the solution and 2.1 when adapting on the trend.

time scales of $O(10^2)$ days. For example, Liu and Schneider (2010) have used simple Held–Suarez type physics to realistically model the general atmospheric circulation of Saturn. This model uses height-dependent Rayleigh damping to represent boundary-layer friction and a height- and latitude-dependent Newton cooling relaxation of potential temperature to a prescribed radiative-convective equilibrium. The temperature relaxation includes parameters accounting for cooling at the surface and top of the atmosphere as well as a tropopause. The Held–Suarez general circulation experiment adds a qualitatively new aspect to the Rossby wave and baroclinic instability tests we considered above: it includes physics source terms for the temperature and momentum equations. None of the previous simulations have included these sorts of physics terms, either in three dimensions or in two dimensions on the sphere or on the plane. Nevertheless, we expect the grid adaptation to track the effect of the source terms either directly through their effect on the trend, or indirectly through their effect on the prognostic variables.

We compare low and high resolution runs since previous work has suggested that the choice of coarsest can affect the result. We also want to explore how grid compression changes with resolution (as in the baroclinic test case). The low resolution run has coarsest scale $J_{\text{min}} = 4$ and finest resolution $J = 6$ (about 120 km or 1°) with tolerance $\varepsilon = 0.04$. The high resolution run has coarsest scale $J_{\text{min}} = 6$ and finest resolution $J = 8$ (about 30 km or $1/4^\circ$) with tolerance $\varepsilon = 0.02$. The low resolution case is run on 40 cores and the high resolution case is run on 320 cores. As in Wan et al. (2013), both cases start from the Jablonowski and Williamson (2006) zonally symmetric initial condition with random noise of magnitude 1 $\mathrm{m/s}$ added to the zonal wind. In both cases the simulations are first run non-adaptively for 200 days at at the coarsest resolution and then

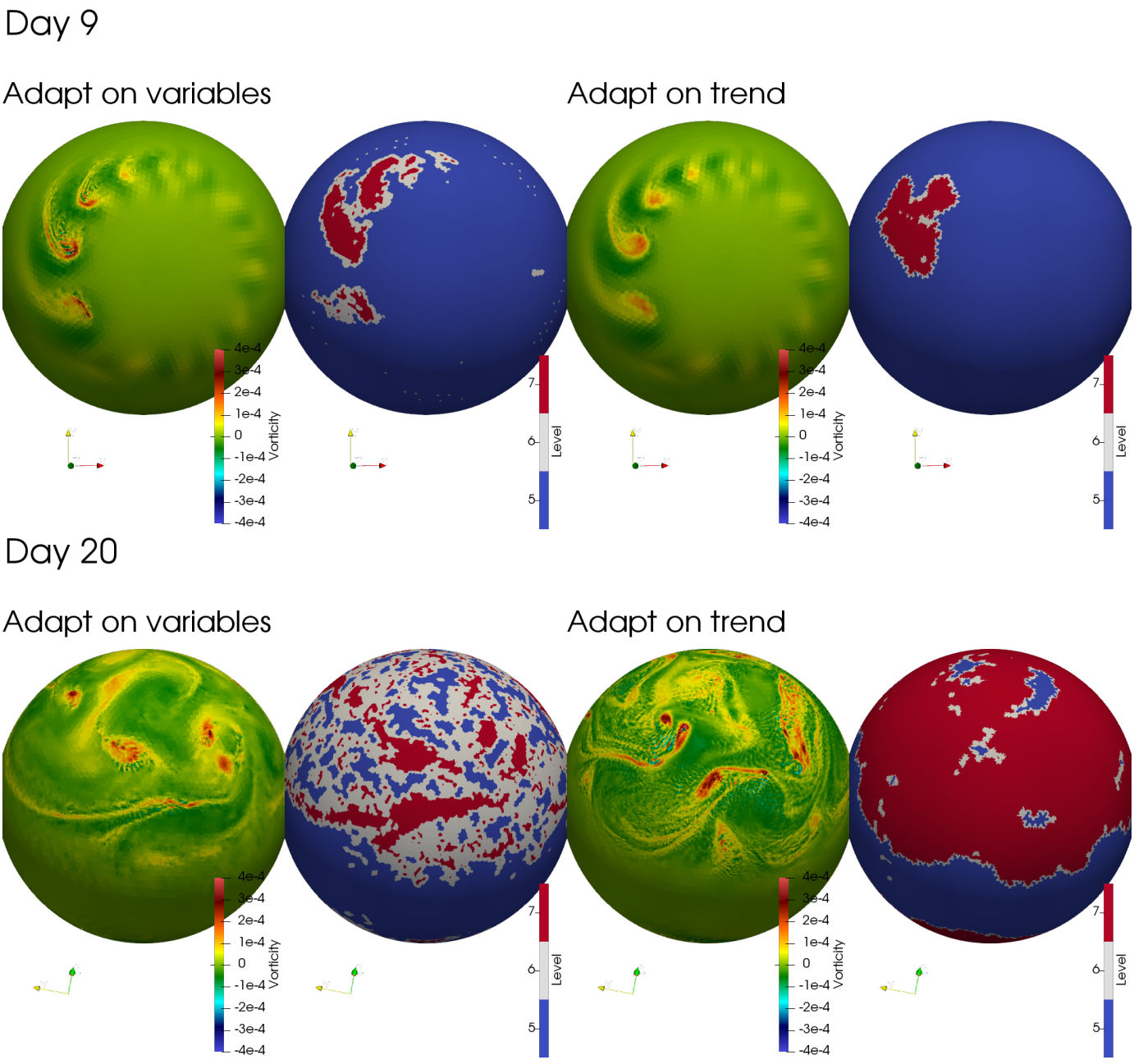

Figure 10. Comparison at days 9 and 20 of the grid compression ratios for the baroclinic instability when adapting on the trend or adapting on the solution. The relative error threshold was set to $\varepsilon = 0.04$ (fixed) when adapting on the solution and $\varepsilon = 2.6$ when adapting on the trend. In both cases the resolution is $j = 5, 6, 7$ and there is no additional diffusion. The tolerances were set to achieve similar compression ratios at 9 days. The vorticity field is shown at hybrid vertical level 4 (about 870 hPa).

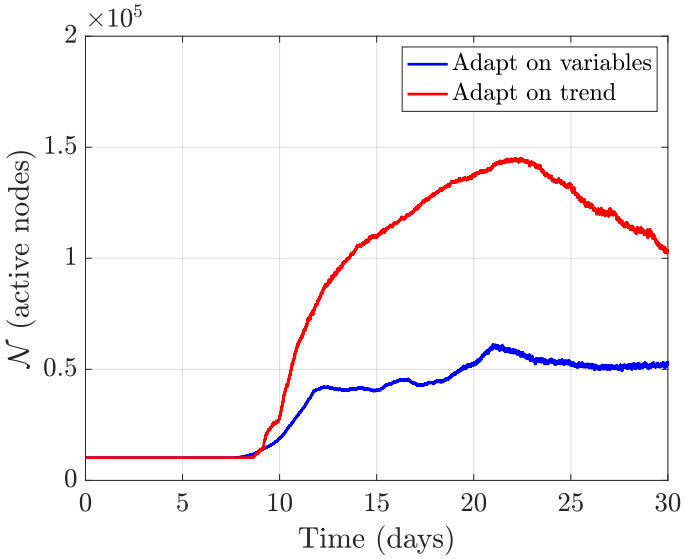

Figure 11. Grid compression ratios at the beginning of the baroclinic instability for maximum resolutions $J = 7$ $(1/2^\circ)$, $J = 8$ $(1/4^\circ)$, $J = 9$ $(1/8^\circ)$ and $J = 10$ $(1/16^\circ)$. In all cases the coarsest resolution is $J = 5$ (2°). The grids are adapted on the trend and there is no diffusion. Note that the $J = 10$ uses a higher relative error threshold $\varepsilon = 4$ than the others, which use $\varepsilon = 2$. Compression ratios increase significantly with increasing resolution, reaching as high as 200 times at day 9 for the maximum resolution case. Even at $J = 7$ the code achieves a compression ratio of about 20 at day 9.

restarted adaptively with the maximum resolution. The time step is adaptive with CFL criterion one. The maximum possible grid compression ratio for both cases is 21.

We use 19 vertical σ pressure levels concentrated at the top and bottom of the atmosphere. The vertical grid is remapped using a piecewise parabolic method with WENO limiting every ten time steps.

Small scale noise is damped with $p = 2$ hyperdiffusion with diffusion constant $K_\phi = 3.48\times10^{15}\,\mathrm{m}^4/\mathrm{s}$, $K_\delta = 3.48\times10^{16}\,\mathrm{m}^4/\mathrm{s}$, $K_\omega = 2.17\times10^{14}\,\mathrm{m}^4/\mathrm{s}$ for the low resolution run and $K_\phi = 2.33\times10^{14}\,\mathrm{m}^4/\mathrm{s}$, $K_\delta = 5.98\times10^{14}\,\mathrm{m}^4/\mathrm{s}$, $K_\omega = 1.46\times10^{13}\,\mathrm{m}^4/\mathrm{s}$. The diffusion is applied each time step in the main trend routine. The source terms for the potential temperature and the velocity representing cooling and Rayleigh damping are implemented as a separate Euler step.

Mean and variance statistics are calculated using a parallel version of Welford's inline algorithm (Chan et al., 1983) by first interpolating the solution to the finest grid from checkpoints saved every 24 hours. The second-order statistics are essentially converged after 200 days (the first order statistics converge much more quickly).

A typical low resolution result is shown in figure 12 (top). The average grid compression ratio at this low resolution is only 1.9 ± 0.1 with $\varepsilon = 0.04$. The adaptive algorithm is able to track the development and evolution of the fine scale filamentary vortex structures over long times. Note that since the adapted grid is the union of adapted grids over all vertical levels, the adapted grid does not necessarily correspond exactly to the structures at the vertical level 11 (about 250 hPa) at the level of the upper atmosphere jets shown in the figure.

Figure 13 shows standard first and second order statistics averaged zonally and over time for the low resolution run. These statistics are qualitatively similar to those of Dubos et al. (2015), Wan et al. (2013) and Lin (2004). (Lin (2004) is the only one that uses Lagrangian vertical coordinates.) The main quantitative difference is in the slightly lower magnitude of the eddy kinetic energy. This difference is due partly to the use of Lagrangian vertical coordinates, where the remapping introduces additional dissipation, and partly to the additional dissipation generated by the adaptivity at these relatively low resolutions. Lin (2004) did not show the eddy kinetic energy, but he reported a maximum variance of zonal wind of about 300 $\mathrm{m}^2/\mathrm{s}^2$, slightly larger than the 284 $\mathrm{m}^2/\mathrm{s}^2$ we find here (not shown), and similar to the 301 $\mathrm{m}^2/\mathrm{s}^2$ we find for the high resolution case discussed below. Note that the original Held and Suarez (1994) paper did not present results for eddy kinetic energy.

The choice of remapping has a significant influence on the eddy kinetic energy. For example, we found that changing from a piecewise linear remapping to a piecewise parabolic remapping increases the maximum eddy kinetic energy by 53 $\mathrm{m}^2/\mathrm{s}^2$ for a non-adaptive simulation with resolution $J = 5$ (240 km). Similarly, Lin (2004) found that including a monotonicity constraint in the remapping lowered the maximum variance of the zonal velocity by about 20 $\mathrm{m}^2/\mathrm{s}^2$. We would also like to emphasize that piecewise constant remapping gives qualitatively incorrect results (e.g. zonal jets are too high), presumably because it is too dissipative, and is not a good choice for this test case.

Larger tolerances ε effectively increase dissipation, which decreases maximum eddy kinetic energy, while choosing a smaller tolerance leads to essentially no compression in the low resolution run. (The other statistics are much less sensitive to ε.) Note that a fixed (non-adaptive) resolution $J_{\min} = 4$ (4°) gives a maximum eddy kinetic energy of only 280 $\mathrm{m}^2/\mathrm{s}^2$, much less than the published values of over 400 $\mathrm{m}^2/\mathrm{s}^2$ at higher resolutions of 2° and 1°. This suggests that the coarsest grid may be the main factor behind the lower eddy kinetic energy we observe here at higher compression ratios.

In contrast, figure 12 (bottom) shows typical results from the high resolution simulation at $\varepsilon = 0.02$. The grid compression is clearly much higher than in the low resolution run, and the fine scales are limited to a small neighbourhood of the high intensity vorticity filaments. The average grid compression ratio for the high resolution simulation is 7.6 ± 0.5, exactly four times as large as in the low resolution case. Both the compression and the fluctuations in the compression are much higher than in the low resolution run. These results suggest that the adaptive method is useful primarily at higher resolutions and for more turbulent flows. This confirms our earlier observations for two-dimensional shallow water turbulence on the sphere (see figures 17 and 18 of Aechtner et al., 2015). The advantages of the adaptive method should be even greater for maximum resolutions higher than $J = 8$ ($1/4^\circ$).

Figure 14 shows standard first and second order statistics averaged zonally and over time for the high resolution run. The results are qualitatively very similar to the low resolution run, with the main differences being more intense eddy momentum and eddy kinetic energy. The negative high altitude zonal jet is also a bit stronger.

The results presented here suggest that the eddy kinetic energy is quite sensitive to the coarsest resolution and relative error tolerance chosen and that, not surprisingly, the adaptive method does not provide significant benefits for low resolution simulations. However, dynamic adaptivity makes much higher resolutions runs feasible at a significantly reduced computational cost (both in terms of memory and cpu time). This advantage should increase for larger maximum resolutions unattainable using

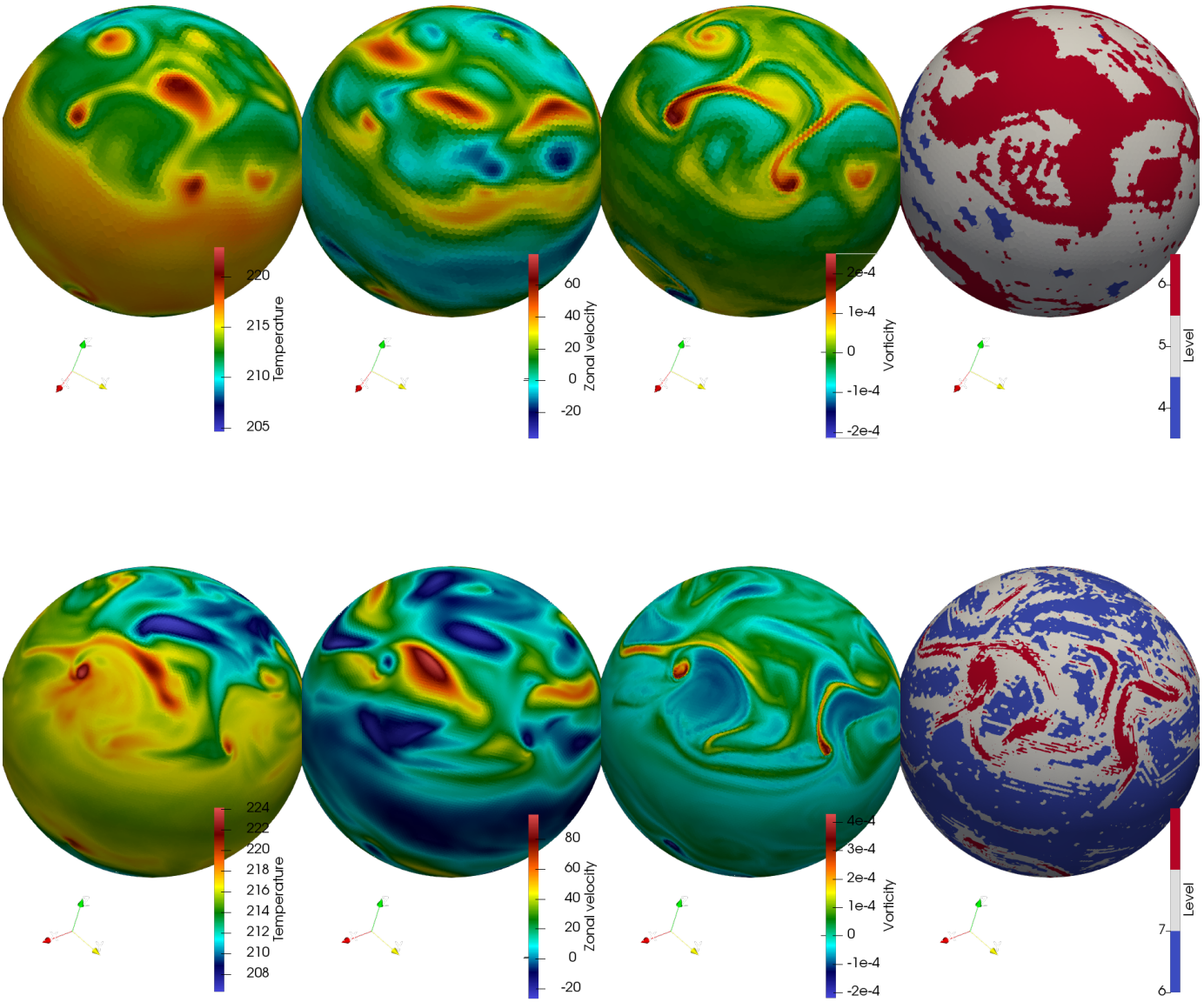

Figure 12. Typical results for the low resolution (top) and high resolution (bottom) Held and Suarez general circulation test case at 250 hPa. The grid is adapted on the solution with relative error tolerance $\varepsilon = 0.04$ in the low resolution case and $\varepsilon = 0.02$ in the high resolution case. The grid compression ratio is 2.0 for the low resolution case and 7.5 for the high resolution case.

non-adaptive codes. The ability to restart a simulation adaptively at a higher resolution (e.g. add three levels of refinement) allows events of interest to be explored at very high local resolutions, even if it is only for relatively short times.

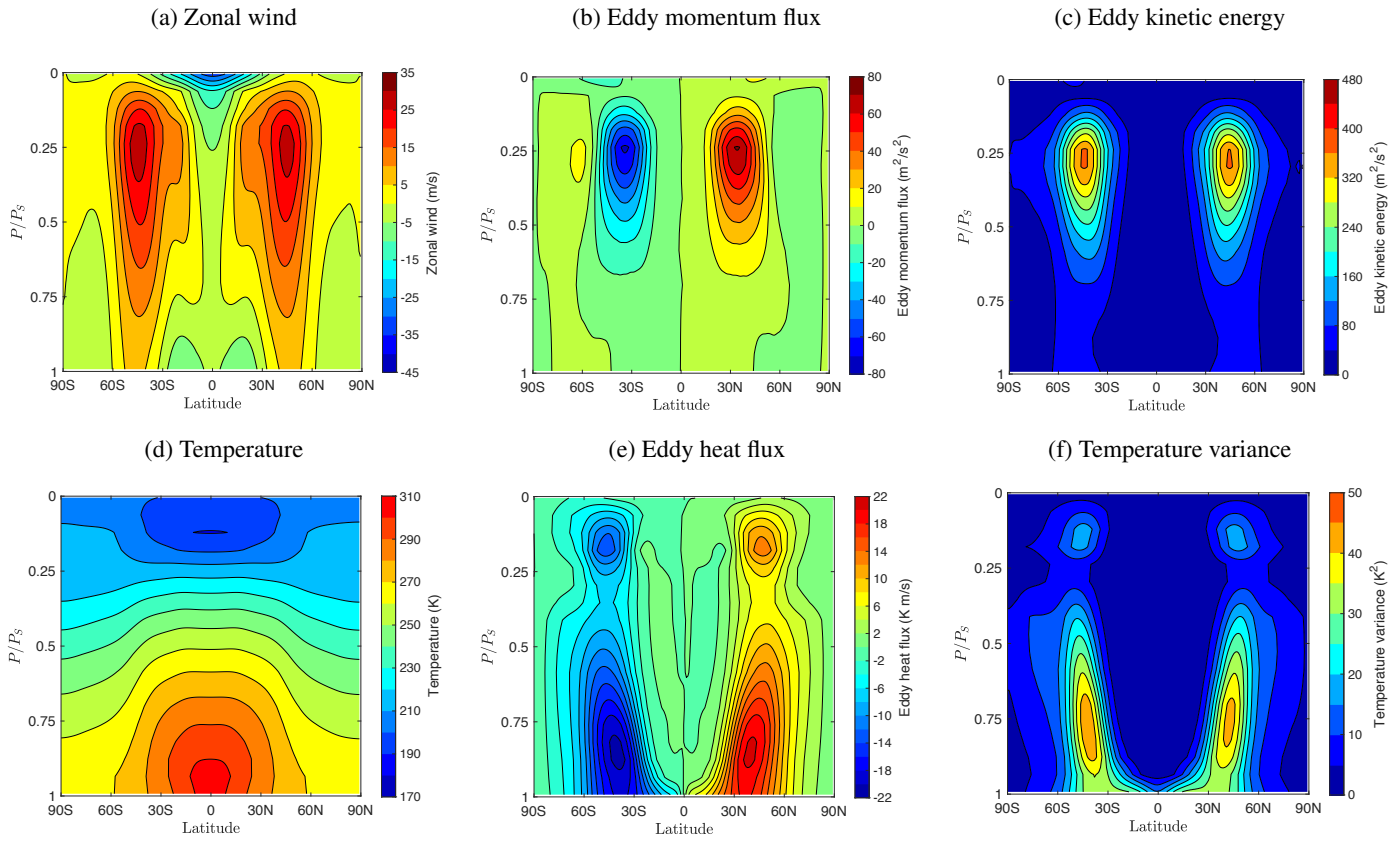

Figure 13. Time–zonal statistics for the low resolution Held and Suarez (1994) test case with scales $J = 4, 5, 6$ (maximum resolution 1°). The grid is adapted on the solution error with tolerance $\varepsilon = 0.04$. Statistics are averaged over 400 days after day 200 by interpolating saved data to the finest grid.

6 Conclusions

This paper introduces `wavetrisk`: a new adaptive dynamical core. `wavetrisk` uses the hydrostatic `dynamico` discretization of Dubos et al. (2015) with a Lagrangian vertical coordinate. Second-generation discrete wavelet transforms provide control of the relative error of the solution in each vertical layer at each time step. The adaptive grid is uniform vertically and is composed of columns of varying horizontal sizes. In addition to the horizontal adaptivity, the vertical coordinates are remapped onto a hybrid σ-pressure coordinate using an arbitrary Lagrangian Eulerian (ALE) scheme. In principle, ALE allows r-adaptivity of the vertical coordinates by optimizing the target grid at each remap. h-adaptivity may also be possible in the future by deactivating some vertical layers (the so-called “dormant layers" used in ocean modelling).s

The code is parallelized via domain decomposition using `mpi` and the data is stored in a hybrid quad tree–patch data structure. The computational load is re-balanced at each checkpoint save. We demonstrate excellent strong parallel scaling up to at least 2560 cores in the perfectly balanced case and close to linear scaling (exponent 0.78) up to at least 640 cores in

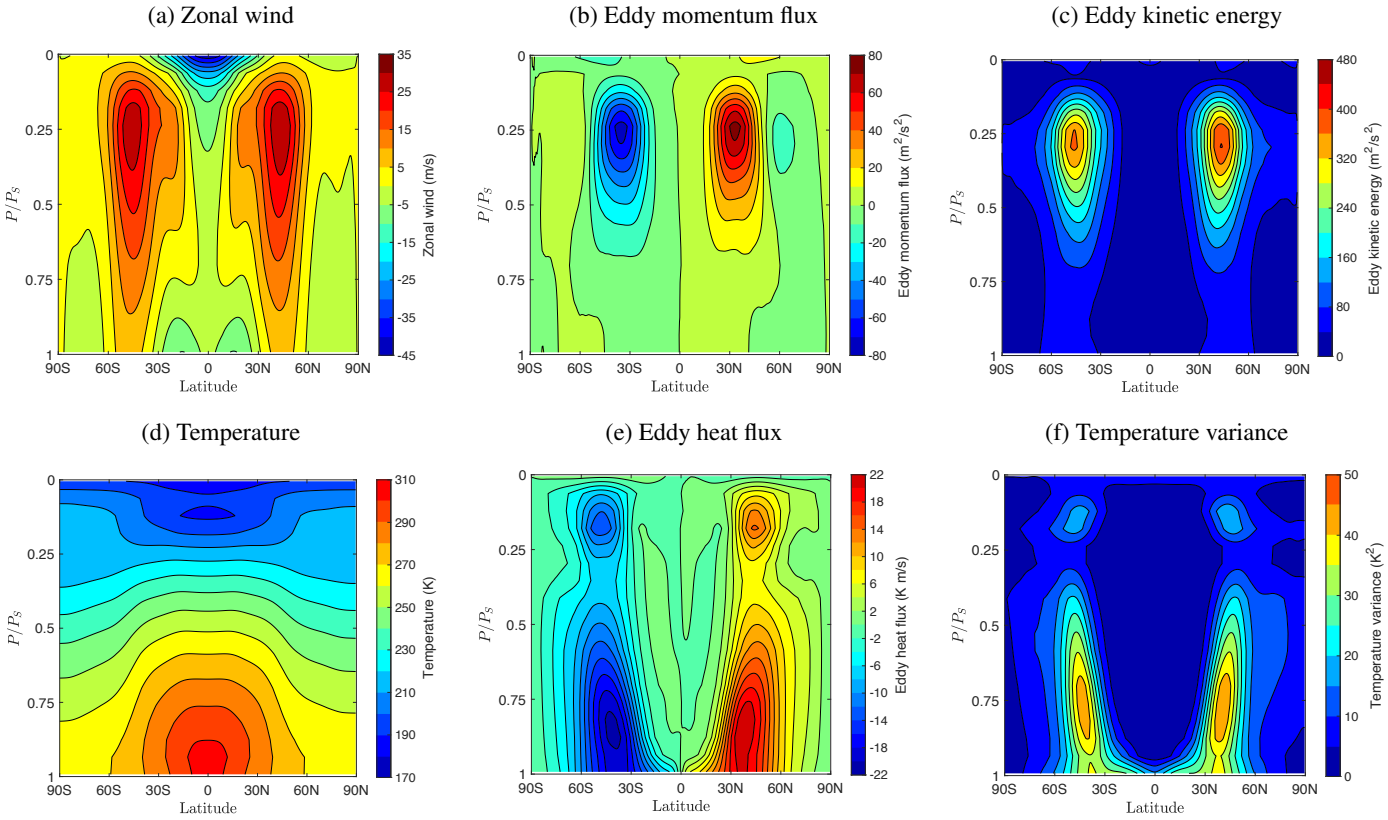

Figure 14. Time–zonal statistics for the high resolution Held and Suarez (1994) test case with scales $J = 6, 7, 8$ (maximum resolution $1/4^\circ$). The grid is adapted on the solution error with tolerance $\varepsilon = 0.02$. Statistics are averaged over 200 days after day 700 by interpolating saved data to the finest grid.

an unbalanced case. `wavetrisk` is three to four times slower per active grid point than the non-adaptive `dynamico` code, which suggests that we require a grid compression ratio of more than four for adaptivity to provide an advantage in cpu time.

The grid may be adapted based on the wavelet coefficients (interpolation errors) of either the prognostic variables themselves (mass density, mass-weighted potential temperature and velocity) or of their associated tendencies. The desired relative error tolerance is multiplied by the appropriate norm (of the variables or tendencies) computed at each time step to control the relative error. Note that the adaptivity is necessarily dissipative compared to an equivalent simulation with uniform resolution on the finest grid since some energy is lost when computational elements are coarsened. However, by construction, this dissipation is controlled and may be reduced by decreasing the tolerance or increasing the maximum allowed resolution. In previous work (Aechtner et al., 2015) we explored how the effective Reynolds number of a turbulent flow depends on the choice of error tolerance parameter.

We have validated the code on three standard benchmarks: a mountain induced Rossby wave train, baroclinic instability of a jet stream and the Held and Suarez general circulation experiment. These tests show that `wavetrisk` correctly captures the

dynamics, including rapidly developing instabilities, with only a small portion of the total grid points available on a similar non-adaptive gird. The grid compression ratio can reach over 200 in ideal cases (e.g. the start of the baroclinic instability with five levels of refinement) and is advantageous at sufficiently high resolutions even in more homogeneous flows like Held and Suarez.

Because adaptive climate simulation is a new field, we have deliberately included many options in our adaptive algorithm. `wavetrisk` can adapt on errors in the solution or on errors in the trend, it can run with no diffusion, Laplacian diffusion, or hyperdiffusion. The vertical grid can be re-gridded (using a large selection of remapping schemes) each time step, or only when a level becomes too narrow. And, of course, we can choose different relative tolerances ε and maximum and minimum grid resolutions. In many cases, the code is stable without diffusion and with grids adapted either on the solution or the trend. However, our test cases suggest that adapting on the solution (rather than the trend) generally gives more accurate and faster solutions for a given number of grid points. Including a small amount of diffusion stabilizes the code and reduces the number of active grid points by reducing grid-level noise.

It is clear that the main application of `wavetrisk` is for simulations at maximum resolutions unattainable by non-adaptive dynamical cores. In future work we will use `wavetrisk` to simulate simple Held–Suarez type climates at much higher resolutions and for longer times and investigate the behaviour of more sophisticated physics parameterizations in adaptive simulations. We are also developing an ocean variant of `wavetrisk` that will improve the ALE formulation of the vertical coordinate and use penalization for bathymetry and coastlines. This work builds on the shallow water ocean model we presented in Kevlahan et al. (2015).

Code availability. WAVETRISK-1.0 is published under the Creative Commons 4.0 license at https://doi.org/10.5281/zenodo.3459710. The current latest version of the code is WAVETRISK-1.1, which includes some bug fixes and two incompressible cases.

Author contributions. Both NKRK and TD have contributed to the research and paper preparation.

Competing interests. The authors have no competing interests.

Acknowledgements. NKRK would like to thank NSERC for Discovery Grant funding and Compute Canada for computing time and the Université Grenoble–Alpes and CNRS for visiting professorships. This work was supported by the French national programme LEFE/INSU. The authors would like to thank Matthias Aechtner for his major contributions to the computational foundations of `wavetrisk` which are outlined in the papers on the shallow water equations version of the code on the sphere.

References

Aechtner, M., Kevlahan, N.-R., and Dubos, T.: A conservative adaptive wavelet method for the shallow water equations on the sphere, Q.J.R. Meteorol. Soc., 141, 1712–1726, https://doi.org/DOI:10.1002/qj.2473, published online 5 December 2014, 2015.

Behrens, J.: Adaptive Atmospheric Modelling, Springer, 2009.

Bryan, G. L., Norman, M. L., O'Shea, B. W., Abel, T., Wise, J. H., Turk, M. J., Reynolds, D. R., Collins, D. C., Wang, P., Skillman, S. W., Smith, B., Harkness, R. P., Bordner, J., Kim, J.-h., Kuhlen, M., Xu, H., Goldbaum, N., Hummels, C., Kritsuk, A. G., Tasker, E., Skory, S., Simpson, C. M., Hahn, O., Oishi, J. S., So, G. C., Zhao, F., Cen, R., Li, Y., and Enzo Collaboration: ENZO: An Adaptive Mesh Refinement Code for Astrophysics, The Astrophysical Journal Supplement Series, 211, 19, https://doi.org/10.1088/0067-0049/211/2/19, 2014.

Chan, T., Golub, G., and LeVeque, R.: Algorithms for computing the sample variance: Analysis and recommendations, The American Statistician, 37, 242–247, 1983.

Domingues, M. O., Gomes, S. M., Roussel, O., and Schneider, K.: An adaptive multiresolution scheme with local time stepping for evolutionary PDEs, J. Comput. Phys., 227, 3758–3780, https://doi.org/10.1016/j.jcp.2007.11.046, 2008.

Dubos, T. and Kevlahan, N.-R.: A conservative adaptive wavelet method for the shallow water equations on staggered grids, Q.J.R. Meteorol. Soc., 139, 1997–2020, 2013.

Dubos, T., Dubey, S., Tort, M., Mittal, R., Meurdesoif, Y., and Hourdin, F.: DYNAMICO-1.0, an icosahedral hydrostatic dynamical core designed for consistency and versatility, Geosci. Model Dev., 8, 3131–3150, https://doi.org/10.5194/gmd-8-3131-2015, 2015.

Engwirda, D. and Kelley, M.: A WENO-type slope-limiter for a family of piecewise polynomial methods, 2016.

Ferguson, J.: Bridging Scales in 2- and 3-Dimensional Atmospheric Modeling with Adaptive Mesh Refinement, Ph.D. thesis, University of Michigan, 2018.

Ferguson, J., Jablonowski, C.and Johansen, H., McCorquodale, P., Colella, P., and Ullrich, P.: Analyzing the Adaptive Mesh Refinement (AMR) Characteristics of a High-Order 2D Cubed-Sphere Shallow-Water Model, Mon. Weather Rev., 144, 4641–4666, https://doi.org/10.1175/MWR-D-16-0197.1, 2016.

Harrison, Edward J., J.: Three-Dimensional Numerical Simulations of Tropical Systems Utilizing Nested Finite Grids., Journal of Atmospheric Sciences, 30, 1528–1543, https://doi.org/10.1175/1520-0469(1973)030<1528:TDNSOT>2.0.CO;2, 1973.

Heikes, R. P., Randall, D. A., and Konor, C. S.: Optimized Icosahedral Grids: Performance of Finite-Difference Operators and Multigrid Solver, Mon. Weather Rev., 2013.

Hejazialhosseini, B., Rossinelli, D., Bergdorf, M., and Koumoutsakos, P.: High order finite volume methods on wavelet-adapted grids with local time-stepping on multicore architectures for the simulation of shock-bubble interactions, J. Comput. Phys., 229, 8364–8383, https://doi.org/10.1016/j.jcp.2010.07.021, 2010.

Held, I. and Suarez, M.: A proposal for the intercomparison of the dynamical cores of atmospheric general-circulation models, Bull. Amer. Meteor. Soc., 75, 1825–1830, https://doi.org/10.1175/1520-0477(1994)075<1825:APFTIO>2.0.CO;2, 1994.

Jablonowski, C. and Williamson, D.: A baroclinic instability test case for atmospheric model dynamical cores, Q. J. R. Meteorol. Soc., 132, 2943–2975, 2006.

Jablonowski, C., Lauritzen, P. H., Nair, R., and Taylor, M.: Idealized test cases for the dynamical cores of Atmospheric General Circulation Models: A proposal for the NCAR ASP 2008 summer colloquium, Tech. rep., 2008.

Jablonowski, C., Oehmke, R. C., and Stout, Q. F.: Block-structured adaptive meshes and reduced grids for atmospheric general circulation models, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367, 4497–4522, 2009.

Jones, R. W.: Vortex Motion in a Tropical Cyclone Model., Journal of Atmospheric Sciences, 34, 1518–1527, https://doi.org/10.1175/1520-0469(1977)034<1518:VMIATC>2.0.CO;2, 1977.

Kavcic, I. and Thuburn, J.: A Lagrangian vertical coordinate version of the ENDGame dynamical core. Part II: Evaluation of Lagrangian conservation properties, Q.J.R. Meteorol. Soc., https://doi.org/doi.org/10.1002/qj.3375, 2018.

Kevlahan, N. and Vasilyev, O.: An adaptive wavelet collocation method for fluid–structure interaction at high Reynolds numbers, SIAM J. Sci. Comput., 26, 1894–1915, 2005.

Kevlahan, N. K. R., Dubos, T., and Aechtner, M.: Adaptive wavelet simulation of global ocean dynamics using a new Brinkman volume penalization, Geoscientific Model Development, 8, 3891–3909, https://doi.org/10.5194/gmd-8-3891-2015, 2015.

Kopera, M. and Giraldo, F.: Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations, J. Comput. Phys, 275, 92–117, https://doi.org/10.1016/j.jcp.2014.06.026, 2014.

Liandrat, J. and Tchamitchian, P.: Resolution of the 1D Regularized Burgers Equation Using a Spatial Wavelet Approximation, Tech. rep., NASA Contractor Report 187480, ICASE Report 90-83, NASA Langley Research Center, Hampton VA 23665-5225, 1990.

Lin, S.-J.: A ??Vertically Lagrangian?? Finite-Volume Dynamical Core for Global Models, Monthly Weather Review, 132, 2293–2307, 2004.

Liu, J. and Schneider, T.: Mechanisms of Jet Formation on the Giant Planets, J. Atmos. Sci., 67, 3652–3672, https://doi.org/10.1175/2010JAS3492.1, 2010.

McCorquodale, P., Ullrich, P., Johansen, H., and Colella, P.: A adaptive multiblock high-order finite-volume method for solving the shallow-water equations on the sphere, Comm. App. Math. Comp. Sci., 10, 121–162, https://doi.org/10.2140/camcos.2015.10.121, 2015.

Mehra, M. and Kevlahan, N. K.-R.: An adaptive wavelet collocation method for the solution of partial differential equations on the sphere, J. Comput. Phys., 227, 5610–5632, https://doi.org/10.1016/j.jcp.2008.02.004, 2008.

Mignone, A., Zanni, C., Tzeferacos, P., van Straalen, B., Colella, P., and Bodo, G.: The PLUTO Code for Adaptive Mesh Computations in Astrophysical Fluid Dynamics, The Astrophysical Journal Supplement Series, 198, 7, https://doi.org/10.1088/0067-0049/198/1/7, 2012.

Naddei, F., de la Llave Plata, M., Couaillier, V., and Coquel, F.: A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous Galerkin methods, Journal of Computational Physics, 376, 508–533, 2019.

Petersen, M., Jacobsen, D., Ringler, T., Hecht, M., and Maltrud, M.: Evaluation of the arbitrary Lagrangian?Eulerian vertical coordinate method in the MPAS-Ocean model, Ocean Modelling, 86, 93–113, 2015.

Popinet, S. and Rickard, G.: A tree-based solver for adaptive ocean modelling, Ocean Model., 16, 224–249, https://doi.org/10.1016/j.ocemod.2006.10.002, 2007.

Popinet, S., Rickard, G., and Delaux, S.: Quadtree-adaptive global atmospheric modelling on parallel systems, weather and Climate Prediction on Next Generation Supercomputers, Exeter, UK, 22- 25 October, 2012. https://www.newton.ac.uk/files/seminar/20121024100510409-153402.pdf, 2012.

Ringler, T. D., Thuburn, J., Klemp, J. B., and Skamarock, W. C.: A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured C-grids, J. Comput. Phys., 229, 3065–3090, https://doi.org/10.1016/j.jcp.2009.12.007, 2010.

Roussel, O. and Schneider, K.: Coherent Vortex Simulation of weakly compressible turbulent mixing layers using adaptive multiresolution methods, J. Comput. Phys., 229, 2267–2286, https://doi.org/10.1016/j.jcp.2009.11.034, 2010.

Schneider, K. and Vasilyev, O.: Wavelet Methods in Computational Fluid Dynamics, Annual Review of Fluid Mechanics, 42, 473–503, https://doi.org/10.1146/annurev-fluid-121108-145637, 2010.

Shchepetkin, A.: A finite volume grid remapping procedure with small inherent Diffusion, institute of Geophysics and Planetary Physics, University of California at Los Angeles, 2001.

Skamarock, W. and Klemp, J.: Adaptive Grid Refinement for Two-Dimensional and Three-Dimensional Nonhydrostatic Atmospheric Flow, Monthly Weather Review, 121, 788–804, 1993.

Skamarock, W., Oliger, J., and Street, R. L.: Adaptive grid refinement for numerical weather prediction, Journal of Computational Physics, 80, 27–60, 1989.

Spiteri, R. and Ruuth, S.: A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM J. Numer. Anal., 40, 469–491, 2002.

Sweldens, W.: The lifting scheme: A construction of second generation wavelets, SIAM J. Math. Anal., 29, 511–546, 1998.

Tomita, H. and Sato, M.: A new dynamical framework of nonhydrostatic global model using the icosahedral grid, Fluid Dyn. Res., 34, 357–400, 2004.

Wan, H., Giorgetta, M. A., Zang, G., Restelli, M., Majewski, D., Bonaventura, L., Fröhlich, K., Reinert, D., Ripodas, P., Kornblueh, L., and Förstner, J.: The ICON-1.2 hydrostatic atmospheric dynamical core on triangular grids ? Part 1: Formulation and performance of the baseline version, Geosci. Model Dev., 6, 735–763, 2013.

Xu, G.: Discrete Laplace–Beltrami operator on sphere and optimal spherical triangulations, Internat. J. Comput. Geom. Appl., 16, 75–93, 2006.

Yakhot, V. and Sreenivasan, K.: Anomalous scaling of structure functions and dynamic constraints on turbulence simulations, J. Stat. Phys., 121, 823–841, 2005.