INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 0000; **00**:1–27
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld

Efficient Multigrid Preconditioners for Atmospheric Flow Simulations at High Aspect Ratio

Andreas Dedner[1], Eike Müller[2*] and Robert Scheichl[2]

[1]*Mathematics Institute, Zeeman Building, University of Warwick, Coventry CV4 7AL*
[2]*Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, United Kingdom*

SUMMARY

Many problems in fluid modelling require the efficient solution of highly anisotropic elliptic partial differential equations (PDEs) in "flat" domains. For example, in numerical weather- and climate-prediction an elliptic PDE for the pressure correction has to be solved at every time step in a thin spherical shell representing the global atmosphere. This elliptic solve can be one of the computationally most demanding components in semi-implicit semi-Lagrangian time stepping methods which are very popular as they allow for larger model time steps and better overall performance. With increasing model resolution, algorithmically efficient and scalable algorithms are essential to run the code under tight operational time constraints. We discuss the theory and practical application of bespoke geometric multigrid preconditioners for equations of this type. The algorithms deal with the strong anisotropy in the vertical direction by using the tensor-product approach originally analysed by Börm and Hiptmair [Numer. Algorithms, 26/3 (2001), pp. 219-234]. We extend the analysis to three dimensions under slightly weakened assumptions, and numerically demonstrate its efficiency for the solution of the elliptic PDE for the global pressure correction in atmospheric forecast models. For this we compare the performance of different multigrid preconditioners on a tensor-product grid with a semi-structured and quasi-uniform horizontal mesh and a one dimensional vertical grid. The code is implemented in the Distributed and Unified Numerics Environment (DUNE), which provides an easy-to-use and scalable environment for algorithms operating on tensor-product grids. Parallel scalability of our solvers on up to 20,480 cores is demonstrated on the HECToR supercomputer.

Received . . .

KEY WORDS: high aspect ratio flow; multigrid; elliptic PDEs; atmospheric modelling; parallelisation; convergence analysis

1. INTRODUCTION

Highly efficient solvers for elliptic partial differential equations (PDEs) are required in many areas of fluid modelling, such as numerical weather- and climate- prediction (NWP), subsurface flow simulations [1] and global ocean models [2, 3]. Often these equations need to be solved in "flat" domains with high aspect ratio, representing a subsurface aquifer or the Earth's atmosphere. In both cases the horizontal extent of the area of interest is much larger than the vertical size. For example, the Euler equations, which describe the large scale atmospheric flow, need to be integrated efficiently in the dynamical core of NWP codes like the Met Office Unified Model [4, 5]. Many forecast centres

*Correspondence to: Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, United Kingdom. E-mail: e.mueller@bath.ac.uk

Contract/grant sponsor: Natural Environment Research Council (NERC); contract/grant number: NE/J005576/1, NE/K006754/1

such as the Met Office and European Centre for Medium-Range Weather Forecasts (ECMWF) use semi-implicit semi-Lagrangian (SISL) time stepping [6,7] to advance the atmospheric fields forward in time because it allows for larger model time steps and thus better computational efficiency. However, this method requires the solution of an anisotropic elliptic PDE for the pressure correction in a thin spherical shell at every time step. In this paper we solve a discrete form of this PDE based on a first order accurate cell-centred finite volume scheme. As the elliptic solve can account for a significant fraction of the total model runtime, it is important to use algorithmically efficient and parallel scalable algorithms.

Suitably preconditioned Krylov-subspace and multigrid methods (see e.g. [8,9]) have been shown to be highly efficient for the solution of elliptic PDEs encountered in numerical weather- and climate prediction (see [4, 10–21] and the comprehensive review in [22]). Multigrid methods are algorithmically optimal, i.e. the number of iterations required to solve a PDE to the accuracy of the discretisation error is independent of the grid resolution. However - as far as we are aware - multigrid algorithms are currently not widely implemented operationally in atmospheric models and one of the aims of this paper is to demonstrate that they can be used very successfully in fluid simulations at high aspect ratio. Whereas "black-box" algebraic multigrid (AMG) [23, 24] solvers such as the ones implemented in the DUNE-ISTL [25] and Hypre [26, 27] libraries can be applied under very general circumstances on unstructured grids and automatically adapt to potential anisotropies, they suffer from additional setup costs and lead to larger matrix stencils on the coarse levels. On (semi-) structured grids which are typical in many atmospheric and oceanographic applications, geometric multigrid algorithms usually give much better performance, as they can be adapted to the structure of the problem by the developer. In contrast to AMG algorithms which explicitly store the matrix on each level, it is possible to use a matrix-free approach: instead of reading the matrix from memory, it is reconstructed on-the-fly from a small number of "profiles". This leads to a more regular memory access pattern and significantly reduces the storage costs, in particular if these profiles can be factorised into a horizontal and vertical component. As the code is memory bandwidth limited this also has a direct impact on the performance of the solver. Robust geometric multigrid methods, such as the one in Hypre [28, 29], adapt the smoother or coarse grid transfer operators to deal with very general anisotropies in the problem (see also [30–35]). However, this robustness comes at a price and these methods are often computationally expensive and difficult to parallelise.

In the problems we consider, the tensor-product structure of the underlying mesh and the grid-aligned anisotropy make it possible to use the much simpler but highly efficient tensor-product multigrid approach described for example in [36, 37]: line-relaxation in the strongly coupled direction is combined with semi-coarsening in the other directions only. The implementation is straightforward: in addition to an obvious modification of the intergrid operators, every smoother application requires the solution of a tridiagonal system of size n_r in each vertical column with n_r grid cells. The tridiagonal solve requires $\mathcal{O}(n_r)$ operations and hence the total cost per iteration is still proportional to the total number of unknowns. The method is also inherently parallel as in atmospheric applications domain decomposition is typically in the horizontal direction only.

In [38] this method was analysed theoretically for equations with a strong vertical anisotropy on a two dimensional tensor-product grid. The authors show that optimal convergence of the tensor-product multigrid algorithm in two dimensions follows from the optimal convergence of the standard multigrid algorithm for a set of one-dimensional elliptic problems in the horizontal direction. While the original work in [38] applies in two dimensions, it has been extended to three dimensions in [39] and the algorithm has recently been applied successfully to three dimensional problems in atmospheric modelling in [20, 21]. Although the proof in [38] relies on the coefficients in the PDE to factorise exactly into horizontal-only and vertical-only contributions, we stress that this property is not required anywhere in the implementation of the multigrid algorithm. In practice we expect the algorithm to work well also for approximately factorising coefficients and under suitable assumptions we are able to also prove this rigorously. To demonstrate this numerically, we carry out experiments for the elliptic PDE arising from semi-implicit semi-Lagrangian time stepping in the dynamical core of atmospheric models such as the Met Office Unified Model [4, 5], where the coefficients only factorise approximately but the multigrid convergence is largely

DOI: 10.1002/fld

unaffected. Alternatively, we also investigate approximate factorisations of the atmospheric profiles and then apply the tensor product multigrid algorithm to the resulting, perturbed pressure equation to precondition iterative methods for the original system, such as a simple Richardson iteration or BiCGStab [40]. As the operator is usually "well behaved" in this direction (i.e. it is smooth and does not have large variations on small length scales), the multigrid algorithm will converge in a very small number of iterations.

An additional advantage of applying the multigrid method only to the perturbed problem with factorised profiles is the significant reduction in storage requirements for the matrix. As the algorithm is memory bound and the cost of a matrix application or a tridiagonal solve depends on the efficiency with which the matrix can be read from memory this leads to performance gains in the preconditioner: we find that the time per iteration can be reduced by around 20%, but this has to be balanced with a possibly worse convergence rate. Nevertheless, our numerical experiments show, that in some cases the factorised preconditioner can be faster overall. On novel manycore computer architectures, such as GPUs, where around 30-40 floating point operations can be carried out per global memory access, we expect the performance gains from this matrix-free tensor-product implementation to be even more dramatic. If the matrix is stored in tensor product format and the local stencil is calculated on-the-fly, the costs for the matrix construction can essentially be neglected compared to the cost of reading fields from memory.

For example, consider the sparse matrix-vector product $\boldsymbol{v} = A\boldsymbol{u}$ and assume that only the n_{nz} non-zero entries of the $n \times n$ matrix A are stored. In particular for a finite volume discretisation on a three dimensional grid there will be on average $n_s = n_{nz}/n = 7$ nonzero entries per matrix row. If we assume perfect caching of the input vector $\boldsymbol{u}$, then a matrix-explicit implementation requires n reads to load the vector $\boldsymbol{u}$, n writes to store the resulting vector $\boldsymbol{v}$ and n_{nz} reads to load the non-zero matrix entries, and hence $2n + n_{nz}$ memory transactions in total. This is reduced to $2n$ for a matrix-free implementation where A does not have to be read from memory, resulting in a speedup of $\frac{2n+n_{nz}}{2n} = 1 + \frac{n_s}{2}$, which is 4.5 for the finite-volume discretisation. If we assume the other extreme, i.e. no caching of the input vector $\boldsymbol{u}$, then loading this vector requires n_{nz} memory reads instead of n. In this case the gain from the matrix-free implementation is $\frac{n+2n_{nz}}{n+n_{nz}} = \frac{1+2n_s}{1+n_s}$, i.e. 1.875 in the finite volume case. The benefits of this matrix-free implementation on GPUs have recently been shown in a similar context in [41, 42].

In state-of-the-art global weather prediction models the horizontal resolution is of the order of tens of kilometres with the aim of reducing this to around one kilometre in the next decade (the number of vertical grid cells is typically around 100). The resulting problems with $10^9 - 10^{11}$ degrees of freedom can only be solved on operational timescales if their scalability can be guaranteed on massively parallel computers. In addition to the sequential algorithmic performance we demonstrate the parallel scalability of our solvers on HECToR, the UK's national supercomputer which is hosted and managed by the Edinburgh Parallel Computing Centre (EPCC). We find that our solvers show very good weak scaling on up to 20,480 AMD Opteron cores and can solve a linear system with 11 billion unknowns in less than 5 seconds (reducing the residual by five orders of magnitude).

The work presented here builds on our earlier papers, in particular [22], where we investigated the performance and parallel scalability of different solvers for a characteristic elliptic model problem which reproduces the main characteristics of the atmospheric pressure correction equation in a simplified geometry (one panel of a cubed sphere grid). We modelled the strong vertical anisotropy and correct dependence of the coefficients on the resolution and timestep-size. We compared different solvers, including AMG solvers from the DUNE- and hypre- libraries to matrix free single-level and geometric multigrid algorithms based on [38]. We found that multigrid methods are faster than Krylov subspace iterations with single level preconditioners, such as vertical line relaxation. Since it avoids explicit storage of the matrix, the matrix-free implementation of the tensor-product geometric multigrid algorithm turned out to be significantly faster than the AMG codes and showed very good weak- and strong scalability on up to 65,536 CPU cores. In [42] this was extended to a multi-GPU implementation of the tensor-product multigrid solver and the performance and scalability was demonstrated by solving equations with 0.5×10^{12} unknowns on up to 16,384 GPUs of the TITAN supercomputer. We quantified the absolute performance of the algorithm and find

DOI: 10.1002/fld

that we can achieve a "useful bandwidth" of $25\% - 50\%$ of the peak (guaranteed-not-to-exceed) value. The measured floating point performance was just below one PetaFLOP. In a socket-to-socket comparison, the K40 Kepler GPUs on Titan were about $4\times$ faster than the AMD Opteron CPUs on HECToR.

Here we extend the work in [22, 42] by studying the full pressure correction equation with representative background profiles which arise from the linearisation of the Navier-Stokes equations around a reference state. The same equation is used in the Met Office ENDGame Dynamical core [5]. The problem is discretised in a thin spherical shell describing the global atmosphere; this is realised by an extruded icosahedral mesh.

Due to the increased complexity of the problem we based our code on an advanced C++ framework for grid-based calculations. To achieve good performance while keeping the implementation modular and maintainable, the algorithms used in this paper are implemented in the Distributed and Unified Numerics Environment (DUNE) [43, 44]. DUNE is an object oriented C++ framework and provides easy to use interfaces to common parallel grid implementations such as ALUGrid [45–47] and UG [48]. Due to the modular structure of the library and because we can rely on the underlying parallel grid implementations, the implementation of our solvers on tensor-product grids is straightforward. Throughout the code performance is guaranteed by using generic metaprogramming based on C++ templates. Most importantly this approach avoids overheads from the very frequent calls to short methods. They can be inlined at compile time since all necessary information is provided via templates, while at the same time allowing the user to write highly modular code. The efficiency of this approach was demonstrated in [44], where the overhead of the object oriented DUNE interface layer was quantified, and the efficiency of the DUNE library is also confirmed by our performance results in section 5.3 below.

Structure This paper is organised as follows. In Section 2 we describe the pressure correction equation arising in semi-implicit semi-Lagrangian time stepping in atmospheric models and discuss the discretisation of the resulting linear PDE with particular emphasis on the tensor-product structure of the grid. The theory of the tensor-product multigrid algorithm is reviewed in Section 3 where we extend the analysis in [38] to three dimensions following [39]. In this section we also prove the convergence of the preconditioned Richardson iteration for non-factorising profiles. The grid structure and the discretisation of the equation as well as the implementation of our algorithms in the DUNE framework are described in Section 4. Numerical results for different test cases are presented together with parallel scaling tests in Section 5. We conclude and present ideas for future work in Section 6. Some more technical aspects can be found in the appendix. In particular, the intergrid operators of the multigrid algorithm are described in detail in Appendix A.

2. ELLIPTIC PDE FOR PRESSURE CORRECTION IN ATMOSPHERIC MODELS

The elliptic PDE which arises in semi-Lagrangian semi-implicit time stepping in atmospheric forecast models is derived for example in [5] for the ENDGame dynamical core of the Unified Model. The work in this paper is based on a cell-centred finite volume discretisation. We assume that the diffusion coefficient is known everywhere in space and can be evaluated at the cell faces. In the following, we outline the main steps in the construction of the corresponding linear algebraic problem.

The Euler equations describe large scale atmospheric flow as a set of coupled non-linear differential equations for the velocity $\boldsymbol{v}$, (Exner-) pressure π, potential temperature θ and density

DOI: 10.1002/fld

ρ.

$$\begin{aligned}
\frac{D\boldsymbol{v}}{Dt} &= -c_p\theta\nabla\pi + \boldsymbol{R}_v \qquad \text{(Momentum equation)} \\
\frac{D\theta}{Dt} &= R_\theta \qquad \text{(Thermodynamic equation)} \\
\frac{D\rho}{Dt} &= -\rho\nabla\cdot v \qquad \text{(Mass conservation)} \\
\rho\theta &= \Gamma\pi^\gamma \qquad \text{(Equation of state)}
\end{aligned} \tag{1}$$

The R-terms describe external- and sub-gridscale- forcings such as gravity and unresolved convection. The constants Γ and γ are defined as

$$\Gamma \equiv p_0/R_d, \qquad \gamma \equiv \frac{1-\kappa}{\kappa}, \qquad \kappa \equiv R_d/c_p, \tag{2}$$

where p_0 is a reference pressure; c_p and R_d are the specific heat capacity and specific gas constant of dry air. System (1) can be written schematically for the state vector $\Phi = \{\boldsymbol{v}, \pi, \theta, \rho\}$ as

$$\frac{D\Phi}{Dt}(\boldsymbol{x}, t) = \mathcal{N}[\Phi(\boldsymbol{x}, t)]. \tag{3}$$

Advection is described in the semi-Lagrangian [7] framework, i.e. material time derivatives $D\Phi/Dt$ are replaced by

$$\frac{D\Phi}{Dt}(\boldsymbol{x}, t) \mapsto \frac{\Phi^{(t+\Delta t)}(\boldsymbol{x}) - \Phi^{(t)}(\boldsymbol{x}_D)}{\Delta t} \tag{4}$$

where $\boldsymbol{x}_D$ is the departure point of a parcel of air at time t which is advected to position $\boldsymbol{x}$ at time $t + \Delta t$. The right-hand-side of (3) is treated semi-implicitly [6]. Because of the small vertical grid spacing and the resulting large Courant number of vertical sound waves, vertical advection needs to be treated fully implicitly, but some of the other terms are evaluated at the previous time step and thus treated explicitly; we write $\mathcal{N} = \mathcal{N}^{(\text{impl.})} + \mathcal{N}^{(\text{expl.})}$. We use the θ-method with off-centering parameter μ for implicit time stepping and replace

$$\begin{aligned}
\mathcal{N}[\Phi(\boldsymbol{x}, t)] &= \mathcal{N}^{(\text{impl.})}[\Phi(\boldsymbol{x}, t)] + \mathcal{N}^{(\text{expl.})}[\Phi(\boldsymbol{x}, t)] \\
&\mapsto \mu\mathcal{N}^{(\text{impl.})}[\Phi^{(t+\Delta t)}(\boldsymbol{x})] + (1-\mu)\mathcal{N}^{(\text{impl.})}[\Phi^{(t)}(\boldsymbol{x})] + \mathcal{N}^{(\text{expl.})}[\Phi^{(t)}(\boldsymbol{x})]
\end{aligned} \tag{5}$$

and in the following we always assume that $\mu = \frac{1}{2}$ which corresponds to the scheme described in [49]. By eliminating the potential temperature, density and all velocities from the resulting equation, one (non-linear) equation for the pressure $\pi^{(t+\Delta t)}$ at the next time step can be obtained[†]. To solve this equation via (inexact) Newton iteration, all fields are linearised around a suitable reference state (which can for example be the atmospheric fields at the previous time step) denoted by subscript "ref". To this end the pressure at the next time step is written as $\pi^{(t+\Delta t)}(\boldsymbol{x}) \equiv \pi(\boldsymbol{x}) = \pi_{\text{ref}}(\boldsymbol{x}) + \pi'(\boldsymbol{x})$ with analogous expressions for $\theta^{(t+\Delta t)}$ and $\rho^{(t+\Delta t)}$; the reference velocities $\boldsymbol{v}_{\text{ref}}$ are assumed to be zero. It should, however, be kept in mind that the linearisation does not need to be "exact" as the non-linear equation can be solved with an inexact Newton iteration. In particular, some terms can be moved to the right hand side which is equivalent to treating them explicitly or lagging them in the non-linear iteration. Naturally, there will be a tradeoff between faster convergence of the Newton iteration and the cost of the inversion of the linear operator; for example, in [5] all couplings to non-direct neighbours, which can be large in the case of steep orography, are moved to the RHS to reduce the size of the stencil of the linear operator. While these considerations are relevant for the optimisation of the non-linear solve in a particular model, in this article we focus on the solution of the linear equation, which is the computationally most expensive component of the Newton iteration.

[†]Mathematically this is equivalent to forming the pressure Schur complement of the equation.

DOI: 10.1002/fld

Once the Exner pressure $\pi^{(t+\Delta t)}$ has been calculated, the evaluation of the remaining atmospheric fields at the next time step is straightforward and does not require any additional (non-)linear solves. In contrast to explicit time stepping methods the Courant number can be chosen significantly larger than 1, which makes semi-implicit semi-Lagrangian time stepping very popular in operational models. However, because of the short advective time scale and to ensure that large scale flow is described correctly, the Courant number is usually limited to around 10, i.e. the implicit time step size is no more than one order of magnitude larger than what would be allowed in an explicit method. To evaluate the overall performance of the method, the benefits of a larger time step would have to be balanced against the additional cost for the elliptic solve.

2.1. Linear equation

For ease of notation we simply write $\pi \equiv \pi^{(t+\Delta t)}$ in the following and drop the time indices. Then the non linear equation for π is of the form

$$\mathcal{N}(\pi) = \mathcal{R}. \tag{6}$$

To solve this equation iteratively we expand all fields around a reference state (which can, for example, be given by the fields at the previous time step) to obtain a linear operator $\mathcal{L}$. As discussed above, in practise some terms might be lagged in the non-linear iteration, i.e. moved to the right hand side of the linear equation. At each step k of the nonlinear iteration we write $\pi_k = \pi_{\text{ref}} + \pi'_k$ for the approximate solution to (6) and update the pressure as follows:

$$\begin{aligned}&\text{Solve} \quad \mathcal{L}\pi'_k = \tilde{\mathcal{R}}_{k-1} := (\mathcal{R} - \mathcal{L}\pi_{\text{ref}}) - \delta\mathcal{N}(\pi_{k-1}) \quad \text{for } \pi'_k \quad \text{with} \quad \delta\mathcal{N} \equiv \mathcal{N} - \mathcal{L},\\ &\text{Update} \quad \pi_k = \pi_{\text{ref}} + \pi'_k.\end{aligned}$$

Every iteration requires the solution of a linear equation $\mathcal{L}\pi'_k = \tilde{\mathcal{R}}_{k-1}$ for the pressure correction π'_k, which we denote as π' in the following. To construct the linear operator $\mathcal{L}$ we proceed as follows: starting from (1) the semi-Lagrangian framework in (4) is used for horizontal advection and vertical advection is treated implicitly (to ensure that mass is exactly conserved, advection is treated implicitly in all three spatial dimensions in the mass equation). The right hand sides are expanded according to (5). We linearise around reference profiles θ_{ref}, π_{ref} and ρ_{ref} which fulfil the equation of state $\rho_{\text{ref}}\theta_{\text{ref}} = (\pi_{\text{ref}})^\gamma$, i.e. write $\theta = \theta_{\text{ref}} + \theta'$ etc. and assume that the velocity expansion is around zero $\boldsymbol{v}_{\text{ref}} = 0$. If we split up the velocity into a tangential- and vertical- component $\boldsymbol{v} = (\boldsymbol{v}_{\mathcal{S}}, w)$ the time-discretised Euler equations in (1) finally become in a spherical geometry

$$\boldsymbol{v}_{\mathcal{S}} = \boldsymbol{R}'_{u_{\mathcal{S}}} - \mu\Delta t c_p \frac{1}{r}\left(\theta_{\text{ref}}\boldsymbol{\nabla}_{\mathcal{S}}\pi' + (\boldsymbol{\nabla}_{\mathcal{S}}\pi_{\text{ref}})\theta'\right), \tag{7}$$

$$w = R'_w - \mu\Delta t c_p \left(\theta_{\text{ref}}\partial_r\pi' + (\partial_r\pi_{\text{ref}})\theta'\right), \tag{8}$$

$$\theta' = R'_\theta - \mu\Delta t(\partial_r\theta_{\text{ref}})w, \tag{9}$$

$$\rho' = R'_\rho - \mu\Delta t\left(\frac{1}{r^2}\partial_r\left(r^2\rho_{\text{ref}}w\right) + \frac{1}{r}\left(\boldsymbol{\nabla}_{\mathcal{S}}\cdot(\rho_{\text{ref}}\boldsymbol{v}_{\mathcal{S}})\right)\right), \tag{10}$$

$$\pi' = \frac{\pi_{\text{ref}}}{\gamma}\left(\frac{\rho'}{\rho_{\text{ref}}} + \frac{\theta'}{\theta_{\text{ref}}}\right), \tag{11}$$

where $\partial_r = \langle\hat{\boldsymbol{n}}, \boldsymbol{\nabla}\rangle$ is the normal component of the derivative and $\boldsymbol{\nabla}_{\mathcal{S}} = r\left(\boldsymbol{\nabla} - \langle\hat{\boldsymbol{n}}, \boldsymbol{\nabla}\rangle\right)$ is the component tangential to a unit sphere $\mathcal{S}$ with outer normal $\hat{\boldsymbol{n}}$. Any terms that depend on the current time step are absorbed in the R'-terms. We then rewrite (11) as a function of ρ' and insert it together with (7) into (10) to obtain an equation with w, ρ' and π' only

$$\begin{aligned}-\frac{\theta'}{\theta_{\text{ref}}} + \gamma\frac{\pi'}{\pi_{\text{ref}}} = R''_\rho &- \mu\Delta t\frac{\partial_r(r^2\rho_{\text{ref}}w)}{r^2\rho_{\text{ref}}}\\ &+ (\mu\Delta t)^2 c_p\frac{\boldsymbol{\nabla}_{\mathcal{S}}\cdot(\rho_{\text{ref}}\theta_{\text{ref}}(\boldsymbol{\nabla}_{\mathcal{S}}\pi')) + \boldsymbol{\nabla}_{\mathcal{S}}\cdot(\rho_{\text{ref}}(\boldsymbol{\nabla}_{\mathcal{S}}\pi_{\text{ref}})\theta')}{r^2\rho_{\text{ref}}}.\end{aligned} \tag{12}$$

Prepared using fldauth.cls
DOI: 10.1002/fld

By solving (8) and (9) for w and θ' we obtain

$$w = \Lambda_{\rm ref}\left(f_2 - \mu\Delta t c_p\theta_{\rm ref}(\partial_r\pi')\right), \qquad \theta' = \Lambda_{\rm ref}\left(f_3 + (\mu\Delta t)^2 c_p\theta_{\rm ref}(\partial_r\theta_{\rm ref})(\partial_r\pi')\right) \tag{13}$$

where $\Lambda_{\rm ref} = \left(1 + (\mu\Delta t)^2\left(N_{\rm ref}\right)^2\right)^{-1}$ arises from the implicit treatment of vertical advection and the (squared) vertical buoyancy (or Brunt-Väisälä-) frequency is given by

$$\left(N_{\rm ref}\right)^2 = -c_p(\partial_r\pi_{\rm ref})(\partial_r\theta_{\rm ref}) = g\frac{\partial_r\theta_{\rm ref}}{\theta_{\rm ref}}. \tag{14}$$

The functions f_2 and f_3 only depend on the fields at the current time step. We rescale the vertical coordinate r by the radius of the earth $R_{\rm earth}$ and the potential temperature by a reference temperature T_0 at ground level to make it dimensionless. Finally, we multiply equation (12) by $\rho_{\rm ref}$ and denote the typical horizontal velocity by

$$c_h \equiv \sqrt{\gamma}c_s \qquad \text{where} \quad c_s = \sqrt{c_pT_0/\gamma}$$

is the speed of sound in a parcel of air with temperature T_0. Furthermore we introduce the dimensionless quantity

$$\omega = \frac{c_h\mu\Delta t}{R_{\rm earth}}. \tag{15}$$

After eliminating w and θ from (12) with the help of (13) we obtain a second order equation for the pressure correction π':

$$\begin{aligned} &-\omega^2\left\{\Lambda_{\rm ref}\rho_{\rm ref}(\partial_r\theta_{\rm ref})(\partial_r\pi') + \frac{1}{r^2}\left[\partial_r\left(r^2\Lambda_{\rm ref}\rho_{\rm ref}\theta_{\rm ref}(\partial_r\pi')\right) + \boldsymbol{\nabla}_{\mathcal{S}}\cdot\left(\rho_{\rm ref}\theta_{\rm ref}(\boldsymbol{\nabla}_{\mathcal{S}}\pi')\right)\right]\right\} \\ &-\omega^4\frac{1}{r^2}\boldsymbol{\nabla}_{\mathcal{S}}\cdot\left(\Lambda_{\rm ref}\rho_{\rm ref}(\boldsymbol{\nabla}_{\mathcal{S}}\pi_{\rm ref})(\partial_r\theta_{\rm ref})(\partial_r\pi')\right) + \gamma\frac{\rho_{\rm ref}}{\pi_{\rm ref}}\pi' = RHS \end{aligned} \tag{16}$$

The $\mathcal{O}(\omega^4)$ term arises due to the last term in (7). In [5] this term is not included in the linear operator since all terms which stem from reference profiles that do not depend exclusively on the vertical coordinate are neglected. To be consistent with this approach, the $\mathcal{O}(\omega^4)$ term is assumed to be moved to the right hand side of the linear equation in the following. The first two terms in the curly brackets are the sum of a vertical advection and a vertical diffusion term.

In contrast, in [5], the linear pressure correction equation is derived from the discretised Euler equations. However, it can be shown that (16) is identical to the continuum limit of equation (67) in [5] if the latter is written down explicitly in spherical coordinates. Denoting the unknown pressure correction π' by u, as is common in the mathematical literature, the elliptic operator can be written as

$$\begin{aligned} \mathcal{L}u &= -\omega^2\boldsymbol{\nabla}\cdot(\underline{\alpha}\boldsymbol{\nabla}u) - \omega^2\boldsymbol{\xi}\cdot\boldsymbol{\nabla}u + \beta u \\ &= -\omega^2\begin{pmatrix}\partial_r, & \frac{1}{r}\boldsymbol{\nabla}_{\mathcal{S}}\end{pmatrix}^T\begin{pmatrix}\alpha_r & 0\\ 0 & \alpha_{\mathcal{S}}\,\mathrm{Id}_{2\times2}\end{pmatrix}\begin{pmatrix}\partial_r\\ \frac{1}{r}\boldsymbol{\nabla}_{\mathcal{S}}\end{pmatrix}u - \omega^2\begin{pmatrix}\xi_r, & 0\end{pmatrix}^T\begin{pmatrix}\partial_r\\ \frac{1}{r}\boldsymbol{\nabla}_{\mathcal{S}}\end{pmatrix}u + \beta u \end{aligned} \tag{17}$$

where $\mathrm{Id}_{2\times2}$ is the 2×2 identity matrix. The equation is solved in a thin spherical shell, $\Omega = \mathcal{S}\times[1,1+H]$ and $H = D/R_{\rm earth} \ll 1$ is the ratio of the thickness of the atmosphere and the radius of the earth. The solution $u = u(r,\hat{\boldsymbol{r}})$ depends on the coordinates $r\in[1,1+H]$ and $\hat{\boldsymbol{r}}\in\mathcal{S}$. In contrast to global latitude-longitude grids, on quasi-uniform grids the ratio between the smallest and largest grid spacing is bounded. To ensure that the horizontal acoustic Courant number $\approx \omega/h$ (where h is the smallest grid spacing) remains unchanged as the horizontal resolution is increased, the time step size Δt has to decrease linearly with h. A simple scaling argument shows that the vertical advection term is much smaller than the diffusion term at high resolution.

The functions $\alpha_r(r,\hat{\boldsymbol{r}})$, $\alpha_{\mathcal{S}}(r,\hat{\boldsymbol{r}})$, $\xi_r(r,\hat{\boldsymbol{r}})$ and $\beta(r,\hat{\boldsymbol{r}})$ are referred to as "profiles" in the following and can be obtained from the background fields $\pi_{\rm ref}$, $\theta_{\rm ref}$ and $\rho_{\rm ref}$ by comparing the elliptic operators in (16) and (17):

$$\alpha_r = r^2\Lambda_{\rm ref}\rho_{\rm ref}\theta_{\rm ref}\left(= r^2\Lambda_{\rm ref}\alpha_{\mathcal{S}}\right), \quad \alpha_{\mathcal{S}} = \rho_{\rm ref}\theta_{\rm ref}, \quad \xi_r = \Lambda_{\rm ref}\rho_{\rm ref}(\partial_r\theta_{\rm ref}), \quad \beta = \gamma\frac{\rho_{\rm ref}}{\pi_{\rm ref}}. \tag{18}$$

DOI: 10.1002/fld

2.2. Iterative solvers

After discretisation, the Helmholtz equation in (17) can be written as a large algebraic system of the form

$$A\mathbf{u} = \mathbf{f}. \tag{19}$$

where the finite-dimensional field vector $\mathbf{u}$ represents the pressure correction in the entire atmosphere. If we assume that the horizontal resolution is around 1 kilometre and $\mathcal{O}(100)$ vertical grid levels are used, each atmospheric variable has $\mathcal{O}(10^{10})$ degrees of freedom. Problems of this size can only be solved with highly efficient iterative solvers and on massively parallel computers. Current forecast models, such as the Met Office Unified Model, use suitable preconditioned Krylov subspace methods (see e.g. [8] for an overview) such as BiCGStab [40, 50]. Due to the flatness of the domain the equation is highly anisotropic: typical grid spacings in the horizontal direction are at the order of tens of kilometres, whereas the distance between vertical levels can be as small as a few metres close to the ground. While this anisotropy is partially compensated by the ratio $\alpha_{\mathcal{S}}/\alpha_r = r^{-2}\Lambda_{\mathrm{ref}}^{-1}$ in (18), it remains large in particular for small time steps Δt for which $\Lambda_{\mathrm{ref}} \to 1$ (recall that we chose units such that $r \approx 1$).

As discussed in the literature [4, 16, 17, 19], a highly efficient preconditioner for Krylov methods in this case is vertical line relaxation. This amounts to a block Jacobi or block SOR iteration where the degrees of freedom in one vertical column are relaxed simultaneously by solving a tridiagonal equation. However, (geometric) multigrid algorithms have also been considered by the atmospheric modelling community [10–15, 18, 20, 21, 51] and recently some of the authors have demonstrated their superior behaviour for a simplified model equation [22, 42].

For the problems considered here we find that while the multigrid-algorithm is robust as the Courant number increases, this is not the case for a Krylov-subspace method with a single-level preconditioner, which even breaks down for large Courant numbers. More detailed numerical results of this comparison are given in Section 5.5.

3. TENSOR-PRODUCT MULTIGRID FOR ANISOTROPIC PROBLEMS

Efficient algorithms for the solution of anisotropic equations have been studied extensively in the multigrid literature. For general anisotropies in convection dominated problems, robust schemes have been designed by adapting the smoother (see e.g. [30, 31]) or the coarsening strategy and the restriction/prolongation operators (see e.g. [29, 35]). For example, in [32–34] alternating approximate plane- and line- smoothers are discussed. Alternatively, if algebraic multigrid (AMG) [23, 24] is used, the coarse grids and smoothers will automatically adapt to any anisotropies and the method can even be applied on unstructured grids. However, AMG has additional setup costs for the coarse grids and explicitly stores the coarse grid matrices. This has a significant impact on the performance in bandwidth-dominated applications. While these "black-box" approaches work well for very general problems and do not require anisotropies to be grid-aligned, they can be computationally expensive and are more challenging to parallelise. Nevertheless, impressive results have recently been achieved with the AMG solvers both in the DUNE [52, 53] and Hypre libraries [54], which have been shown to scale to 100,000s of CPU cores. The problem is simplified significantly in the case of grid-aligned anisotropies, which are typical in atmospheric- and ocean-modelling applications. It has long been known that if the problem is anisotropic in one direction only, this can be dealt with effectively by either adapting the smoother or coarsening strategy (see e.g. [55–57] and also the discussion for simple anisotropic model problems in [9]).

Both methods can be combined as for example discussed in [36–38] where the solution of two dimensional anisotropic problems with grid-aligned anisotropies is studied. By using line-relaxation in the r-direction together with semi-coarsening in the x-direction only, the multigrid solver is robust with respect to anisotropies in both the x- and r-direction as long as they are grid-aligned. In the following we will refer to multigrid algorithms which combine horizontal semi-coarsening with vertical line relaxation in the strongly coupled direction as *tensor product multigrid* (TPMG) methods (both in 2D and in 3D).

DOI: 10.1002/fld

In [38] the convergence of such a tensor-product multigrid solver for elliptic equations of the form

$$\mathcal{L}u = -\operatorname{div}(\underline{\alpha}\operatorname{grad} u) = -\begin{pmatrix}\partial_r & \partial_x\end{pmatrix}\begin{pmatrix}\alpha_r(r,x) & 0\\ 0 & \alpha_x(r,x)\end{pmatrix}\begin{pmatrix}\partial_r\\ \partial_x\end{pmatrix}u(r,x) = f(r,x)$$

in a two dimensional domain $\Omega = [0,1]\times[0,1]$ is analysed under the assumption that the coefficients in the diagonal 2×2 matrix can be factorised, i.e.

$$\alpha_r(r,x) = \alpha_r^r(r)\alpha_r^x(x), \qquad \alpha_x(r,x) = \alpha_x^r(r)\alpha_x^x(x). \tag{20}$$

The authors show that the tensor product multigrid algorithm applied to this problem converges uniformly provided the standard multigrid algorithm with point relaxation and uniform coarsening converges uniformly for one dimensional (horizontal) operators of the form

$$\mathcal{L}^x(\lambda_j)u^x(x) = -\partial_x\left(\alpha_r^x(x)\partial_x u^x(x)\right) + \lambda_j\ \alpha_x^x(x)u^x(x)$$

where the positive values λ_j are the eigenvalues of the vertical Galerkin matrices. In particular, they analyse the strongly anisotropic case of $\alpha_r \gg \alpha_x$, which arises for example in the case of a polar grid on a disk with a (small) hole at the origin.

3.1. Tensor-product multigrid preconditioners

Based on these observations, we propose two approaches for solving the pressure correction equation in (17). In both cases we use an iterative method such as a Richardson iteration or BiCGStab and precondition it with the tensor-product multigrid algorithm.

Tensor-product multigrid with full, non-factorising profiles ($\mathrm{TPMG}^{(\mathrm{full})}$) Often the profiles encountered in atmospheric flow simulations only factorise approximately. Although the theory in [38] applies only if the coefficient functions $\alpha_r(r,x)$ and $\alpha_x(r,x)$ can be written as the product of a vertical and a horizontal function as in (20), this assumption is not used anywhere in the implementation. Our numerical experiments demonstrate that good convergence can be achieved even in the non-factorising case where we use the full operator in the multigrid preconditioner.

Tensor-product multigrid with approximately factorised profiles ($\mathrm{TPMG}^{\otimes}$) Given a set of profiles, we explicitly construct an approximate factorisation and use the resulting operator in the multigrid preconditioner; we apply a matrix-free approach, where the local stencil is reconstructed on-the-fly from the profiles. Depending on the quality of the factorisation, this may lead to a slight increase in the number of iterations of the underlying iterative solver. However, in terms of computational cost this increase is usually offset by a reduction in the amount of data that needs to be transferred from main memory. As the algorithm is memory bound, this will translate directly into performance gains. If the profiles factorise, for each horizontal cell and edge, $\mathcal{O}(1)$ entries which describe the horizontal coupling need to be stored. In the vertical direction four vectors of length n_r are required *for the entire grid*. Hence, in this case the matrix can be reconstructed from $\mathcal{O}(n_r)+\mathcal{O}(n_{\mathcal{S}})$ values where $n_{\mathcal{S}}$ is the number of horizontal grid cells. This should be compared to $\mathcal{O}(n_r\times n_{\mathcal{S}})$ data transfers for constructing the matrix in the $\mathrm{TPMG}^{(\mathrm{full})}$ approach. We also prove formally in the following section that the Richardson iteration with $\mathrm{TPMG}^{\otimes}$ preconditioner converges if the non-factorising part of the operator is small.

3.2. Convergence of Tensor-product multigrid

The convergence theory for conforming finite element discretisations and for factorising profiles in a spherical shell is a straightforward generalisation of the two dimensional case in [38] and is written down in detail in [39] based on the multigrid convergence theory in [58]. In the following we outline the proof for the pressure correction in (17). This is done in two steps: we first argue that if the profiles factorise and the advection term is dropped, the resulting symmetric positive definite

Prepared using fldauth.cls
DOI: 10.1002/fld

equation can be solved efficiently with a tensor-product multigrid iteration. We then show that if this factorising operator is used as a preconditioner for a Richardson iteration, the method converges also for the non-factorising equation provided the non-factorising contribution is sufficiently small.

For the numerical experiments presented in this paper we use a cell-centred finite volume discretisation, which is typical in many atmospheric modelling codes, but differs from the conforming finite element discretisation which is assumed in the proof. However, as already remarked in [38], in many cases finite difference and finite volume schemes agree with finite element schemes if a specific quadrature formula is used. In particular, in [39] it is shown that the finite-volume scheme on a orthogonal cubic grid is equivalent to a trilinear finite element discretisation on a shifted grid, if a suitable quadrature rule is used. For this reason we believe that the theoretical analysis below also gives a good justification as to why the method works well here.

3.2.1. Factorising case Consider the following PDE in the spherical shell $\Omega = \mathcal{S} \times [1, 1+H]$:

$$\mathcal{L}^{\otimes} u = -\omega^2 \boldsymbol{\nabla} \cdot [\underline{\alpha}(r, \hat{\boldsymbol{r}}) \boldsymbol{\nabla} u(r, \hat{\boldsymbol{r}})] + \beta(r, \hat{\boldsymbol{r}}) u(r, \hat{\boldsymbol{r}}) = f(r, \hat{\boldsymbol{r}}) \tag{21}$$

with $r \in [1, 1+H]$, $\hat{\boldsymbol{r}} \in \mathcal{S}$. This should be compared to (17); for simplicity we do not consider the vertical advection term in this section, as it may in general destroy the positive definiteness of the problem. However, for high horizontal resolution this term is small and can be treated as a perturbation. We further assume that the 3×3 matrix $\underline{\alpha}$ and the function β have the following form that factorises into the product of a horizontal and a vertical function:

$$\underline{\alpha} = \begin{pmatrix} \alpha_r(r, \hat{\boldsymbol{r}}) & 0 \\ 0 & \underline{\alpha}_{\mathcal{S}}(r, \hat{\boldsymbol{r}}) \end{pmatrix} = \begin{pmatrix} \alpha_r^r(r)\alpha_r^{\mathcal{S}}(\hat{\boldsymbol{r}}) & 0 \\ 0 & \alpha_{\mathcal{S}}^r(r)\underline{\alpha}_{\mathcal{S}}^{\mathcal{S}}(\hat{\boldsymbol{r}}) \end{pmatrix}, \qquad \beta(r, \hat{\boldsymbol{r}}) = \beta^r(r)\beta^{\mathcal{S}}(\hat{\boldsymbol{r}}).$$

We also require that $\beta^{\mathcal{S}}(\hat{\boldsymbol{r}}) = \alpha_r^{\mathcal{S}}(\hat{\boldsymbol{r}})$, which is satisfied for all factorisations that we use in our numerical experiments. The 2×2 matrix $\underline{\alpha}_{\mathcal{S}}^{\mathcal{S}}$ is required to be symmetric positive definite and we assume $\alpha_{\mathcal{S}}^r, \alpha_r^r, \alpha_r^{\mathcal{S}}, \beta^r > 0$.

To discretise the problem, we choose finite element spaces $V^{\mathcal{S}}$ over $\mathcal{S}$ and V^r over $[1, 1+H]$ and tensorise them to obtain the product space $V \equiv V^r \otimes V^{\mathcal{S}}$ over Ω. We write $n_r \equiv \dim V^r$ and $n_{\mathcal{S}} \equiv \dim V^{\mathcal{S}}$. For any two functions $u(r, \hat{\boldsymbol{r}}) = u^r(r)u^{\mathcal{S}}(\hat{\boldsymbol{r}})$, $v(r, \hat{\boldsymbol{r}}) = v^r(r)v^{\mathcal{S}}(\hat{\boldsymbol{r}})$ in V the bilinear form $a : V \times V \to \mathbb{R}$ associated with the operator $\mathcal{L}^{\otimes}$ in (21) can be expressed in terms of the bilinear forms

$$\begin{aligned}
a^r(u^r, v^r) &\equiv \int_1^{1+H} \alpha_r^r(r) \partial_r u^r(r) \partial_r v^r(r)\, dr, \\
b^r(u^r, v^r) &\equiv \int_1^{1+H} \beta^r(r) u^r(r) v^r(r)\, dr, \\
m^r(u^r, v^r) &\equiv \int_1^{1+H} \alpha_{\mathcal{S}}^r(r) u^r(r) v^r(r)\, dr, \\
a^{\mathcal{S}}(u^{\mathcal{S}}, v^{\mathcal{S}}) &\equiv \int_{\mathcal{S}} \left\langle \underline{\alpha}_{\mathcal{S}}^{\mathcal{S}}(\hat{\boldsymbol{r}}) \boldsymbol{\nabla}_{\mathcal{S}} u^{\mathcal{S}}(\hat{\boldsymbol{r}}), \boldsymbol{\nabla}_{\mathcal{S}} v^{\mathcal{S}}(\hat{\boldsymbol{r}}) \right\rangle\, d\hat{\boldsymbol{r}} \text{ and} \\
m^{\mathcal{S}}(u^{\mathcal{S}}, v^{\mathcal{S}}) &\equiv \int_{\mathcal{S}} \alpha_r^{\mathcal{S}}(\hat{\boldsymbol{r}}) u^{\mathcal{S}}(\hat{\boldsymbol{r}}) v^{\mathcal{S}}(\hat{\boldsymbol{r}})\, d\hat{\boldsymbol{r}}
\end{aligned} \tag{22}$$

as

$$a(u, v) = \left(\omega^2 a^r(u^r, v^r) + b^r(u^r, v^r)\right) m^{\mathcal{S}}(u^{\mathcal{S}}, v^{\mathcal{S}}) + \omega^2 m^r(u^r, v^r) a^{\mathcal{S}}(u^{\mathcal{S}}, v^{\mathcal{S}}).$$

Using the Kronecker product, the Galerkin-matrix representation $A^{\otimes}$ of the bilinear form $a(\cdot, \cdot)$ can then be expressed in terms of the Galerkin matrices of the bilinear forms in (22), i.e.

$$A^{\otimes} = \left(\omega^2 A^r + B^r\right) \otimes M^{\mathcal{S}} + \omega^2 M^r \otimes A^{\mathcal{S}}.$$

Here $A^r, M^r, B^r \in \mathbb{R}^{n_r \times n_r}$ correspond to $a^r(\cdot, \cdot)$, $m^r(\cdot, \cdot)$ and $b^r(\cdot, \cdot)$ respectively and describe the vertical derivative- and mass- matrices. Analogously the derivative and mass matrix in the horizontal direction are described by $A^{\mathcal{S}}, M^{\mathcal{S}} \in \mathbb{R}^{n_{\mathcal{S}} \times n_{\mathcal{S}}}$, which correspond to $a^{\mathcal{S}}(\cdot, \cdot)$ and $m^{\mathcal{S}}(\cdot, \cdot)$.

DOI: 10.1002/fld

To use the tensor-product multigrid approach, we further assume that there is a nested sequence

$$V_0^{\mathcal{S}} \subset V_1^{\mathcal{S}} \subset \cdots \subset V_L^{\mathcal{S}} \equiv V^{\mathcal{S}} \tag{23}$$

of finite element spaces over $\mathcal{S}$, where the subscript ℓ denotes the multigrid level; for the icosahedral and cubed sphere grid this hierarchy naturally exists. We then use $V_\ell \equiv V^r \otimes V_\ell^{\mathcal{S}}$ to discretise the full three dimensional problem on the multigrid level ℓ, i.e. we do not coarsen in the vertical direction. The line smoother then corresponds to collectively relaxing all degrees of freedom in each of the n_r-dimensional subspaces $\operatorname{span}\{\{\psi_{\ell,k}\} \otimes V^r\}$ where $\psi_{\ell,k}$ are the nodal basis functions on level ℓ.

The two-dimensional prolongation $P_\ell^{\mathcal{S}} : V_\ell^{\mathcal{S}} \to V_{\ell+1}^{\mathcal{S}}$ and restriction $R_\ell^{\mathcal{S}} \equiv \left(P_\ell^{\mathcal{S}}\right)^T : V_{\ell+1}^{\mathcal{S}} \to V_\ell^{\mathcal{S}}$ naturally induce intergrid transfer operators between the three dimensional spaces V_ℓ and $V_{\ell+1}$ by $P_\ell \equiv \mathrm{Id} \otimes P_\ell^{\mathcal{S}}$, $R_\ell \equiv \mathrm{Id} \otimes R_\ell^{\mathcal{S}}$. On each multigrid level the matrix $A_\ell^\otimes$ can be constructed recursively using the Galerkin product $A_\ell^\otimes \equiv R_\ell A_{\ell+1}^\otimes P_\ell$ and it is easy to see that $A_\ell^\otimes$ and the (block-)smoother $W_\ell^\otimes$ can be written as

$$\begin{aligned} A_\ell^\otimes &= \left(\omega^2 A^r + B^r\right) \otimes M_\ell^{\mathcal{S}} + \omega^2 M^r \otimes A_\ell^{\mathcal{S}}, \\ W_\ell^\otimes &= \left(\omega^2 A^r + B^r\right) \otimes W_\ell^{M,\mathcal{S}} + \omega^2 M^r \otimes W_\ell^{A,\mathcal{S}}. \end{aligned}$$

In the case of weighted block-Jacobi relaxation, for example, the matrices $W_\ell^{M,\mathcal{S}}$ and $W_\ell^{A,\mathcal{S}}$ are the weighted diagonals of $M_\ell^{\mathcal{S}}$ and $A_\ell^{\mathcal{S}}$. One V-cycle of the tensor product multigrid algorithm can now be written down compactly as follows.

Algorithm 1 Tensor Product Multigrid V-cycle. Input: system matrix $A_\ell^\otimes$, RHS $\mathbf{f}_\ell$, initial guess $\mathbf{u}_\ell$

1: *Pre-Smoothing:* ν_ℓ^{pre} steps of $\mathbf{u}_\ell \;\to\; (W_\ell^\otimes)^{-1}\mathbf{f}_\ell + \left(I - (W_\ell^\otimes)^{-1} A_\ell^\otimes\right)\mathbf{u}_\ell$
2: **if** $\ell > 0$ **then**
3: *Residual Calculation:* $\mathbf{r}_\ell = \mathbf{f}_\ell - A_\ell^\otimes \mathbf{u}_\ell$
4: *Recursion:* Apply algorithm with $A_{\ell-1}^\otimes$, $\mathbf{f}_{\ell-1} = R_{\ell-1}\mathbf{r}_\ell$, $\mathbf{u}_{\ell-1} = 0$
5: *Coarse Grid Correction:* $\mathbf{u}_\ell \;\to\; \mathbf{u}_\ell + P_{\ell-1}\mathbf{u}_{\ell-1}$
6: **end if**
7: *Post-Smoothing:* ν_ℓ^{post} steps of $\mathbf{u}_\ell \;\to\; (W_\ell^\otimes)^{-1}\mathbf{f}_\ell + \left(I - (W_\ell^\otimes)^{-1} A_\ell^\otimes\right)\mathbf{u}_\ell$

On the finest level L, this V-cycle is applied to the right hand side $\mathbf{f}_L = \mathbf{f}$ of the original problem until the residual error is reduced below a certain tolerance. We typically choose the numbers of smoothing steps to be $\nu_\ell^{\text{pre}} = 2$ and $\nu_\ell^{\text{post}} = 2$, for $\ell > 0$, and $\nu_0^{\text{pre}} + \nu_0^{\text{post}} = 1$ on the coarsest grid. To simply apply a few steps of the smoother on the coarsest grid is sufficient because the CFL condition ensures that the system matrix $A_0^\otimes$ on the coarsest grid is dominated by the mass matrix term $B_\ell^r \otimes M_\ell^{\mathcal{S}}$ and thus well-conditioned. For more details on the algorithm which was used for our numerical experiments see Section 4.2.

Reduction of the theory to two dimensions The crucial idea in [38] is now that it is possible to construct a set of n_r invariant $n_{\mathcal{S}}$-dimensional subspaces such that the convergence of the tensor product multigrid method for the problem in $\Omega \subset \mathbb{R}^3$ can be analysed by independently studying the convergence of a standard multigrid algorithm in each of these subspaces over $\mathcal{S} \subset \mathbb{R}^2$. This can be seen as follows: because both A^r and M^r are positive definite, there exists an eigenbasis $\mathbf{e}_j^r$, $j = 1, \ldots, n_r$, of V^r and a corresponding set of strictly positive eigenvalues λ_j such that

$$\left(\omega^2 A^r + B^r\right)\mathbf{e}_j^r = \lambda_j M^r \mathbf{e}_j^r, \qquad \left\langle M^r \mathbf{e}_j^r, \mathbf{e}_k^r \right\rangle = \delta_{j,k} \qquad \text{for all} \quad j,k \in \{1,\ldots,N\}. \tag{24}$$

It follows from simple identities for the inner product on tensor product spaces that

$$\begin{aligned} \left\langle \mathbf{e}_k^r \otimes \mathbf{u}^{\mathcal{S}}, A_\ell^\otimes(\mathbf{e}_j^r \otimes \mathbf{v}^{\mathcal{S}}) \right\rangle &= \left\langle \mathbf{e}_k^r \otimes \mathbf{u}^{\mathcal{S}}, \left(\omega^2 A^r + B^r\right)\mathbf{e}_j^r \otimes M_\ell^{\mathcal{S}}\mathbf{v}^{\mathcal{S}} + M^r \mathbf{e}_j^r \otimes A_\ell^{\mathcal{S}}\mathbf{v}^{\mathcal{S}} \right\rangle \\ &= \delta_{jk} \left\langle \mathbf{u}^{\mathcal{S}}, \left(\lambda_j M_\ell^{\mathcal{S}} + A_\ell^{\mathcal{S}}\right)\mathbf{v}^{\mathcal{S}} \right\rangle \end{aligned}$$

DOI: 10.1002/fld

and so the subspaces spanned by the different $\mathbf{e}_j^r$ are $A_\ell^\otimes$-orthogonal, with a similar property for the smoother matrix $W_\ell^\otimes$. As we do not coarsen in the vertical direction, the intergrid operators P_ℓ and R_ℓ do not mix different subspaces. For each j the space $\mathrm{span}\{\mathbf{e}_j^r\} \otimes V_\ell^\mathcal{S}$ is trivially isomorphic to $V_\ell^\mathcal{S}$ and each of the n_r independent subspaces corresponds to a two dimensional problem on $\mathcal{S}$ with the following matrix representation of the linear operator and smoother:

$$A_{\ell,j}^\mathcal{S} \equiv \omega^2 A_\ell^\mathcal{S} + \lambda_j M_\ell^\mathcal{S}, \qquad\qquad W_{\ell,j}^\mathcal{S} \equiv \omega^2 W_\ell^{A,\mathcal{S}} + \lambda_j W_\ell^{M,\mathcal{S}}.$$

In particular, $A_{\ell,j}^\mathcal{S}$ is the Galerkin matrix which is obtained from discretising the bilinear form $\omega^2 a^\mathcal{S}(u^\mathcal{S}, v^\mathcal{S}) + \lambda_j m^\mathcal{S}(u^\mathcal{S}, v^\mathcal{S})$ on $V_\ell^\mathcal{S}$. This bilinear form is the weak formulation of the following two dimensional operator:

$$\mathcal{L}_j^\mathcal{S} u^\mathcal{S}(\hat{\boldsymbol{r}}) = -\omega^2 \boldsymbol{\nabla}_\mathcal{S} \cdot \left(\underline{\alpha}_\mathcal{S}^\mathcal{S}(\hat{\boldsymbol{r}}) \boldsymbol{\nabla}_\mathcal{S} u^\mathcal{S}(\hat{\boldsymbol{r}})\right) + \lambda_j \alpha_r^\mathcal{S}(\hat{\boldsymbol{r}}) u^\mathcal{S}(\hat{\boldsymbol{r}}) \tag{25}$$

Convergence of two dimensional multigrid According to Theorem 10.7.15 in [59], the multigrid V-cycle converges for each of the two dimensional operators $\mathcal{L}_j^\mathcal{S}$, $j = 1, \ldots, N$ if there exists a C_A such that the smoothing property

$$A_{\ell,j}^\mathcal{S} \leq W_{\ell,j}^\mathcal{S} \tag{26}$$

and the approximation property

$$0 \leq \left(A_{\ell+1,j}^\mathcal{S}\right)^{-1} - P_\ell^\mathcal{S} \left(A_{\ell,j}^\mathcal{S}\right)^{-1} R_\ell^\mathcal{S} \leq C_A \left(W_{\ell+1,j}^\mathcal{S}\right)^{-1} \tag{27}$$

are satisfied on all levels $\ell = 1, \ldots, L$.

The smoothing property (26) is automatically satisfied for (sufficiently damped) point Jacobi and SOR smoothers (Remark 4.6.5 in [39]). To see this, denote the matrix consisting only of the diagonal entries of $A_{\ell,j}^\mathcal{S}$ by $D_{\ell,j}^\mathcal{S}$ and use $W_{\ell,j}^\mathcal{S} = \rho_{\mathrm{relax}}^{-1} D_{d,j}^\mathcal{S}$, i.e. weighted point Jacobi relaxation. The relaxation parameter is chosen such that $\rho_{\mathrm{relax}} \leq ||(D_{\ell,j}^\mathcal{S})^{-1} A_{\ell,j}^\mathcal{S}||^{-1}$ where $||\cdot||$ is the spectral norm. Then (26) follows by definition from the equivalence $-\mathrm{Id} \leq X \leq \mathrm{Id} \Leftrightarrow ||X|| \leq 1$ applied to $X = (W_{\ell,j}^\mathcal{S})^{-1} A_{\ell,j}^\mathcal{S}$.

A proof of the approximation property is significantly harder and we will not give it here (see Lemma 10.7.8 and Remark 10.7.13 in [59]). It depends on some minimal regularity assumptions on the profiles $\underline{\alpha}_\mathcal{S}^\mathcal{S}(\hat{\boldsymbol{r}})$ and $\alpha_r^\mathcal{S}(\hat{\boldsymbol{r}})$. The constant C_A may depend on the contrast, i.e. the maximum variation of the profiles. We stress again that we use quasi-uniform grids for the horizontal discretisation (see the review in [60] for a discussion of grids considered in meteorological application). In contrast to latitude-longitude grids, where the convergent grid lines near the pole introduce an additional horizontal anisotropy, the ratio between the smallest and largest grid spacing is bounded from below in the grids we consider. Hence the simple block-Jacobi and block-SOR smoothers which relax all degrees of freedom in one vertical column simultaneously will be efficient and no additional horizontal plane smoothing or selective semi-coarsening as described in [20] is required.

As the two dimensional equations are solved on the unit sphere, the operator $\mathcal{L}_j^\mathcal{S}$ could become near-singular if $\lambda_j \to 0$. However, it is easy to see that this is not the case. As noted in Section 2.1 we require the scaling $\omega \propto \Delta t \propto h_L$ to keep the Courant number fixed as the horizontal resolution increases. Therefore the second order term in (25) is of order 1 and hence the relative importance of the two terms in (25) is independent of grid resolution. It follows that all the eigenvalues λ_j of (24) are of order 1. It is a reasonable assumption that the profiles $\underline{\alpha}_\mathcal{S}^\mathcal{S}(\hat{\boldsymbol{r}})$, $\alpha_r^\mathcal{S}(\hat{\boldsymbol{r}})$ are "well-behaved" in the sense that they are dominated by large scale variations due to global weather systems, small scale phenomena such as strong local variations carry substantially less energy. In this case we expect the spectrum of $\mathcal{L}_j^\mathcal{S}$ to be bounded from above and below by two constants which are independent of h_L.

Convergence of three dimensional multigrid As argued above, the three dimensional problem can be decoupled into a set of n_r two-dimensional problems. Due to the particular form of the

DOI: 10.1002/fld

smoother and of the prolongation/restriction matrices, it is in fact easy to verify that the smoothing property and the approximation property

$$A_\ell^\otimes \leq W_\ell^\otimes \quad \text{and} \qquad 0 \leq \left(A_{\ell+1}^\otimes\right)^{-1} - P_\ell^h \left(A_\ell^\otimes\right)^{-1} R_\ell^h \leq C_A \left(W_{\ell+1}^\otimes\right)^{-1},$$

for the tensor product multigrid algorithm for the original 3D problem on Ω follow directly from the respective properties (26) and (27) for the 2D problems on $\mathcal{S}$, for all $j = 1, \ldots, N$.

Theorem 1
Let us assume that (26) and (27) are satisfied, for all $j = 1, \ldots, N$, and let $M^\otimes$ denote the iteration matrix for one step of the tensor product multigrid V-cycle defined above, i.e.

$$\mathbf{u} - \mathbf{u}_\otimes^* \;\mapsto\; M^\otimes \left(\mathbf{u} - \mathbf{u}_\otimes^*\right)$$

where $\mathbf{u}_\otimes^*$ is the exact solution of the equation $A^\otimes \mathbf{u}_\otimes^* = \mathbf{f}$. Then the convergence rate

$$\rho_A^\otimes = ||M^\otimes||_{A^\otimes} \leq \frac{C_A}{C_A + 2(\nu_{\text{pre}} + \nu_{\text{post}})} \;<\; 1$$

independent of h_L, where $||\cdot||_{A^\otimes}$ is the energy norm induced by $A^\otimes$.

This is the main result given and proved for the two dimensional case in [38, Theorem 2]. As we have seen above, the proof extends directly also to three dimensions and to our pressure correction problem here. In that case the assumptions of the theorem are satisfied as discussed above.

3.2.2. Non-factorising case We now assume that the matrix A can be written as the sum of a perfectly factorising symmetric positive definite matrix $A^\otimes$ and a small correction δA, namely $A = A^\otimes + \delta A$. We quantify the deviation from perfect factorisation by $\Delta = ||\left(A^\otimes\right)^{-1} \delta A||_{A^\otimes}$ and assume that $\Delta < 1$. We also assume that the theory in Section 3.2.1 applies and the multigrid iteration for the factorising operator $A^\otimes$ converges, i.e. the error is reduced by a factor $\rho_A^\otimes < 1$ in every multigrid V-cycle. The Richardson iteration for the full operator A preconditioned with μ multigrid V-cycle cycles for $A^\otimes$ can then formally be written as

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \left[\mathrm{Id} - \left(M^\otimes\right)^\mu\right] \left(A^\otimes\right)^{-1} \left(\mathbf{f} - A\mathbf{u}^{(k)}\right).$$

Then at every step the error $\mathbf{u}^{(k)} - \mathbf{u}^*$ to the exact solution $\mathbf{u}^* := A^{-1}\mathbf{f}$ is reduced by a factor

$$\begin{aligned} \rho_A \;&=\; ||\,\mathrm{Id} - \left[\mathrm{Id} - \left(M^\otimes\right)^\mu\right] \left(A^\otimes\right)^{-1} A||_{A^\otimes} \\ &\leq\; ||\left(A^\otimes\right)^{-1} A - \mathrm{Id}\,||_{A^\otimes} + ||M^\otimes||_{A^\otimes}^\mu ||\left(A^\otimes\right)^{-1} A||_{A^\otimes} \;\leq\; \Delta + (1+\Delta)(\rho_A^\otimes)^\mu. \end{aligned} \tag{28}$$

Thus, for an arbitrary $\Delta < 1$ the convergence rate ρ_A is less than 1, provided the number of V-cycles $\mu > \log_{\rho_A^\otimes}\left((1-\Delta)/(1+\Delta)\right)$. On the other hand, if we only apply one V-cycle ($\mu = 1$), then a convergence rate $\rho_A < 1$ can still be guaranteed provided $\Delta < (1 - \rho_A^\otimes)/(1 + \rho_A^\otimes)$. Similar results can also be proved for the convergence of Krylov solvers, such as BiCGStab, preconditioned with μ multigrid V-cycle cycles for $A^\otimes$.

It is usually difficult to quantify Δ for a specific problem, but in Section 5.2 we study a model problem where we vary the size of Δ via an explicit parameter and study the performance of the solver as Δ increases.

4. DISCRETISATION AND IMPLEMENTATION

In practise, and as we demonstrate in the following, the tensor product preconditioners will be efficient for a wider range of problems not covered by the formal theory. We now describe the discretisation and DUNE implementation of the solvers we used in our numerical experiments.

DOI: 10.1002/fld

4.1. Grid structure and discretisation

For simplicity we use a simple finite volume discretisation for all numerical experiments in this work. More complex schemes such as mimetic mixed finite elements are also currently under consideration for the development of dynamical cores [61, 62] and might require the solution of the equation in a different pressure space, such as higher order DG space. However, the basic ideas described in this work can still be applied.

Grids used in meteorological applications (and also in many ocean models [2, 3]) usually have a tensor-product structure. They consist of a semi-structured two dimensional horizontal grid on the surface of the sphere and a one-dimensional vertical grid which is often graded to achieve higher resolution near the surface. In particular, each three dimensional grid cell $E = (T, k)$ can be uniquely identified by the corresponding horizontal cell T and a vertical index $k \in 1, \ldots, n_r$. This tensor-product structure in itself has important implications for the performance of any implementation: while it might be necessary to use indirect indexing for the horizontal grid, the vertical grid can always be addressed directly. As typically the number of vertical levels is large with $n_r \gtrsim 100$, the cost of indirect addressing in the horizontal direction can be "hidden" [63], a phenomenon which we have confirmed numerically for our solvers in Section 5.1. Furthermore fields can be stored such that the levels in each column are stored consecutively in memory, which leads to efficient cache utilisation (however, as discussed in [41] a different memory layout has to be used on GPU architectures where the vertically-consecutive storage would prevent coalesced memory access in the tridiagonal solve). To be able to use the geometric multigrid solvers described in this work, we also assume that the horizontal grid has a natural hierarchy; this is true for the icosahedral grids which are used in our numerical tests where each triangular coarse grid cell consist of four smaller triangles on the next-finer multigrid level. In contrast to a simple longitude-latitude grid, these semi-structured grids have no pole-problem, i.e. the ratio between the size of the largest and smallest grid spacing is bounded. This implies that there is no additional horizontal anisotropy which would further complicate the construction of a solver (however, as has been shown in [20, 21], the tensor-product multigrid approach can still be applied for longitude-latitude grids if the horizontal coarsening strategy is adapted appropriately).

In the finite volume discretisation any continuous field $u(r, \hat{\boldsymbol{r}})$ is approximated by its average value in a grid cell. In particular, for each *horizontal* grid cell T we store one vector $\mathbf{u}_T$ of length n_r representing the field in the vertical column. In this cell the discrete equation (19) for the n_r-vector $\mathbf{u}_T$ can be written as

$$(\underline{A}\mathbf{u})_T = \underline{A}_T \mathbf{u}_T + \sum_{T' \in \mathcal{N}(T)} \underline{A}_{TT'} \mathbf{u}_{T'} = \mathbf{f}_T, \tag{29}$$

where the sum runs over all horizontal neighbours $T' \in \mathcal{N}(T)$ of T. In this expression $\underline{A}_T$ and $\underline{A}_{TT'}$ are $n_r \times n_r$ tridiagonal- and diagonal matrices of the form

$$A_T = \mathrm{tridiag}(\mathbf{a}_T, \mathbf{b}_T, \mathbf{c}_T), \qquad A_{TT'} = \mathrm{diag}(\mathbf{d}_{TT'}). \tag{30}$$

Both matrices can be reconstructed on-the-fly from a number of scalar quantities, which are obtained from a discrete approximation of the profiles in (18) and geometric factors. This on-the-fly reconstruction of matrix elements reduces the amount of main memory access, in particular if the factorising profiles in the $\mathrm{TPMG}^{\otimes}$ preconditioner are used. For each horizontal cell T the explicit expressions of the diagonals $\mathbf{a}_T$, $\mathbf{d}_{TT'}$ and upper- and lower- subdiagonals $\mathbf{b}_T$, $\mathbf{c}_T$ depend on whether the profiles can be factorised or not and are given explicitly in the next section. A block-SOR iteration with overrelaxation factor ρ_{relax} can then be written as

$$\mathbf{u}_T \leftarrow \mathbf{u}_T + \rho_{\mathrm{relax}} \left(\underline{A}_T\right)^{-1} \left(\mathbf{b}_T - (\underline{A}\mathbf{u})_T\right) \tag{31}$$

and requires a tridiagonal solve in each vertical column to apply the inverse of the matrix $\underline{A}_T$ to the residual. This can be implemented using the Thomas algorithm [64].

DOI: 10.1002/fld

4.2. Multigrid algorithm

In the multigrid algorithm the operator is re-discretised on every grid level. This is straightforward since we assume that the profiles α_r, $\alpha_{\mathcal{S}}$, ξ_r and β are known at every point in space. In practice the profiles are relatively smooth (i.e. without large, high-contrast jumps) and can be obtained as a linear interpolation of data from a Unified Model run. Re-discretising the operator on every grid level also leads to much smaller stencils than the Galerkin product approach in AMG.

Unless otherwise stated, in our numerical experiments we always use 6 multigrid levels with two vertical line-SOR pre- and post- smoothing steps on each level ($\nu^{\text{pre}} = \nu^{\text{post}} = 2$); the overrelaxation factor in the smoother was set to $\rho_{\text{relax}} = 1$. One smoother iteration is used to solve the coarse grid problem. We use linear interpolation to prolongate the solution to the next-finer grid ($2h \mapsto h$). The right hand side, which in each cell represents a cell-integral of a field, is restricted to the next-coarser level ($h \mapsto 2h$) by summing the fine grid values of all four fine grid cells comprising the coarse grid cell. Thus, these intergrid-operations, which are described in more detail in Appendix A, only require interpolation and summation in the horizontal direction.

4.3. Matrix-free DUNE Implementation

All code was implemented using the DUNE library [43, 44], which provides a set of C++ classes for solving PDEs using grid based methods. In particular, it provides interfaces to (parallel) grid implementations such as ALUGrid [45–47] and UGGrid [48]. The implementation of the grids is separated from data which is attached to the grid by the user via mapper functions between different grid entities (cells, edges, vertices) and the local data arrays. In our case we used the DUNE-grid module to implement a two dimensional host grid and then attached a whole column of length n_r to each horizontal grid cell T. We represent the matrix as follows: in the non-factorising case ($\text{TPMG}^{(\text{full})}$), we store a vector $\hat{\boldsymbol{\beta}}_T$ of length n_r at each horizontal cell to represent the zero order term, two vectors $(\hat{\boldsymbol{\alpha}}_r)_T$ and $(\hat{\boldsymbol{\xi}}_r)_T$ of length $n_r + 1$ to represent the vertical diffusion and advection terms, and one vector $(\hat{\boldsymbol{\alpha}}_{\mathcal{S}})_{TT'}$ of length n_r at each horizontal edge TT'. The explicit form of these vectors is obtained by a standard finite volume discretisation of the problem. The vectors $\mathbf{a}_T$, $\mathbf{b}_T$, $\mathbf{c}_T$ and $\mathbf{d}_{TT'}$ in (30) are

$$\begin{aligned} d_{TT',k} &= -(\hat{\alpha}_{\mathcal{S}})_{TT',k}, & d_{T,k} &= \sum_{T' \in \mathcal{N}(T)} \tilde{d}_{TT',k}, \\ b_{T,k} &= -(\hat{\alpha}_r)_{T,k+1} - (\hat{\xi}_r)_{T,k+1}, & c_{T,k} &= -(\hat{\alpha}_r)_{T,k+1} + (\hat{\xi}_r)_{T,k+1}, \\ a_{T,k} &= \hat{\beta}_{T,k} - (b_{T,k} + c_{T,k} + d_{T,k}). \end{aligned} \tag{32}$$

In the factorising case ($\text{TPMG}^{\otimes}$) it is only necessary to store *scalars* $\hat{\beta}^{\mathcal{S}}_T$, $(\hat{\alpha}^{\mathcal{S}}_r)_T$, $(\hat{\xi}^{\mathcal{S}}_r)_T$ and $(\hat{\alpha}^{\mathcal{S}}_{\mathcal{S}})_{TT'}$ on the horizontal cells and edges. In addition to this, four vectors of length n_r and $n_r + 1$ ($\hat{\boldsymbol{\beta}}^r$, $\hat{\boldsymbol{\alpha}}^r_{\mathcal{S}}$, $\hat{\boldsymbol{\alpha}}^r_r$ and $\hat{\boldsymbol{\xi}}^r_r$) which arise from the vertical discretisation need to be stored once for the entire grid. Similarly to (32) the matrix entries in (30) can be calculated on the fly as

$$\begin{aligned} d_{TT',k} &= -(\hat{\alpha}^r_{\mathcal{S}})_k(\hat{\alpha}^{\mathcal{S}}_{\mathcal{S}})_{TT'}, & d_{T,k} &= \sum_{T' \in \mathcal{N}(T)} \tilde{d}_{TT',k} \\ b_{T,k} &= -(\hat{\alpha}^r_r)_{k+1}(\hat{\alpha}^{\mathcal{S}}_r)_T - (\hat{\xi}^r_r)_{k+1}(\hat{\xi}^{\mathcal{S}}_r)_T, & c_{T,k} &= -(\hat{\alpha}^r_r)_{k+1}(\hat{\alpha}^{\mathcal{S}}_r)_T + (\hat{\xi}^r_r)_{k+1}(\hat{\xi}^{\mathcal{S}}_r)_T, \\ a_{T,k} &= \hat{\beta}^r_k\hat{\beta}^{\mathcal{S}}_T - (b_{T,k} + c_{T,k} + d_{T,k}). \end{aligned} \tag{33}$$

The scalars $\hat{\beta}^{\mathcal{S}}_T$, $(\hat{\alpha}^{\mathcal{S}}_r)_T$, $(\hat{\xi}^{\mathcal{S}}_r)_T$ and $(\hat{\alpha}^{\mathcal{S}}_{\mathcal{S}})_{TT'}$ only need to be read once per vertical column and the associated cost can be hidden together with the cost of indirect addressing on the horizontal grid for large enough n_r. Moreover, the vectors $\hat{\boldsymbol{\beta}}^r$, $\hat{\boldsymbol{\alpha}}^r_{\mathcal{S}}$, $\hat{\boldsymbol{\alpha}}^r_r$ and $\hat{\boldsymbol{\xi}}^r_r$ require only a small amount of memory and can be cached. In summary, the cost of memory access for the matrix is likely to be significantly smaller than the cost of accessing field vectors such as $\mathbf{u}_T$ and $\mathbf{b}_T$ when solving the tridiagonal system in (31) or in the matrix vector product.

The DUNE-grid interface provides iterators over the horizontal grid cells and over the neighbours of each cell. To implement for example the sparse matrix vector product (SpMV) in (29) we iterate

DOI: 10.1002/fld

over all horizontal grid cells T, and then in each cell we loop over the edges TT' for all neighbours T' to read the profiles stored on the cells and edges from memory and construct the matrices $\underline{A}_T$ and $\underline{A}_{TT'}$. These are then applied to the local vectors $\mathbf{u}_T$ and $\mathbf{u}_{T'}$ to evaluate $(\underline{A}\mathbf{u})_T$, which requires inner loops over the vertical levels. Of all grids that are currently available through the DUNE interface we found that only ALUGrid can be used to represent a two-dimensional sphere embedded in three dimensional space. Unfortunately the scalability of ALUGrid is very limited because in a parallel implementation the entire grid is stored on each processor. Alternatively we used a three dimensional UGGrid implementation for a thin spherical shell consisting of one vertical layer to represent the unit sphere. Based on the coarsest grid, finer multigrid levels can be constructed by refinement in the horizontal direction only. Any geometric quantities in this thin three dimensional grid can then be related to the corresponding values on the two dimensional grid by simple scaling factors. We implemented both a gnomonic cubed sphere grid [65] and an icosahedral grid, for which the grid points are projected onto the sphere, and all numerical results reported in this work were obtained with the icosahedral grid. We assume that on each multigrid level the cells of the two-dimensional grid are flat, i.e. strictly speaking the grid is a polyhedron. While this differs from curvilinear grids where the interior of each cell is projected onto the sphere, it is a very good approximation for high resolution. In particular it does not cause any problems on the coarser multigrid levels since those are only used to accelerate the fine grid solution.

As is typical in atmospheric applications, parallel domain decomposition is in the horizontal direction only. As the DUNE host grids that we used are already inherently parallel, parallelisation of the code was straightforward by calling the relevant halo exchange routines when necessary. Load balancing was achieved by choosing the problem size such that the number of cells on the coarsest level is identical to the number of processors and each processor "owns" one coarse grid cell and the corresponding child cells. While at first sight this might cause a problem for large core counts because the coarsest level still has a relatively large number of degrees of freedom and the multigrid hierarchy is very shallow, it turns out that the zero order term in the Helmholtz equation (17) averts potential problems. This is because relative to the zero order term the importance of the horizontal diffusion term decreases with a factor of four on each coarse level, and so after a small number of coarsening steps the problem is well conditioned and can be solved by a very small number of smoothing iterations. An alternative and more physical explanation is that any interactions in the continuous PDE in (17) are exponentially damped with an intrinsic length scale ω and hence it is not necessary to coarsen the grid beyond this scale. This has been confirmed numerically for a simplified test problem in [22], where it has been shown that as little as four multigrid levels still give very good convergence for typical grid spacings and time step sizes. In the parallel scaling tests in this work we typically used 6 or 7 multigrid levels and one iteration of the smoother to solve the coarse grid problem.

5. NUMERICAL RESULTS

In the following we study the performance of the two tensor-product preconditioners $\mathrm{TPMG}^{(\mathrm{full})}$ and $\mathrm{TPMG}^{\otimes}$ described in Section 3.1 applied to two test cases in atmospheric flow simulation. We confirm the optimality and robustness of $\mathrm{TPMG}^{(\mathrm{full})}$ even for non-factorising profiles, compare the performance of the two variants and study their parallel scalability. Unless stated otherwise, all runs (including the sequential tests) were carried out on the phase 3 configuration of the HECToR supercomputer, which consists 2816 compute nodes with two 16-core AMD Opteron 2.3GHz Interlagos processors each. The entire cluster contains 90,112 cores in total. The code was compiled with version 4.6.3 of the gnu C compiler.

The tolerance in the iterative solver was set to 10^{-5}, i.e. we iterate until the residual has been reduced by at least five orders of magnitude. The number of vertical levels was set to $n_r = 128$, which is typical for current meteorological applications. We note, however, that all runtimes are directly proportional to n_r.

Prepared using fldauth.cls
DOI: 10.1002/fld

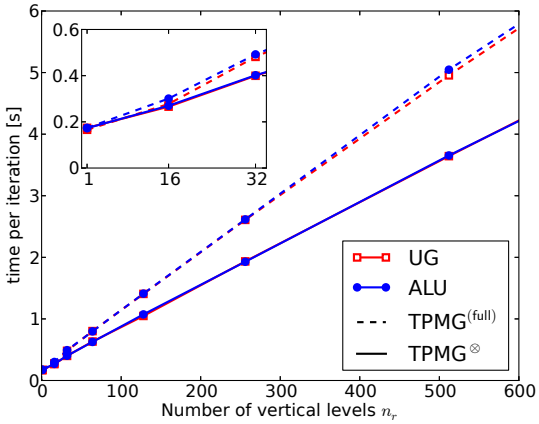

Figure 1. Time per iteration for different numbers n_r of vertical levels for two grid implementations (UGGrid in red, open squares and ALUGrid in blue, filled circles) on an icosahedral grid; results for shown both for the $\mathrm{TPMG}^{(\mathrm{full})}$ (dashed lines) and $\mathrm{TPMG}^{\otimes}$ (solid lines) preconditioner.

5.1. Overhead from indirect addressing

While data in one vertical column is stored consecutively in memory and can be addressed directly, in general indirect addressing has to be used in the horizontal directions. However, as the horizontal lookup is only required once per column, the relative penalty for this will be very small provided n_r is large enough. As discussed in [63], in this case the overhead from indirect addressing can be "hidden" behind work in the vertical direction. To verify this we ran our solver with two different DUNE grid implementations and measured the time per iteration for different numbers of vertical levels. We expect this time to depend on n_r as follows

$$t_{\mathrm{iter}}(\mathrm{grid}, n_r) = (C_0 + C_{\mathrm{grid}}) + q \cdot n_r \tag{34}$$

where C_{grid} is the overhead of indirect addressing and depends on the grid implementation. The constant C_0 encapsulates any other work which is only done once per column and both C_0 and the slope q are independent of the horizontal grid. Figure 1 shows the results for the ALUGrid and UGGrid implementation and confirms the linear dependency in (34). As can be seen from this plot, for both preconditioners $\mathrm{TPMG}^{(\mathrm{full})}$ and $\mathrm{TPMG}^{\otimes}$ the overhead from indirect addressing C_{grid} and the additional overhead C_0 together are at the order of less than 20% as soon as $n_r \gtrsim 100$. Incidentally both DUNE grid implementations that we tested are equally efficient. We stress that in both grids data in adjacent vertical columns is not necessarily stored consecutively in memory. Not surprisingly, the slope q is larger for the more expensive $\mathrm{TPMG}^{(\mathrm{full})}$ preconditioner. The results in this section also confirm that performance tests carried out on a directly addressed horizontal grid, such as the results in [22], can be generalised to indirectly addressed grids.

5.2. Test Case I: Balanced zonal flow

We first test our solver with the profiles from a simplified meteorological test problem which corresponds to a balanced atmosphere with constant buoyancy frequency and zonal flow with one jet in each hemisphere. The advantage of this test case is that the deviation of the atmospheric profiles from a perfect factorisation can be controlled by varying a single parameter. In [66] it is shown that under the assumption that the velocity field points in the longitudinal direction and the buoyancy frequency N defined in (14) is constant, a solution of the Euler equations is given by

$$\begin{aligned} \pi(\hat{\boldsymbol{r}}, r) &= \frac{\epsilon + E^{\mathcal{S}}(\phi)E^r(r)}{1+\epsilon}, & \theta(\hat{\boldsymbol{r}}, r) &= T_0 \left(E^{\mathcal{S}}(\phi)E^r(r)\right)^{-1}, \\ \rho(\hat{\boldsymbol{r}}, r) &= \frac{p_0}{R_d T_0} \left[\pi(\hat{\boldsymbol{r}}, r)\right]^{\gamma} E^{\mathcal{S}}(\phi)E^r(r), & u(\hat{\boldsymbol{r}}, r) &= u_{\mathcal{S}}(\phi) \end{aligned} \tag{35}$$

DOI: 10.1002/fld

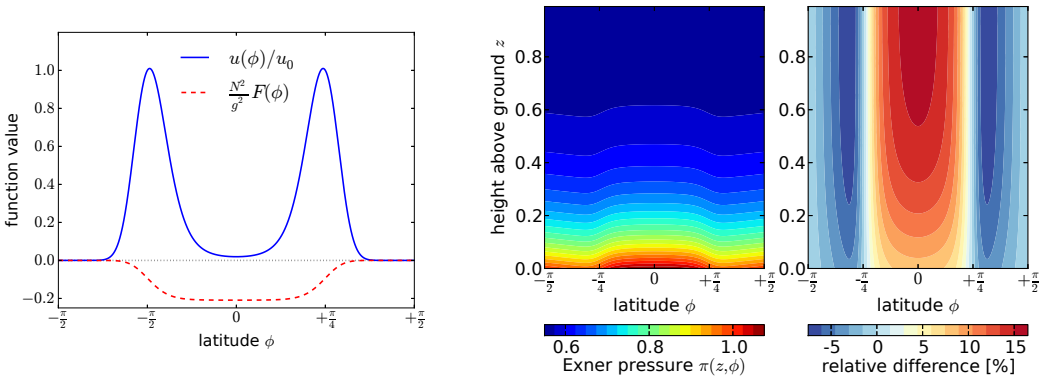

Figure 2. Left: Velocity field $u(\phi)$ and jet function F defined in eqns. (36) and (37) for $N = 0.028s^{-1}$ ($\epsilon = 1.23$). Right: Exner pressure π and relative difference $\frac{\pi^{\otimes}-\pi}{\pi}$ in the (ϕ, z)-plane for the same value of N. The height above ground is measured in units of the depth of the atmosphere.

where the functions $E^{\mathcal{S}}(\phi)$ and $E^{r}(r)$ are defined as

$$E^{\mathcal{S}}(\phi) = \exp\left[-\frac{N^2}{g^2}F(\phi)\right], \qquad E^{r}(r) = \exp\left[-\frac{N^2 R_{\text{earth}}}{g}(r-1)\right].$$

In the horizontal direction the profiles only vary in the latitudinal direction $\phi \in [-\pi/2, \pi/2]$. The parameter ϵ is related to the buoyancy frequency by $\epsilon \equiv \left(\frac{N}{N^*}\right)^2 - 1$ with $N^* \equiv \frac{\sqrt{c_p T_0}}{g} = \frac{c_h}{g}$. The function $F(\phi)$ is related to the velocity field $u_{\mathcal{S}}(\phi)$ as

$$\frac{dF(\phi)}{d\phi} = 2R_{\text{earth}}\Omega_{\text{earth}}u_{\mathcal{S}}(\phi)\sin\phi + u_{\mathcal{S}}(\phi)^2\tan\phi \tag{36}$$

with angular velocity $\Omega_{\text{earth}} = 2\pi/(24 \cdot 3600)s^{-1}$. For our numerical experiments we choose the velocity such that it corresponds to two jets with peak velocity $u_0 = 100ms^{-1}$ in the mid latitudes ($\phi_M = \pi/4$, $\sigma = 0.1$):

$$u_{\mathcal{S}}(\phi) = u_0\frac{\cos\phi}{\cos\phi_M}\exp\left[-\frac{(\cos\phi - \cos\phi_M)^2}{2\sigma^2}\right] \tag{37}$$

as plotted together with the corresponding $F(\phi)$ in Figure 2. If we fix the reference pressure and temperature to physically realistic values $p_0 = 10,000Pa$ and $T_0 = 273K$, the only free parameter in (35) is the buoyancy frequency. In particular if N is identical to N^*, i.e. $\epsilon = 0$, the first term in the expression for the Exner pressure in (35) vanishes and all profiles factorise exactly.

In the following we present numerical results for a range of buoyancy frequencies between $N = N^* = 0.01873s^{-1}$ and $N = 0.028s^{-1}$. As a preconditioner we use both a multigrid algorithm with the full model operator and the tensor-product multigrid algorithm with an approximate factorisation of the Exner pressure

$$\pi^{\otimes}(\hat{\boldsymbol{r}}, r) = \pi^{\mathcal{S}}(\hat{\boldsymbol{r}})\pi^{r}(r) \equiv \frac{\epsilon + E^{r}(r)}{1+\epsilon} \cdot E^{\mathcal{S}}(\phi) \tag{38}$$

which reduces to the expression in (35) for $\epsilon = 0$. Both the Exner pressure and the relative difference $\frac{\pi^{\otimes}-\pi}{\pi}$, which is an indicator of the quality of the factorisation, are plotted for $N = 0.028s^{-1}$ in the (z, ϕ) plane in Figure 2. As can be seen from this figure, the relative difference between the profiles can be larger than 15%.

The time per iteration is shown in Figure 3 (left) for two grid implementations. Both a preconditioned Richardson iteration and BiCGStab are used with one multigrid V-cycle as a preconditioner. It is important to note that BiCGStab requires two applications of the preconditioner

DOI: 10.1002/fld

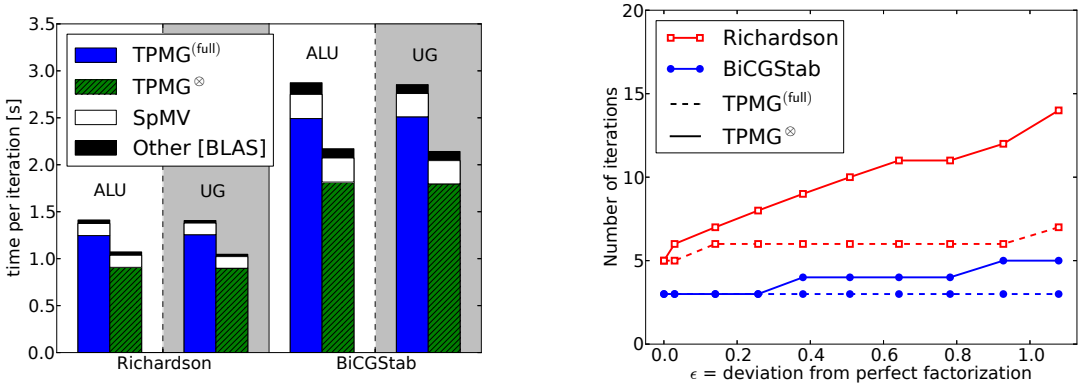

Figure 3. Breakdown of the time per iteration for two different iterative solvers and grid implementations (left) and number of iterations (right) to reduce the relative residual by at least five orders of magnitude for the idealised balanced flow testcase. The multigrid preconditioner is used with both the full, non-factorising profiles ($\mathrm{TPMG}^{(\mathrm{full})}$, blue columns and dashed curves) and the approximate factorisation ($\mathrm{TPMG}^{\otimes}$, hatched green columns and solid curves) in (38). A matrix-free implementation was used for all parts of the algorithm; for the SpMV operation we always used the non-factorising code. In all cases a problem with $n_r = 128$ and $2.6 \cdot 10^6$ total degrees of freedom was solved sequentially on HECToR.

and two sparse matrix-vector products per iteration, while the Richardson iteration only requires one of each, and not surprisingly the figure demonstrates that most of the time is taken up by the multigrid preconditioner in all cases. The number of iterations for each of the combinations is plotted in Figure 3 (right) for a range of ϵ.

First of all we note the almost perfect robustness of the full preconditioner $\mathrm{TPMG}^{(\mathrm{full})}$ for this test problem where the profiles strongly deviate from the factorising case, but the convergence of preconditioned Richardson iteration and preconditioned BiCGStab are essentially not affected. The practically observed convergence rate for the V-cycle (in the Richardson iteration) is around $\rho_A = 0.1$. This confirms the theoretical results in Sections 3.2.1 and 3.2.2. BiCGStab converges in approximately half the number of iterations than Richardson, as expected. In terms of time per iteration, the multigrid preconditioner with factorised profiles ($\mathrm{TPMG}^{\otimes}$) can be up to 25% faster than the algorithm with non-factorising profiles ($\mathrm{TPMG}^{(\mathrm{full})}$). However, this comes at the expense of an increase in the number of iterations for larger values of ϵ that can be seen in Figure 3 (right). While for the Richardson iteration the increase is almost threefold if $\mathrm{TPMG}^{\otimes}$ is used, this is much less dramatic for BiCGStab where $\mathrm{TPMG}^{\otimes}$ only requires twice as many iterations as $\mathrm{TPMG}^{(\mathrm{full})}$ for the largest ϵ.

Finally, the total solution time is shown in Figure 4. As expected, the total solution time for solvers with $\mathrm{TPMG}^{\otimes}$ preconditioner grows as ϵ increases. However, as the time per iteration is about 25% smaller for this preconditioner, for small ϵ the total solution time is also reduced by a similar factor. The most robust solver appears to be BiCGStab, which gives the best overall performance for large ϵ, even with the factorising preconditioner $\mathrm{TPMG}^{\otimes}$.

5.3. Performance analysis

To analyse the gains that can be obtained with the factorising implementation, we counted the number of floating point operations and memory references in our code both for the factorising- and the non-factorising implementation. The relevant numbers for carrying out one sparse matrix-vector product (SpMV) and one cycle of the multigrid algorithm are given in Table I. We always assume perfect caching for the vectors, i.e. each entry is only read/written once per iteration over the grid, any memory bandwidth derived from this number is known as "useful bandwidth" and should be interpreted as a conservative lower bound. We also assume that in the factorising implementation the cost of reading the matrix from memory can be neglected. As can be seen from this table, the number of memory references is reduced by almost a factor two in the factorising implementation,

DOI: 10.1002/fld

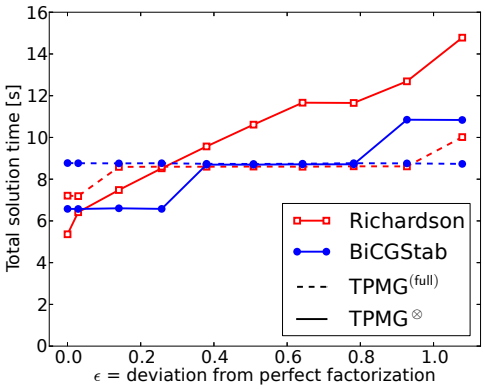

Figure 4. Total time required to reduce the relative residual by at least five orders of magnitude for the idealised balanced flow testcase. The multigrid preconditioner is used with both the full, non-factorising profiles ($\mathrm{TPMG}^{(\mathrm{full})}$, dashed curves) and the approximate factorisation ($\mathrm{TPMG}^{\otimes}$, solid curves) in (38). In all cases a problem with $n_r = 128$ and $2.6 \cdot 10^6$ total degrees of freedom was solved sequentially on one node of the HECToR supercomputer.

Table I. Number of FLOPs and memory references per cell for the sparse matrix-vector product (SpMV) and one multigrid V-cycle. The arithmetic intensity (number of FLOPs per memory reference) is shown in the last row.

	SpMV		Multigrid	
	factorising	non-factorising	$\mathrm{TPMG}^{\otimes}$	$\mathrm{TPMG}^{(\mathrm{full})}$
# FLOPs / cell	25.00	17.00	301.33	205.33
# memory references / cell	2.00	9.00	72.00	136.00
Arithmetic intensity	12.50	1.89	4.19	1.51

Table II. Measured floating point performance and useful memory bandwidth for the sparse matrix-vector product (SpMV) and one multigrid V-cycle

	SpMV		Multigrid	
	factorising	non-factorising	$\mathrm{TPMG}^{\otimes}$	$\mathrm{TPMG}^{(\mathrm{full})}$
Performance [GFLOPs/s]	0.94	0.37	0.88	0.43
Useful memory Bandwidth [GB/s]	0.60	1.57	1.68	2.28

while the arithmetic intensity is significantly increased ($6.6\times$ for SpMV and $2.8\times$ for the multigrid V-cycle). Note, however, that we assume perfect caching, so in reality (and as suggested by our measurements) the arithmetic intensity will be lower. On modern processors typically $\mathcal{O}(10)$ floating point operations can be carried out in the time it takes to read one double precision number from memory. We hence believe that the factorising implementation is still not arithmetically intensive enough to be classified as compute bound.

Using those numbers we then quantified the floating point performance and useful memory bandwidth in Table II by dividing by the measured times in Figure 3. Not very surprisingly the floating point performance is more than doubled for the factorising implementation. The useful bandwidth is about 1GB/s which indicates that our assumption of perfect caching is probably too simplistic since the theoretical peak bandwidth is $\mathcal{O}(10\,\mathrm{GB/s})$.

DOI: 10.1002/fld

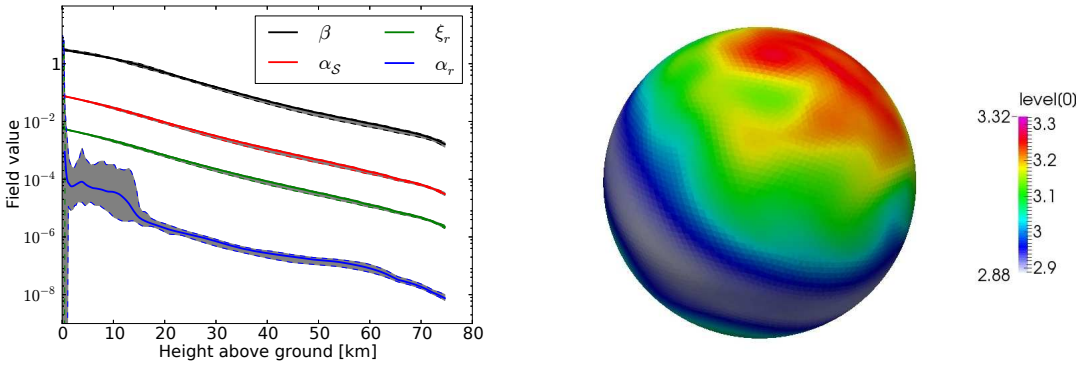

Figure 5. Left: Dependency of the horizontal average of different profiles with height (solid lines, from top to bottom: β, $\alpha_{\mathcal{S}}$, ξ_r and α_r). The horizontal variation is also represented by gray bands between the minimum and maximum value on each grid level (dashed curves). Right: Zero-order term $\beta = \gamma\rho/\pi$ on the lowest grid level. The horizontal variation in the field is at the order of 10%.

5.4. Test Case II: Aquaplanet

While the runs in the previous section were carried out under idealised and not necessarily realistic conditions, we also tested our solver for profiles obtained from common meteorological test cases. We first obtained the profiles π, θ and ρ from an aquaplanet run of the Met Office Unified Model[‡]. While these fields contain significantly more variation than the idealised profiles described in Section 5.2 and also describe phenomena such as convection near the ground and baroclinic instabilities, they are largely "well behaved" in the sense that most of them can be factorised approximately into a horizontal and a vertical variation. To quantify this further, we plot for each of the profiles the average, minimum and maximum over the horizontal grid on each vertical level in Figure 5 (right). For most profiles the horizontal variation is small and the average value decays exponentially with height; see for example Figure 5 (left), which shows the profile β on the lowest grid level. The only exception is α_r which shows significant horizontal variation in the lower atmosphere. This is mainly due to the fact that, as can be seen from the explicit expressions in (18), this profile contains the buoyancy frequency and hence vertical derivatives of the potential temperature, which can vary significantly from column to column due to convection in the lower atmosphere.

We found that for these more typical profiles the factorising preconditioner $\mathrm{TPMG}^{\otimes}$ causes both solvers to diverge. An easy fix for this is to factorise all profiles except α_r. We denote the resulting preconditioner with partial factorisation, where we keep the full non-factorising profile for α_r, as $\mathrm{TPMG}^{(\mathrm{partial})}$. As Table III demonstrates, this increases the time per iteration by just over 5% relative to the fully factorising case ($\mathrm{TPMG}^{\otimes}$), but it is still significantly smaller than in the non-factorised case ($\mathrm{TPMG}^{(\mathrm{full})}$).

The numbers of iterations and total solution times are shown in Table IV. We find that again the tensor product multigrid method converges extremely fast and robustly even though the profiles do not factorise, needing no more than 5 to 7 V-cycles to reduce the residual by 5 orders of magnitude. In this case the problem is solved fastest with the BiCGStab solver and the $\mathrm{TPMG}^{(\mathrm{partial})}$ preconditioner. In total we find that, as in the idealised test case with small ϵ, the (partially) factorised multigrid preconditioner can again lead to performance gains. As outlined in the introduction, these gains may be more significant on novel manycore architectures, such as GPUs, where the cost of memory references relative to one floating point operation is even larger.

[‡]For technical reasons we used the wet density, such that the equation of state is not satisfied, but this should not have a significant impact on our conclusions.

DOI: 10.1002/fld

Table III. Time per iteration and speedups relative to $\mathrm{TPMG}^{(\mathrm{full})}$ for different solvers and preconditioners. In all cases a problem with $n_r = 128$ and $2.6 \cdot 10^6$ total degrees of freedom was solved sequentially on one node of HECToR using the ALUGrid implementation.

Solver	$\mathrm{TPMG}^{(\mathrm{full})}$		$\mathrm{TPMG}^{(\mathrm{partial})}$		$\mathrm{TPMG}^{\otimes}$	
	t_{iter}	speedup	t_{iter}	speedup	t_{iter}	speedup
Richardson	1.43	—	1.11	$1.29\times$	1.05	$1.36\times$
BiCGStab	2.88	—	2.26	$1.27\times$	2.14	$1.35\times$

Table IV. Performance of different solvers for an aquaplanet run. A problem with $n_r = 128$ and $2.6 \cdot 10^6$ total degrees of freedom was solved sequentially on HECToR using the ALUGrid implementation.

	# iterations ($\|\|r\|\|/\|\|r_0\|\|$)		total time	
Solver	$\mathrm{TPMG}^{(\mathrm{full})}$	$\mathrm{TPMG}^{(\mathrm{partial})}$	$\mathrm{TPMG}^{(\mathrm{full})}$	$\mathrm{TPMG}^{(\mathrm{partial})}$
Richardson	5 ($9.1 \cdot 10^{-6}$)	7 ($4.8 \cdot 10^{-6}$)	7.94	7.28
BiCGStab	3 ($5.1 \cdot 10^{-7}$)	3 ($5.2 \cdot 10^{-6}$)	8.81	6.94

Preconditioner	Courant number	n_{iter}	t_{iter}	t_{total}
Multigrid	≈ 2	2	1.99	4.10
	≈ 4	3	2.01	6.17
	≈ 8	3	2.00	6.12
Single level	≈ 2	7	1.28	9.07
	≈ 4	12	1.33	16.08
	≈ 8	—	—	—

Table V. Efficiency and performance of a single-level smoother and the multigrid-preconditioner for different values of the Courant number; for the largest Courant number the single-level preconditioner did not converge. All runs were carried out on a single core of a Intel E8400 3.0GHz processor.

5.5. Comparison to single-level method

We also compared the efficiency of our multigrid-preconditioner with a single-level method. For this we use the same aquaplanet setup as in the previous section, but in the BiCGStab solver we replace the multigrid preconditioner by 4 iterations of the vertical line smoother (recall that we also use 4 smoothing steps in the multigrid algorithm). In both preconditioners considered in this section we do not factorise the profiles, i.e. we use the $\mathrm{TPMG}^{(\mathrm{full})}$ setup in the multigrid iteration. In Tab. V the number of iterations, time per iteration and the total solution times are shown for different values of the Courant-number $c_s \Delta t / \Delta x$. Note that for all other numerical experiments in this paper we use a Courant-number of around 8, i.e. the largest value in the table for which the single-level method does not converge. For smaller Courant numbers the multigrid preconditioner is still at least twice as fast as the single level method.

5.6. Parallel scaling tests

In addition to studying the sequential performance of the solvers, and in particular ensuring that they are algorithmically efficient, it is crucial to guarantee their parallel scalability on large computer clusters. For this we carried out scaling tests of our solvers for the balanced flow testcase described in Section 5.2 with $\epsilon = 0.14$; in contrast to the previous runs we always used 7 multigrid levels so that on the coarsest level each processor stores one vertical column of data. In Figure 6 (left) the

Prepared using fldauth.cls
DOI: 10.1002/fld

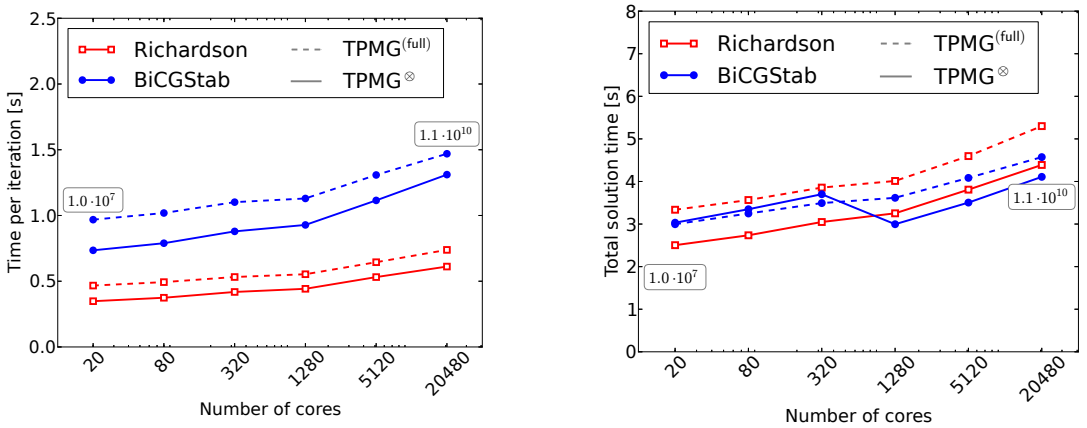

Figure 6. Weak scaling of the time per iteration (left) and total solution time (right) on the HECToR supercomputer. The number of degrees of freedom varies from 10 million to 11 billion.

weak scaling of the time per iteration on the HECToR supercomputer is shown for up to 20,480 cores, the largest problem that was solved has just over 10^{10} degrees of freedom. Each node of HECToR consists of two 16 core AMD Opteron 2.3GHz Interlagos processors processors; for the runs we always requested all cores per node (`#PBS -l mppnppn=32`). We find that the number of iterations does not increase with the core count, and even drops in some cases. The Richardson solver requires seven iterations to reduce the residual by five orders of magnitude for both preconditioners, whereas BiCGStab requires 4 ($\mathrm{TPMG}^{\otimes}$) and 3 ($\mathrm{TPMG}^{(\mathrm{full})}$) iterations for the same residual reduction. Consequently the total solution time in Figure 6 (right) shows the same excellent weak scaling.

6. CONCLUSION

In this work we discussed several multigrid preconditioners for anisotropic problems in flow simulations in "flat" domains with high aspect ratio. The algorithms are based on the tensor-product multigrid approach proposed and analysed for two-dimensional problems with separable coefficients in [38]. We extended the method and its analysis to three dimensional problems and via a perturbation argument also to non-separable coefficients. We demonstrated the excellent performance of tensor-product multigrid for two model PDEs arising in semi-implicit semi-Lagrangian time stepping in atmospheric modelling. The numerical tests confirm the theoretically predicted optimality and effectivity of the method. The practically observed convergence rates are around $\rho_A = 0.1$. The tests also show that under certain conditions a preconditioner based on an approximate factorisation of the atmospheric profiles can reduce the total solution time. We found this to be the case both for an idealised flow scenario and for a more realistic aquaplanet test case. We also demonstrated the excellent weak parallel scaling on up to 20,480 cores of the HECToR supercomputer. Overall our work demonstrates that bespoke multigrid preconditioners are highly efficient for solving the pressure correction equation encountered in NWP models.

There are several ways to further improve this work: so far all tests have been carried out without any orography. It is known that steep gradients can lead to deteriorating performance of the non-linear iteration and we plan to study this by looking at the full non-linear solve for more realistic model problems. For simplicity we used a finite volume discretisation, but more advanced approaches such as higher-order mixed finite elements can also be used in this framework. This will require the solution of a suitable pressure correction equation in higher order FEM spaces. The parallel performance can also be further improved by, for example, overlapping calculations and communications and strong scaling tests should also be carried out. Finally, the performance gains from approximate factorisations of the matrix are expected to be significantly higher on GPU systems and hence on such architectures its use may be more justified and more efficient for a wider class of profiles.

DOI: 10.1002/fld

ACKNOWLEDGEMENTS

This work was funded as part of the NERC project on Next Generation Weather and Climate Prediction (NGWCP), grant numbers NE/J005576/1 and NE/K006754/1. We gratefully acknowledge input from discussions with our collaborators in the Met Office Dynamics Research group and the GungHo! project, in particular Tom Allen, Terry Davies, Markus Gross and Nigel Wood. Ian Boutle kindly provided the aquaplanet UM output used in Section 5.4. We would like to thank all DUNE developers and in particular Oliver Sander for his help with extending the parallel scalability of the UG grid implementation. This work made use of the facilities of HECToR, the UK's national high-performance computing service, which is provided by UoE HPCx Ltd at the University of Edinburgh, Cray Inc and NAG Ltd, and funded by the Office of Science and Technology through EPSRC's High End Computing Programme.

A. INTERGRID OPERATORS

In this appendix we describe the prolongation and restriction operations in our multigrid algorithm. Recall that the grids we use are polygonal and not curvilinear, and care has to be taken when prolongating fields to the next-finer grid level. The schemes described in the following work on grids with an arbitrary number n_{Nb} of neighbours and any number n_{child} of fine grid cells per coarse grid cell, in particular for the icosahedral grids used in this paper. We also successfully tested our code on cubed sphere grids.

A.1. Prolongation: Linear interpolation scheme

To prolongate the fields on one multigrid level to the next finer level we use the following linear interpolation.

For a given coarse grid cell on the horizontal host grid, let $\boldsymbol{x}_0^{(c)}$ be the coordinate of its centre and $\boldsymbol{x}_i^{(c)}$, with $i = 1, \ldots, n_{\text{Nb}}$, the coordinates of the centres of its neighbours. Furthermore, let $u_0^{(c)}$ be the field value associated with the coarse grid cell and let $u_i^{(c)}$ be the values of the field at the centres of the neighbouring cells. Similarly for each of the fine children cells which comprise the coarse cell, let $\boldsymbol{x}_j^{(f)}$, $j = 1, \ldots, n_{\text{child}}$, be the coordinates of their centres. To calculate the corresponding values $u_j^{(f)}$ on the fine grid we proceed as follows:

Let Π be the two-dimensional plane defined by the (flat) coarse grid cell. Denote with

$$P(\boldsymbol{x}) \equiv \frac{\left(\boldsymbol{x} - \boldsymbol{x}_0^{(c)}\right)_\perp}{\left|\left|\left(\boldsymbol{x} - \boldsymbol{x}_0^{(c)}\right)_\perp\right|\right|_2} \left|\left|\boldsymbol{x} - \boldsymbol{x}_0^{(c)}\right|\right|_2 \tag{39}$$

the orthogonal projection ($\perp$) of any point $\boldsymbol{x}$ onto Π, followed by scaling with the factor $\left|\left|\boldsymbol{x} - \boldsymbol{x}_0^{(c)}\right|\right|_2 / \left|\left|\left(\boldsymbol{x} - \boldsymbol{x}_0^{(c)}\right)_\perp\right|\right|_2$ which guarantees that $||P(\boldsymbol{x})||_2 = \left|\left|\boldsymbol{x} - \boldsymbol{x}^{(c)}\right|\right|_2$, where $||\cdot||_2$ is the Euclidean norm. Note in particular that $P(\boldsymbol{x}_0^{(c)}) = 0$ and that $P(\boldsymbol{x}) \approx \left(\boldsymbol{x} - \boldsymbol{x}_0^{(c)}\right)_\perp$ on the finer levels as $h \to 0$ and the curvature decreases. For each fine grid cell j, let $\boldsymbol{x}_{a_1}^{(c)}$, $\boldsymbol{x}_{a_2}^{(c)}$, $a_1, a_2 \in \{1, \ldots, n_{\text{Nb}}\}$ be the centres $\boldsymbol{x}_a^{(c)}$ of the coarse grid neighbour cells for which $\left|\left|P(\boldsymbol{x}_j^{(f)}) - P(\boldsymbol{x}_a^{(c)})\right|\right|_2$ is minimal, see Figure 7. The fine grid value $u_j^{(f)}$ is then chosen such that the point $(P(\boldsymbol{x}_j^{(f)}), u_j^{(f)}) \in \Pi \times \mathbb{R}$ lies in the plane defined by the three vectors $(P(\boldsymbol{x}_0^{(c)}), u_0^{(c)})$, $(P(\boldsymbol{x}_{a_1}^{(c)}), u_{a_1}^{(c)})$ and $(P(\boldsymbol{x}_{a_2}^{(c)}), u_{a_2}^{(c)})$. In other words, for each fine grid cell we identify the two coarse grid neighbour cells which are closest to this cell; we then obtain the fine grid value as a linear interpolation of the data on those two coarse grid neighbour cells and the parent cell.

A.2. Restriction: Cell integral

To restrict the fields we use a simple cell summation

$$u_0^{(c)} = \sum_{j=1}^{n_{\text{child}}} u_j^{(f)}. \tag{40}$$

For vanishing curvature this operation conserves the cell integral for linear functions.

DOI: 10.1002/fld

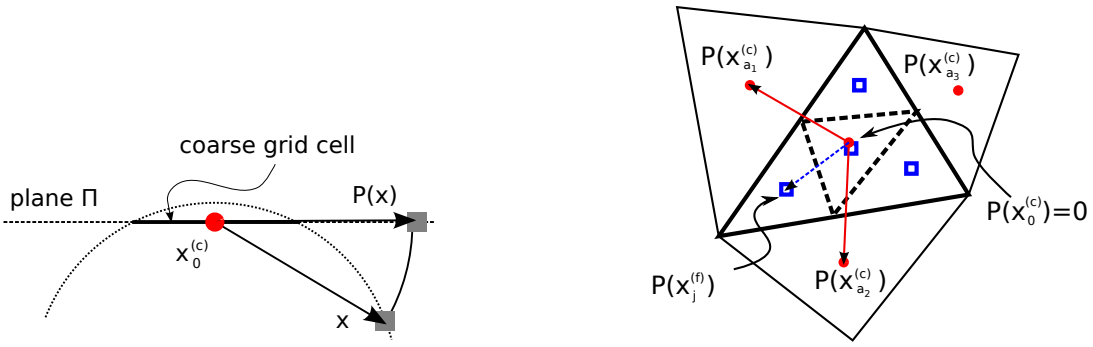

Figure 7. Projection $P(\boldsymbol{x})$ (left) and linear interpolation (right). Centres of the coarse grid cell and its neighbours are marked by solid red circles; centres of the fine grid cells are shown as empty blue squares.

REFERENCES

1. Lacroix S, Vassilevski Y, Wheeler J, Wheeler M. Iterative Solution Methods for Modeling Multiphase Flow in Porous Media Fully Implicitly. *SIAM J. Sci. Comput.* 2003; **25**(3):905–926, doi:10.1137/S106482750240443X.
2. Marshall J, Adcroft A, Hill C, Perelman L, Heisey C. A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *J. Geophys. Res.* 1997; **102**:5753–5766, doi:10.1029/96JC02775.
3. Fringer OB, Gerritsen M. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. *Ocean Modelling* 2006; **14**:139–173, doi:10.1016/j.ocemod.2006.03.006.
4. Davies T, Cullen MJP, Malcolm AJ, Mawson MH, Staniforth A, White AA, Wood N. A new dynamical core for the Met Office's global and regional modelling of the atmosphere. *Q. J. Roy. Meteor. Soc.* 2005; **131**(608):1759–1782, doi:10.1256/qj.04.101.
5. Wood N, Staniforth A, White A, Allen T, Diamantakis M, Gross M, Melvin T, Smith C, Vosper S, Zerroukat M, *et al.*. An inherently mass-conserving semi-implicit semi-Lagrangian discretisation of the deep-atmosphere global nonhydrostatic equations. *Q. J. Roy. Meteor. Soc.* 2013; doi:10.1002/qj.2235. Published online December 4th 2013.
6. Kwizak M, Robert AJ. A semi-implicit scheme for grid point atmospheric models of the primitive equations. *Mon. Weather Rev.* 1971; **99**:32–36, doi:10.1175/1520-0493(1971)099<0032:ASSFGP>2.3.CO;2.
7. Robert A. A stable numerical integration scheme for the primitive meteorological equations. *Atmos. Ocean* 1981; **19**(1):35–46, doi:10.1080/07055900.1981.9649098.
8. Saad Y. *Iterative Methods for Sparse Linear Systems, Second Edition*. SIAM: Philadelphia, 2003.
9. Trottenberg U, Oosterlee CW, Schüller A. *Multigrid*. Academic Press, 2001.
10. Barros SR. *Optimierte Mehrgitterverfahren für zwei- und dreidimensionale elliptische Randwertaufgaben in Kugelkoordinaten*. Berichte der Gesellschaft für Mathematik und Datenverarbeitung, Oldenbourg Verlag, 1989.
11. Bates JR, Semazzi FHM, Higgins RW, Barros SRM. Integration of the shallow-water equations on the sphere using a vector semi-lagrangian scheme with a multigrid solver. *Mon. Weather Rev.* 1990; **118**(8):1615–1627, doi:10.1175/1520-0493(1990)118<1615:IOTSWE>2.0.CO;2.
12. Bowman KP, Huang J. A multigrid solver for the Helmholtz-equation on a semiregular grid on the sphere. *Mon. Weather Rev.* 1991; **119**(3):769–775, doi:10.1175/1520-0493(1991)119<0769:AMSFTH>2.0.CO;2.
13. Adams JC. MUDPACK – Multigrid portable Fortran software for the efficient solution of linear elliptic partial differential equations. *Appl. Math. Comput.* 1989; **34**(2):113–146, doi:10.1016/0096-3003(89)90010-6.
14. Adams JC. MUltigriD software for elliptic partial differential equations: MUDPACK. *Technical Report NCAR/TN-357+STR*, National Center for Atmospheric Research 1991.
15. Adams JC, Garcia R, Gross B, Hack J, Haidvogel D, Pizzo V. Applications of multigrid software in the atmospheric sciences. *Mon. Weather Rev.* 1992; **120**:1447–1458, doi:10.1175/1520-0493(1992)120<1447:AOMSIT>2.0.CO;2.
16. Skamarock WC, Smolarkiewicz PK, Klemp JB. Preconditioned conjugate-residual solvers for Helmholtz equations in nonhydrostatic models. *Mon. Weather Rev.* 1997; **125**(4):587–599, doi:10.1175/1520-0493(1997)125<0587:PCRSFH>2.0.CO;2.
17. Thomas SJ, Malevsky AV, Desgagn M, Benoit R, Pellerin P, Valin M. Massively parallel implementation of the mesoscale compressible community model. *Parallel Comput.* 1997; **23**:2143–2160, doi:10.1016/S0167-8191(97)00105-1.
18. Hess R, Joppich W. A comparison of parallel multigrid and a fast Fourier transform algorithm for the solution of the Helmholtz equation in numerical weather prediction. *Parallel Comput.* 1997; **22**, doi:10.1016/S0167-8191(96)00058-0.
19. Qaddouri A, Côté J. Preconditioning for an iterative elliptic solver on a vector processor. *High Performance Computing for Computational Science (VECPAR 2002), Lecture Notes in Computer Science*, vol. 2565, Palma J, Sousa A, Dongarra J, Hernndez V (eds.), Springer, Berlin, 2003; 451–455, doi:10.1007/3-540-36569-9_22.
20. Buckeridge SD, Scheichl R. Parallel geometric multigrid for global weather prediction. *Numer. Linear Algebr.* 2010; **17**(2-3):325–342, doi:10.1002/nla.699.
21. Buckeridge SD, Cullen MJP, Scheichl R, Wlasak M. A robust numerical method for the potential vorticity based control variable transform in variational data assimilation. *Q. J. Roy. Meteor. Soc.* 2011; **137**(657):1083–1094.

22. Müller EH, Scheichl R. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. *Quarterly Journal of the Royal Meteorological Society* 2014; **140**(685):2608–2624, doi: 10.1002/qj.2327.
23. Brandt A, McCormick SF, Ruge JW. Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity And Its Applications*, Evans DJ (ed.), 1984; 258–283.
24. Stüben K. *Algebraic multigrid (AMG). An introduction with applications*. Fraunhofer (GMD), Sankt Augustin, Germany, 1999.
25. Blatt M. A Parallel Algebraic Multigrid Method for Elliptic Problems with Highly Discontinuous Coefficients. PhD Thesis, University of Heidelberg 2010.
26. Falgout RD, Meier-Yang U. Hypre: a Library of High Performance Preconditioners. *Lecture Notes in Computer Science*, vol. 2331, Sloot PMA, Tan CJK, Dongarra JJ, Hoekstra AG (eds.), Springer, 2002; 632–641, doi: 10.1007/3-540-47789-6_66.
27. Falgout R, Jones J, Yang U. The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners. *Numerical Solution of Partial Differential Equations on Parallel Computers, Lecture Notes in Computational Science and Engineering*, vol. 51, Bruaset A, Tveito A (eds.). Springer Berlin Heidelberg, 2006; 267–294, doi:10.1007/3-540-31619-1_8.
28. Ashby SF, Falgout RD. A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. *Nuclear Science and Engineering* 1996; **124**(1):145–159.
29. Schaffer S. A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients. *SIAM J. Sci. Comput.* 1998; **20**(1):228–242, doi:10.1137/S1064827595281587.
30. Bey J, Wittum G. Downwind numbering: Robust multigrid for convection-diffusion problems. *Appl. Numer. Math.* 1997; **23**(1):177–192, doi:10.1016/S0168-9274(96)00067-0.
31. Hackbusch W, Probst T. Downwind Gauss-Seidel smoothing for convection dominated problems. *Numer. Linear Algebr.* 1997; **4**(2):85–102, doi:10.1002/(SICI)1099-1506(199703/04)4:2<85::AID-NLA100>3.0.CO;2-2.
32. Dendy JE. Two multigrid methods for three-dimensional problems with discontinuous and anisotropic coefficients. *SIAM J. Sci. Stat. Comp.* 1987; **8**(5):673–685, doi:10.1137/0908059.
33. Oosterlee C. A GMRES-based plane smoother in multigrid to solve 3D anisotropic fluid flow problems. *J. Comput. Phys.* 1997; **130**(1):41–53, doi:http://dx.doi.org/10.1006/jcph.1996.5442.
34. Reisinger C, Wittum G. On multigrid for anisotropic equations and variational inequalities "Pricing multi-dimensional European and American options". *Comput. Visual. Sci.* 2004; **7**(3-4):189–197, doi:10.1007/s00791-004-0149-9.
35. De Zeeuw PM. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *J Comput. Appl. Math.* 1990; **33**(1):1–27, doi:10.1016/0377-0427(90)90252-U.
36. Dendy JE, Ida M, Rutledge J. A semicoarsening multigrid algorithm for SIMD machines. *SIAM J. Sci. Stat. Comp.* 1992; **13**(6):1460–1469, doi:10.1137/0913082.
37. Pflaum C. Fast and robust multilevel algorithms. *Habilitation*, Universität Würzburg 1998.
38. Börm S, Hiptmair R. Analysis of tensor product multigrid. *Numer. Algorithms* 2001; **26**:219–234, doi: 10.1023/A:1016686408271.
39. Buckeridge SD. Numerical Solution of Weather and Climate Systems. PhD Thesis, University of Bath 2010.
40. van der Vorst HA. BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* Mar 1992; **13**(2):631–644, doi:10.1137/0913035.
41. Müller E, Guo X, Scheichl R, Shi S. Matrix-free GPU implementation of a preconditioned conjugate gradient solver for anisotropic elliptic PDEs. *Computing and Visualization in Science* 2013; **16**(2):41–58, doi:10.1007/s00791-014-0223-x.
42. Müller E, Scheichl R, Vainikko E. Petascale elliptic solvers for anisotropic PDEs on GPU clusters 2015.
43. Bastian P, Blatt M, Dedner A, Engwer C, Klöfkorn R, Kornhuber R, Ohlberger M, Sander O. A generic grid interface for parallel and adaptive scientific computing. Part II: implementation and tests in DUNE. *Computing* 2008; **82**(2-3):121–138, doi:10.1007/s00607-008-0004-9.
44. Bastian P, Blatt M, Dedner A, Engwer C, Klöfkorn R, Ohlberger M, Sander O. A generic grid interface for parallel and adaptive scientific computing. Part I: abstract framework. *Computing* 2008; **82**(2-3):103–119, doi: 10.1007/s00607-008-0003-x.
45. Schupp B. Entwicklung eines effizienten Verfahrens zur Simulation kompressibler Strömungen in 3D auf Parallelrechnern. PhD Thesis, Albert-Ludwigs-Universität Freiburg Dec 1999.
46. Dedner A, Rohde C, Schupp B, Wesenberg M. A parallel, load-balanced MHD code on locally-adapted, unstructured grids in 3D. *Comput. Visual. Sci.* 2004; **7**:79–96, doi:10.1007/s00791-004-0140-5.
47. Burri A, Dedner A, Klöfkorn R, Ohlberger M. An efficient implementation of an adaptive and parallel grid in DUNE. *Proceedings of 2nd Russian-German Advanced Research Workshop on Computational Science and High Performance Computing, Stuttgart*, 2005.
48. Bastian P, Birken K, Johannsen K, Lang S, Neuss N, Rentz-Reichert H, Wieners C. UG — A flexible software toolbox for solving partial differential equations. *Comput. Visual. Sci.* 1997; **1**(1):27–40, doi: 10.1007/s007910050003.
49. Crank J, Nicolson P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Adv. Comput. Math.* 1996; **6**:207–226, doi:10.1007/BF02127704.
50. Wesseling P. *Principles of Computational Fluid Dynamics*. Lecture Notes in Computer Science, Springer, 2001.
51. Chen SH, Sun WY. Application of the multigrid method and a flexible hybrid coordinate in a nonhydrostatic model. *Mon. Weather Rev.* 2001; **129**(11):2660–2676, doi:10.1175/1520-0493(2001)129<2660:AOTMMA>2.0.CO;2.
52. Ippisch O, Blatt M. Scalability test of $\mu\varphi$ and the parallel algebraic multigrid solver of DUNE-ISTL. *Jülich Blue Gene/P Extreme Scaling Workshop 2011, Technical Report FZJ-JSC-IB-2011-02*, Mohr B, Frings W (eds.), 2011.
53. Blatt M, Ippisch O, Bastian P. A massively parallel algebraic multigrid preconditioner based on aggregation for elliptic problems with heterogeneous coefficients. *arXiv preprint arXiv:1209.0960* 2012; .

54. Baker AH, Falgout RD, Kolev TV, Yang UM. Scaling hypre's multigrid solvers to 100,000 cores. *High-Performance Scientific Computing*. Springer, 2012; 261–279.
55. Mulder WA. A new multigrid approach to convection problems. *J. Comput. Phys.* 1989; **83**(2):303–323, doi: http://dx.doi.org/10.1016/0021-9991(89)90121-6.
56. Mavriplis DJ. Directional coarsening and smoothing for anisotropic Navier-Stokes problems. *Electronic Transactions on Numerical Analysis* 1997; **6**:182–197.
57. Oosterlee CW, Gaspar FJ, Washio T, Wienands R. Multigrid line smoothers for higher order upwind discretizations of convection-dominated problems. *J. Comput. Phys.* 1998; **139**(2):274–307, doi:10.1006/jcph.1997.5854.
58. Hackbusch W. *Multi-grid Methods and Applications*. Springer Series in Computational Mathematics, Springer-Verlag, 1985.
59. Hackbusch W. *Iterative Solution of Large Sparse Systems of Equations*. Applied Mathematical Sciences, Springer, 1993.
60. Staniforth A, Thuburn J. Horizontal grids for global weather and climate prediction models: A review. *Q. J. Roy. Meteor. Soc.* 2012; **138**(662):1–26, doi:10.1002/qj.958.
61. Cotter CJ, Shipton J. Mixed finite elements for numerical weather prediction. *J. Comput. Phys.* Aug 2012; **231**(21):7076–7091, doi:10.1016/j.jcp.2012.05.020.
62. Cotter CJ, Thuburn J. A finite element exterior calculus framework for the rotating shallow-water equations. *J. Comput. Phys.* 2014; **257**:1506–1526, doi:10.1016/j.jcp.2013.10.008.
63. MacDonald J A Eand Middlecoff, Henderson T, Lee JL. A general method for modeling on irregular grids. *Int. J. High. Perform. C.* 2011; **25**(4):392–403, doi:10.1177/1094342010385019.
64. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes: The Art of Scientific Computing, 3rd Edition*. Cambridge University Press: New York, 2007.
65. Sadourny R. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Mon. Weather Rev.* 1972; **100**(2):136–144, doi:10.1175/1520-0493(1972)100<0136:CFAOTP>2.3.CO;2.
66. Davies T. Balanced Flows Nov 2005. Technical Report (internal).