Multigrid preconditioners for the mixed finite element dynamical core of the LFRic atmospheric model

Received: 31 January 2020 | Revised: 24 June 2020 | Accepted: 19 July 2020 | Published on: 6 October 2020

DOI: 10.1002/qj.3880

Quarterly Journal of the Royal Meteorological Society RMetS

RESEARCH ARTICLE

Multigrid preconditioners for the mixed finite element dynamical core of the LFRic atmospheric model

Christopher Maynard[1,2] | Thomas Melvin[1] | Eike Hermann Müller[3]

[1]Met Office, Exeter, UK

[2]Department of Computer Science, University of Reading, UK

[3]Department of Mathematical Sciences, University of Bath, UK

Correspondence
Eike H. Müller, Department of Mathematical Sciences, University of Bath, Bath, BA2 7AY, UK.
Email: e.mueller@bath.ac.uk

Funding information
NERC, Grant/Award Number: NE/K006762/1; Bath Institute for Mathematical Innovation

Abstract

Due to the wide separation of time-scales in geophysical fluid dynamics, semi-implicit time integrators are commonly used in operational atmospheric forecast models. They guarantee the stable treatment of fast (acoustic and gravity) waves, while not suffering from severe restrictions on the time-step size. To propagate the state of the atmosphere forward in time, a nonlinear equation for the prognostic variables has to be solved at every time step. Since the nonlinearity is typically weak, this is done with a small number of Newton or Picard iterations, which in turn require the efficient solution of a large system of linear equations with $\mathcal{O}(10^6 - 10^9)$ unknowns. This linear solve is often the computationally most costly part of the model. In this article an efficient linear solver for the LFRic next-generation model currently being developed by the Met Office is described. The model uses an advanced mimetic finite element discretisation which makes the construction of efficient solvers challenging as compared to models using standard finite-difference and finite-volume methods. The linear solver hinges on a bespoke multigrid preconditioner of the Schur-complement system for the pressure correction. By comparing it to Krylov subspace methods, the superior performance and robustness of the multigrid algorithm is demonstrated for standard test cases and realistic model set-ups. In production mode, the model will have to run in parallel on hundreds of thousands of processing elements. As confirmed by numerical experiments, one particular advantage of the multigrid solver is its excellent parallel scalability due to its avoidance of expensive global reduction operations.

KEYWORDS

atmosphere, dynamics, finite element discretisation, linear solvers, multigrid, numerical methods, numerical weather prediction, parallel scalability

1 | INTRODUCTION

Operational models for numerical climate and weather prediction must solve the equations of fluid dynamics in a very short space of time. State-of-the art implementations rely on accurate spatial discretisations and efficient time-stepping algorithms. To make efficient use of modern supercomputers they have to exploit many levels of parallelism (such as SIMD/SIMT, threading on a single node and message-passing on distributed memory systems) and scale to hundreds of thousands of processing elements. Semi-implicit time integrators are commonly employed since they allow the stable treatment of fast acoustic waves, which carry very little energy but have to be included in a fully compressible formulation. The implicit treatment of the acoustic modes allows the model to be run with a relatively large time step. The size of the time step is only restricted by advection, which is generally around one order of magnitude slower horizontally than acoustic oscillations and two orders of magnitude slower vertically. The main computational cost of semi-implicit models is the repeated solution of a very large sparse system of linear equations. While standard iterative solution algorithms exist, the linear system is ill-conditioned, which leads to the very slow convergence of Krylov subspace methods. This is a particularly serious problem for massively parallel implementations due to their very large number of global reduction operations (arising from vector dot products and norms in each Krylov iteration). Preconditioners, which solve an approximate version of the linear system, overcome this issue and dramatically reduce the number of Krylov iterations and global communications. The construction of an efficient preconditioner is non-trivial and requires careful exploitation of the specific properties of the system to be solved. For global atmospheric models, two key features that have to be taken into account are (a) the high aspect ratio arising from the shallow domain, and (b) the finite speed of sound in compressible formulations, which limits the effective distance over which different points in the domain are correlated during one time step. Typically, the preconditioner is based on a Schur-complement approach. This reduces the problem to an elliptic equation for the pressure correction, which can then be solved with standard methods.

1.1 | Multigrid Solver

Hierarchical methods such as Multigrid solver (Trottenberg *et al.*, 2001) are often employed for the solution of elliptic systems, as they have a computational complexity which grows linearly with the number of unknowns. Müller and Scheichl (2014) contains a recent review of linear solvers in atmospheric modelling (see also Steppeler *et al.*, 2003). There, the performance of a multigrid solver based on the tensor-product algorithm described in Börm and Hiptmair (2001) was applied to a simplified model system which is representative of the linear system for the pressure correction. The key idea is to use a vertical line relaxation smoother together with semi-coarsening in the horizontal direction only. Furthermore, due to the finite speed of sound in compressible models, it is sufficient to use a relatively small number of multigrid levels of $L \approx \log_2 \mathrm{CFL_h}$, where $\mathrm{CFL_h} = c_s \Delta t / \Delta x$ is the horizontal acoustic Courant number. The much higher vertical acoustic Courant number $\mathrm{CFL_v} = c_s \Delta t / \Delta z$ does not cause any problems as vertical sound propagation is treated exactly by the line relaxation smoother. Since advective transport is about an order of magnitude slower than acoustic pressure oscillations, $\mathrm{CFL_h} \approx 10$ and $L \approx 4$ irrespective of the model resolution. As was demonstrated in Müller and Scheichl (2014) and Sandbach *et al.* (2015), this "shallow" multigrid works well, and avoids expensive global communications. It also significantly simplifies the parallel decomposition, since it is only necessary to (horizontally) partition the coarsest grid, which still has a large number of cells and allows a relatively fine-grained domain decomposition. For example, one coarse grid cell could be assigned to a node on a supercomputer, exploiting additional shared-memory parallelism on the cells of the $4^{L-1} \approx 4^3 = 64$ fine grid cells.

The tensor-product multigrid algorithm was applied to more realistic model equations in Dedner *et al.* (2016), and its performance on a cluster with 16,384 graphics processing units (GPUs) was demonstrated in Müller *et al.* (2015b).

1.2 | Solvers for finite element discretisations

One challenge of standard latitude–longitude models, which is becoming more pronounced with increasing model resolution, is the convergence of grid lines at the Poles. Due to the resulting small grid cells at high latitudes, this leads to severe time-step constrictions, slow solver convergence and poor parallel scalability due to global coupling at the Poles. To overcome this problem, there has been a push towards using different meshes which avoid this issue (see the review in Staniforth and Thuburn, 2012). However, ensuring the accurate discretisation of the continuous equations and the exact conservation of certain physical quantities on these non-orthogonal grids requires advanced discretisations. While low-order finite-volume methods (Ringler *et al.*, 2010; Thuburn and Cotter, 2012; Thuburn *et al.*, 2013) and high-order collocated spectral element methods (Fournier *et al.*, 2004;

Giraldo and Restelli, 2008) do exist, the mimetic finite element approach developed in Cotter and Shipton (2012), Cotter and Thuburn (2014) and Thuburn and Cotter (2015) for the shallow-water equations is particularly attractive since it generalises to arbitrary discretisation order, has good wave dispersion properties, avoids spurious computational modes and allows the approximate conservation of certain physical quantities in the discrete equations. At lowest order on orthogonal meshes it reduces to the extensively studied C-grid staggering in the horizontal direction.

This article builds on the work of Melvin *et al.* (2019), which describes the recently developed GungHo dynamical core employed in the LFRic model. The mimetic finite element discretisation used there is combined with a vertical discretisation (described in Natale *et al.*, 2016; Melvin *et al.*, 2018), which is similar to Charney–Phillips staggering, and a mass-conserving finite-volume advection scheme.

A particular challenge of mimetic finite element discretisations is the significantly more complex structure of the discretised linear equation system, which has to be solved repeatedly at every time step. For traditional finite-difference and finite-volume discretisations on structured grids the Schur-complement can be formed, and – provided the resulting pressure equation is solved to sufficiently high accuracy – the preconditioner is exact. However, this is not possible for the finite element discretisations considered here, since the velocity mass matrix is not (block-) diagonal. Instead, the linear system is preconditioned by constructing an approximate Schur complement using velocity mass lumping. As demonstrated for a gravity-wave system in Mitchell and Müller (2016), this method is efficient if one V-cycle of the same bespoke tensor-product multigrid algorithm is used to solve the pressure system. As shown there, the method also works for next-to-lowest-order discretisations if a p-refinement is used on the finest level of the multigrid hierarchy.

In this article it is shown how the method can be extended to solve the full equations of motion, that is, the Euler equations for a perfect gas in a rotating frame. The efficiency of the multigrid algorithm is demonstrated by alternatively solving the pressure correction equation with a Krylov subspace method. As will be shown by running on hundreds of thousands of processing cores and solving problems with more than one billion (10^9) unknowns, the multigrid also improves the parallel scalability since – in contrast to the Krylov method – the multigrid V-cycle does not require any global reductions.

1.3 | Implementation

To achieve optimal performance, an efficient implementation is required. In a continuously diversifying hardware landscape the code has to be performance portable. In general, the LFRic model uses an implementation which is based on the separation-of-concerns approach described in Adams *et al.* (2019). The composability of iterative methods and preconditioners is exploited to easily swap components of the complex hierarchical solver in an object-oriented Fortran 2003 framework (see sect. 6 in Adams *et al.*, 2019).

1.4 | Structure

This article is organised as follows. Once the research is put into context by reviewing related work in Section 2, the mixed finite element discretisation is described and the construction of a Schur-complement preconditioner for the linear system is discussed in Section 3. The properties of the elliptic pressure operator are used to construct a bespoke multigrid preconditioner. After outlining the parallel implementation in the LFRic framework in Section 4, numerical results for performance and parallel scalability are presented in Section 5. Conclusions are drawn and future work is discussed in Section 6.

2 | CONTEXT AND RELATED WORK

2.1 | Semi-implicit time-stepping methods

One of the perceived drawbacks of semi-implicit models is the additional complexity required to solve a large nonlinear problem. Iterative solvers introduce global communications, which potentially limits scalability and performance; this can become a serious issue for operational forecast systems that run on large supercomputers and have to deliver results on very tight time-scales. Nevertheless, the comprehensive review of linear solver techniques for atmospheric applications in Müller and Scheichl (2014) shows that semi-implicit models deserve serious consideration. Looking at actively developed dynamical cores that target massively parallel supercomputers, of the 11 non-hydrostatic implementations compared in the recent Dynamical Core Model Intercomparison Project (DCMIP-2016) presented in Ullrich *et al.* (2017), two are semi-implicit: the Canadian GEM finite difference code (Yeh *et al.*, 2002) and FVM (Kühnlein *et al.*, 2019), the next-generation finite-volume version of the Integrated Forecasting System (IFS) (Temperton *et al.*, 2001; Wedi *et al.*, 2015) developed at the European Centre for Medium-Range Weather Forecasts (ECMWF); the current spectral-transform model used by ECMWF is

also semi-implicit. To solve the linear equation system these models use different approaches. IFS employs a global spectral transform, which diagonalises the operator in Fourier spaces. Although this approach inherently involves expensive all-to-all communications, scalability can be improved by exploiting properties of the spherical harmonics (Wedi *et al.*, 2013). The solution of the pressure system in IFS-FVM with a generalised conjugate residual (GCR) method is described in Smolarkiewicz and Szmelter (2011); the preconditioner exactly inverts the vertical part of the operator, similarly to the line relaxation strategy which is used in the smoother for the multigrid algorithm in this work. To solve the three-dimensional elliptic boundary value problem in the GEM model, a vertical transform is used to reduce it to a set of decoupled two-dimensional problems, which are solved iteratively (see Côté *et al.*, 1998).

All of the above semi-implicit models use second-order accurate finite-difference/finite-volume discretisations or the spectral-transform approach. In contrast, actively developed massively parallel high-order spectral element codes include NUMA (Giraldo *et al.*, 2013), which uses an implicit–explicit (IMEX) time integrator, the CAM-SE/HOMME dynamical core used by Dennis *et al.* (2012) in the ACME climate model, and Tempest (Ullrich, 2014; Guerra and Ullrich, 2016). Collocating the quadrature points with nodal points in the continuous Galerkin (CG) formulation of NUMA results in a diagonal-velocity mass matrix, which allows the construction of a Schur-complement pressure system. This system is then solved with an iterative method. This is in contrast to the mixed finite element approach employed here, for which the velocity mass matrix is non-diagonal. To address this issue, the outer system is solved iteratively and preconditioned with an approximate Schur complement based on a lumped mass matrix. It should be noted, however, that the construction of an efficient linear solver for the discontinuous Galerkin (DG) version of NUMA is significantly more challenging, since the numerical flux augments the velocity mass matrix by artificial diffusion terms. Overcoming this problem is a topic of current research, and it is argued in Peraire *et al.* (2010) and Kang *et al.* (2020) that hybridisable DG methods appear to be particularly suitable. As discussed below, applying a similar hybridised approach to the mixed finite element formulation is a promising direction for future work.

While the fully implicit version of NUMA has been optimised on modern chip architectures (see Abdi *et al.*, 2019), the massively parallel scaling tests in Müller *et al.* (2015a) are reported for the horizontally explicit vertically implicit (HEVI) variant of the model, in which only the vertical couplings are treated implicitly. The same HEVI time integrator can also be used by the Tempest dynamical core. Again this vertically implicit solver is equivalent to the block-Jacobi smoother in the fully implicit multigrid algorithm and the preconditioner in Kühnlein *et al.* (2019).

The discretisation used by the semi-implicit GUSTO code developed at Imperial College London is based on Natale *et al.* (2016), Yamazaki *et al.* (2017) and Shipton *et al.* (2018), and is very similar to the one used in this work. In contrast to LFRic, which is developed for operational use, GUSTO is a research model implemented in the Firedrake Python code-generation framework described in Rathgeber *et al.* (2017). It uses the iterative solvers and preconditioners from the PETSc library (see Balay *et al.*, 1997) to solve the linear system. By default, the elliptic pressure operator is inverted with a black-box algebraic multigrid (AMG) algorithm. While in Mitchell and Müller (2016) AMG has been shown to give comparable performance to the bespoke geometric multigrid preconditioners developed here, using off-the-shelf AMG libraries in the LFRic code is not feasible due to their incompatible parallelisation strategy. It would also introduce undesirable software dependencies for a key component of the model.

2.2 | Parallel multigrid and atmospheric models

Multigrid algorithms allow the solution of ill-conditioned elliptic partial differential equations in a time that is proportional to the number of unknowns in the system. Due to this algorithmically optimal performance, they are often the method of choice for large-scale applications in geophysical modelling. The hypre library (Falgout and Yang, 2002) contains massively parallel multigrid implementations, including BoomerAMG, and has been shown to scale to hundreds of thousands of cores in Baker *et al.* (2012). Similarly, the scalability of the AMG solver in the DUNE library (Blatt and Bastian, 2006) has been demonstrated in Ippisch and Blatt (2011), and Notay and Napov (2015) describe another highly parallel AMG implementation. In Gmeiner *et al.* (2014), massively parallel multigrid methods based on hybrid hierarchical grids are used to solve problems with 10^{12} unknowns on more than 200,000 compute cores.

While these results clearly show the significant potential of parallel multigrid algorithms, it is evident from the review in Müller and Scheichl (2014) that they are rarely used in semi-implicit atmospheric models. An exception is the recent implementation of the MPAS model. In Sandbach *et al.* (2015), it is shown that a semi-implicit method with a multigrid solver can be competitive with fully explicit time integrators. A conditional semi-coarsening multigrid for the ENDGame dynamical core (Wood *et al.*, 2014) used by the Met Office is described in Buckeridge

and Scheichl (2010) and is currently implemented in the Unified Model code. Another recent application of the multigrid in a non-hydrostatic model is given in Yi (2018). Two-dimensional multigrid solvers for the Poisson equation on different spherical grids relevant for atmospheric modelling are compared in Heikes *et al.* (2013). More importantly, the work in Yang *et al.* (2016) showed that domain-decomposition-based multigrid algorithms can be used to solve the Euler equations with $0.77 \cdot 10^{12}$ unknowns. That paper received the 2016 Gordon Prize for showing that the code scales to 10 million cores and achieves 7.95 PetaFLOP performance on the TaihuLight supercomputer. Note, however, that none of the models described in this section are based on the advanced finite element discretisations that are used in this work.

3 | METHODS

In the following, the mimetic finite element discretisation of the Euler equations in the LFRic dynamical core is reviewed. By exploiting the structure of the pressure correction equation in the approximate Schur-complement solver, an efficient tensor-product multigrid algorithm is constructed.

3.1 | Continuous equations

The dynamical core of the model solves the Euler equations for a perfect gas in a rotating frame:

$$\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} &= -(\nabla \times \boldsymbol{u}) \times \boldsymbol{u} - 2\boldsymbol{\Omega} \times \boldsymbol{u} \\
&\quad - \frac{1}{2}\nabla(\boldsymbol{u} \cdot \boldsymbol{u}) - \nabla\Phi - c_p\theta\nabla\Pi, \\
\frac{\partial \rho}{\partial t} &= -\nabla \cdot (\rho \boldsymbol{u}), \\
\frac{\partial \theta}{\partial t} &= -\boldsymbol{u} \cdot \nabla\theta, \\
\Pi^{\frac{1-\kappa}{\kappa}} &= \frac{R}{p_0}\rho\theta.
\end{aligned} \tag{1}$$

At every point in time the state of the atmosphere $\boldsymbol{x} = (\boldsymbol{u}, \rho, \theta, \Pi)$ is described by the three-dimensional fields for (vector-valued) velocity $\boldsymbol{u}$, density ρ, potential temperature θ and (Exner) pressure Π. In Equation 1 Φ is the geopotential such that $\nabla\Phi = -\boldsymbol{g}$, where the vector $\boldsymbol{g}$ denotes the gravitational acceleration and the Earth's rotation vector is denoted by $\boldsymbol{\Omega}$. R is the gas constant per unit mass and $\kappa = R/c_p$, where c_p is the specific heat at constant pressure; p_0 is a reference pressure. The equations are solved in a domain D, which describes either the global atmosphere or a local area model (LAM); for further details on the relevant boundary conditions see Melvin *et al.* (2019). While this article describes the development of multigrid solvers for global models, the method can be easily adapted for LAMs.

3.2 | Finite element discretisation

To discretise Equation 1 in space, the mimetic finite element discretisation from Cotter and Shipton (2012) and Natale *et al.* (2016) is used. For this, four principal function spaces $\mathbb{W}_i$, $i = 0, 1, 2, 3$, of varying degrees of continuity are constructed. These function spaces are related by the de Rham complex (Bott and Tu, 2013)

$$\mathbb{W}_0 \xrightarrow{\nabla} \mathbb{W}_1 \xrightarrow{\nabla\times} \mathbb{W}_2 \xrightarrow{\nabla\cdot} \mathbb{W}_3. \tag{2}$$

Pressure and density are naturally discretised in the entirely discontinuous space $\mathbb{W}_3$, while the space $\mathbb{W}_2$, which describes vector-valued fields with a continuous normal component, is used for velocity. At order p on hexahedral elements the space $\mathbb{W}_2$ is the Raviart–Thomas space RT_p and $\mathbb{W}_3$ is the scalar discontinuous Galerkin space Q_p^{DG}. As will be important later on, note that the space $\mathbb{W}_2 = \mathbb{W}_2^h \oplus \mathbb{W}_2^z$ can be written as the direct sum of a component $\mathbb{W}_2^z$ which only contains vectors pointing in the vertical direction and the space $\mathbb{W}_2^h$ such that the elements of $\mathbb{W}_2^h$ are purely horizontal vector fields. In the absence of orography, these two spaces are orthogonal in the sense that

$$\int_{\mathsf{D}} \boldsymbol{u}^{(h)} \cdot \boldsymbol{u}^{(z)}\, dV = 0 \quad \text{for all } \boldsymbol{u}^{(h)} \in \mathbb{W}_2^h \text{ and } \boldsymbol{u}^{(z)} \in \mathbb{W}_2^z.$$

Note that $\mathbb{W}_2^z$ is continuous in the vertical direction and discontinuous in the tangential direction, whereas $\mathbb{W}_2^h$ is continuous in the horizontal direction only. To discretise the potential temperature field, an additional space $\mathbb{W}_\theta$ is introduced. $\mathbb{W}_\theta$ is the scalar-valued equivalent of $\mathbb{W}_2^z$ and has the same continuity. The lowest order ($p = 0$) function spaces are shown in Figure 1. Choosing suitable basis functions $\boldsymbol{v}_j(\chi) \in \mathbb{W}_2$, $\sigma_j(\chi) \in \mathbb{W}_3$ and $w_j(\chi) \in \mathbb{W}_\theta$ that depend on the spatial coordinate χ, the discretised fields at the nth model time step t can be written as follows:

$$\begin{aligned}
\boldsymbol{u}^n(\chi) &= \sum_j \tilde{u}_j^n \boldsymbol{v}_j(\chi) \in \mathbb{W}_u, \\
\rho^n(\chi) &= \sum_j \tilde{\rho}_j^n \sigma_j(\chi) \in \mathbb{W}_3, \\
\theta^n(\chi) &= \sum_j \tilde{\theta}_j^n w_j(\chi) \in \mathbb{W}_\theta,
\end{aligned}$$

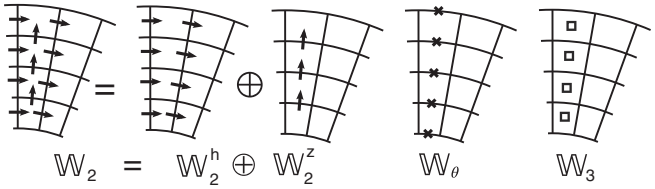

FIGURE 1 Function spaces used in the finite element discretisation

$$\Pi^n(\chi) = \sum_j \tilde{\Pi}_j^n \sigma_j(\chi) \in \mathbb{W}_3.$$

For each quantity a the corresponding vector of unknowns is written as $\tilde{a} = [a_1, a_2, \dots]$.

3.3 | Linear system

The time discretisation described in Melvin *et al.* (2019) is semi-implicit and uses a finite-volume transport scheme. This requires the solution of a nonlinear equation to obtain the state vector $\tilde{\boldsymbol{x}}^{n+1} = (\tilde{u}^{n+1}, \tilde{\rho}^{n+1}, \tilde{\theta}^{n+1}, \tilde{\Pi}^{n+1})$ at the next time step. The nonlinear system can be written compactly as

$$\mathcal{R}(\tilde{\boldsymbol{x}}^{n+1}) = 0. \tag{3}$$

Equation 3 is solved iteratively with a quasi-Newton method. For this a sequence of states $\tilde{\boldsymbol{x}}^{(k)}$, $k = 0,1,2, \dots, N_{\mathrm{NL}}$ with $\tilde{\boldsymbol{x}}^{(0)} = \tilde{\boldsymbol{x}}^n$, $\tilde{\boldsymbol{x}}^{(N_{\mathrm{NL}})} = \tilde{\boldsymbol{x}}^{(n+1)}$ is constructed such that

$$\mathcal{L}(\tilde{\boldsymbol{x}}^*)\tilde{\boldsymbol{x}}' = -\mathcal{R}(\tilde{\boldsymbol{x}}^{(k)}) \quad \text{with} \quad \tilde{\boldsymbol{x}}' = \tilde{\boldsymbol{x}}^{(k+1)} - \tilde{\boldsymbol{x}}^{(k)}. \tag{4}$$

The linear operator $\mathcal{L}(\tilde{\boldsymbol{x}}^*)$, which needs to be inverted in every Newton step, is an approximation to the Jacobian of $\mathcal{R}$. Following Wood *et al.* (2014), it is obtained by linearisation around a reference state $\tilde{\boldsymbol{x}}^* = (0, \tilde{\rho}^*, \tilde{\theta}^*, \tilde{\Pi}^*)$, which is updated at every time step. Introducing $\tilde{\boldsymbol{x}}' = (\tilde{u}', \tilde{\rho}', \tilde{\theta}', \tilde{\Pi}')$ and following Melvin *et al.* (2019), the linear system in Equation 4 can be written down in matrix form as

$$\begin{pmatrix} M_2^{\mu,C} & & -P_{2\theta}^{\Pi^*} & -G^{\theta^*} \\ D^{\rho^*} & M_3 & & \\ P_{\theta 2}^{\theta^*} & & M_\theta & \\ & -M_3^{\rho^*} & -P_{3\theta}^* & M_3^{\Pi^*} \end{pmatrix} \begin{pmatrix} \tilde{u}' \\ \tilde{\rho}' \\ \tilde{\theta}' \\ \tilde{\Pi}' \end{pmatrix} = \begin{pmatrix} -\mathcal{R}_u \\ -\mathcal{R}_\rho \\ -\mathcal{R}_\theta \\ -\mathcal{R}_\Pi \end{pmatrix}. \tag{5}$$

The exact form of the individual operators in Equation 5 is given in Melvin *et al.* (2019) and the matrix D^{ρ^*} is defined as $D^{\rho^*}\tilde{u} := D(\tilde{f}^*) \equiv D(\tilde{\rho}^*\tilde{u})$, where the mass flux f^* is defined as the product of the reference density ρ^* sampled at velocity nodal points pointwise multiplied by the velocity field $\tilde{u}$. Note that the expression in Melvin *et al.* (2019, their eqn. 81) for $\mathcal{R}_\Pi$ is incorrect and should be

$$\mathcal{R}_\Pi \equiv \left\langle \hat{\sigma}, \det \mathbf{J} \left[1 - \frac{p_0}{R} \frac{(\hat{\Pi}^{(k)})^{\frac{1-\kappa}{\kappa}}}{\hat{\rho}^{(k)}\hat{\theta}^{(k)}} \right] \right\rangle \tag{6}$$

such that both the linearised left-hand side and the nonlinear right-hand side are non-dimensionalised. To interpret the different operators, it is instructive to also write down the continuum equivalent of the equations for the state $(\boldsymbol{u}', \rho', \theta', \Pi')$:

$$\begin{aligned} &\boldsymbol{u}' + \tau_u \Delta t \left(\mu \hat{\boldsymbol{z}} (\hat{\boldsymbol{z}} \cdot \boldsymbol{u}') + 2\boldsymbol{\Omega} \times \boldsymbol{u}' \right) \\ &\quad + \tau_u \Delta t c_p (\theta' \nabla \Pi^* + \theta^* \nabla \Pi') = \boldsymbol{r}_u, \\ &\rho' + \tau_\rho \Delta t \nabla \cdot (\rho^* \boldsymbol{u}') = r_\rho, \\ &\theta' + \tau_\theta \Delta t \boldsymbol{u}' \cdot \nabla \theta^* = r_\theta, \\ &\frac{\Pi'}{\Pi^*} - \frac{\kappa}{1-\kappa} \left(\frac{\rho'}{\rho^*} + \frac{\theta'}{\theta^*} \right) = r_\Pi, \end{aligned} \tag{7}$$

where $\tau_{u,\rho,\theta} = \frac{1}{2}$ are relaxation parameters, and ρ^*, θ^* and Π^* are the continuous reference profiles around which the equation is linearised. The unit normal vector in the vertical direction is denoted as $\hat{\boldsymbol{z}}$ and the quantity μ is defined as $\mu = 1 + \tau_u \tau_\theta \Delta t^2 N^2$ with the Brunt–Väisälä frequency $N^2 = g(\partial_z \theta^*)/\theta^*$. In contrast to Wood *et al.* (2014), the horizontal couplings are not neglected in $P_{\theta 2}^{\theta^*}$, and will only be dropped in the approximate Schur complement constructed in Section 3.4. The block-diagonal entries in the 4×4 matrix in Equation 5 are modified mass matrices of the $\mathbb{W}_2$, $\mathbb{W}_3$ and $\mathbb{W}_\theta$ spaces, possibly weighted by reference profiles ($M_3^{\Pi^*}$, similar to the off-diagonal $M_3^{\rho^*}$). The term in the upper-left corner of the matrix in Equation 5 is the velocity mass matrix augmented by contributions from Rayleigh damping (optionally only applied to the vertical component of the velocity vector near the model lid; see Melvin *et al.*, 2019) and the implicit treatment of the Coriolis term,

$$M_2^{\mu,C} = M_2 + \tau_u \Delta t (M_\mu + M_C), \tag{8}$$

where

$$(M_C)_{ij} = 2 \int_{\mathsf{D}} \boldsymbol{v}_i \cdot (\Omega \times \boldsymbol{v}_j)\, dV.$$

While M_3, $M_3^{\Pi^*}$ and $M_3^{\rho^*}$ do not contain couplings to unknowns in neighbouring cells, and M_θ only couples between unknowns in the same vertical column, $M_2^{\mu,C}$ contains couplings in all directions. This prevents the exact solution of Equation 5 with a Schur-complement approach as in Wood *et al.* (2014), since the inverse of $M_2^{\mu,C}$ is dense. Instead, the system in Equation 5 is solved with an iterative Krylov subspace solver, which only requires application of the sparse operator $\mathcal{L}(\tilde{\boldsymbol{x}}^*)$ itself. The solver is

preconditioned with the approximate Schur complement described in the following section.

3.4 | Schur-complement preconditioner

To obtain an approximate solution of the linear system in Equation 5, first all instances of the mass matrix M_θ are replaced by a lumped, diagonal version $\mathring{M}_\theta$ such that the diagonal entries of $\mathring{M}_\theta$ are the row sums of M_θ. As in Wood *et al.* (2014), only the part of $P_{\theta 2}^{\theta^*}$ that acts on the vertical part of the velocity field is kept. The resulting operator $P_{\theta 2}^{\theta^*,z}$ maps from the subspace $\mathbb{W}_2^z \subset \mathbb{W}_2$ to $\mathbb{W}_\theta$.

Algebraically, the following steps correspond to multiplication by the upper block-triangular matrix (Step 1), solution of the block-diagonal matrix (Step 2) and back-substitution through multiplication by the lower block-triangular matrix (Step 3) in the Schur-complement approach (Zhang, 2006).

Step 1a. Use

$$\tilde{\theta}' = \mathring{M}_\theta^{-1}(-P_{\theta 2}^{\theta^*,z}\tilde{u}' - \mathcal{R}_\theta) \tag{9}$$

to eliminate θ. In the resulting 3×3 system, replace the matrix $\overline{M}_2^{\mu,C} := M_2^{\mu,C} + P_{2\theta}^{\Pi^*}\mathring{M}_\theta^{-1}P_{\theta 2}^{\theta^*,z}$ on the diagonal by a lumped diagonal approximation $\mathring{M}_2^{\mu,C}$ such that the diagonal entries of $\mathring{M}_2^{\mu,C}$ are the row sums of $\overline{M}_2^{\mu,C}$. This leads to a system for $\tilde{u}'$, $\tilde{\rho}'$ and $\tilde{\Pi}'$ only:

$$\begin{pmatrix} \mathring{M}_2^{\mu,C} & & -G^{\theta^*} \\ D^{\rho^*} & M_3 & \\ P_{3\theta}^*\mathring{M}_\theta^{-1}P_{\theta 2}^{\theta^*,z} & -M_3^{\rho^*} & M_3^{\Pi^*} \end{pmatrix} \begin{pmatrix} \tilde{u}' \\ \tilde{\rho}' \\ \tilde{\Pi}' \end{pmatrix} = \begin{pmatrix} -\overline{\mathcal{R}}_{\boldsymbol{u}} \\ -\mathcal{R}_\rho \\ -\overline{\mathcal{R}}_\Pi \end{pmatrix}, \tag{10}$$

with

$$\begin{aligned} \overline{\mathcal{R}}_{\boldsymbol{u}} &= \mathcal{R}_u - P_{2\theta}^{\Pi^*}\mathring{M}_\theta^{-1}\mathcal{R}_\theta, \\ \overline{\mathcal{R}}_\Pi &= \mathcal{R}_\Pi - P_{3\theta}^*\mathring{M}_\theta^{-1}\mathcal{R}_\theta. \end{aligned} \tag{11}$$

Note that – in contrast to Equation 5 – the (block-) diagonal entries of the 3×3 system in Equation 10 have sparse inverses, and it is possible to form the exact Schur complement.

Step 1b. Similarly, eliminate density from Equation 10 using

$$\tilde{\rho}' = M_3^{-1}(-D^{\rho^*}\tilde{u}' - \mathcal{R}_\rho) \tag{12}$$

to obtain a 2×2 system for $\tilde{u}'$ and $\tilde{\Pi}'$. Finally, eliminate velocity with

$$\tilde{u}' = (\mathring{M}_2^{\mu,C})^{-1}(G^{\theta^*}\tilde{\Pi}' - \overline{\mathcal{R}}_u) \tag{13}$$

to get an equation for the pressure increment only:

$$H\tilde{\Pi}' = \mathcal{B} = -\overline{\mathcal{R}}_\Pi + (\mathring{M}_2^{\mu,C})^{-1}\overline{\mathcal{R}}_u - M_3^{\rho^*}M_3^{-1}\mathcal{R}_\rho. \tag{14}$$

The Helmholtz operator $H : \mathbb{W}_3 \to \mathbb{W}_3$ is defined as

$$H = M_3^{\Pi^*} + (P_{3\theta}^*\mathring{M}_\theta^{-1}P_{\theta 2}^{\theta^*,z} + M_3^{\rho^*}M_3^{-1}D^{\rho^*})(\mathring{M}_2^{\mu,C})^{-1}G^{\theta^*}. \tag{15}$$

Step 2: Approximately solve the Helmholtz equation in Equation 14 for $\tilde{\Pi}'$. For this, one multigrid V-cycle as described in Section 3.6 is used.

Step 3: Given $\tilde{\Pi}'$, recover $\tilde{u}'$, $\tilde{\rho}'$ and $\tilde{\theta}'$ using Equations 13,12 and 9.

3.5 | Structure of the Helmholtz operator

Understanding the structure of the Helmholtz operator H is crucial for the construction of a robust multigrid algorithm for the approximate solution of Equation 14. The tensor-product multigrid method which will be used here was first described for simpler equations and discretisations in Börm and Hiptmair (2001) and applied to mixed finite element problems in atmospheric modelling in Mitchell and Müller (2016).

First, consider the sparsity pattern of the Helmholtz operator H. In each cell of the grid, it contains couplings to its four direct horizontal neighbours. In addition, it couples to the two cells immediately above and the two cells immediately below. Including the self-coupling, this results in a nine-cell stencil, independent of the order of discretisation p (see Figure 2). Second, it is important to take into account how the components of H depend on the time-step size Δt, the horizontal grid spacing Δx and the vertical grid spacing Δz. For this, first note that the weighted weak derivative D^{ρ^*} can be decomposed into a vertical and horizontal part $D^{\rho^*} = D^{\rho^*,z} + D^{\rho^*,h}$ with $D^{\rho^*,z} : \mathbb{W}_2^z \to \mathbb{W}_3$ and $D^{\rho^*,h} : \mathbb{W}_2^h \to \mathbb{W}_3$. Since the lumped mass matrix is diagonal, it is the sum of two terms, $\mathring{M}_2^{\mu,C} = \mathring{M}_2^{\mu,C,z} + \mathring{M}_2^{\mu,C,h}$ with $\mathring{M}_2^{\mu,C,h} : \mathbb{W}_2^h \to \mathbb{W}_2^h$ and $\mathring{M}_2^{\mu,C,z} : \mathbb{W}_2^z \to \mathbb{W}_2^z$. Using this decomposition, the Helmholtz operator can be written as the sum of four terms:

$$\begin{aligned} H = &\underbrace{M_3^{\Pi^*}}_{H_0} + \underbrace{P_{3\theta}^*\mathring{M}_\theta^{-1}P_{\theta 2}^{\theta^*,z}(\mathring{M}_2^{\mu,C,z})^{-1}G^{\theta^*}}_{D_1^z} \\ &+ \underbrace{M_3^{\rho^*}M_3^{-1}D^{\rho^*,z}(\mathring{M}_2^{\mu,C,z})^{-1}G^{\theta^*}}_{D_2^z} \\ &+ \underbrace{M_3^{\rho^*}M_3^{-1}D^{\rho^*,h}(\mathring{M}_2^{\mu,C,h})^{-1}G^{\theta^*}}_{D_2^h}. \end{aligned} \tag{16}$$

In order to interpret the different parts of H it is constructive to derive the corresponding Schur-complement operator for the continuous linear system given in

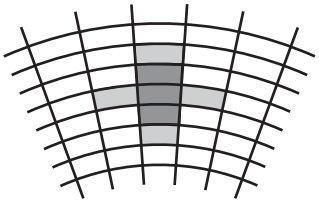

FIGURE 2 Stencil of the Helmholtz operator H (all grey cells) and of the operator $\hat{H}_z$ (dark grey cells). For clarity, only a two-dimensional cross-section of the stencil is shown; in three dimensions H has nine entries (instead of seven as for the standard near-neighbour stencil) and H_z has three en tries

Equation 7. To be consistent with the substitution $P^{\rho^*}_{\theta 2} \mapsto P^{\rho^*,z}_{\theta 2}$ above, the third part of Equation 7 is replaced by $\theta' + \tau_\theta \Delta t u'_z \partial_z \theta^* = r_\theta$. The resulting pressure operator is

$$\begin{aligned} h = & \underbrace{\frac{1}{\Pi^*}}_{h_0} \underbrace{-\tau_\rho \tau_u \Delta t^2 c_p \frac{\kappa}{1-\kappa} \mu^{-1} (\partial_z \theta^*) \partial_z}_{d_1^z} \\ & \underbrace{- \tau_\rho \tau_u \Delta t^2 c_p \frac{\kappa}{1-\kappa} \frac{\partial_z (\mu^{-1} \rho^* \theta^* \partial_z)}{\rho^*}}_{d_2^z} \\ & \underbrace{-\tau_\rho \tau_u \Delta t^2 c_p \frac{\kappa}{1-\kappa} \frac{\nabla_h \cdot (\rho^* \theta^* \nabla_h)}{\rho^*}}_{d_2^h}, \end{aligned} \tag{17}$$

where ∇_h is the horizontal gradient operator. Up to scaling by the local volume of the grid cell it is therefore possible to identify $H_0 \doteq h_0$, $D_1^z \doteq d_1^z$, $D_2^z \doteq d_2^z$ and $D_2^h \doteq d_2^h$. With the definitions of the linear operators in Melvin *et al.* (2019), it is easy to see that the different parts of H depend on the grid spacing and time-step size as

$$\begin{aligned} H_0 &\propto \Delta x^2 \cdot \Delta z, \\ D_1^z, D_2^z &\propto \frac{\Delta x^2 \cdot \Delta t^2}{\Delta z}, \\ D_2^h &\propto \Delta z \cdot \Delta t^2. \end{aligned} \tag{18}$$

Further observe that with $T^* = \Pi^* \theta^*$ and the ratio of the specific heat capacities $\kappa/(1-\kappa) = c_p/c_v$ the (squared) speed of sound is given by $c_s^2 = c_p/c_v R T^* = c_p \frac{\kappa}{1-\kappa} \Pi^* \theta^*$. Assuming that the reference profiles are slowly varying, this implies that the ratios $d_1^z : h_0, d_2^z : h_0$ and $d_2^h : h_0$ scale as

$$\begin{aligned} d_1^z : h_0 &\propto \frac{N^2 c_s^2}{g} \Delta t^2, \\ d_2^z : h_0 &\propto c_s^2 \Delta t^2, \\ d_2^h : h_0 &\propto c_s^2 \Delta t^2. \end{aligned} \tag{19}$$

Combining this with Equation 18 results in the estimates

$$\begin{aligned} D_1^z : H_0 &\propto \frac{N^2 c_s^2}{g} \cdot \frac{\Delta t^2}{\Delta z}, \\ D_2^z : H_0 &\propto \left(\frac{c_s \Delta t}{\Delta z} \right)^2 = \mathrm{CFL}_\mathrm{v}^2, \\ D_2^h : H_0 &\propto \left(\frac{c_s \Delta t}{\Delta x} \right)^2 = \mathrm{CFL}_\mathrm{h}^2, \end{aligned}$$

where CFL_v and CFL_h are the vertical and horizontal acoustic Courant numbers. Note also that the relative size of D_2^z and D_2^h is given by the squared aspect ratio $(\Delta x/\Delta z)^2$, and the relative size of D_1^z and D_2^z decreases $\propto \Delta z$ as the vertical grid spacing goes to zero.

To proceed further, the Helmholtz operator is split into two parts, $H = H_z + \delta H$, such that H_z contains the couplings to neighbouring cells in the vertical direction only. If the degrees of freedom are ordered consecutively in the vertical direction, H_z is a block-diagonal matrix. Each block describes the couplings in one vertical column; furthermore, solution of the system $H_z \tilde{\Pi} = r_\Pi$ for some right-hand side r_Π requires the independent solution of a block-pentadiagonal system in each column. Following the scaling arguments above, and observing that the operators D_1^z and D_2^z only contribute to H_z, it can be seen that for high aspect ratios $\Delta z \ll \Delta x$, the dominant part of the operator H is given by H_z. This observation is crucial for the following construction of a robust tensor-product multigrid algorithm.

3.6 | Multigrid solver

Starting from some initial guess, an approximate solution of Equation 14 can be obtained with a block-Jacobi iteration S. To avoid the expensive block-pentadiagonal solve, the next-to-nearest-neighbour couplings in H_z are dropped to obtain a block-tridiagonal $\hat{H}_z$; see Figure 2. With this matrix, one iteration of the block-Jacobi method is

$$\tilde{\Pi}' \mapsto \tilde{\Pi}' + \omega \hat{H}_z^{-1} (\mathcal{B} - H \tilde{\Pi}'), \tag{20}$$

where ω is an over-relaxation factor. The shorthand $\tilde{\Pi}' \mapsto \mathsf{BlockJacobi}(H, \mathcal{B}, \tilde{\Pi}', \omega, n_\mathrm{Jac})$ is used for n_Jac applications of the block-Jacobi iteration in Equation 20. Multiplication by $\hat{H}_z^{-1}$ in Equation 20 corresponds to the solution of the block-tridiagonal linear system, which can be carried out independently in each vertical column. The tridiagonal solve can be done, for example, with the Thomas algorithm (see, e.g., Press *et al.*, 2007). When applying H to $\tilde{\Pi}'$ to

calculate the residual $\mathcal{B} - H\tilde{\Pi}'$ in Equation 20, the vertical terms D_2^z and D_1^z in Equation 16 are treated exactly.

It is well known that stationary methods such as the Jacobi iteration converge extremely slowly since they only reduce the high-frequency error components. This issue is overcome by multigrid methods (Trottenberg *et al.*, 2001, contains a comprehensive treatment of the topic), which construct a hierarchy of grids with associated (nested) finite element function spaces, in particular $\mathbb{W}_3 = \mathbb{W}_3^{(1)} \supset \mathbb{W}_3^{(2)} \supset \ldots \supset \mathbb{W}_3^{(L)}$. Following the tensor-product approach in Börm and Hiptmair (2001), the grid is only coarsened in the horizontal direction. By applying a small number of smoother iterations on each level, the error is reduced at all length scales. In the following the index $\ell \in \{1, \ldots, L\}$ is used to label the multigrid level, with $L = 1$ corresponding to the fine-grid level on which the solution is to be found. Let $\{\tilde{\Pi}'^{(\ell)}\}$ and $\{\mathcal{B}^{(\ell)}\}$ be the set of solution vectors and right-hand sides on all levels, with $\tilde{\Pi}'^{(1)} = \tilde{\Pi}'$ and $\mathcal{B}^{(1)} = \mathcal{B}$. Since the function spaces are nested, the obvious prolongation $\mathfrak{P} : \mathbb{W}_3^{(\ell+1)} \to \mathbb{W}_3^{(\ell)}$ from a coarse space to the next-finest multigrid level is the natural injection

$$\mathfrak{P} : \Pi'^{(\ell+1)} \mapsto \Pi'^{(\ell)}_{\mathfrak{P}}, \tag{21}$$

with

$$\Pi'^{(\ell)}_{\mathfrak{P}}(\chi) = \Pi'^{(\ell+1)}(\chi) \qquad \text{for all points} \chi \in \mathsf{D}.$$

The corresponding linear operator acting on the degrees-of-freedom-vector (dof-vector) $\tilde{\Pi}'^{(\ell+1)}$ can be written as $\mathsf{Prolongate}(\tilde{\Pi}'^{(\ell+1)})$. Equation 21 naturally induces a restriction $\mathfrak{R} : \mathbb{W}_3^{(\ell)*} \to \mathbb{W}_3^{(\ell+1)*}$ on the corresponding dual spaces (denoted by *):

$$\mathfrak{R} : r^{(\ell)} \mapsto r^{(\ell+1)},$$

with

$$r^{(\ell+1)}(\Pi'^{(\ell+1)}) = r^{(\ell)}(\mathfrak{P}(\Pi'^{(\ell+1)}))$$

for all functions $\Pi'^{(\ell+1)} \in \mathbb{W}_3^{(\ell+1)}$. The corresponding linear operator acting on the vector $\mathcal{R}^{(\ell)}$ representing the dual one-form $r^{(\ell)}$ is written as $\mathsf{Restrict}(\mathcal{R}^{(\ell)})$; note that the level-dependent residual $\mathcal{R}^{(\ell)}$ is different from the quantities that appear on the right-hand side of Equation 5. The Helmholtz operators on the coarse levels are constructed by representing the reference profiles on those levels and re-discretising the operator. This is more efficient than assembling it via the expensive Galerkin triple-matrix product.

Based on these ingredients, it is now possible to write down the recursive multigrid V-cycle in Algorithm 1. Starting from some initial guess $\tilde{\Pi}' = \tilde{\Pi}'^{(1)}$ and the right-hand side $\mathcal{B}^{(1)} = \mathcal{B}$ on the finest level, this reduces the error by recursively solving the residual equation on the next-coarsest level.

Algorithm 1. Multigrid V-cycle, $\Sigma^\ell = \{H^{(\ell)}, \tilde{\Pi}'^{(\ell)}, \mathcal{B}^{(\ell)}, \mathcal{R}^{(\ell)}\}$ $\mathsf{MGVcycle}(\Sigma^{(\ell)}, \ell, \omega, n_{\text{pre}}, n_{\text{post}})$

if $\ell = L$ **then**
  $\tilde{\Pi}'^{(L)} \mapsto \mathsf{CoarseSolve}(H^{(L)}, \mathcal{B}^{(L)})$
else
  $\tilde{\Pi}'^{(\ell)} \mapsto \mathsf{BlockJacobi}(H^{(\ell)}, \mathcal{B}^{(\ell)}, \tilde{\Pi}'^{(\ell)}, \omega, n_{\text{pre}})$
  $\mathcal{R}^{(\ell)} \mapsto \mathcal{B}^{(\ell)} - H^{(\ell)}\tilde{\Pi}'^{(\ell)}$
  $\mathcal{B}^{(\ell+1)} \mapsto \mathsf{Restrict}(\mathcal{R}^{(\ell)})$
  $\tilde{\Pi}'^{(\ell+1)} \mapsto 0$
  $\tilde{\Pi}'^{(\ell+1)} \mapsto \mathsf{MGVcycle}(\{\Sigma^{(\ell)}\}, \ell+1, \omega, n_{\text{pre}}, n_{\text{post}})$
  $\tilde{\Pi}'^{(\ell)} \mapsto \tilde{\Pi}'^{(\ell)} + \mathsf{Prolongate}(\tilde{\Pi}'^{(\ell+1)})$
  $\tilde{\Pi}'^{(\ell)} \mapsto \mathsf{BlockJacobi}(H^{(\ell)}, \mathcal{B}^{(\ell)}, \tilde{\Pi}'^{(\ell)}, \omega, n_{\text{post}})$
end if

The natural way of (approximately) solving the equation on the coarsest level $\ell = L$ is by inverting the matrix directly or by applying a Krylov subspace method. In a parallel implementation this requires expensive global communications. As explained in Sandbach *et al.* (2015) and Müller and Scheichl (2014), this is not necessary in our case. To see this, observe that the relative size of the zero-order term H_0 and the second derivative in the horizontal direction D_2^h is proportional to the squared, inverse grid spacing Δx. Since the grid spacing doubles on each subsequent level, the relative size of the two terms in the Helmholtz operator reduces to $4^{-(\ell-1)}\mathrm{CFL}_\mathrm{h}^2$ where, as above, CFL_h is the acoustic Courant number in the horizontal direction. As the vertical terms are treated exactly in the block-Jacobi smoother, the condition number of the Helmholtz operator will be $\mathcal{O}(1)$ on levels $\ell \gtrsim \log_2(\mathrm{CFL}_\mathrm{h}) + 1$. Hence, it is sufficient to pick $L = \log_2(\mathrm{CFL}_\mathrm{h}) + 1$ and simply apply a few iterations of the block-Jacobi smoother. For typical atmospheric applications, $\mathrm{CFL}_\mathrm{h} \approx 10$ and hence it is sufficient to work with $L \approx 4$ levels. As demonstrated in Müller and Scheichl (2014), this shallow multigrid approach also greatly reduces global communications.

3.7 | Computational complexity

Although the multigrid method requires additional calculations on the coarse levels, its computational complexity is proportional to the number of unknowns. The time spent in one multigrid V-cycle in Algorithm 1 is dominated by two contributions: the multiplication with the matrix $H^{(\ell)}$ and the vertical solve, i.e., the application of $(\hat{H}_z^{(\ell)})^{-1}$. These operations are required in the residual calculation

and the block-Jacobi algorithm, which is used for both pre-/post-smoothing and the approximate solution of the coarse-level system with n_{coarse} block-Jacobi iterations. Let C_H and $C_{\hat{H}_z}$ be the cost per unknown for those two operations. For a pressure system with N unknowns the computational cost per multigrid V-cycle is

$$\begin{aligned}\text{Cost}_{\text{V-cycle}} &= ((n_{\text{pre}} + n_{\text{post}})(C_H + C_{\hat{H}_z}) + C_H)N\sum_{\ell=1}^{L-1} 4^{-\ell+1} \\ &\quad + n_{\text{coarse}}(C_H + H_{\hat{H}_z})N4^{-L} \\ &\approx \frac{4}{3}((n_{\text{pre}} + n_{\text{post}})(C_H + C_{\hat{H}_z}) + C_H)N, \qquad (22)\end{aligned}$$

where the approximation in the last line is valid for $4^{-L+1} \ll 1$.

In contrast, solving the Helmholtz pressure system with n_{iter} iterations of a BiCGStab method, preconditioned by $\hat{H}_z^{-1}$, involves a cost of

$$\text{Cost}_{\text{BiCGStab}} = (2n_{\text{iter}}(C_H + C_{\hat{H}_z}) + C_H)N.$$

While the multigrid V-cycle contains more nearest-neighbour parallel communications in the form of halo exchanges, these can easily be overlapped with computations via asynchronous calls to the message passing interface (MPI). In contrast, the BiCGStab solver requires four global sums per iteration (including one for monitoring convergence in the residual norm). While communication-avoiding variants of Krylov solvers exist (see the overview in Hoemmen, 2010, and recent work on BiCGStab in Carson *et al.*, 2013), this will not overcome the fundamental issue. Several BiCGStab iterations with global communications are required to achieve the same reduction in the pressure residual as in a multigrid V-cycle. As the numerical results in Section 5.2 demonstrate, this reduces the scalability of Krylov subspace solvers for the pressure correction equation.

3.8 | Memory requirements

The memory requirements of the different solvers for the pressure equation are quantified by counting the number of dof-vectors they need to store. Since the matrix H has a nine-point stencil and the LU-factorisation of the tridiagonal matrix $\hat{H}_z$ is required in the block-Jacobi iteration in Equation 20, storing the matrix requires the same memory as 12 dof-vectors. As can be seen from Algorithm 1, on each level of the multigrid hierarchy the three vectors $\tilde{\Pi}^{(\ell)}$, $\mathcal{B}^{(\ell)}$ and $\mathcal{R}^{(\ell)}$ are stored in addition to the Helmholtz matrix, resulting in a total memory requirement of 15 dof-vectors on each level. However, since the number of unknowns is reduced by a factor of 4 in each coarsening step, the memory requirements on the coarser levels are significantly reduced. In the standalone multigrid iteration, two additional vectors are required on the finest level to monitor convergence. Assuming that a dof-vector on the finest level contains N unknowns, this results in a total memory requirement of

$$\text{Memory}_{\text{Multigrid}} = 15\left(1 + \frac{1}{4} + \frac{1}{16} + \ \ldots\ \right)N + 2N < 22N$$

for the multigrid method. This should be compared to the BiCGStab solver: in addition to the solution, right-hand side and matrix, this uses eight temporary vectors, resulting in a total storage requirement of

$$\text{Memory}_{\text{BiCGStab}} = 22N.$$

The memory requirements of other solver combinations considered in this work are in the same ballpark. Using a standalone block-Jacobi iteration requires the storage of 16 dof-vectors, whereas the equivalent of no more than 28 dof-vectors has to be stored if BiCGStab is preconditioned with a multigrid V-cycle.

4 | IMPLEMENTATION

As described in Adams *et al.* (2019), the LFRic code is designed around a separation-of-concerns philosophy originally introduced in this context in Ford *et al.* (2013). It provides well-defined abstractions for isolating high-level scientific code from computational issues related to low-level optimisation and parallelisation, with the aim of achieving performance portability on different parallel hardware platforms. The model developer (an atmospheric scientist or numerical algorithm specialist) writes two kinds of code:

- *local kernels*, which describe the operations that are executed in one vertical column of the mesh;
- high-level *algorithms*, which orchestrate the kernel calls.

The PSyclone code-generation system (see Ford and Porter, 2019) automatically generates optimised wrapper subroutines for the parallel execution of the kernels over the grid. PSyclone can generate code for execution on distributed memory machines via MPI and shared-memory parallelisation with OpenMP, as well as mixed-mode parallelisation; it also supports threaded implementations on GPUs. Depending on the data dependencies, which are specified by attaching access descriptors to the kernels,

appropriate MPI calls (e.g., for halo exchanges) are automatically inserted into the generated code.

As is common in atmospheric modelling codes, and consistent with the tensor-product multigrid algorithm described in the article, the grid is only partitioned in the horizontal direction and the elementary kernels operate on contiguous data stored in individual vertical columns. By using a structured memory layout in the vertical direction, any costs of indirect addressing in the horizontal direction from logically unstructured grids can be hidden; this was already observed in MacDonald *et al.* (2011) and confirmed for tensor-multigrid solvers in Dedner *et al.* (2016).

To implement the solvers described in this article, the user will have to write the iterative solver algorithm and kernels for applying the appropriate finite element matrices or carrying out the block-tridiagonal solves in each column.

4.1 | Solver Application Programming Interface

For the complex solvers and preconditioners described in this article the parameter space is very large: different Krylov solvers for the outer mixed system in Equation 4 and for the Helmholtz pressure equation in Equation 14 will lead to varying overall runtimes. The performance of the multigrid preconditioner depends on the number of levels, the value of the over-relaxation parameter, the number of pre- and post-smoothing steps, and the choice of coarse-level solver. To explore different configurations and allow for the easy switching of components, an object-oriented framework for iterative solvers has been implemented in LFRic as described in Adams *et al.* (2019). Similar to the DUNE Iterative Solver Template Library (Blatt and Bastian, 2006) and PETSc (Balay *et al.*, 1997), this avoids re-implementation of the iterative solver and aids reproducibility. Based on this library of iterative solvers, the user has to provide problem-specific linear operators and preconditioner objects.

For this, three abstract data types are defined in the code:

- a *vector type*, which supports linear algebra operations such as `axpy` updates ($\boldsymbol{y} \mapsto \boldsymbol{y} + \alpha\boldsymbol{x}$) and dot products ($\beta = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_j x_j y_j$);
- a *linear operator type*, which acts on vectors $\boldsymbol{x}$ and implements the operation $\boldsymbol{y} \mapsto A\boldsymbol{x}$;
- a *preconditioner type*, which implements the operation $\boldsymbol{x} \mapsto P\boldsymbol{y}$, where $\boldsymbol{x}$ approximately solves the equation $A\boldsymbol{x} = \boldsymbol{y}$.

This allows the implementation of different Krylov subspace solvers, which are parametrised over the linear operator A and the corresponding preconditioner P, both of which are derived from their respective abstract base types. So far, the following solvers of the general form K(A,P,[ϵ]) (where ϵ is a tolerance on the target residual reduction) are available in LFRic:

- conjugate gradient, CG(A, P, ϵ);
- generalised minimal residual, GMRES(A, P, ϵ);
- stabilised biconjugate gradient, BiCGStab(A, P, ϵ);
- generalised conjugate residual, GCR(A, P, ϵ);
- a null-solver (preconditioner only), PreOnly(A,P).

All solvers operate on instances of a concrete field_vector type, which is derived from the abstract base-vector type and contains a collection of dof-vectors. More specifically, to implement the linear solver with the Schur-complement preconditioner for the linear system described in Section 3.3, an operator A_{mixed}, which represents the matrix in Equation 5 and acts on a field_vector for the state $\tilde{\boldsymbol{x}}' = (\tilde{u}', \tilde{\rho}', \tilde{\theta}', \tilde{\Pi}')$, was created. The corresponding Schur-complement preconditioner P_{mixed} acts on field_vectors of the same form and, as discussed in Section 3.4, contains a call to a Krylov subspace method $\mathrm{K}_H(A_H,P_H)$ for solving the Helmholtz problem in Equation 14. Here the operator A_H represents the Helmholtz operator in Equation 15 and P_H is a preconditioner; both A_H and P_H act on single-component field_vector objects of the form $\tilde{\boldsymbol{x}}'_H = (\tilde{\Pi}')$. Both the multigrid preconditioner $P_H^{(\mathrm{MG})}(L, \omega, n_{\text{pre}}, n_{\text{post}})$ described in Section 3.6 and a single-level method $P_H^{(\mathrm{Jac})}(\omega, n_{\text{Jac}})$, which corresponds to n_{Jac} applications of the block-Jacobi iteration in Equation 20, were implemented.

Thus the general nested solver can be written as

$$\mathsf{K}_{\text{mixed}}(A_{\text{mixed}}, P_{\text{mixed}}(\mathsf{K}_H(A_H, P_H, \epsilon_H)), \epsilon). \tag{23}$$

5 | RESULTS

To identify the most promising preconditioner, first the performance of different solvers for the pressure correction in Equation 14 is explored on a relatively small number of compute cores in Section 5.1, before massively parallel scaling tests for a smaller subset of solver configurations are presented in Section 5.2. Finally, robustness with respect to the time-step size is quantified in Section 5.3. All tests were run on the Met Office Cray XC40 supercomputer using the Aries interconnect. Each node comprises dual-socket, 18-core Broadwell Intel Xeon processors, that

TABLE 1 Pressure solver configurations used in Section 5.1

Pressure solver	$\mathbf{K}_H$	ϵ_H	P_H
Line relaxation	PreOnly	—	$P_H^{(\mathrm{Jac})}(0.8, 10)$
		10^{-2}	$P_H^{(\mathrm{Jac})}(1.0, 1)$
Krylov (ϵ_H)	BiCGStab	10^{-4}	$P_H^{(\mathrm{Jac})}(1.0, 1)$
		10^{-6}	$P_H^{(\mathrm{Jac})}(1.0, 1)$
		—	$P_H^{(\mathrm{MG})}(1, 0.8, 2, 2)$
MG(L)	PreOnly	—	$P_H^{(\mathrm{MG})}(2, 0.8, 2, 2)$
		—	$P_H^{(\mathrm{MG})}(3, 0.8, 2, 2)$
		—	$P_H^{(\mathrm{MG})}(4, 0.8, 2, 2)$
		10^{-2}	$P_H^{(\mathrm{MG})}(4, 0.8, 2, 2)$
Krylov-MG (ϵ_H, L)	BiCGStab	10^{-4}	$P_H^{(\mathrm{MG})}(4, 0.8, 2, 2)$
		10^{-6}	$P_H^{(\mathrm{MG})}(4, 0.8, 2, 2)$

is, 36 CPU cores per node. The model was compiled with the Intel 17 Fortran compiler (version 17.0.0.098).

5.1 | Algorithmic performance and pressure solver comparison

In all cases a GCR solver with a tolerance of $\epsilon = 10^{-6}$ is used to solve the mixed system in Equation 4; sometimes this will also be referred to as the "outer solve" below. Although other methods are available for this option, an investigation of the mixed solver is not the focus of this article and so only this method, as used in Melvin *et al.* (2019), is considered. To test the algorithmic performance of the solver, the model is run on the baroclinic-wave test case of Ullrich *et al.* (2014), which models the development of midlatitude atmospheric wave dynamics. Apart from the semi-implicit solver, the model set-up is the same as described in Melvin *et al.* (2019) with the following exceptions:

1. To improve long time-step stability, the continuity equation is handled in an (iterated) implicit manner, instead of an explicit one as in Melvin *et al.* (2019); that is, their eqn. 22 becomes

$$\langle \sigma, \delta_t \rho \rangle = -\Delta t \langle \sigma, \nabla \cdot \overline{\boldsymbol{F}}^{\alpha} \rangle \tag{24}$$

   with $\overline{\boldsymbol{F}}^{\alpha} \equiv \alpha \boldsymbol{u}^{n+1}\rho^{n+1} + (1-\alpha)\boldsymbol{u}^n\rho^n$ sampled pointwise.
2. To improve the accuracy of the advection operator over non-uniform meshes, a two-dimensional horizontal polynomial reconstruction of the potential temperature field is used instead of the one-dimensional reconstruction of Melvin *et al.* (2019). This reconstruction follows the method of Thuburn and Cotter (2012), except here the polynomial is always evaluated at fixed points in space instead of at Gauss points of the swept area as in Thuburn and Cotter (2012).

The model is run on a C192 mesh ($6 \times 192 \times 192$ cells) with $\approx 50km$ horizontal resolution and 30 levels in the vertical following a quadratic stretching such that the smallest vertical grid spacing is $\approx 200\,m$. This results in $39.3 \cdot 10^6$ total degrees of freedom, with $6.6 \cdot 10^6$ pressure unknowns. The time step is $\Delta t = 1200\ s$, which results in a horizontal-wave Courant number of $\mathrm{CFL_h} = c_s \Delta t / \Delta x \approx 7.9$ with $c_s = 340\,\mathrm{m \cdot s^{-1}}$. The vertical Courant number is $\mathrm{CFL_v} = c_s \Delta t / \Delta z \approx 1800$ (near the surface).

Different methods are used to solve the pressure correction equation in Equation (14):

1. **Line relaxation**: 10 iterations of the block-Jacobi solver.
2. **MG(L)**: Single geometric multigrid V-cycle with a varying number of levels $L = 1,2,3,4$ and a block-Jacobi line smoother on each level.
3. **Krylov(ϵ_H)** and **Krylov-MG(ϵ_H, L)**: BiCGStab iteration with a relative tolerance of $\epsilon_H = 10^{-2}, 10^{-3}, 10^{-6}$ and one of the following preconditioners:
   a. One iteration of the block-Jacobi method [Krylov(ϵ_H)]
   b. Single geometric multigrid V-cycle with $L = 4$ levels [Krylov-MG(ϵ_H, L)]

Following the notation in Equation 23, the solver configurations are summarised in Table 1. The Krylov solver with $\epsilon_H = 10^{-6}$ corresponds to the solver set-up used in Melvin *et al.* (2019). Two pre- and post-smoothing steps with an over-relaxation parameter $\omega = 0.8$ are used in the multigrid algorithm; the number of smoothing steps on the coarsest level is $n_{\mathrm{coarse}} = 4$.

TABLE 2 Number of iterations and time per linear solve for both the outer mixed solve $t^{(m)}_{solve}$ and the inner pressure solve $t^{(p)}_{solve}$

	Mixed solve		Pressure solve		Solver
Pressure solver	**Iterations**	$\boldsymbol{t^{(m)}_{solve}}$	**Iterations**	$\boldsymbol{t^{(p)}_{solve}}$	**Set-up**
Line relaxation	24.54	0.33	10	0.0075	0.018
Krylov(10^{-2})	15.10	0.39	15.12	0.0196	0.020
Krylov(10^{-3})	15.03	0.52	22.59	0.0276	0.019
Krylov(10^{-6})	14.03	0.96	52.50	0.0571	0.018
MG(1)	35.29	0.38	—	0.0047	0.018
MG(2)	19.51	0.23	—	0.0058	0.027
MG(3)	15.24	0.19	—	0.0067	0.026
MG(4)	15.12	0.20	—	0.0073	0.026
Krylov-MG(10^{-2},4)	15.04	0.26	1.58	0.0117	0.028
Krylov-MG(10^{-3},4)	15.03	0.34	2.21	0.0165	0.026
Krylov-MG(10^{-6},4)	15.03	0.63	4.37	0.0350	0.026

Notes: Set-up time per mixed solve for the linear operators is given in the final column. All numbers are averaged over the total run of the baroclinic-wave test case on the C192 mesh described in Section 5.1. Times are given in seconds.

The test is run for 8 days of simulation time (768 time steps) on 64 nodes of the Cray XC40 with six MPI ranks per node and six OpenMP threads per rank.

The accuracy of the linear system is governed by the solution obtained for Equation 5, which in all cases uses the same GCR solver and relative tolerance $\epsilon = 10^{-6}$. Therefore, there is very little difference in the solutions obtained from the different solver set-ups in Table 1, with the maximum difference in surface pressure after 8 days of simulation being $\approx 0.0001\%$ as compared to the Krylov(10^{-6}) configuration.

Table 2 lists the average number of outer, mixed-solver iterations per semi-implicit solve and the average number of iterations per pressure solve for each of the set-ups described above and summarised in Table 1. Note that in each time step the mixed solver is called four times to solve a nonlinear problem and Table 2 shows the average times for a *single* linear solve; these times are visualised in Figure 3. For completeness, results for the one-level multigrid method are also reported. Although in essence this simply corresponds to four applications of the line smoother, since $n_{pre} + n_{post} = 4$ this guarantees that the same number of fine-level smoother iterations is used for all multigrid results. The multigrid methods (provided more than one level is used) result in a significant reduction in the time taken for each linear solve and, compared to the Krylov methods, require roughly the same number of outer iterations. In particular, solving the pressure correction equation to a relatively tight tolerance of $\epsilon_H = 10^{-6}$ does not reduce the number of outer GCR iterations. This implies that the main error in the approximate Schur-complement solve is due to mass lumping, and not to an inexact solution of the pressure equation. Overall, a single multigrid V-cycle with $L = 3$ levels gives the best performance. Increasing the number of multigrid levels further does not provide any advantage since the problem is already well-conditioned after two coarsening steps, and adding further coarse levels will make the method slightly more expensive. Although stand-alone

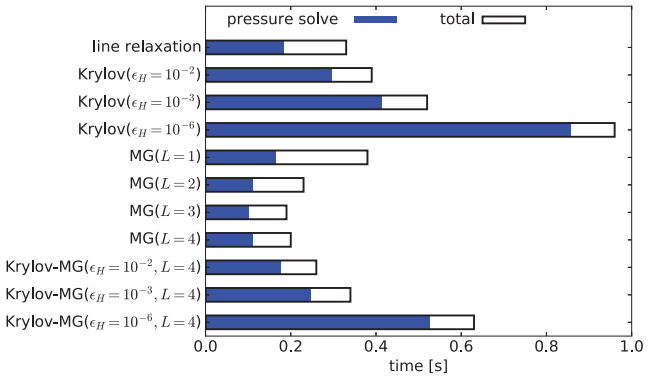

FIGURE 3 Breakdown of solver times for the baroclinic-wave test case on the C192 mesh described in Section 5.1. Both the total time per solve and the time spent in the pressure solver (filled bars) are shown

TABLE 3 Node configurations and local problem sizes for parallel strong scaling runs in Section 5.2

			Local unknowns	
Nodes	**Threads**	**Local columns**	**Mixed**	**Pressure**
384	13,824	24×24	103,968	17,280
864	31,104	16×16	46,528	7,680
1,536	55,296	12×12	26,352	4,320
3,456	124,416	8×8	11,872	1,920

line relaxation provides a cheap pressure solver (without any global sums), this is offset by a significant increase in the number of outer iterations such that the overall cost is not competitive with the multigrid method. In fact (looking at the final column of Table 2), 10 iterations of the block-Jacobi solver are slightly more expensive than the four-level multigrid V-cycle with two pre-/post-smoothing steps; this is not too far off the theoretical cost estimate in Equation 22. As the results for the Krylov-MG method show, there is no advantage in wrapping the multigrid V-cycle in a Krylov solver.

Although the use of the multigrid preconditioner significantly improves the speed of the linear solver, it does come with additional set-up costs. Principally, these come from two areas that are not covered in the solver timings in Figure 3. The first area is the set-up costs of the multigrid hierarchy, such as reading in the multiple meshes, computing the intergrid mappings and computation of certain temporally invariant operators, such as the mass matrices on every grid level. The additional cost of this area is marginal and does not show itself in the overall model runtimes for the tests in Figure 3. The second area is the computation of operators in Equation 16 that depend upon the reference state (those denoted with a * in Equation 16) and the restriction of the reference state (Π^*, ρ^*, θ^*) between different levels of the multigrid hierarchy. The average cost per mixed solve of computing these operators for all solver configurations is given in the final column of Table 2. If any multigrid levels are used then the cost of computing these operators increases by ≈50%, but adding more levels after the first does not alter the cost significantly. This increase is not a significant fraction of the overall model runtime and is more than outweighed by the savings from the solver; additionally, all these computations are local to the processing element and so would be expected to scale well.

Results (not shown) using the alternative iterative methods listed in Section 4.1 for the pressure solver with $\epsilon_H = 10^{-2}$ show similar performance to the BiCGStab solver presented in Table 2. The only exceptions are the Jacobi and precondition-only methods, which take approximately twice as long to run as using BiCGStab for the pressure solver. Since the linear problem is not symmetric, the conjugate gradient method does not converge.

5.2 | Massively parallel scalability

To examine parallel scalability, two representative pressure solvers are chosen, namely BiCGStab with a prescribed tolerance of $\epsilon_H = 10^{-2}$ (denoted "Kr2" hereafter) and a stand-alone multigrid V-cycle with three levels (denoted "MG"). For comparison, BiCGStab with a prescribed tolerance of $\epsilon_H = 10^{-6}$ (denoted "Kr6") is also included in the following study. The scaling tests are run on a C1152 cubed sphere mesh ($6 \times 1152 \times 1152$ cells) with ≈9 km horizontal resolution and 30 vertical levels with the same stretched grid and vertical CFL_v as in Section 5.1. The total number of degrees of freedom is $1.4 \cdot 10^9$ with $2.4 \cdot 10^8$ pressure unknowns. The time step is set to 205 s such that the horizontal-wave Courant number is again $CFL_h = c_s \Delta t / \Delta x \approx 8.0$, as in Section 5.1. In a strong scaling experiment the model is run for 400 time steps[1] on up to 3,456 nodes of the Cray XC40, keeping the global problem size fixed. Each node is configured to run with six MPI ranks and six OpenMP threads per rank; for the largest node count this corresponds to $3{,}456 \times 36 = 12{,}4416$ threads. In this case the local state vector has around 10,000 degrees of freedom and there are only around 2,000 pressure unknowns per core. In Fischer (2015), an analysis of solver algorithms is combined with a performance model for the computation and communication costs. According to this analysis, strong scaling is expected to break down when the local number of unknowns is smaller than $\mathcal{O}(10^4 - 10^5)$. It is therefore expected that the largest node count considered in this article is well within the strong scaling limit. Table 3 summarises the resulting local problem sizes for all node counts.

The average times spent in the mixed and pressure solvers are shown in Table 4. Note that these times are now reported for an entire time step, that is, aggregated over

[1]Due to time constraints, the model is only run for 250 time steps with the Kr6 solver on 384 nodes.

Nodes	MG		Kr2		Kr6	
	$T^{(m)}_{solve}$	$T^{(p)}_{solve}$	$T^{(m)}_{solve}$	$T^{(p)}_{solve}$	$T^{(m)}_{solve}$	$T^{(p)}_{solve}$
384	4.70	2.34	15.1	10.3	35.0	26.3
864	3.14	1.30	8.82	5.52	14.0	10.9
1,536	1.63	0.563	5.96	3.69	16.7	10.5
3,456	1.15	0.285	5.71	3.34	11.1	7.24

Notes: All results are measured in seconds and are averaged over 400 time steps.
T(m)solve denotes the average total time spent in the mixed, outer solve for each time step;
T(p)solve is the average time spent in the inner pressure solve.

TABLE 4 Strong scaling of the time spent in the linear solver on different numbers of nodes for the C1152 mesh test case in Section 5.2

four linear solves. This allows a quantitative comparison to the overall communication cost per time step as reported in Fig. 6.

It should be noted that the Aries network deployed on the Cray XC40 uses an adaptive routing algorithm for the messages (Mendygral *et al.*, 2019). This ensures the best utilisation of all the network links across the machine, but may not be optimal for a given task. Moreover, the path taken by messages is not the same each time, resulting in variation in the time taken. Ideally, a statistical measure such as the minimum or mean would be taken over many measurements to discount such variation. However, jobs running on such large numbers of nodes would be computationally expensive and the large number of potential paths through the network would require a large (and expensive) statistical sample, which was not feasible for this study. Consequently, only a single result for each set-up is reported.

Figure 4 quantifies the strong scaling of the time spent in the inner pressure solver. Rather than plotting the absolute times (which can be read off from Table 4), the parallel efficiency and the relative cost of the Krylov subspace solvers (Kr2 and Kr6) compared to the multigrid (MG) are shown. Let $T^{(p)}_{solve}(N)$ be the time spent in the pressure solve on N nodes. The parallel efficiency PE relative to $N_{ref} = 384$ nodes is defined as

$$PE = \frac{T^{(p)}_{solve}(N_{ref})/T^{(p)}_{solve}(N)}{N/N_{ref}}. \quad (25)$$

As the unhatched bars in Figure 4 show, the multigrid solver scales extremely well to 3,456 nodes. Both Krylov solvers (▧ hatching for Kr2 and ▨ hatching for Kr6) show significantly worse scaling; that the trend is not smooth can be attributed to network variability. The upper panel of Figure 4 shows the relative performance of the multigrid solver and the two Krylov methods for increasing node counts. More specifically, the ratio R = Kr/MG is obtained by dividing the time spent in one of the Krylov solvers by the time for the multigrid solver. The MG solver is much faster than either Kr2 or Kr6, especially for large

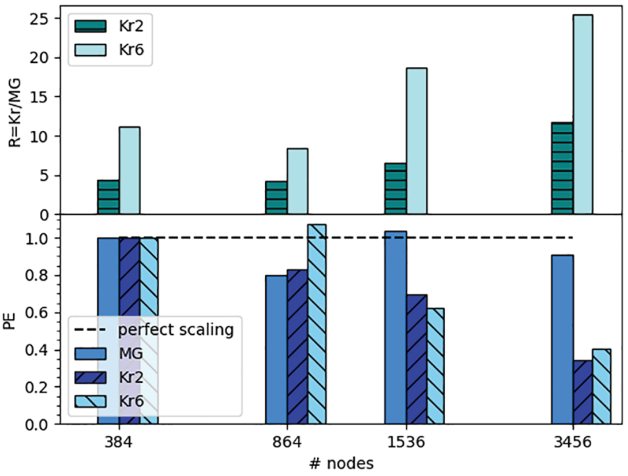

FIGURE 4 Strong scaling of the time spent in the inner pressure solver for the C1152 mesh test case in Section 5.2. The lower panel shows the parallel efficiency PE compared to 384 nodes as defined in Equation 25. The horizontal dashed line corresponds to perfect scaling where PE = 1. The relative cost of the Krylov solvers (Kr2 and Kr6) compared to the multigrid V-cycle (MG) is shown at the top.

node counts where the superior scalability of the multigrid algorithm pays off: for 3,456 nodes, the MG solver is more than 11 times faster than Kr2 and 25 times faster than Kr6. As will be discussed below, this can be at least partially explained by the fact that the multigrid does not rely on costly global reductions, which are required in each iteration of the Krylov subspace solvers. However, even for 384 nodes the relative advantage of MG is more than 4 times that of the Kr2 solver and 11 times that of Kr6.

Figure 5 shows the same quantities as in Figure 4, but for the outer, mixed solve. Again the parallel efficiency is given in the lower panel, while the upper panel quantifies the relative advantage of the MG pressure solver as compared to Kr2 and Kr6. Compared to Figure 4, the strong parallel efficiency of the outer solve drops to less than 40% for all pressure solvers. This can be explained by the additional parallel communications in the outer solver, which have a particularly strong impact for MG. As can be seen

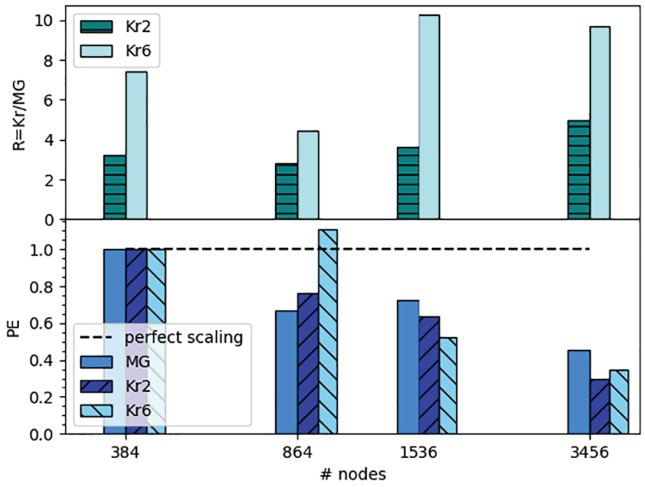

FIGURE 5 Strong scaling of the time spent in the outer, mixed solver for the C1152 mesh test case in Section 5.2. The lower panel shows the parallel efficiency PE compared to 384 nodes as defined in Equation 25. The horizontal dashed line shows perfect scaling where PE = 1. The relative cost of the Krylov pressure solvers (Kr2 and Kr6) compared to the multigrid V-cycle (MG) is shown at the top

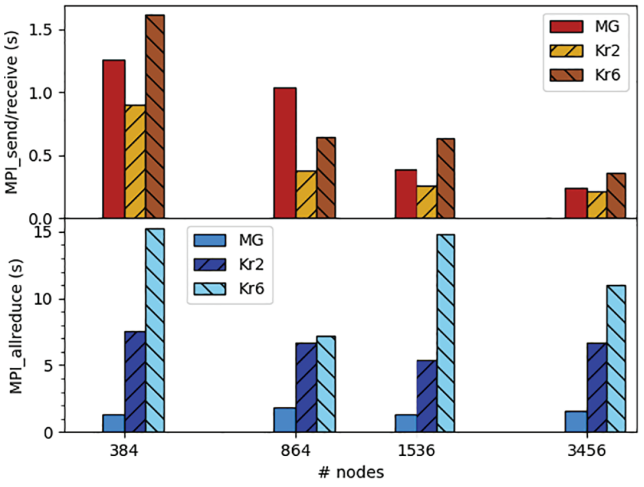

FIGURE 6 Strong scaling of the average communication costs per time step for the C1152 test case in Section 5.2. The upper panel shows the cost of MPI send/receive operations during halo exchange; the lower panel shows the time spent in global sums. All times are measured with the CrayPAT profiling tool

from Table 4, this is to be expected since the multigrid solver accounts for a relatively smaller fraction of the outer solve time. However, as can be read off from the top panel of Figure 5, the MG pressure solver still improves performance relative to Kr2 and Kr6: on the largest node count the multigrid leads to a 4× reduction in runtime compared to Kr2 (9× for Kr6) and even on 384 nodes the relative advantage of MG is 3× for Kr2 (7× for Kr6).

To understand the differences in parallel scalability for the different pressure solvers, reported in Fig. 6 shows the communication costs for increasing numbers of nodes. This includes both local communications in halo exchanges (top panel) and all-to-all communications in global reductions (bottom panel). The numbers were collected with the CrayPAT profiler in sampling mode. For technical reasons it was not possible to limit the measurements to the solver routine. Instead, the measured data is aggregated across all the calls to the relevant MPI library functions for the entire model run. However, from the profiling data, the time spent in the semi-implicit solver varies from 58–72% of the time per time step for MG, 79–87% for Kr2 and 89–92% for Kr6. Moreover, almost all the calls to the global sum take place in the solver routines. Thus it is reasonable to conclude that the communication costs are dominated by the solver.

The lower panel in Figure 6 shows τ, the time spent in the global sums per time step. On the largest node count, the approximate times are $\tau \approx 2\ s$ for MG (unhatched), $\tau \approx 7\ s$ for Kr2 (▨ hatching) and $\tau \approx 12\ s$ for Kr6 (▧ hatching). Evidently Kr2 and Kr6 spend much more time in global communications. This is readily explained by the fact that the number of global sums in the pressure solve is proportional to the number of Krylov solver iterations. In contrast, the MG V-cycle does not require any global sums and the superior scaling of the MG pressure solver itself (Figure 4) can be largely accounted for by this absence of the global communication. Moreover, from Figure 6 it can be seen especially for Kr6 the variation in the cost of the global sum is large. To quantify this, define the variation ν_t as

$$\nu_t = \frac{t_{\max} - t_{\min}}{t_{\max} + t_{\min}}, \tag{26}$$

where $t_{\min}$ and $t_{\max}$ are the smallest and largest measured global communication times across all four considered node counts. For the different methods the numerical values are $\nu_t = 0.18$ for MG, $\nu_t = 0.17$ for Kr2 and $\nu_t = 0.36$ for Kr6. Apart from the longer time-to-solution, the large variation in global communication costs due to network variability is another disadvantage of the Kr6 method. Again, this can be explained by the significant fraction of time spent in global sums during the pressure solve. For MG, global sums are only required in the outer solve and consequently the runtime variation is much smaller.

Although the trend in the data is not entirely clear due to the network variability, the Kr2 solver (▨ hatching) spends the least amount of time in nearest-neighbour communications. Since Kr6 (▧ hatching) requires more iterations to converge, it will also require more halo exchanges. The nearest-neighbour communication cost of the MG V-cycle (unhatched) lies somewhere between these two extremes, which can be attributed to additional halo exchanges on the coarser multigrid levels. Note, however,

that limiting the number of levels to $L = 3$ in the shallow multigrid approach avoids the inclusion of very coarse levels with a poor computation-to-communication ratio. The cost of halo exchange decreases with the number of nodes. This is again plausible since (as long as the message size is not too small) local communication is bandwidth-bound and thus scales with the amount of data that is sent. The third column of Table 3 shows that the size of the halo reduces by a factor of 3 as the number of nodes increases from 384 to 3,456.

The multigrid solver scales better than might be expected from Fischer (2015), which predicts that for the largest node count the local problems are so small that strong scaling should break down. This can be at least partially attributed to the fact that the analysis in Fischer (2015) is for a three-dimensional decomposition, whereas in the present work the decomposition is only two-dimensional. Surface-to-volume scaling implies that for small local volumes, there is less communication in two days than in three days.

As discussed in Section 5.1, there are extra costs for the set-up of the multigrid solver. The dominant costs arise from the computation of operators in Equation 16, which depend on the reference state and the restriction of the reference state between levels in the multigrid hierarchy. The cost of this computation is small compared to the overall runtimes, in some cases less than 1%, which is then not automatically captured in the profile. However, from the incomplete data captured it is apparent that, firstly, the computation of the operators scales very well with the processor number (as it is proportional to the problem size), and secondly, the excess cost for the multigrid is small, roughly a 50% increase as compared to the Krylov methods. Moreover, the excess is itself small compared to the cost of the multigrid pressure solve, about 6–8%.

5.3 | Robustness with respect to the time-step size

Finally, the impact of variations in the horizontal acoustic Courant number $\mathrm{CFL_h}$ on the performance of the model is studied. As can be inferred from the analysis in Section 3.5, the condition number of the Helmholtz operator H in Equation 15 depends strongly on $\mathrm{CFL_h}$. In other words, for a fixed grid spacing the condition number will increase with the time-step size, which can potentially make the pressure solver more costly. To quantify the impact of this, the model was run on 384 nodes with time-step sizes of varying length. Aside from this, the set-up is the same as in Section 5.2. Table 5 shows the number of outer mixed and pressure solver iterations for $\mathrm{CFL_h}$ between 4 and 8. Results for $\mathrm{CFL_h} = 4,6$ are averaged over the first 100 time steps. The final column of Table 5 shows the average time spent in the linear solver in each time step. Unfortunately it was not possible to choose larger time-step sizes due to instabilities resulting from the CFL limit imposed by the explicit advection scheme (for the simulations in Section 5.1 with the horizontal wave $\mathrm{CFL_h} \approx 8$, the advective Courant number grows from $\mathrm{CFL_{adv,h}} \approx 0.6$ early in the simulation to around $\mathrm{CFL_{adv,h}} \approx 1.3$ at the end of the simulation; the vertical advective Courant number is $\mathrm{CFL_{adv,v}} \approx 1.8$ by the end of the simulation). For the Kr2 and Kr6 solvers the number of inner, pressure iterations is also given. It can be observed that the number of outer iterations is largely insensitive to increases in the time-step size. In particular, using one multigrid V-cycle for the pressure solve gives good results for the range of $\mathrm{CFL_h}$ considered here. Any increase in the wall-clock time for the configurations using Krylov subspace pressure solvers can be clearly attributed to the growing number of inner iterations. Relative to the MG set-up, which only requires exactly one multigrid V-cycle in the pressure solve, the cost of the Kr2 and Kr6 solvers increases for larger values of $\mathrm{CFL_h}$. This is readily explained by the fact that the number of pressure solver iterations grows with the condition number. Empirically it can be observed that this number of iterations approximately doubles when $\mathrm{CFL_h}$ is increased from 4 to 8. While values of $\mathrm{CFL_h} \approx 10$ are typical atmospheric simulations, even for smaller time-step sizes the multigrid pressure solver shows a clear advantage compared to Krylov subspace methods.

TABLE 5 Average number of solver iterations for the outer mixed and inner pressure solves for increasing horizontal acoustic Courant number

		Iterations		
$\mathrm{CFL_h}$	Solver	Mixed	Pressure	$T^{(m)}_{solve}$
4	MG	12.9	—	3.96
4	Kr2	13.2	9.4	9.13
4	Kr6	12.0	26.2	16.29
6	MG	13.6	—	4.98
6	Kr2	13.3	13.2	10.31
6	Kr6	12.3	40.4	22.16
8	MG	14.0	—	4.70
8	Kr2	13.3	17.3	15.15
8	Kr6	12.6	54.2	34.96

Notes: Results are reported for the C1152 test case in Section 5.2. The final column lists the average time spent in the mixed, outer solve for each time step.

6 | CONCLUSION

This article describes the construction of a Schur-complement preconditioner with a multigrid pressure solver for the mixed-finite element LFRic numerical forecast model. Due to the presence of a velocity mass matrix–matrix, an additional outer solver iteration is necessary, which makes the solver significantly more complex than in simpler finite-difference models on structured latitude–longitude grids. By exploiting the structure of the Helmholtz operator on the highly anisotropic global grid, it is possible to build a highly efficient bespoke multigrid method using the tensor-product approach in Börm and Hiptmair (2001). Using only a relatively small number of multigrid levels further improves parallel scalability.

The numerical results presented here confirm the conclusions from earlier studies in idealised contexts, as described, for example, in Mitchell and Müller (2016): compared to Krylov subspace methods, solving the pressure correction equation with a single multigrid V-cycle leads to significantly better overall performance. Running a baroclinic test case with more than 1 billion (10^9) unknowns on 124,416 threads, the multigrid reduces the time spent in the outer, mixed solve by a factor of around 4. Since it requires significantly fewer global reductions, the multigrid also scales better in parallel. In contrast to Krylov subspace-based pressure solvers, the multigrid is robust with respect to changes in the time-step size.

There are several avenues for future work. While the multigrid has been demonstrated to be consistently better than any other considered solver, there are likely further small, problem-specific optimisations that can be achieved, for example, by tuning the parameters or including additional terms in the approximate Helmholtz operator by using a modified mass lumping strategy. While the focus of this article was on the lowest-order discretisation, Mitchell and Müller (2016) have demonstrated that the approach can be easily extended to higher orders by including an additional *p*-refinement step in the multigrid hierarchy. LFRic already provides the necessary code infrastructure. Following the promising results shown here, extending the approach from global simulations to local area models will require careful treatment of the boundary conditions, but is likely to lead to similar performance gains. Finally, an obvious downside of the approximate Schur-complement approach pursued here is the need for an additional outer solve of the mixed problem. As has been shown recently in Gibson *et al.* (2020), this can be avoided by using a hybridised mixed finite element discretisation. In fact, for a gravity-wave test case a solver based on this hybridised approach has already been implemented in LFRic. Although not yet available currently, an efficient preconditioner for this solver is being actively developed.

ACKNOWLEDGEMENTS

The authors thank the GungHo network (NERC grant NE/K006762/1) for financial support during collaborative visits. Part of this work was carried out during a secondment of E.H.M. funded by the Bath Institute for Mathematical Innovation.

ORCID

Thomas Melvin https://orcid.org/0000-0003-3933-0851
Eike Hermann Müller https://orcid.org/0000-0003-3006-3347

REFERENCES

Abdi, D.S., Giraldo, F.X., Constantinescu, E.M., Carr, L.E., Wilcox, L.C. and Warburton, T.C. (2019) Acceleration of the IMplicit–EXplicit nonhydrostatic unified model of the atmosphere on manycore processors. *The International Journal of High Performance Computing Applications*, 33(2), 242–267.

Adams, S.V., Ford, R.W., Hambley, M., Hobson, J.M., Kavčič, I., Maynard, C.M., Melvin, T., Hermann Müller, E.H., Mullerworth, S., Porter, A.R., Rezny, M., Shipway, B.J. and Wong, R. (2019) LFRic: meeting the challenges of scalability and performance portability in weather and climate models. *Journal of Parallel and Distributed Computing*, 132, 383–396.

Baker, A.H., Falgout, R.D., Kolev, T.V. and Yang, U.M. (2012). Scaling hypre's multigrid solvers to 100,000 cores, In: Berry M. et al. (eds) *High-Performance Scientific Computing*, pp. 261–279. London: Springer.

Balay, S., Gropp, W.D., McInnes, L.C. and Smith, B.F. (1997) Efficient management of parallelism in object oriented numerical software libraries, In: Arge E., Bruaset A.M., Langtangen H.P. (eds) *Modern Software Tools in Scientific Computing*, pp. 163–202. Boston, MA: Birkhäuser Press. https://doi.org/10.1007/978-1-4612-1986-6_8

Blatt, M. and Bastian, P. (2006) The iterative solver template library. In: Kågström B., Elmroth E., Dongarra J., Waśniewski J. (eds) *Applied Parallel Computing. State of the Art in Scientific Computing*. PARA 2006. Lecture Notes in Computer Science, vol 4699, pp. 666–675. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-75755-9_82

Börm, S. and Hiptmair, R. (2001) Analysis of tensor product multigrid. *Numerical Algorithms*, 26(3), 219–234.

Bott, R. and Tu, L.W. (2013) *Differential Forms in Algebraic Topology*. Vol. 82. New York: Springer Science & Business Media.

Buckeridge, S. and Scheichl, R. (2010) Parallel geometric multigrid for global weather prediction. *Numerical Linear Algebra with Applications*, 17(2–3), 325–342.

Carson, E., Knight, N. and Demmel, J. (2013) Avoiding communication in nonsymmetric Lanczos-based Krylov subspace methods. *SIAM Journal on Scientific Computing*, 35(5), S42–S61.

Côté, J., Gravel, S., Méthot, A., Patoine, A., Roch, M. and Staniforth, A. (1998) The operational CMC–MRB global environmental multiscale (GEM) model. Part I: design considerations and formulation. *Monthly Weather Review*, 126(6), 1373–1395.

Cotter, C.J. and Shipton, J. (2012) Mixed finite elements for numerical weather prediction. *Journal of Computational Physics*, 231(21), 7076–7091.

Cotter, C.J. and Thuburn, J. (2014) A finite element exterior calculus framework for the rotating shallow-water equations. *Journal of Computational Physics*, 257, 1506–1526.

Dedner, A., Müller, E. and Scheichl, R. (2016) Efficient multigrid preconditioners for atmospheric flow simulations at high aspect ratio. *International Journal for Numerical Methods in Fluids*, 80(1), 76–102.

Dennis, J.M., Edwards, J., Evans, K.J., Guba, O., Lauritzen, P.H., Mirin, A.A., St-Cyr, A., Taylor, M.A. and Worley, P.H. (2012) CAM-SE: a scalable spectral element dynamical core for the Community Atmosphere Model. *The International Journal of High Performance Computing Applications*, 26(1), 74–89.

Falgout, R.D. and Yang, U.M. (2002). hypre: a library of high performance preconditioners, In: Sloot P.M.A., Hoekstra A.G., Tan C.J.K., Dongarra J.J. (eds) *Computational Science - ICCS 2002. ICCS 2002. Lecture Notes in Computer Science*, vol 2331, pp. 632–641. Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-47789-6_66

Fischer, P.F. (2015). Scaling limits for PDE-based simulation (invited), *22nd AIAA Computational Fluid Dynamics Conference, 22–26 Jun 2015, Dallas, TX*. https://doi.org/10.2514/6.2015-3049, (to appear in print).

Ford, R., Glover, M.J., Ham, D.A., Maynard, C.M., Pickles, S.M., Riley, G.D. and Wood, N. (2013). Gung Ho: a code design for weather and climate prediction on exascale machines, *Exascale Applications and Software Conference, EASC2013, Edinburgh, 9–11 April 2013*. Edinburgh.

Ford, R. and Porter, A. (2019) *PSyclone Code Generation System*. https://github.com/stfc/PSyclone/wiki.

Fournier, A., Taylor, M.A. and Tribbia, J.J. (2004) The Spectral Element Atmosphere Model (SEAM): high-resolution parallel computation and localized resolution of regional dynamics. *Monthly Weather Review*, 132, 726–748.

Gibson, T.H., Mitchell, L., Ham, D.A. and Cotter, C.J. (2020) Slate: extending Firedrake's domain-specific abstraction to hybridized solvers for geoscience and beyond. *Geoscientific Model Development*, 13(2), 735–761.

Giraldo, F.X., Kelly, J.F. and Constantinescu, E.M. (2013) Implicit-explicit formulations of a three-dimensional nonhydrostatic unified model of the atmosphere (NUMA). *SIAM Journal on Scientific Computing*, 35(5), B1162–B1194.

Giraldo, F.X. and Restelli, M. (2008) A study of spectral element and discontinuous Galerkin methods for the Navier–Stokes equations in nonhydrostatic mesoscale atmospheric modeling: equation sets and test cases. *Journal of Computational Physics*, 227(8), 3849–3877. https://doi.org/10.1016/j.jcp.2007.12.009.

Gmeiner, B., Köstler, H., Stürmer, M. and Rüde, U. (2014) Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters. *Concurrency and Computation: Practice and Experience*, 26(1), 217–240.

Guerra, J.E. and Ullrich, P.A. (2016) A high-order staggered finite-element vertical discretization for non-hydrostatic atmospheric models. *Geoscientific Model Development*, 9(5), 2007–2029. https://doi.org/10.5194/gmd-9-2007-2016.

Heikes, R.P., Randall, D.A. and Konor, C.S. (2013) Optimized icosahedral grids: performance of finite-difference operators and multigrid solver. *Monthly Weather Review*, 141(12), 4450–4469.

Hoemmen, M. (2010). Communication-avoiding Krylov subspace methods. PhD Thesis, UC Berkeley.

Ippisch, O. and Blatt, M. (2011). Scalability test of $\mu\phi$ and the parallel algebraic multigrid solver of DUNE-ISTL, *Jülich Blue Gene/P Extreme Scaling Workshop, Jülich Supercomputing Centre, 14–16 Feb 2011*, pp. FZJ–JSC-IB-2011-02.

Kang, S., Giraldo, F.X. and Bui-Thanh, T. (2020) IMEX HDG-DG: a coupled implicit hybridized discontinuous Galerkin and explicit discontinuous Galerkin approach for shallow water systems. *Journal of Computational Physics*, 401, 109010

Kühnlein, C., Deconinck, W., Klein, R., Malardel, S., Piotrowski, Z.P., Smolarkiewicz, P.K., Szmelter, J. and Wedi, N.P. (2019) FVM 1.0: a nonhydrostatic finite-volume dynamical core for the IFS. *Geoscientific Model Development*, 12(2), 651–676.

MacDonald, A.E., Middlecoff, J., Henderson, T. and Lee, J.-L. (2011) A general method for modeling on irregular grids. *The International Journal of High Performance Computing Applications*, 25(4), 392–403.

Melvin, T., Benacchio, T., Thuburn, J. and Cotter, C. (2018) Choice of function spaces for thermodynamic variables in mixed finite-element methods. *Quarterly Journal of the Royal Meteorological Society*, 144(712), 900–916.

Melvin, T., Benacchio, T., Shipway, B., Wood, N., Thuburn, J. and Cotter, C. (2019) A mixed finite-element, finite-volume, semi-implicit discretization for atmospheric dynamics: Cartesian geometry. *Quarterly Journal of the Royal Meteorological Society*, 145(724), 2835–2853.

Mendygral, P., Wichmann, N., Roweth, D., Kandalla, K. and McMahon, K. (2019). Characterizing full-system network performance and congestion management capabilities with improved network benchmarks, *Cray User Group 2019 Proceedings*. https://cug.org/proceedings/cug2019_proceedings/includes/files/pres125s1.pdf.

Mitchell, L. and Müller, E.H. (2016) High level implementation of geometric multigrid solvers for finite element problems: applications in atmospheric modelling. *Journal of Computational Physics*, 327, 1–18.

Müller, A., Kopera, M., Marras, S., Wilcox, L.C., Isaac, T. and Giraldo, F.X. (2015a) Strong scaling for numerical weather prediction at petascale with the atmospheric model NUMA. *The International Journal of High Performance Computing Applications*, 33, 411–426.

Müller, E.H. and Scheichl, R. (2014) Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. *Quarterly Journal of the Royal Meteorological Society*, 140(685), 2608–2624.

Müller, E.H., Scheichl, R. and Vainikko, E. (2015b) Petascale solvers for anisotropic PDEs in atmospheric modelling on GPU clusters. *Parallel Computing*, 50, 53–69.

Natale, A., Shipton, J. and Cotter, C.J. (2016) Compatible finite element spaces for geophysical fluid dynamics. *Dynamics and Statistics of the Climate System*, 1, 1

Notay, Y. and Napov, A. (2015) A massively parallel solver for discrete Poisson-like problems. *Journal of Computational Physics*, 281, 237–250.

Peraire, J., Nguyen, N. and Cockburn, B. (2010). A hybridizable discontinuous Galerkin method for the compressible Euler and Navier–Stokes equations, *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 04 January 2010 – 07 January 2010. Orlando, Florida*. pp. 363.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (2007) *Numerical Recipes: The Art of Scientific Computing* (3rd edition). New York, NY: Cambridge University Press.

Rathgeber, F., Ham, D.A., Mitchell, L., Lange, M., Luporini, F., McRae, A.T.T., Bercea, G.-T., Markall, G.R. and Kelly, P.H.J. (2017) Firedrake: automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software (TOMS)*, 43(3), 24

Ringler, T.D., Thuburn, J., Klemp, J.B. and Skamarock, W.C. (2010) A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured C-grids. *Journal of Computational Physics*, 229(9), 3065–3090.

Sandbach, S., Thuburn, J., Vassilev, D. and Duda, M.G. (2015) A semi-implicit version of the MPAS-Atmosphere dynamical core. *Monthly Weather Review*, 143(9), 3838–3855.

Shipton, J., Gibson, T.H. and Cotter, C.J. (2018) Higher-order compatible finite element schemes for the nonlinear rotating shallow water equations on the sphere. *Journal of Computational Physics*, 375, 1121–1137.

Smolarkiewicz, P.K. and Szmelter, J. (2011) A nonhydrostatic unstructured-mesh soundproof model for simulation of internal gravity waves. *Acta Geophysica*, 59(6), 1109

Staniforth, A. and Thuburn, J. (2012) Horizontal grids for global weather and climate prediction models: a review. *Quarterly Journal of the Royal Meteorological Society*, 138(662), 1–26.

Steppeler, J., Hess, R., Schättler, U. and Bonaventura, L. (2003) Review of numerical methods for nonhydrostatic weather prediction models. *Meteorology and Atmospheric Physics*, 82(1–4), 287–301.

Temperton, C., Hortal, M. and Simmons, A. (2001) A two-time-level semi-Lagrangian global spectral model. *Quarterly Journal of the Royal Meteorological Society*, 127(571), 111–127.

Thuburn, J. and Cotter, C.J. (2012) A framework for mimetic discretization of the rotating shallow-water equations on arbitrary polygonal grids. *SIAM Journal on Scientific Computing*, 34(3), B203–B225.

Thuburn, J. and Cotter, C.J. (2015) A primal–dual mimetic finite element scheme for the rotating shallow water equations on polygonal spherical meshes. *Journal of Computational Physics*, 290, 274–297.

Thuburn, J., Cotter, C.J. and Dubos, T. (2013) A mimetic, semi-implicit, forward-in-time, finite volume shallow water model: comparison of hexagonal–icosahedral and cubed-sphere grids. *Geoscientific Model Development*, 7, 909–929. https://doi.org/10.5194/gmd-7-909-2014.

Trottenberg, U., Oosterlee, C.W. and Schuller, A. (2001) *Multigrid*. London: Academic Press.

Ullrich, P.A. (2014) A global finite-element shallow-water model supporting continuous and discontinuous elements. *Geoscientific Model Development*, 7, 3017–3035.

Ullrich, P.A., Jablonowski, C., Kent, J., Lauritzen, P.H., Nair, R., Reed, K.A., Zarzycki, C.M., Hall, D.M., Dazlich, D., Heikes, R., Konor, C., Randall, D., Dubos, T., Meurdesoif, Y., Chen, X., Harris, L., Christian, K., Lee, V., Qaddouri, A., Girard, C., Giorgetta, M., Reinert, D., Klemp, J., Sang-Park, H., Skamarock, W., Miura, H., Ohno, T., Yoshida, R., Walko, R., Reinecke, A. and Viner, K. (2017) DCMIP2016: a review of non-hydrostatic dynamical core design and intercomparison of participating models. *Geoscientific Model Development*, 10, 4477–4509.

Ullrich, P.A., Melvin, T., Jablonowski, C. and Staniforth, A. (2014) A proposed baroclinic wave test case for deep and shallow atmosphere dynamical cores. *Quarterly Journal of the Royal Meteorological Society*, 140, 1590–1602. https://doi.org/10.1002/qj.2241.

Wedi, N.P., Bauer, P., Denoninck, W., Diamantakis, M., Hamrud, M., Kuhnlein, C., Malardel, S., Mogensen, K., Mozdzynski, G. and Smolarkiewicz, P.K. (2015) *The Modelling Infrastructure of the Integrated Forecasting System: Recent Advances and Future Challenges*. European Centre for Medium-Range Weather Forecasts. Technical Memorandum no. 760.

Wedi, N.P., Hamrud, M. and Mozdzynski, G. (2013) A fast spherical harmonics transform for global NWP and climate models. *Monthly Weather Review*, 141(10), 3450–3461.

Wood, N., Staniforth, A., White, A., Allen, T., Diamantakis, M., Gross, M., Melvin, T., Smith, C., Vosper, S., Zerroukat, M. and Thuburn, J. (2014) An inherently mass-conserving semi-implicit semi-Lagrangian discretization of the deep-atmosphere global non-hydrostatic equations. *Quarterly Journal of the Royal Meteorological Society*, 140(682), 1505–1520.

Yamazaki, H., Shipton, J., Cullen, M.J.P., Mitchell, L. and Cotter, C.J. (2017) Vertical slice modelling of nonlinear Eady waves using a compatible finite element method. *Journal of Computational Physics*, 343, 130–149.

Yang, C., Xue, W., Haohuan, F., You, H., Wang, X., Ao, Y., Liu, F., Lin, G., Xu, P., Wang, L., Yang, G. and Zheng, W. (2016). 10M-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, 13–18 Nov. 2016*, pp. 6.

Yeh, K.-S., Côté, J., Gravel, S., Méthot, A., Patoine, A., Roch, M. and Staniforth, A. (2002) The CMC–MRB global environmental multiscale (GEM) model. Part III: nonhydrostatic formulation. *Monthly Weather Review*, 130(2), 339–356.

Yi, T.-H. (2018) Time integration of unsteady nonhydrostatic equations with dual time stepping and multigrid methods. *Journal of Computational Physics*, 374, 873–892.

Zhang, F. (2006) *The Schur Complement and its Applications*. New York: Springer Science & Business Media.

How to cite this article: Maynard C, Melvin T, Müller E. Multigrid preconditioners for the mixed finite element dynamical core of the LFRic atmospheric model. *Q J R Meteorol Soc*. 2020;146:3917–3936. https://doi.org/10.1002/qj.3880