Improving the scalability of a high-order atmospheric dynamics solver based on the deal. II library

Giuseppe Orlando, Tommaso Benacchio, Luca Bonaventura

Improving the scalability of a high-order atmospheric dynamics solver based on the `deal.II` library

Giuseppe Orlando[(1)],
Tommaso Benacchio[(2)], Luca Bonaventura[(3)]

[(1)] CMAP, CNRS, École polytechnique, Institut Polytechnique de Paris
Route de Saclay, 91120 Palaiseau, France
giuseppe.orlando@polytechnique.edu

[(2)] Weather Research, Danish Meteorological Institute
Sankt Kjelds Plads 11, 2100 Copenhagen, Denmark
tbo@dmi.dk

[(3)] Dipartimento di Matematica, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
luca.bonaventura@polimi.it

Keywords: Numerical Weather Prediction, Non-conforming meshes, Flows over orography, IMEX schemes, `deal.II`

Abstract

We present recent advances on the massively parallel performance of a numerical scheme for atmosphere dynamics applications based on the `deal.II` library. The implicit-explicit discontinuous finite element scheme is based on a matrix-free approach, meaning that no global sparse matrix is built and only the action of the linear operators on a vector is actually implemented. Following a profiling analysis, we focus on the performance optimization of the numerical method and describe the impact of different preconditioning and solving techniques in this framework. Moreover, we show how the use of the latest version of the `deal.II` library and of suitable execution flags can improve the parallel performance.

1 Introduction

Efficient numerical simulations of atmospheric flows are key to viable weather and climate forecasts and several related practical applications. Medium-range global numerical weather predictions (NWP) up to ten days ahead are typically required to complete within a one-hour operational threshold in the forecast cycles of meteorological centres. In addition, the computational meshes employed for these forecasts are being refined to head towards the km-scale, while experimental limited-area suites already reach the hectometric scale [10]. Hence, in the current context of increasing demand of computational resources driven by increasingly high spatial resolutions, massively parallel model scalability is a fundamental requirement.

In this work, we present recent advances in the parallel performance of a Implicit-Explicit Discontinuous Galerkin (IMEX-DG) solver [12] employed for atmosphere dynamics applications [13, 15, 14], following up on earlier analyses by the authors [16]. The implementation is carried out in the framework of the `deal.II` library [1, 2, 3]. First, we show how the use of suitable preconditioning techniques can significantly improve the performance of the elliptic pressure solver. More specifically, we measure the impact of a novel preconditioning technique for the linear system associated with the pressure field that exploits the numerical formulation of the problem. Next, we assess the impact of activating execution flags and of using the most recent version of the library (9.6.2, [1]). Finally, we present the result of employing a different strategy for the solution of a linear system.

The paper is structured as follows. In Section 2, we briefly review the model equations and the relevant details of the numerical methodology used in this work. The profiling and parallel performance analysis in numerical experiments with a three-dimensional benchmark of atmospheric flow are presented in Section 3. Finally, some conclusions are reported in Section 4.

2 The model equations and some details of the numerical method

The compressible Euler equations of gas dynamics represent the most comprehensive mathematical model for atmospheric flows [20]. Considering for simplicity the dry, non-rotating case, the mathematical model reads as follows:

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\,\mathbf{u}) = 0 \\ &\frac{\partial (\rho\,\mathbf{u})}{\partial t} + \nabla\cdot(\rho\,\mathbf{u}\otimes\mathbf{u}) + \nabla\, p = \rho\mathbf{g} \\ &\frac{\partial (\rho E)}{\partial t} + \nabla\cdot[(\rho h + \rho k)\,\mathbf{u}] = \rho\mathbf{g}\cdot\mathbf{u}, \end{aligned} \tag{1}$$

where $t \in (0, T_f]$ is the temporal coordinate, supplied with suitable initial and boundary conditions on the computational domain. Here $\otimes$ denotes the tensor product, T_f is the final time, ρ is the density, $\mathbf{u}$ is the fluid velocity, and p is the pressure. Moreover, $\mathbf{g} = -g\mathbf{k}$ is the acceleration of gravity, with $g = 9.81\,\mathrm{m\,s^{-2}}$ and $\mathbf{k}$ being the upward pointing unit vector in the standard Cartesian frame of reference. Finally, E denotes the total energy per unit of mass, h is the specific enthalpy and $k = 1/2\,|\mathbf{u}|^2$ is the specific kinetic energy. System (1) is complemented by the equation of state of ideal gases, given by $\rho e = \frac{1}{\gamma-1}p$, where e denotes the specific internal energy $e = E - k$ and the isentropic exponent γ is taken equal to 1.4.

The time discretization is based on an Implicit-Explicit (IMEX) Runge-Kutta (RK) method [6], which is a widely employed approach for ODE systems that include both stiff and non-stiff components. IMEX-RK schemes are represented compactly by the companion Butcher tableaux:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^\top \end{array} \qquad \begin{array}{c|c} \tilde{\mathbf{c}} & \tilde{\mathbf{A}} \\ \hline & \tilde{\mathbf{b}}^\top \end{array}$$

with $\mathbf{A} = \{a_{lm}\}, \mathbf{b} = \{b_l\}, \mathbf{c} = \{c_l\}$, $l, m = 1 \dots s$, denoting the coefficients of the explicit method, $\tilde{\mathbf{A}} = \{\tilde{a}_{lm}\}, \tilde{\mathbf{b}} = \left\{\tilde{b}_l\right\}$, and $\tilde{\mathbf{c}} = \{\tilde{c}_l\}$, $l, m = 1 \dots s$ denoting the coefficients of the implicit method, and s representing the number of stages of the method. We refer, e.g., to [6, 19] for a detailed analysis of the order and coupling conditions of the two companion schemes. A generic IMEX-RK l stage with time step Δt for the Euler equations reads therefore as follows:

$$\begin{aligned} \rho^{(l)} &= \rho^n - \sum_{m=1}^{l-1} a_{lm} \Delta t \, \nabla \cdot \left(\rho^{(m)} \, \mathbf{u}^{(m)} \right) \\ \rho^{(l)} \mathbf{u}^{(l)} &+ \tilde{a}_{ll} \Delta t \, \nabla \, p^{(l)} = \mathbf{m}^{(l)} \\ \rho^{(l)} E^{(l)} &+ \tilde{a}_{ll} \Delta t \, \nabla \cdot \left(h^{(l)} \rho^{(l)} \, \mathbf{u}^{(l)} \right) = \hat{e}^{(l)}, \end{aligned} \tag{2}$$

where we have set

$$\mathbf{m}^{(l)} = \rho^n \, \mathbf{u}^n - \sum_{m=1}^{l-1} a_{lm} \Delta t \, \nabla \cdot \left(\rho^{(m)} \, \mathbf{u}^{(m)} \otimes \mathbf{u}^{(m)} \right) - \sum_{m=1}^{l-1} \tilde{a}_{lm} \Delta t \, \nabla \, p^{(m)} \tag{3}$$

$$\hat{\mathbf{e}}^{(l)} = \rho^n E^n - \sum_{m=1}^{l-1} \tilde{a}_{lm} \Delta t \, \nabla \cdot \left(h^{(m)} \rho^{(m)} \, \mathbf{u}^{(m)} \right)) - \sum_{m=1}^{l-1} a_{lm} \Delta t \, \nabla \cdot \left(k^{(m)} \rho^{(m)} \, \mathbf{u}^{(m)} \right) \tag{4}$$

Notice that, substituting formally $\rho^{(l)} \, \mathbf{u}^{(l)}$ into the energy equation and taking into account the definitions $\rho E = \rho e + \rho k$ and $h = e + p/\rho$, the following nonlinear Helmholtz-type equation for the pressure is obtained:

$$\begin{aligned} \rho^{(l)} \left[e(p^{(l)}, \rho^{(l)}) + k^{(l)} \right] &- \tilde{a}_{ll}^2 \Delta t^2 \, \nabla \cdot \left[\left(e(p^{(l)}, \rho^{(l)}) + \frac{p^{(l)}}{\rho^{(l)}} \right) \nabla p^{(l)} \right] \\ &+ \tilde{a}_{ll} \Delta t \, \nabla \cdot \left[\left(e(p^{(l)}, \rho^{(l)}) + \frac{p^{(l)}}{\rho^{(l)}} \right) \mathbf{m}^{(l)} \right] = \hat{e}^{(l)}, \end{aligned} \tag{5}$$

which is solved through a fixed point procedure. We refer to [12, 17, 18] for further references and details on the theoretical properties of the method.

The spatial discretization is based on a Discontinuous Galerkin (DG) method [5], which combines high-order accuracy and flexibility in a highly data-local framework. In particular, this method is particularly well suited for mesh adaptive approaches [13, 15]. The shape functions correspond to the products of Lagrange interpolation polynomials for the support points of $(r+1)$-order Gauss-Lobatto quadrature rule in each coordinate direction, with r denoting the polynomial degree. The discrete formulation of method (2) can therefore be expressed as

$$\mathbf{A}^{(l)} \mathbf{U}^{(l)} + \mathbf{B}^{(l)} \mathbf{P}^{(l)} = \mathbf{F}^{(l)} \tag{6}$$

$$\mathbf{C}^{(l)} \mathbf{U}^{(l)} + \mathbf{D}^{(l)} \mathbf{P}^{(l)} = \mathbf{G}^{(l)}, \tag{7}$$

where $\mathbf{U}^{(l)}$ represents the vector of degrees of freedom of the velocity field, while $\mathbf{P}^{(l)}$ is the vector of the degrees of freedom of the pressure field. We refer to [13, 17, 16] for the detailed expression of all the matrices and vectors. We report only the expression of the matrix $A^{(l)}$ that will be useful for the discussion in Section 3:

$$A_{ij}^{(l)} = \sum_{K \in \mathcal{T}_\mathcal{H}} \int_K \rho^{(l)} \boldsymbol{\varphi}_j \cdot \boldsymbol{\varphi}_i \mathrm{d}\mathbf{x}, \tag{8}$$

where $\mathcal{T}_\mathcal{H}$ represents the computational mesh, $\boldsymbol{\varphi}_i$ denotes the basis function of the space of polynomial functions employed to discretize the velocity, and $\mathrm{d}\mathbf{x}$ is the spatial element of integration.

Formally, from (6) one can derive

$$\mathbf{U}^{(l)} = \left(\mathbf{A}^{(l)}\right)^{-1} \left(\mathbf{F}^{(l)} - \mathbf{B}^{(l)}\mathbf{P}^{(l)}\right), \tag{9}$$

and obtain

$$\mathbf{D}^{(l)}\mathbf{P}^{(l)} + \mathbf{C}^{(l)} \left(\mathbf{A}^{(l)}\right)^{-1} \left(\mathbf{F}^{(l)} - \mathbf{B}^{(l)}\mathbf{P}^{(l)}\right) = \mathbf{G}^{(l)}. \tag{10}$$

The above system is then solved following the fixed point procedure described in [12]. In Sections 3.1 and 3.3 we describe the impact of using different preconditioning and solving techniques for system (10).

3 Numerical results

In this Section, we show results of simulations of an idealized three-dimensional test case of atmospheric flow over orography [11, 15] using the numerical method described in the previous Section and focusing on its scalability. The simulations have been run using up to 1024 2x AMD EPYC Rome 7H12 64c 2.6GHz CPUs at MeluXina HPC facility[1] and OpenMPI 4.1.5 has been employed. The compiler is GCC version 12.3 and the Vectorization level is 256 bits.

We consider a three-dimensional configuration of a flow over a bell-shaped hill, already studied in [11, 13, 15]. The computational domain is $(0, 60) \times (0, 40) \times (0, 16)$ km. The mountain profile is defined as follows:

$$h(x, y) = h_c \left[1 + \left(\frac{x - x_c}{a_c}\right)^2 + \left(\frac{y - y_c}{a_c}\right)^2\right]^{-\frac{3}{2}}, \tag{11}$$

with $h_c = 400\,\mathrm{m}, a_c = 1\,\mathrm{km}, x_c = 30\,\mathrm{km}$, and $y_c = 20\,\mathrm{km}$. The buoyancy frequency is $N = 0.01\,\mathrm{s}^{-1}$, whereas the background velocity is $\overline{u} = 10\,\mathrm{m\,s}^{-1}$. The final time is $T_f = 1\,\mathrm{h}$. The initial conditions read as follows [4]:

$$p = p_{ref} \left\{1 - \frac{g}{N^2}\Gamma\frac{\rho_{ref}}{p_{ref}}\left[1 - \exp\left(-\frac{N^2 z}{g}\right)\right]\right\}^{1/\Gamma}, \quad \rho = \rho_{ref}\left(\frac{p}{p_{ref}}\right)^{1/\gamma} \exp\left(-\frac{N^2 z}{g}\right), \tag{12}$$

where $p_{ref} = 10^5\,\mathrm{Pa}$ and $\rho_{ref} = \frac{p_{ref}}{RT_{ref}}$, with $T_{ref} = 293.15\,\mathrm{K}$. Finally, we set $\Gamma = \frac{\gamma - 1}{\gamma}$. We refer to [15] for the implementation of boundary conditions. Unless differently stated, we consider a) a uniform mesh composed of $N_{el} = 120 \times 80 \times 32 = 307200$ elements with polynomial degree $r = 4$, leading to about 38.5 million unknowns for each scalar variable and a resolution of 125 m; and b) a non-conforming mesh composed of $N_{el} = 204816$ elements, yielding around 25.6 millions of unknowns for each scalar variable and a maximum resolution of 62.5 m. A non-conforming mesh is characterized by neighbouring cells with different resolution on both the horizontal and the vertical direction [15, 16] and its use provides sizeable computational savings [15, 16]. The following optimization flags are employed throughout the runs:

```
-O2 -funroll-loops -funroll-all-loops -fstrict-aliasing -Wno-unused-local-typedefs
```

which are the standard ones in the Release mode of `deal.II` (see https://www.dealii.org/developer/users/cmake_user.html). Unfortunately, due to limitations on computational resources, the scaling analyses reported in this Section could only be performed once for each of the different configurations.

[1] https://docs.lxp.lu/

3.1 Impact of preconditioning technique

The preconditioned conjugate gradient method with a geometric multigrid preconditioner is employed to solve the symmetric positive definite linear systems, while the GMRES solver with a Jacobi preconditioner is employed for the solution of non-symmetric linear systems. A matrix-free approach is employed [2], meaning that no global sparse matrix is built and only the action of the linear operators on a vector is actually implemented. This strategy is very efficient for high-order discretization methods [7]. However, it leads to some difficulties in the preconditioning, since the matrix is not available and standard techniques as ILU cannot be employed. For symmetric positive-definite systems, an efficient preconditioning technique compatible with the matrix-free approach is based on the so-called Chebyshev polynomial that relies on an estimate of the eigenvalues of the matrix so as to damp the eigenvalue range [9].

The approach based on the eigenvalue estimate cannot be directly employed for the solution of (10) because it requires symmetry and positive definiteness of the preconditioned matrix [21]. Moreover, the entries of the matrices $(\mathbf{A}^{(l)})^{-1}$, $\mathbf{D}^{(l)}$, and $\mathbf{C}^{(l)}$ are not readily available because of the matrix-free approach. In a matrix-free framework, a matrix representation of the operator can be obtained by applying the operator on all unit vectors. Apparently, this is a very inefficient procedure because it requires to perform n operator evaluations for a $n \times n$ matrix. In practice, the integration is so efficient that the computation completes without significant overhead [2]. However, using this procedure for the computation of $(\mathbf{A}^{(l)})^{-1}$, $\mathbf{D}^{(l)}$, and $\mathbf{C}^{(l)}$ turns out to be very inefficient because it requires the evaluation of three operators as well as the solution of a linear system at each operator evaluation to compute $(\mathbf{A}^{(l)})^{-1}$. Hence, the simplest solution consists in considering only the contribution provided by $\mathbf{D}^{(l)}$, which takes into account the internal energy. For low Mach number flows, as those considered in this work, this approximation can be considered reliable since velocities are typically low and the internal energy is therefore dominant with respect to the kinetic energy. This is the strategy adopted, e.g., in [15], and we refer to it as *internal energy preconditioner*.

We consider here a novel alternative approach that can be easily implemented in a matrix-free framework and consists of defining an operator that takes into account the underlying nonlinear Helhmholtz-type equation (5) when solving (10). More specifically, we consider the volumetric contribution of the weak form associated to a DG discretization of (5). Hence, the preconditioner is the inverse of the diagonal part of the matrix $\tilde{\mathbf{S}}^{(l)}$ whose entries are

$$\tilde{S}_{ij}^{(l)} = \sum_{K \in \mathcal{T}_\mathcal{H}} \int_K \rho^{(n,l)} e\left(\Psi_j, \rho^{(l)}\right) \Psi_i \mathrm{d}\mathbf{x} + \sum_{K \in \mathcal{T}_\mathcal{H}} \int_K \tilde{a}_{ll}^2 \Delta t^2 \left(e\left(p^{(l)}, \rho^{(l)}\right) + \frac{p^{(l)}}{\rho^{(l)}} \right) \nabla \Psi_j \cdot \nabla \Psi_i \mathrm{d}\mathbf{x}. \tag{13}$$

where Ψ_i denotes the basis function of the space of polynomial functions employed to discretize the pressure. Matrix $\tilde{\mathbf{S}}^{(l)}$ is the sum of two contributions: a first one that considers the internal energy and that coincides with matrix $\mathbf{D}^{(l)}$, and a second one that considers the contribution due to specific enthalpy and takes into account the underlying elliptic character of (5). The pressure $p^{(l)}$ in (13) in the fixed point procedure is evaluated at the previous fixed point iteration. We refer to this strategy as *Helmholtz preconditioner*.

An analysis of time to solution using the two preconditioning techniques reveals that, when using a uniform mesh, the use of the Helmholtz preconditioner gives a relatively modest computational time saving - about 18% with 128 cores (1 node) - that decreases for higher core counts (Figure 1). A more sizeable computational time saving is established for the non-conforming mesh which amounts to 60% employing 128 cores and to 46% employing 2048 cores (Figure 1). Hence, strategy *Helmholtz preconditioner* provides a significant advantage in terms of time-to-solution, in particular for non-conforming meshes. In the following, we will restrict our attention only to this preconditioning technique and to experiments using the non-conforming mesh.

[2] https://www.dealii.org/current/doxygen/deal.II/step_37.html

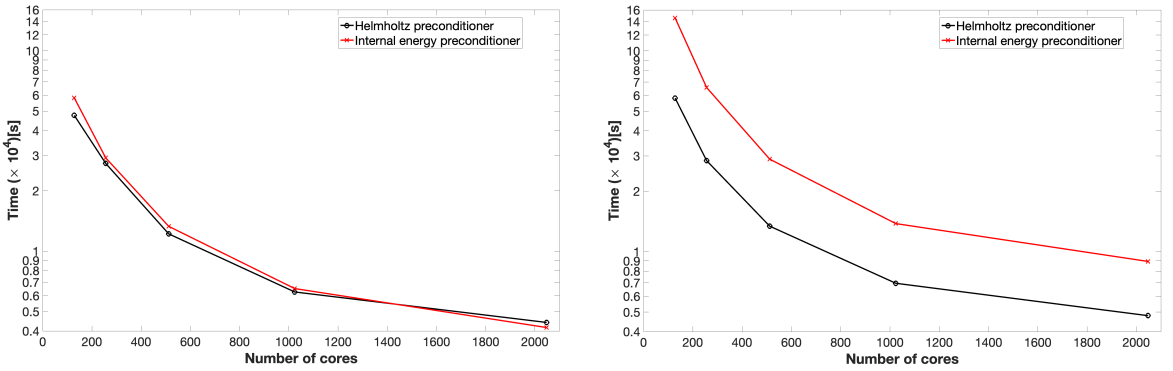

Figure 1: Analysis of preconditioning technique, wall-clock time in the numerical solution of problem (2), flow over a 3D hill test case. Left: uniform mesh. Right: non-conforming mesh. The black lines show the results with the *Helmholtz preconditioner* approach, whereas the red lines show the results with *internal energy preconditioner* approach.

3.2 Impact of execution flags and version

Next, we analyze the impact of some execution flags and of the use of the `deal.II` 9.6.2 version [1] instead of the `deal.II` 9.5.2 version [2]. Thanks to the support of EPICURE team at MeluXina, it has been shown that the execution flags

```
export DEAL_II_NUM_THREADS="$SLURM_CPUS_PER_TASK"
export OMP_NUM_THREADS="$SLURM_CPUS_PER_TASK"
```

improve the parallel performance of the solver in terms of both strong scaling and wall-clock time (Figure 2). However, an even more sizeable improvement in the parallel performance is obtained using the `deal.II` 9.6.2 version, leading to a computational time saving of around 28% compared to using the 9.5.2 version in a run with 2048 cores (i.e. 16 nodes). For the results in the following Section 3.3, `deal.II` 9.6.2 version with the optimal execution flags is therefore employed.

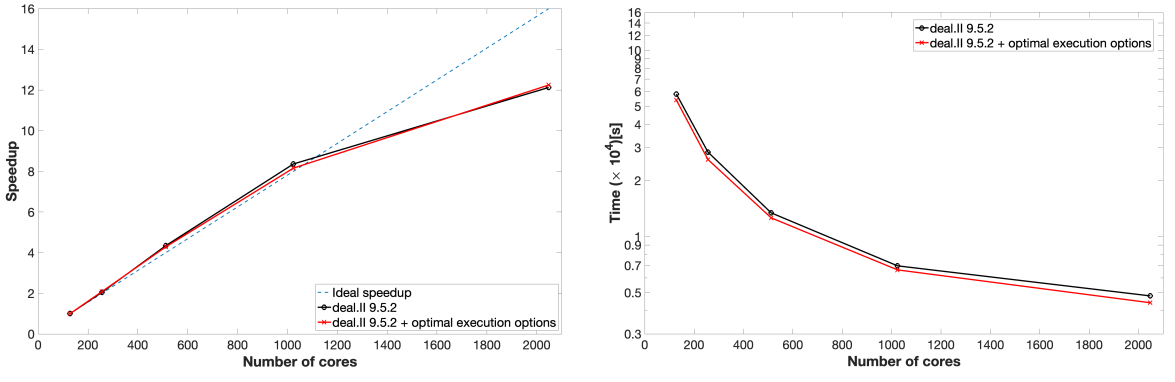

Figure 2: Performance impact of using different execution flags in the numerical solution of problem (2), flow over a 3D hill test case. Left: strong scaling. Right: WT time. The results are obtained with the non-conforming mesh. The black lines show the results with the standard execution flags, whereas the red lines show the results with optimal execution flags.

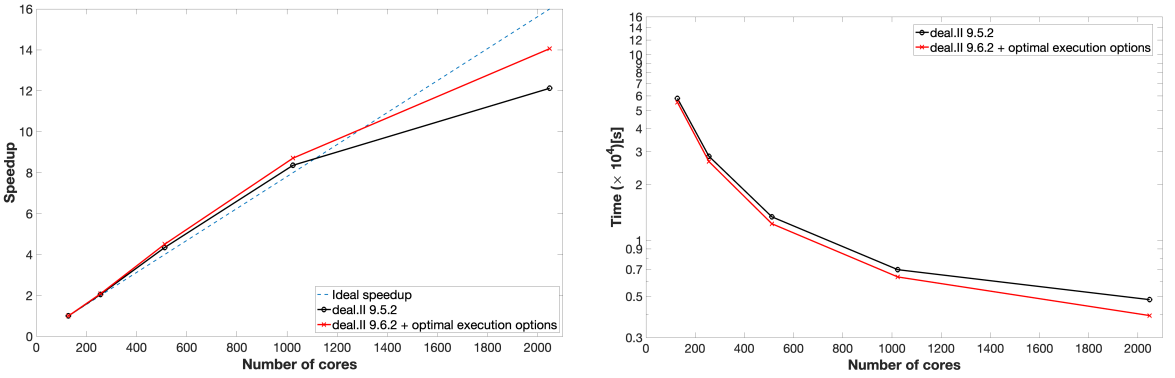

Figure 3: Performance impact of using different `deal.II` versions in the numerical solution of problem (2), flow over a 3D hill test case. Left: strong scaling. Right: WT time. The results are obtained with the non-conforming mesh. The black lines show the results obtained using the `deal.II` 9.6.2 version and with optimal execution flags, whereas the red lines show the results obtained using the `deal.II` 9.5.2 version with standard execution flags.

3.3 Fast evaluation of block-diagonal operators

Finally, we have investigated the distribution of the computational time spent in the different part of the algorithms. A profiling analysis show that, as expected, the vast majority of execution time is spent in solving problem (10) to compute the pressure field (Figure 4). The results are obtained using the non-conforming mesh, but analogous considerations hold for the uniform mesh (not shown). Apart from the computational time spent in computing the pressure field, one can immediately notice an increasing relative cost of computing the velocity field (orange bar in Figure 4) which ranges from about 3.6% with 128 cores to about 13% with 2048 cores. This part of the algorithm consists in computing $(\mathbf{A}^{(l)})^{-1}$ (we refer to [12, 13] for all the details), hence solving a linear system. However, this operation is repeated several times in the fixed point loop because of the matrix-free approach, which explains the increasing computational burden.

The matrix $\mathbf{A}^{(l)}$ (8) is a simple block-diagonal operator. Efficient techniques to evaluate and to invert block-diagonal operators in a DG framework were presented in [7]. However, they require the use of the same number of quadrature points and degrees of freedom along each coordinate direction, since this yields, after a change of basis, diagonal blocks in the algebraic structure of the the block-diagonal operator [7]. More specifically, as explained in [7, 8], the block-diagonal operator is inverted as $\mathbf{S}^{-T}\mathbf{J}^{-1}\mathbf{S}^{-1}$. Here $\mathbf{J}$ is a diagonal matrix, whose entries are equal to the determinant of the Jacobian times the quadrature weight, while $\mathbf{S}$ is a square matrix with basis functions in the row and quadrature points in columns, i.e. $S_{ij} = \varphi_i(\mathbf{x}_j)$. The matrix $\mathbf{S}$ is then constructed as the Kronecker product (tensor product) of small one-dimensional matrices that can be inverted efficiently at each stage. In a nodal DG framework with nodes located at $r+1$ on the quadrature points of a Gauss-Legendre-Lobatto formula, as the one considered here, this means that it is possible to employ a Gauss-Legendre formula with $r+1$ quadrature points along each coordinate direction for the numerical integration.

The main drawback of this approach is that an aliasing error is introduced. Since numerical integration based on the Gauss-Legendre formula with $r+1$ quadrature points integrates exactly polynomials up to degree $2r+1$, the strategy presented in [7] does not introduce aliasing errors for classical mass matrices that consist of the product of two basis function. However, the matrix $\mathbf{A}^{(l)}$ (8) is a modified mass matrix and consists of the product between three polynomials. Hence, an aliasing error is introduced for $r > 1$. The numerical integration for the results shown so far is based on the so-called consistent integration or over-integration (see [14] for further details). More specifically, a Gauss-Legendre formula with $2r+1$ quadrature points

along each coordinate direction is employed. Notice that the choice to consider a polynomial representation for density, velocity, and pressure avoids any polynomial division in the case of the ideal gas law and therefore all the integrals can be computed without aliasing errors thanks to the aforementioned consistent integration. Hence, the choice of the numerical quadrature formula is related to the need to integrate all the terms so as to avoid aliasing errors.

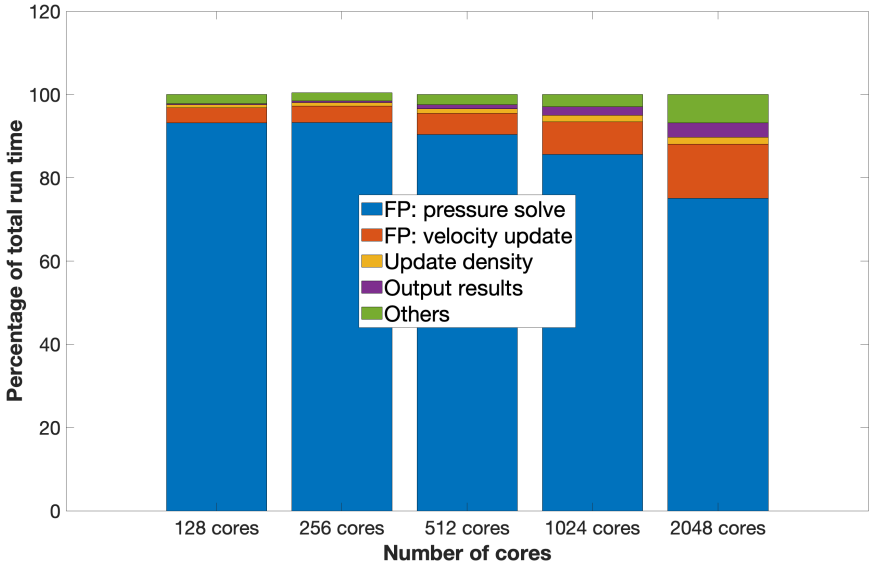

Figure 4: Distribution of the computational time spent in various blocks of the algorithm for the solution of problem (2), flow over a 3D hill test case as a function of number of cores.

The strategy presented in [7] is already available in the `deal.II` library, but it was not considered in our previous work in order to avoid aliasing errors, as mentioned above. In this work we show the impact of this strategy on the numerical results and on the parallel performance. First, we analyze the impact of the use of the fast evaluation of block-diagonal operators in terms of accuracy. We consider a uniform mesh composed of $N_{el} = 60 \times 40 \times 16 = 38400$ elements, i.e. a resolution of 250 m with a final time $T_f = 1$ h. An excellent agreement is established between the two numerical strategies to solve the linear systems (Figure 5). A computational time saving of around the 20% is established using the fast evaluation of the block diagonal operators. Hence, this first test suggests that the fast evaluation of block-diagonal operators provides similar results in terms of accuracy in spite of the aliasing error. We aim to further investigate this matter in future work.

Finally, we show the impact of the fast evaluation of the block-diagonal operators on the parallel performance. One can easily notice how the computational time spent in computing the velocity field is significantly reduced (orange bars in Figure 6). More specifically, the percentage with respect to the run time is reduced to around 0.52% with 2048 cores. The strong scaling (Figure 7, top-left) when using the fast evaluation of the block-diagonal operators (red line) is apparently worse than when using consistent integration (black line) because the evaluation of the block-diagonal operators is so efficient that a small amount of time is required even for a relatively small number of cores (Figure 7, bottom-left). This consideration is further confirmed by the sizeable advantage in time-to-solution - 53% with 128 cores, about 47% with 2048 cores - brought by the fast evaluation of the block-diagonal operators (Figure 7, top-right). Moreover, one can easily notice that most of the computational time is required to solve (10) to compute the the pressure field (Figure 7, bottom-right).

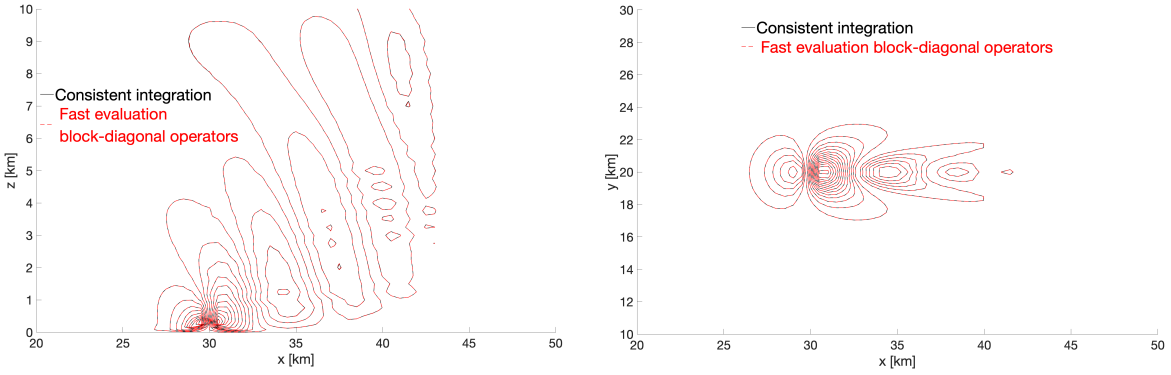

Figure 5: Flow over 3D hill test case, computed vertical velocity at $T_f = 1\,\text{h}$. Left: $x - z$ slice at $y = 20\,\text{km}$. Right: $x - y$ slice at $z = 20\,\text{km}$. Continuous black lines show the results with the consistent integration, whereas dashed red lines report the results with the fast evaluation of block-diagonal operators.

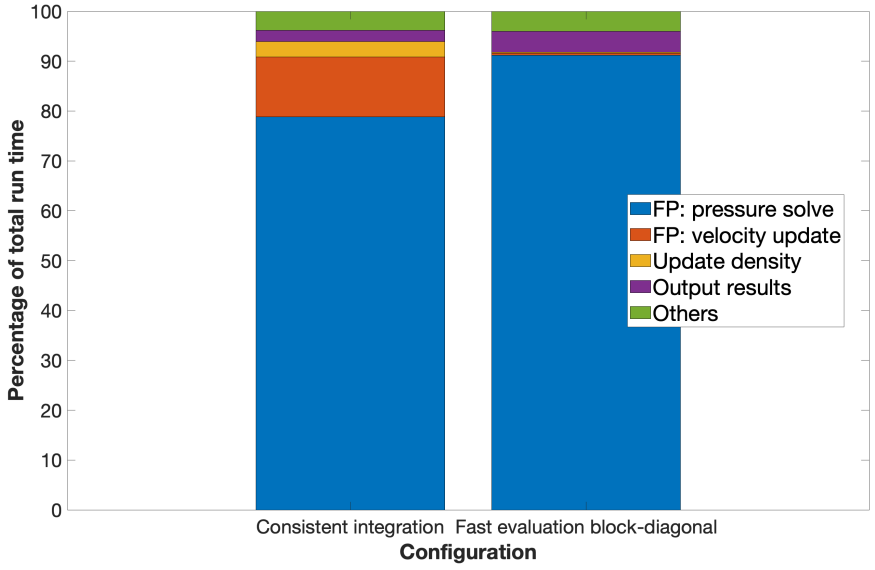

Figure 6: Distribution of the computational time spent in various blocks of the algorithm for the solution of problem (2), flow over a 3D hill test case with 2048 cores. Left: consistent numerical integration. Right: fast evaluation of block-diagonal operators.

4 Conclusions

This paper has reported the recent advances in the parallel performance of the IMEX-DG model for atmospheric dynamics simulations presented in [12, 13] and follows the analysis presented in [16]. First, we have shown the impact of the use of a suitable preconditioner, then the impact in the use of suitable execution flags and latest version of `deal.II`, and, finally, the use of efficient matrix-free techniques for block-diagonal operators [7].

In future work, we aim to further analyze the theoretical properties of the preconditioner, to further investigate the impact of the use of the direct inversion of the block-diagonal operator, and to exploit this technique to build a more suitable algebraic/geometric preconditioner since $(\mathbf{A}^{(l)})^{-1}$ can be readily evaluated. However, we point out that no theoretical result is available about the convergence properties of matrix-free multigrid preconditioners for non-symmetric

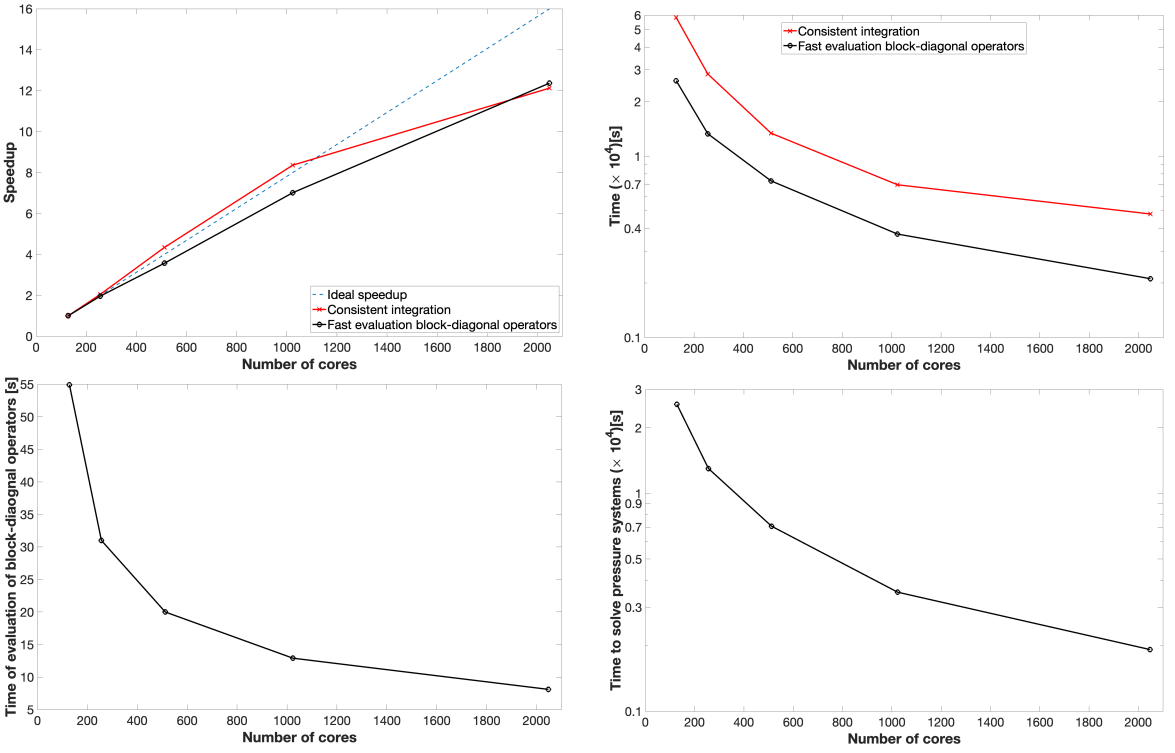

Figure 7: Parallel performance of the solver for problem (2), flow over a 3D hill test case with fast evaluation of block-diagonal operators. Top-left: strong scaling. Top-right: wall-clock times. Bottom-left: wall-clock times of evaluation of block-diagonal operators. Bottom-right: wall-clock times to solve linear systems for the pressure field (10). In the top panels, the black lines show the results with the fast evaluation of block-diagonal operators, while the red lines report the results obtained employing the consistent integration.

linear systems arising from hyperbolic problems. Moreover, we plan to include more complex physical phenomena, in particular moist air, and to develop a three-dimensional dynamical core in spherical geometry including rotation so as to enable the testing on more realistic atmospheric flows. Finally, the data locality and high parallel efficiency of the DG method as well as the matrix-free approach which performs several computations on the fly makes the numerical method employed in this work particularly well-suited for a GPU implementation. Currently the `deal.II` library does not support discontinuous finite elements for GPU use. However, there is an ongoing effort of the `deal.II` community to provide a more complete and efficient GPU infrastructure including support for discontinuous finite elements. We aim to provide and test a GPU implementation once the library will be ready for this purpose.

Acknowledgements

The simulations have been run thanks to the computational resources made available through the EuroHPC JU Benchmark And Development project EHPC-DEV-2024D10-054. We thank the Application Support EPICURE Team and in particular W.A. Mainassara for the support and help. This work has been partly supported by the ESCAPE-2 project, European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 800897).

References

[1] P.C. Africa et al. "The deal.II library, Version 9.6". *Journal of Numerical Mathematics* 32 (2024), pp. 369–380.

[2] D. Arndt et al. "The deal.II library, Version 9.5". *Journal of Numerical Mathematics* 31 (2023), pp. 231–246.

[3] W. Bangerth, R. Hartmann, and G. Kanschat. "deal.II: a general-purpose object-oriented finite element library". *ACM Transactions on Mathematical Software* 33 (2007), pp. 24–51.

[4] T. Benacchio, W.P. O'Neill, and R. Klein. "A blended soundproof-to-compressible numerical model for small-to mesoscale atmospheric dynamics". *Monthly Weather Review* 142 (2014), pp. 4416–4438.

[5] F.X. Giraldo. *An Introduction to Element-Based Galerkin Methods on Tensor-Product Bases.* Springer Nature, 2020.

[6] C.A. Kennedy and M.H. Carpenter. "Additive Runge-Kutta schemes for convection-diffusion-reaction equations". *Applied Numerical Mathematics* 44 (2003), pp. 139–181.

[7] M. Kronbichler and K. Kormann. "Fast matrix-free evaluation of discontinuous Galerkin finite element operators". *ACM Transactions on Mathematical Software (TOMS)* 45 (2019), pp. 1–40.

[8] M. Kronbichler et al. "Comparison of implicit and explicit hybridizable discontinuous Galerkin methods for the acoustic wave equation". *International Journal for Numerical Methods in Engineering* 106 (2016), pp. 712–739.

[9] C. Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". *Journal of research of the National Bureau of Standards* 45 (1950), pp. 255–282.

[10] H.W. Lean et al. "The hectometric modelling challenge: Gaps in the current state of the art and ways forward towards the implementation of 100-m scale weather and climate models". *Quarterly Journal of the Royal Meteorological Society* 150 (2024), pp. 4671–4708.

[11] T. Melvin et al. "A mixed finite-element, finite-volume, semi-implicit discretization for atmospheric dynamics: Cartesian geometry". *Quarterly Journal of the Royal Meteorological Society* 145 (2019), pp. 2835–2853.

[12] G. Orlando, P.F. Barbante, and L. Bonaventura. "An efficient IMEX-DG solver for the compressible Navier-Stokes equations for non-ideal gases". *Journal of Computational Physics* 471 (2022), p. 111653.

[13] G. Orlando, T. Benacchio, and L. Bonaventura. "An IMEX-DG solver for atmospheric dynamics simulations with adaptive mesh refinement". *Journal of Computational and Applied Mathematics* 427 (2023), p. 115124.

[14] G. Orlando, T. Benacchio, and L. Bonaventura. "Impact of curved elements for flows over orography with a Discontinuous Galerkin scheme". *Journal of Computational Physics* 519 (2024), p. 113445.

[15] G. Orlando, T. Benacchio, and L. Bonaventura. "Robust and accurate simulations of flows over orography using non-conforming meshes". *Quarterly Journal of the Royal Meteorological Society* 150 (2024), pp. 4750–4770.

[16] G. Orlando, T. Benacchio, and L. Bonaventura. "Efficient and scalable atmospheric dynamics simulations using non-conforming meshes". *Procedia Computer Science* 255 (2025). Proceedings of the Second EuroHPC user day, pp. 33–42.

[17] G. Orlando and L. Bonaventura. "Asymptotic-preserving IMEX schemes for the Euler equations of non-ideal gases". *Journal of Computational Physics* 529 (2025), p. 113889.

[18] G. Orlando, S. Boscarino, and G. Russo. *A quantitative comparison of high-order asymptotic-preserving and asymptotically-accurate IMEX methods for the Euler equations with non-ideal gases*. 2025. arXiv: 2501.12733.

[19] L. Pareschi and G. Russo. "Implicit-explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxation". *Journal of Scientific computing* 25 (2005), pp. 129–155.

[20] J. Steppeler et al. "Review of numerical methods for nonhydrostatic weather prediction models." *Meteorology and Atmospheric Physics* 82 (2003), pp. 287–301.

[21] R.S. Varga. *Matrix Iterative Analysis*. Springer Science & Business Media, 2009.