Towards dynamic adaptive mesh refinement in Earth system models

Kerstin Hartung, Benedict Geihe

April 29, 2025

1 Motivation

The currently available computing power severely limits the spatial resolution in chemistry climate simulations, even on upcoming exascale machines. This is mainly due to the large number of prognostic variables, which includes chemical tracers. To further enhance the reliability and accuracy of climate projections, smaller scales need to be resolved. Adaptive methods enable a continuously re-adjusted focus of computational power in time and space. This considerably increases the achievable level of detail, while reducing the time to solution and resource consumption. However, adaptivity requires a sophisticated selection of adaptation criteria, algorithms, memory layouts, and communication patterns to fully utilize modern HPC infrastructures.

Discontinuous Galerkin methods present another approach to improve the accuracy of a simulation, as they promise to increase the effective resolution, i.e. by employing high-order polynomials, so that prognostic variables are better resolved, even on coarser meshes. Most of the additional computation is done locally, so that the overall algorithm is ideally suited for parallel execution. The typical lack of robustness of higher-order methods can be remedied by utilizing state-of-the-art entropy stable schemes.

In this conference contribution, we present the setup and the interfaces between MESSy, Trixi.jl and t8code as well as results from a prototypical simulation that showcases the interaction, application, and challenges of dynamic adaptive meshes.

Here, MESSy is a software framework that enables the integration of multiple numerical model components to build regional and global chemistry climate models. t8code is a parallel mesh management library written in C++. Finally, Trixi.jl is a computational fluid dynamics solver, built around a modern Discontinuous Galerkin method, and written in the Julia programming language. This work was performed within the ADAPTEX project (ADAPtive Earth system modelling with strongly reduced computation time for EXascale-supercomputers), which aims to evaluate the potential benefit of dynamic adaptive meshes in ESM.

2 Methods

In the following, we will describe the mathematical and numerical methods as well as the software used. Not all features of the software components are described, but a focus is mainly put on those relevant for the topics raised in the conference contribution.

2.1 Governing equations

The three-dimensional compressible Euler equations are chosen as a model for the resolved dynamics of dry air in atmospheric flows. In Cartesian coordinates and conservative form they read as follows.

$$\partial_t \begin{pmatrix} \rho \\ \rho v \\ \rho e \end{pmatrix} + \nabla \cdot \begin{pmatrix} \rho v \\ \rho v \otimes v + p\mathbb{1} \\ v(\rho e + p) \end{pmatrix} = \begin{pmatrix} 0 \\ -\rho \nabla \Phi_g - 2\rho\Omega \times v \\ -\rho v \cdot \nabla \Phi_g \end{pmatrix} \tag{1}$$

The unknowns are the fluid density ρ, the three-dimensional momentum density ρv in Cartesian components, and the total energy density ρe, given by

$$e = c_v T + \frac{1}{2}|v|^2 \tag{2}$$

with temperature T, and heat capacity at constant volume c_v. Moreover, Φ_g is the gravitational geopotential, Ω denotes Earth's axis of rotation and its angular speed $|\Omega|$, and p is the pressure, related to the energy

densities via

$$p = (\gamma - 1)\left(\rho e - \frac{1}{2}\rho \left\|\vec{v}\right\|^2\right) = \rho R T\,, \tag{3}$$

where an ideal gas is assumed and $\gamma = 1.4$ is the adiabatic index. Alternatively, it is possible to include the geopotential in the energy considered, i. e.

$$e = c_v T + \frac{1}{2}|v|^2 + \Phi_g\,. \tag{4}$$

In this case, the third entry of the right hand side of equation (1) is zero.

2.2 t8code

`t8code` [1] is a mesh management library written in C++. It focuses on dynamic adaptive mesh refinement (AMR), providing efficient algorithms for mesh adaptation, load-balancing, and ghost computations. Several element types and hybrid meshes are supported. `t8code` can handle meshes with over one trillion elements and scales up to one million parallel processes.

2.3 Trixi.jl

`Trixi.jl` [2] is an open-source Julia package for adaptive high-order simulations of conservative and non-conservative systems. It is extensively used in computational fluid dynamics, including applications in compressible fluid flow, astrophysics, aeroacoustics, and atmospheric flows. A central focus of `Trixi.jl` is facilitating research on novel models and methods while ensuring accessibility for new users and high performance for large-scale simulations.

`Trixi.jl` implements high-order discontinuous Galerkin spectral element methods (DGSEM). It supports various mesh types, including one- to three-dimensional meshes, Cartesian, curvilinear, and non-conforming grids. Adaptive mesh refinement is supported via the `p4est` [3] and `t8code` [1] (see Sec. 2.2) libraries. Robustness is ensured by novel entropy- and energy-stable flux formulations with subcell limiting strategies for shock capturing. Time integration is performed using low-storage Runge-Kutta schemes. More details on the numerical methods in `Trixi.jl` can be found in [4, 5].

The single-thread performance was shown to be on par with classical HPC codes in [5]. For parallel execution, `Trixi.jl` combines MPI and multithreading. Scalability has been demonstrated for flow simulations with more than 50 000 CPU cores.

2.4 libtrixi

`Trixi.jl` is written in the modern programming language Julia. Such high-level languages typically offer convenient interfaces to libraries compiled with classical programming languages. For our application, this pattern has be reversed. `MESSy`, written in Fortan, has to call `Trixi.jl`. For this purpose, we created `libtrixi` [6], an interface library for using `Trixi.jl` from C, C++, or Fortran. The API currently allows for controlling the simulation and basis mutual data exchange. Figure 1 gives an overview.

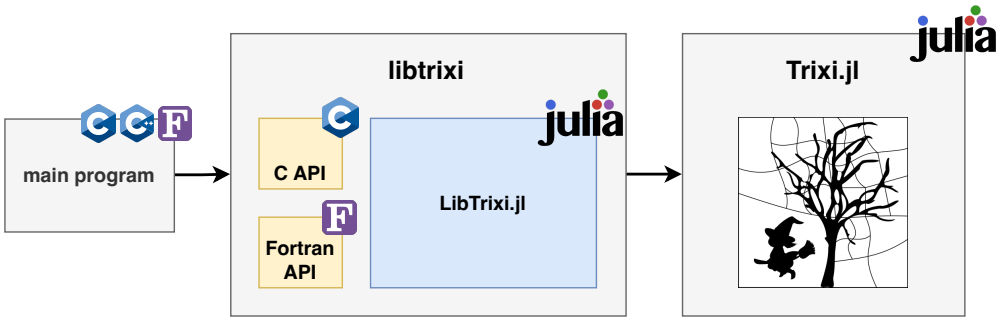

Figure 1: `libtrixi`, an interface library for using `Trixi.jl` from C, C++, or Fortran

2.5 MESSy

MESSy [7, 8], the Modular Earth Submodel System, is a software framework that couples numerical representations of our Earth system (so-called legacy models (e.g., ICON, ECHAM, COSMO)), with specialized ESM components, the so-called submodels (e.g., physical parameterizations, chemistry packages, diagnostics) via generalized interfaces for standardized control and coupling. Currently more than 100 submodels are included in MESSy. MESSy is mainly written in Fortran, but some libraries written in C/C++ are used.

Among a wide variety of applications, MESSy is applied to build global and regional chemistry climate models for studies on atmospheric chemistry and air quality.

As a Chemistry Climate Model (CCM), MESSy usually deals with setups containing some hundred chemical tracers, in some setups up to more than 1000. These chemical tracers are subject to different source and sink processes, thus making a CCM simulation much more demanding than pure dynamical model applications in terms of computing, energy, and memory consumption.

The goal within this work for MESSy is to benefit from dynamic AMR methods by running the dynamical core (and tracer transport) provided by Trixi.jl on an adaptive mesh maintained by t8code. In addition, use of an advanced DG scheme within Trixi.jl further increases the effective resolution per degree of freedom.

3 Setup of prototype for classical test cases

To demonstrate the applicability of the framework, we present results for the classical baroclinic instability [9] and Held-Suarez [10] test cases. Earth's atmosphere is discretized using a cubed sphere mesh, see Figure 2. We use `Trixi.jl`'s Discontinuous Galerkin spectral element method on hexahedral elements with tensor

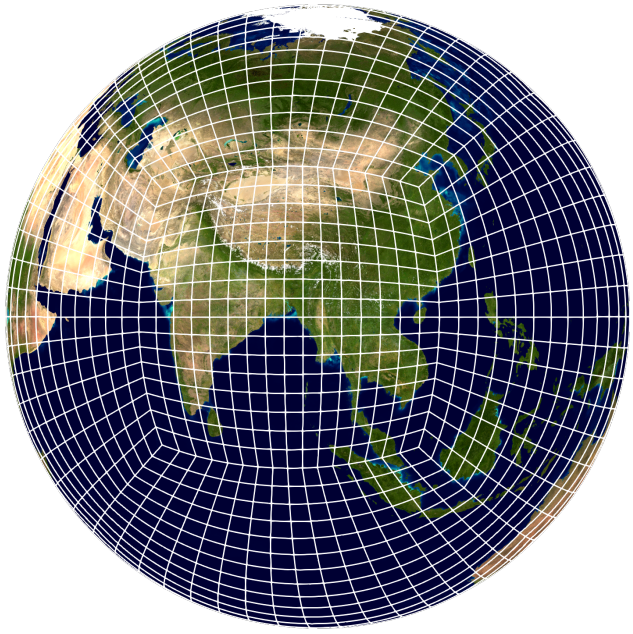

Figure 2: Cubed sphere discretization of Earth's atmosphere [11].

product Lagrange polynomial basis functions and Gauss-Lobatto quadrature. Currently, `Trixi.jl` does not support different discretization schemes in certain dimensions. Likewise, it is also not possible to use implicit time integration in the vertical direction, as is commonly done in atmospheric simulations. The fully-explicit time integrator, taken from the package `OrdinaryDiffEq.jl`, is therefore limited by the vertical cell size. However, for DG methods typically rather coarse meshes can be used.

3.1 Baroclinic instability

Our setup closely follows the exposition in [12]. We use $6 \times 32 \times 32 \times 16$ cells in the cubed sphere, leading to a total of 98 304 mesh cells. With polynomials of degree 5 in each spatial direction, there are 6^3 degrees of freedom for each unknown on every element, amounting to total count of about 2.5 million cells. The adaptive time integration scheme used in this setup settled at a time step length of about 2.14 s. A simple Fortran program was controlling the simulation via `libtrixi`.

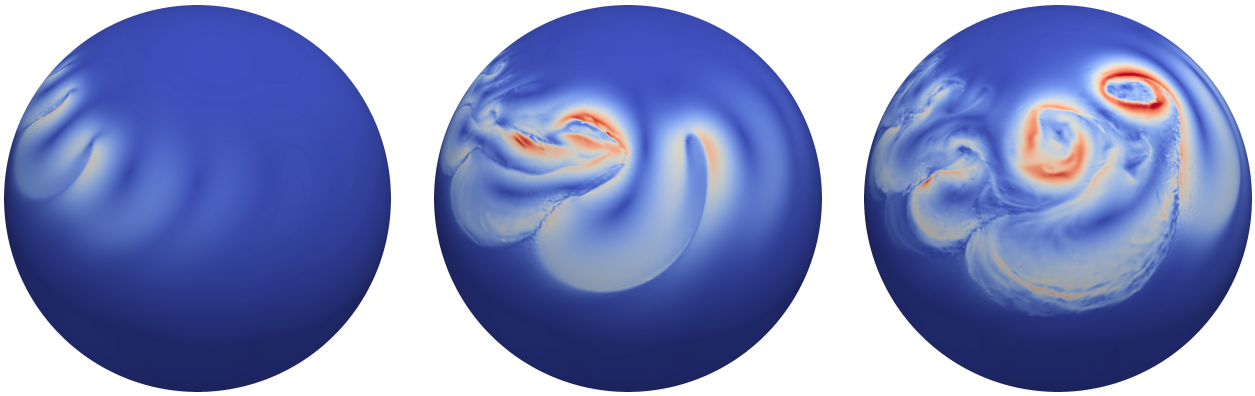

Figure 3: Velocity magnitude near surface after about 7.5, 10 and 12 days for the baroclinic instability test case.

3.2 Held-Suarez test case

For the Held-Suarez test case we follow the approach taken in [13], where in particular a flux differencing scheme, as in `Trixi.jl`, is employed. Furthermore the alternative energy formulation included the gravitational geopotential (4) is considered and special treatment for the remaining gravity term $-\rho\nabla\Phi_g$ in the right hand side of the momentum equation (1) is applied. The discretization uses $6 \times 10 \times 10 \times 8$ cells only, and the time step length was about 6.4 s.

3.3 Tracer advection with AMR

In `Trixi.jl` it is possible to add an arbitrary number of additional species, represented by individual densities, to the underlying equation set. These densities are then evolved analogously to the main flow, but they do not add to it or interact.

In a simple example, we initialize the flow field with a perturbation in density and a constant angular speed on a cubed sphere mesh with $6 \times 4 \times 4 \times 1$ cells. We also use this test case to demonstrate the AMR functionality in `Trixi.jl`. It is straightforward to define an error indicator based on the density. We use a controller which refines cells once a threshold value has been reached. In this case, a hexahedron is split into 8 subcells. The controller refines up to another two levels once a second threshold value is approached.

4 Results

4.1 Baroclinic instability

Figure 3 shows the buildup of the baroclinic instability, visualized by the velocity magnitude. It was shown in [11] that comparable results can already be obtained on a coarse mesh with only $6 \times 8 \times 8 \times 4$ cells. To demonstrate the parallel computing capabilities, we conducted a strong scaling test for this setup, starting from 96 MPI ranks and going up to 6144 ranks. As can be seen in Figure 4, the setup scales well.

4.2 Held-Suarez case

Figure 5 shows the temporal evolution of near-surface temperature in the Held-Suarez test case. Similar to the baroclinic instability test, features on different scales can be resolved.

4.3 AMR tracer advection

Figure 6 shows a few snapshots of the tracer advection and clearly indicates the success of the adaptation procedure. Although the blob is small, compared to the coarse mesh cells, the advected perturbation is well resolved and dynamically tracked. The maximum number of cells during the simulation is 453. A uniform grid with the same resolution would have 49 152 cells.

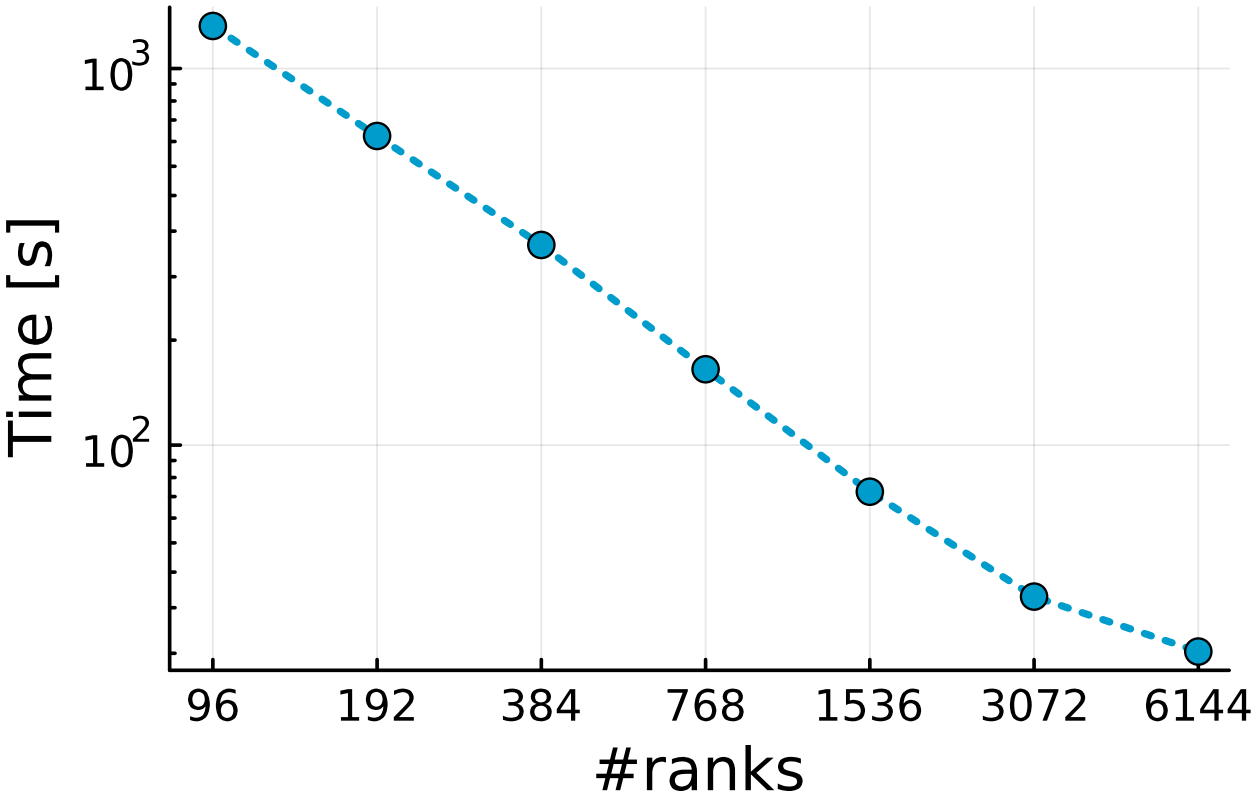

Figure 4: Strong scaling for the baroclinic instability. Each simulation was run for 1 h simulated time.

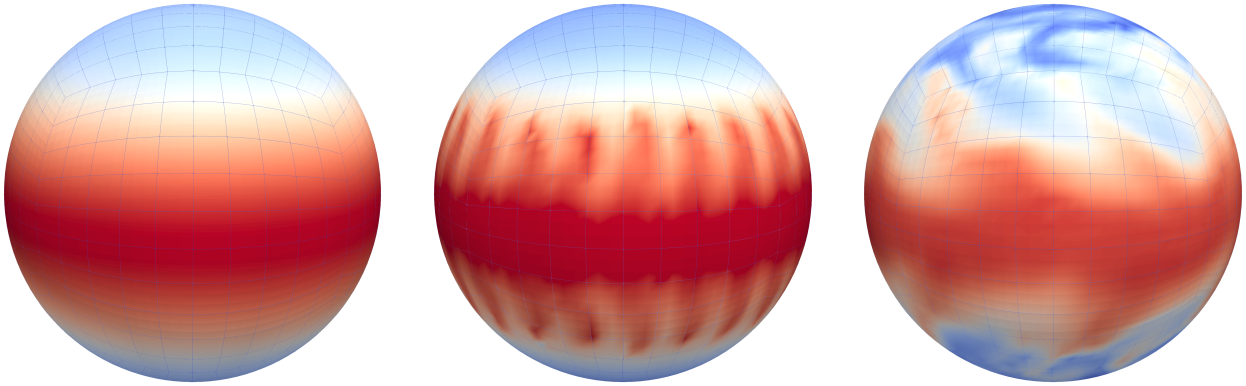

Figure 5: Temperature near surface after about 41, 71 and 127 days for the Held-Suarez test case.

5 Outlook

We are currently looking at additional classical test cases for atmospheric flows. Among them are setups with moist air and cloud microphysics, which require extending the compressible Euler equations used currently. The results will have to be validated, e.g. against the literature. The adaptive scheme will be tested with different error indicators. Here, we have to make sure that the quality of the underlying flow solution is maintained while we refine for a specific prescribed criterion. With respect to the overall infrastructure, the integration into the `MESSy` framework has to be finalized. It will then be possible to directly compare the results of `MESSy` simulations running with the current and the new methods, respectively.

6 Reproducibility

We have created a repository at `https://github.com/trixi-framework/talk-2024-juliacon-libtrixi`, which contains a step-by-step guide on how to set up `libtrixi` and reproduce the results obtained for the

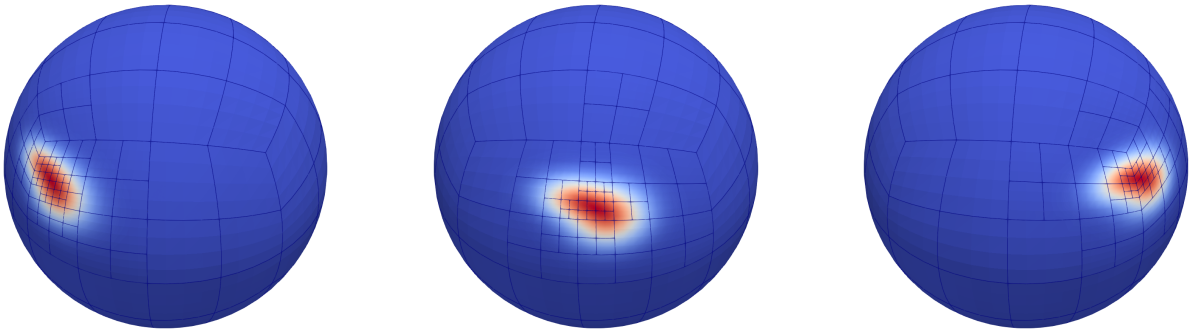

Figure 6: Perturbation in density advected with constant angular speed on an adaptively refined mesh.

baroclinic instability test case.

7 Acknowledgements

This work has been funded by the Federal Ministry of Education and Research (BMBF) within the project „ADAPTEX" under the funding code 16ME0668K, and thereby also through the European Union - NextGenerationEU. The views and opinions expressed are solely those of the author(s) and do not necessarily reflect the views of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them. The authors gratefully acknowledge the resources granted through the ESM partition on the supercomputer JUWELS at the Jülich Supercomputing Centre, Germany.

References

[1] J. Holke, J. Markert, D. Knapp, L. Dreyer, S. Elsweijer, N. Boeing, I. Lilikakis, J. Fußbroich, T. Leistikow, F. Becker, V. Uenlue, O. Albers, C. Burstedde, A. Basermann, C. Hergl, W. Julia, K. Schoenlein, J. Ackerschott, A. Evgenii, Z. Csati, A. Dutka, B. Geihe, P. Kestener, A. Kirby, H. Ranocha, and M. Schlottke-Lakemper. *t8code - modular adaptive mesh refinement in the exascale era*. Version v3.0.1-JOSS. Dec. 2024. DOI: 10.5281/zenodo.14418226. URL: https://doi.org/10.5281/zenodo.14418226.

[2] M. Schlottke-Lakemper, G. J. Gassner, H. Ranocha, A. R. Winters, and J. Chan. *Trixi.jl: Adaptive high-order numerical simulations of hyperbolic PDEs in Julia*. https://github.com/trixi-framework/Trixi.jl. Sept. 2021. DOI: 10.5281/zenodo.3996439.

[3] C. Burstedde, L. C. Wilcox, and O. Ghattas. "p4est : Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees." In: *SIAM Journal on Scientific Computing* 33.3 (Jan. 2011), pp. 1103–1133. DOI: 10.1137/100791634.

[4] M. Schlottke-Lakemper, A. R. Winters, H. Ranocha, and G. J. Gassner. "A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics." In: *Journal of Computational Physics* 442 (June 2021), p. 110467. DOI: 10.1016/j.jcp.2021.110467. arXiv: 2008.10593 [math.NA].

[5] H. Ranocha, M. Schlottke-Lakemper, J. Chan, A. M. Rueda-Ramírez, A. R. Winters, F. Hindenlang, and G. J. Gassner. *Efficient implementation of modern entropy stable and kinetic energy preserving discontinuous Galerkin methods for conservation laws*. 2021. arXiv: 2112.10517.

[6] M. Schlottke-Lakemper, B. Geihe, and G. J. Gassner. *libtrixi*. Version v0.1.5. Mar. 2024. DOI: 10.5281/zenodo.10784457. URL: https://doi.org/10.5281/zenodo.10784457.

[7] P. Jöckel, R. Sander, A. Kerkweg, H. Tost, and J. Lelieveld. "Technical Note: The Modular Earth Submodel System (MESSy) - a new approach towards Earth System Modeling." In: *Atmospheric Chemistry and Physics* 5.2 (2005), pp. 433–444. DOI: 10.5194/acp-5-433-2005. URL: https://acp.copernicus.org/articles/5/433/2005/.

[8] P. Jöckel, A. Kerkweg, A. Pozzer, R. Sander, H. Tost, H. Riede, A. Baumgaertner, S. Gromov, and B. Kern. "Development cycle 2 of the Modular Earth Submodel System (MESSy2)." In: *Geoscientific Model Development* 3.2 (2010), pp. 717–752. DOI: 10.5194/gmd-3-717-2010. URL: https://gmd.copernicus.org/articles/3/717/2010/.

[9] C. Jablonowski and D. Williamson. "A baroclinic instability test case for atmospheric model dynamical cores." In: *Q. J. R. Meteorol. Soc.* (2006). DOI: https://doi.org/10.1256/qj.06.12. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1256/qj.06.12. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1256/qj.06.12.

[10] I. M. Held and M. J. Suarez. "A Proposal for the Intercomparison of the Dynamical Cores of Atmospheric General Circulation Models." In: *Bulletin of the American Meteorological Society* 75.10 (1994), pp. 1825–1830. DOI: 10.1175/1520-0477(1994)075<1825:APFTIO>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/bams/75/10/1520-0477_1994_075_1825_apftio_2_0_co_2.xml.

[11] E. Faulhaber. "Discontinuous Galerkin Methods for Atmospheric Simulations on Hierarchical Meshes in Julia." Master's Thesis. University of Cologne, 2022.

[12] P. A. Ullrich, T. Melvin, C. Jablonowski, and A. Staniforth. "A proposed baroclinic wave test case for deep- and shallow-atmosphere dynamical cores." In: *Quarterly Journal of the Royal Meteorological Society* 140.682 (2014), pp. 1590–1602. DOI: https://doi.org/10.1002/qj.2241. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.2241.

[13] A. N. Souza, J. He, T. Bischoff, M. Waruszewski, L. Novak, V. Barra, T. Gibson, A. Sridhar, S. Kandala, S. Byrne, L. C. Wilcox, J. Kozdon, F. X. Giraldo, O. Knoth, J. Marshall, R. Ferrari, and T. Schneider. "The Flux-Differencing Discontinuous Galerkin Method Applied to an Idealized Fully Compressible Nonhydrostatic Dry Atmosphere." In: *Journal of Advances in Modeling Earth Systems* 15.4 (2023). e2022MS003527 2022MS003527, e2022MS003527. DOI: https://doi.org/10.1029/2022MS003527. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2022MS003527. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022MS003527.