The Journal of Open Source Software

SpeedyWeather.jl: Reinventing atmospheric general circulation models towards interactivity, extensibility and composability

Milan Klöwer[1,2¶], Maximilian Gelbrecht[3,4], Daisuke Hotta[5,6], Justin Willmert[7], Simone Silvestri[1], Gregory L Wagner[1], Alistair White[3,4], Sam Hatfield[8], David Meyer[8], Tom Kimpson[2,9], Navid C Constantinou[10], and Chris Hill[1]

1 Massachusetts Institute of Technology, Cambridge, MA, USA **2** University of Oxford, UK **3** Technical University Munich, Germany **4** Potsdam Institute for Climate Impact Research, Germany **5** Japan Meteorological Agency, Tsukuba, Japan **6** University of Reading, UK **7** University of Minnesota, Minneapolis, MN, USA **8** European Centre for Medium-Range Weather Forecasts, Reading, UK **9** University of Melbourne, Australia **10** Australian National University, Canberra, Australia ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

SpeedyWeather.jl is a library to simulate and analyse the global atmospheric circulation on the sphere. It implements several 2D and 3D models to solve the primitive equations with and without humidity (Figure 1), the shallow water equations (Figure 2), or the barotropic vorticity equations with spherical harmonics. Several simple parameterizations for unresolved physical processes such as precipitation or the boundary layer are implemented, and new ones can be externally defined and passed as an argument to the model constructor. SpeedyWeather.jl is an intermediate-complexity general circulation model (Kucharski et al., 2013) and research playground with an (almost) everything-flexible attitude. It can be thought of as a conceptual reinvention of the Fortran SPEEDY model (Molteni, 2003) in the Julia programming language (Bezanson et al., 2017).

SpeedyWeather.jl internally uses three sub-modules `SpeedyTransforms`, `RingGrids`, and `LowerTriangularMatrices` to perform spherical harmonic transforms and interpolations between various grids and the spectral space. `RingGrids` discretize the sphere on iso-latitude rings and the spectral space is defined by the `LowerTriangularMatrices` of the spherical harmonic coefficients. These three modules are independently usable and therefore make SpeedyWeather.jl, beyond its main purpose of simulating the weather, also a library for the analysis of gridded data on the sphere. Running and analysing simulations can interactively combined, enhancing user experience and productivity.

The user interface of SpeedyWeather.jl is heavily influenced by the Julia ocean model Oceananigans.jl (Ramadhan et al., 2020). A monolithic interface is deliberately avoided, instead, a model is created bottom-up by first defining the discretization and any non-default model components with its respective parameters. All components are then collected into a single model object, which, once initialized, returns a simulation object that contains the entire model state, work arrays and parameters, that can be run, analysed or changed. While these steps can be written into a script for reproducibility, the same steps can be executed and interacted with one-by-one in Julia's read-evaluate-print loop (REPL). We thereby reach an interactivity far beyond a monolithic interface that is limited to the options provided.

To be extendible and composable with new model components, SpeedyWeather.jl relies on Ju-

lia's multiple dispatch programming paradigm (Bezanson et al., 2017). Every model component is defined as a new type. For example, to define a new way how to calculate the precipitation due to the physical process of large-scale condensation, one would define `MyCondensation` as a new subtype of `AbstractCondensation`. One then only needs to extend the `initialize!` and `condensation!` functions for this new type. Passing on `condensation = MyCondensation()` to the model constructor then implements this new model component without the need to branch off or overwrite existing model components. Conceptually similar scientific modelling paradigms have been very successful in the Python-based generic partial differential equation solver Dedalus (Burns et al., 2020) and the Julia ocean model Oceananigans.jl (Ramadhan et al., 2020).

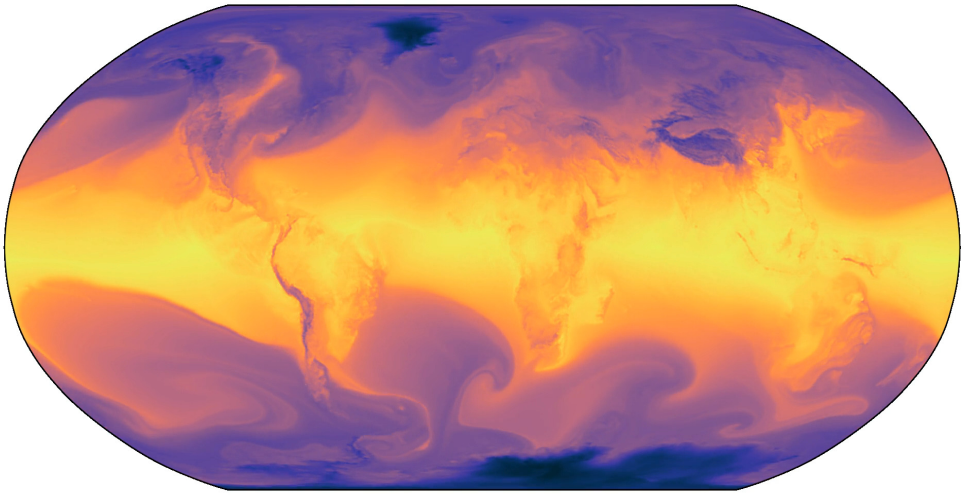

Figure 1: Surface temperature simulated with the primitive equation model in SpeedyWeather.jl. (Figure will be updated)

The dynamical core of SpeedyWeather.jl uses established numerics (Bourke, 1972; Hoskins & Simmons, 1975; Simmons et al., 1978; Simmons & Burridge, 1981), widely adapted in numerical weather prediction. It is based on the spherical harmonic transform with a leapfrog-based semi-implicit time integration (Hoskins & Simmons, 1975) and a Robert-Asselin-Williams filter (Amezcua et al., 2011; Williams, 2011). The spherical harmonic transform is grid-flexible. Any iso-latitude ring-based grid can be used and new grids can be externally defined and passed on as argument. Many grids are already implemented: The conventional Gaussian grid, a regular longitude-latitude grid, the octahedral Gaussian grid (Malardel et al., 2016), the octahedral Clenshaw-Curtis grid (Hotta & Ujiie, 2018), and the HEALPix grid (Górski et al., 2005). Both SpeedyWeather.jl and its spherical harmonic transform are also number format-flexible. 32-bit single-precision floating-point numbers (Float32) are the default as adapted by other modelling efforts Nakano et al. (2018), but Float64 and other custom number formats can be used with a single code basis. Julia will compile to the choice of the number format, the grid, and and other model components just-in-time. A simple parallelisation across vertical layers is supported with Julia's multi threading.

Statement of need

SpeedyWeather.jl is a fresh approach to atmospheric models that have been very influential in many areas of scientific and high-performance computing as well as climate change mitigation and adaptation. Most weather, ocean and climate models are written in Fortran and have been developed over decades. From this tradition follows a specific programming style and associated user interface. Running a simulation in Fortran and analysing the data in Python makes it virtually impossible to interact with various model components interactively. Furthermore, data-driven climate modelling (Rasp et al., 2018, p. Schneider2023), which replaces existing model components with machine learning is difficult due to the lack of established deep learning frameworks in Fortran (Innes et al., 2019). Let alone online learning, which trains a neural network-based component together with the rest of the model, accounting for interactions between components. Gradients, necessary to optimize training, can be computed with automatic differentiation (Moses & Churavy, 2020), but only if differentiable functions in a coherent language framework are provided.

We hope to provide with SpeedyWeather.jl a first test platform for data-driven atmospheric modelling and in general an interactive model that makes difficult problems easy to simulate. Climate models that are user-friendly, trainable, but also easily extensible will suddenly make many complex research ideas possible.

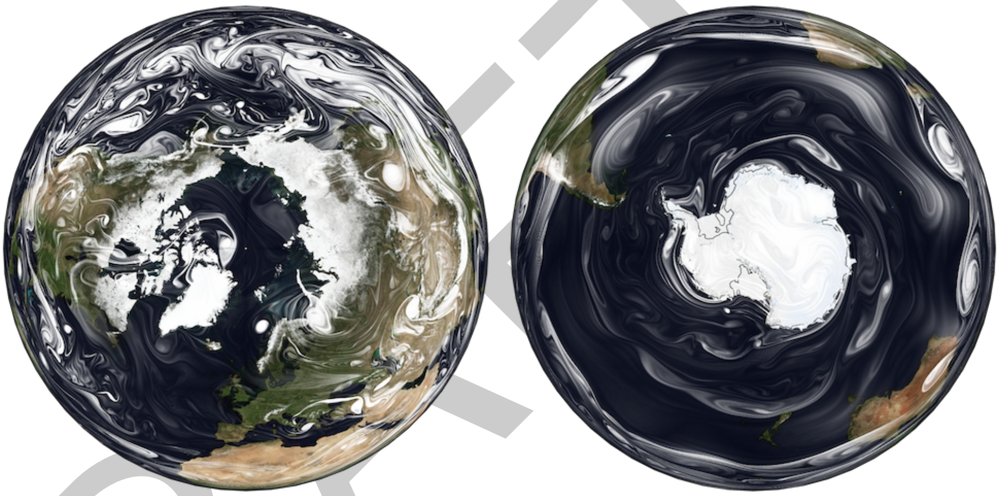

Figure 2: Relative vorticity simulated with the shallow water model in SpeedyWeather.jl. The simulation used a spectral resolution of T1023 (about 20km) and Float32 arithmetic on an octahedral Clenshaw-Curtis grid (Hotta & Ujiie, 2018). Relative vorticity is visualised with Matplotlib (Hunter, 2007) and Cartopy (Met Office, 2010 - 2015) using a transparent-to-white colormap to mimic the appearance of clouds. Underlain is NASA's blue marble from June 2004.

Acknowledgements

We acknowledge contributions from Mosè Giordano, Valentin Churavy and Pietro Monticone who have also committed to the SpeedyWeather.jl repository, and the wider Julia community for help and support. We gratefully acknowledge funding from the National Science Foundation (Chris please add) and the European Research Council under the European Union's Horizon 2020 research and innovation programme for the ITHACA grant (no. 741112).

References

Amezcua, J., Kalnay, E., & Williams, P. D. (2011). The Effects of the RAW Filter on the Climatology and Forecast Skill of the SPEEDY Model. *Monthly Weather Review*, *139*(2), 608–619. https://doi.org/10.1175/2010MWR3530.1

Bezanson, Jeff., Edelman, Alan., Karpinski, Stefan., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Bourke, W. (1972). An Efficient, One-Level, Primitive-Equation Spectral Model. *Monthly Weather Review*, *100*(9), 683–689. https://doi.org/10.1175/1520-0493(1972)100%3C0683:AEOPSM%3E2.3.CO;2

Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., & Brown, B. P. (2020). Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, *2*(2), 023068. https://doi.org/10.1103/PhysRevResearch.2.023068

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., & Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, *622*(2), 759. https://doi.org/10.1086/427976

Hoskins, B. J., & Simmons, A. J. (1975). A multi-layer spectral model and the semi-implicit method. *Quarterly Journal of the Royal Meteorological Society*, *101*(429), 637–655. https://doi.org/10.1002/qj.49710142918

Hotta, D., & Ujiie, M. (2018). A nestable, multigrid-friendly grid on a sphere for global spectral models based on Clenshaw–Curtis quadrature. *Quarterly Journal of the Royal Meteorological Society*, *144*(714), 1382–1397. https://doi.org/10.1002/qj.3282

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W. (2019). *A Differentiable Programming System to Bridge Machine Learning and Scientific Computing* (No. arXiv:1907.07587). arXiv. https://doi.org/10.48550/arXiv.1907.07587

Kucharski, F., Molteni, F., King, M. P., Farneti, R., Kang, I.-S., & Feudale, L. (2013). On the Need of Intermediate Complexity General Circulation Models: A "SPEEDY" Example. *Bulletin of the American Meteorological Society*, *94*(1), 25–30. https://doi.org/10.1175/BAMS-D-11-00238.1

Malardel, S., Wedi, N., Deconinck, N., Diamantakis, M., Kuehnlein, C., Mozdzynski, G., Hamrud, M., & Smolarkiewicz, P. (2016). A new grid for the IFS. In *ECMWF Newsletter*. https://www.ecmwf.int/node/15041.

Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*. https://scitools.org.uk/cartopy

Molteni, F. (2003). Atmospheric simulations using a GCM with simplified physical parametrizations. I: Model climatology and variability in multi-decadal experiments. *Climate Dynamics*, *20*(2), 175–191. https://doi.org/10.1007/s00382-002-0268-2

Moses, W., & Churavy, V. (2020). Instead of Rewriting Foreign Code for Machine Learning, Automatically Synthesize Fast Gradients. *Advances in Neural Information Processing Systems*, *33*, 12472–12485.

Nakano, M., Yashiro, H., Kodama, C., & Tomita, H. (2018). Single Precision in the Dynamical Core of a Nonhydrostatic Global Atmospheric Model: Evaluation Using a Baroclinic Wave Test Case. *Monthly Weather Review*, *146*(2), 409–416. https://doi.org/10.1175/MWR-D-17-0257.1

Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza, A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and friendly geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, *5*(53), 2018. https://doi.org/10.21105/joss.02018

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, *115*(39), 9684–9689.

Simmons, A. J., & Burridge, D. M. (1981). An Energy and Angular-Momentum Conserving Vertical Finite-Difference Scheme and Hybrid Vertical Coordinates. *Monthly Weather Review*, *109*(4), 758–766. https://doi.org/10.1175/1520-0493(1981)109%3C0758:AEAAMC%3E2.0.CO;2

Simmons, A. J., Hoskins, B. J., & Burridge, D. M. (1978). Stability of the Semi-Implicit Method of Time Integration. *Monthly Weather Review*, *106*(3), 405–412. https://doi.org/10.1175/1520-0493(1978)106%3C0405:SOTSIM%3E2.0.CO;2

Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., & Carver, G. (2017). Single Precision in Weather Forecasting Models: An Evaluation with the IFS. *Monthly Weather Review*, *145*(2), 495–502. https://doi.org/10.1175/MWR-D-16-0228.1

Williams, P. D. (2011). The RAW Filter: An Improvement to the Robert–Asselin Filter in Semi-Implicit Integrations. *Monthly Weather Review*, *139*(6), 1996–2007. https://doi.org/10.1175/2010MWR3601.1