

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 Willmert⁷, Simone Silvestri¹, Gregory L Wagner¹, Alistair White^{3,4}, Sam Hatfield⁶, David Meyer⁶, Tom Kimpson^{2,8}, Navid C Constantinou⁹, and Chris Hill¹

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

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))

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 be 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 based on parameter files is avoided in favor of a library-style interface in which users write code to run models rather than merely supplying parameters and input arrays. 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 extensible and composable with new model components, SpeedyWeather.jl relies on Julia'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).

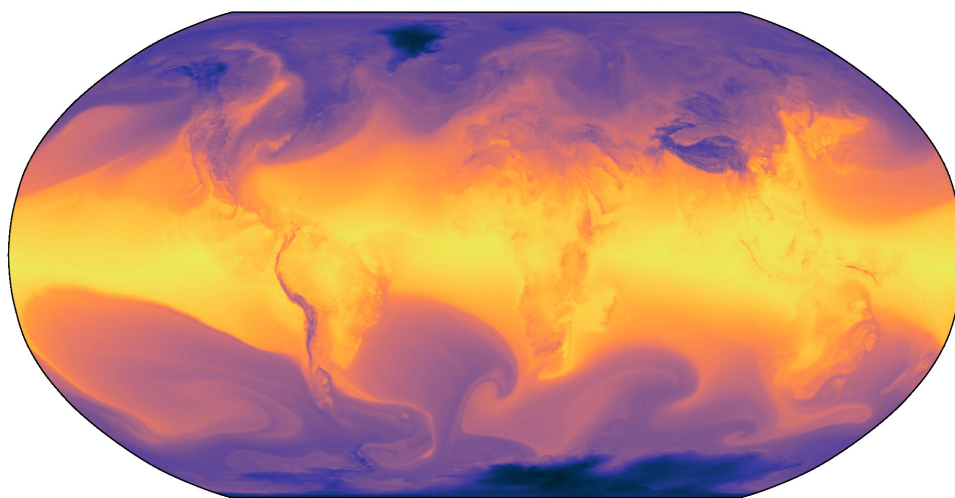


Figure 1: Surface temperature simulated with the primitive equation model in SpeedyWeather.jl. (Figure will be updated)

The dynamical core of SpeedyWeather.jl uses established numerics (Bourke, 1972; Hoskins & Simmons, 1975; Simmons et al., 1978; Simmons & Burridge, 1981), widely adopted 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 (Amezcuca 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 in as an 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 SpeedyTransforms are also number format-flexible. 32-bit single-precision floating-point numbers (Float32) are the default as adopted 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 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

74 user interface. Running a simulation in Fortran and analysing the data in Python makes it
75 virtually impossible to interact with various model components interactively. Furthermore,
76 data-driven climate modelling (Rasp et al., 2018, p. Schneider2023), which replaces existing
77 model components with machine learning is difficult due to the lack of established deep learning
78 frameworks in Fortran (Innes et al., 2019). Let alone online learning, which trains a neural
79 network-based component together with the rest of the model, accounting for interactions
80 between components. Gradients, necessary to optimize training, can be computed with
81 automatic differentiation (Moses & Churavy, 2020), but only if differentiable functions in a
82 coherent language framework are provided.

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

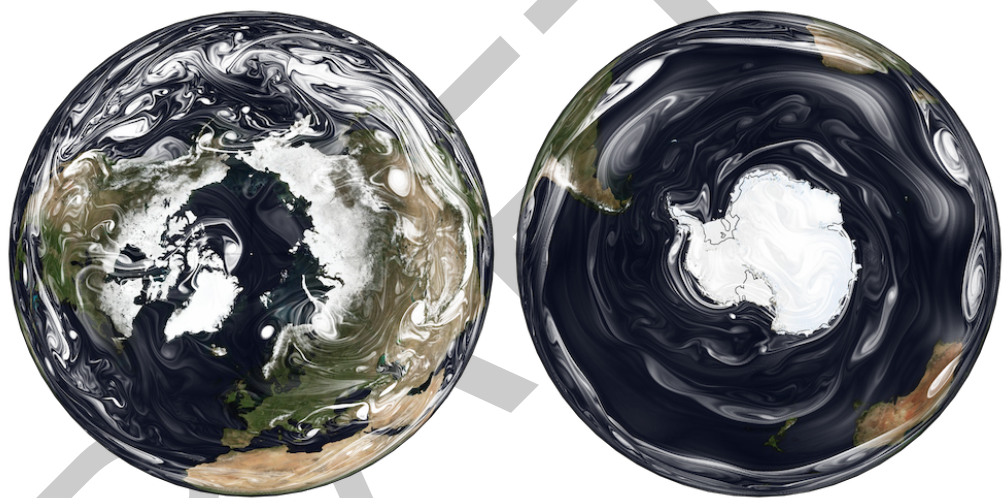


Figure 2: Relative vorticity simulated with the shallow water model in SpeedyWeather.jl. The simulation used a spectral resolution of T1023 (about 20km) and Float32 arithmetic on an octahedral Clenshaw-Curtis grid (Hotta & Ujiie, 2018). Relative vorticity is visualised with Matplotlib (Hunter, 2007) 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.

87 Acknowledgements

88 We acknowledge contributions from Mosè Giordano, Valentin Churavy, and Pietro Monticone
89 who have also committed to the SpeedyWeather.jl repository, and the wider Julia community
90 for help and support. MK acknowledges funding from the National Science Foundation (Chris
91 please add). MK and TK acknowledge funding from the European Research Council under the
92 European Union's Horizon 2020 research and innovation programme for the ITHACA grant (no.
93 741112). NCC acknowledges support by the Australian Research Council DECRA Fellowship
94 DE210100749.

95 References

96 Amezcua, J., Kalnay, E., & Williams, P. D. (2011). The Effects of the RAW Filter on the
97 Climatology and Forecast Skill of the SPEEDY Model. *Monthly Weather Review*, 139(2),
98 608–619. <https://doi.org/10.1175/2010MWR3530.1>

- 99 Bezanson, Jeff., Edelman, Alan., Karpinski, Stefan., & Shah, V. B. (2017). Julia: A Fresh
100 Approach to Numerical Computing. *SIAM Review*, 59(1), 65–98. [https://doi.org/10.1137/](https://doi.org/10.1137/141000671)
101 [141000671](https://doi.org/10.1137/141000671)
- 102 Bourke, W. (1972). An Efficient, One-Level, Primitive-Equation Spectral Model. *Monthly*
103 *Weather Review*, 100(9), 683–689. [https://doi.org/10.1175/1520-0493\(1972\)100%](https://doi.org/10.1175/1520-0493(1972)100%3C0683:AEOPSM%3E2.3.CO;2)
104 [3C0683:AEOPSM%3E2.3.CO;2](https://doi.org/10.1175/1520-0493(1972)100%3C0683:AEOPSM%3E2.3.CO;2)
- 105 Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., & Brown, B. P. (2020). Dedalus:
106 A flexible framework for numerical simulations with spectral methods. *Physical Review*
107 *Research*, 2(2), 023068. <https://doi.org/10.1103/PhysRevResearch.2.023068>
- 108 Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &
109 Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and
110 Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, 622(2), 759.
111 <https://doi.org/10.1086/427976>
- 112 Hoskins, B. J., & Simmons, A. J. (1975). A multi-layer spectral model and the semi-implicit
113 method. *Quarterly Journal of the Royal Meteorological Society*, 101(429), 637–655.
114 <https://doi.org/10.1002/qj.49710142918>
- 115 Hotta, D., & Ujiie, M. (2018). A nestable, multigrid-friendly grid on a sphere for global
116 spectral models based on Clenshaw–Curtis quadrature. *Quarterly Journal of the Royal*
117 *Meteorological Society*, 144(714), 1382–1397. <https://doi.org/10.1002/qj.3282>
- 118 Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &*
119 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 120 Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W.
121 (2019). A Differentiable Programming System to Bridge Machine Learning and Scientific
122 Computing (No. arXiv:1907.07587). arXiv. <https://doi.org/10.48550/arXiv.1907.07587>
- 123 Kucharski, F., Molteni, F., King, M. P., Farneti, R., Kang, I.-S., & Feudale, L. (2013). On
124 the Need of Intermediate Complexity General Circulation Models: A “SPEEDY” Example.
125 *Bulletin of the American Meteorological Society*, 94(1), 25–30. [https://doi.org/10.1175/](https://doi.org/10.1175/BAMS-D-11-00238.1)
126 [BAMS-D-11-00238.1](https://doi.org/10.1175/BAMS-D-11-00238.1)
- 127 Malardel, S., Wedi, N., Deconinck, N., Diamantakis, M., Kuehnlein, C., Mozdzyński, G.,
128 Hamrud, M., & Smolarkiewicz, P. (2016). A new grid for the IFS. In *ECMWF Newsletter*.
129 <https://www.ecmwf.int/node/15041>.
- 130 Met Office. (2010 - 2015). Cartopy: A cartographic python library with a matplotlib interface.
131 <https://scitools.org.uk/cartopy>
- 132 Molteni, F. (2003). Atmospheric simulations using a GCM with simplified physical param-
133 etrizations. I: Model climatology and variability in multi-decadal experiments. *Climate*
134 *Dynamics*, 20(2), 175–191. <https://doi.org/10.1007/s00382-002-0268-2>
- 135 Moses, W., & Churavy, V. (2020). Instead of Rewriting Foreign Code for Machine Learning,
136 Automatically Synthesize Fast Gradients. *Advances in Neural Information Processing*
137 *Systems*, 33, 12472–12485.
- 138 Nakano, M., Yashiro, H., Kodama, C., & Tomita, H. (2018). Single Precision in the Dynamical
139 Core of a Nonhydrostatic Global Atmospheric Model: Evaluation Using a Baroclinic
140 Wave Test Case. *Monthly Weather Review*, 146(2), 409–416. [https://doi.org/10.1175/](https://doi.org/10.1175/MWR-D-17-0257.1)
141 [MWR-D-17-0257.1](https://doi.org/10.1175/MWR-D-17-0257.1)
- 142 Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza,
143 A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and friendly
144 geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, 5(53), 2018.
145 <https://doi.org/10.21105/joss.02018>

- 146 Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes
147 in climate models. *Proceedings of the National Academy of Sciences*, 115(39), 9684–9689.
- 148 Simmons, A. J., & Burridge, D. M. (1981). An Energy and Angular-Momentum Conserving Ver-
149 tical Finite-Difference Scheme and Hybrid Vertical Coordinates. *Monthly Weather Review*,
150 109(4), 758–766. [https://doi.org/10.1175/1520-0493\(1981\)109%3C0758:AEAAMC%3E2.0.CO;2](https://doi.org/10.1175/1520-0493(1981)109%3C0758:AEAAMC%3E2.0.CO;2)
151
- 152 Simmons, A. J., Hoskins, B. J., & Burridge, D. M. (1978). Stability of the Semi-Implicit
153 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](https://doi.org/10.1175/1520-0493(1978)106%3C0405:SOTSIM%3E2.0.CO;2)
154
- 155 Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., & Carver, G. (2017).
156 Single Precision in Weather Forecasting Models: An Evaluation with the IFS. *Monthly*
157 *Weather Review*, 145(2), 495–502. <https://doi.org/10.1175/MWR-D-16-0228.1>
- 158 Williams, P. D. (2011). The RAW Filter: An Improvement to the Robert–Asselin Filter
159 in Semi-Implicit Integrations. *Monthly Weather Review*, 139(6), 1996–2007. <https://doi.org/10.1175/2010MWR3601.1>
160

DRAFT