




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 White¹,^{3,4}, Sam Hatfield⁶, David Meyer^{8,9}, Tom Kimpson^{2,10}, Navid C Constantinou¹¹, and Chris Hill¹

¹ Massachusetts Institute of Technology, Cambridge, MA, USA ² 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 Imperial College London, UK ⁹ European Centre for Medium-Range Weather Forecasts, Bonn, Germany ¹⁰ University of Melbourne, Australia ¹¹ 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Published: unpublished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 analyze the global atmospheric circulation on the sphere. It implements several 2D and 3D models solved with spherical harmonics:

- the primitive equations with and without humidity ([Figure 1](#)),
- the shallow water equations ([Figure 2](#)), and
- the barotropic vorticity equations.

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 RingGrids, LowerTriangularMatrices, and SpeedyTransforms. RingGrids is a module that discretizes the sphere on iso-latitude rings and implements interpolations between various such grids. LowerTriangularMatrices is a module used to define the spectral space of the spherical harmonic coefficients. SpeedyTransforms implements the spectral transform between the grid-point space as defined by RingGrids and the spectral space defined in LowerTriangularMatrices. These three modules are independently usable and therefore make SpeedyWeather.jl, beyond its main purpose of simulating atmospheric motion 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 short scripts 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 their respective parameters. All components are then collected into a single model object which, once initialized, returns a simulation object. A simulation contains everything, the model with all parameters as created

before but also all prognostic and diagnostic variables. Such a simulation can then be run, but also accessed before and after to analyze or visualize the current variables, or individual terms of the equations. One can also adjust parameters or define new model components before resuming the simulation. 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) or in a single Jupyter or Pluto notebook. We thereby achieve an interactivity of a simulation and its various model components far beyond the options provided in a monolithic interface. At the same time, defaults, set to well-established test cases, enable even inexperienced users to run simulations in just a few lines of code.

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 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), the process-oriented climate model CLIMLAB (Rose, 2018), and the Julia ocean model Oceananigans.jl (Ramadhan et al., 2020).

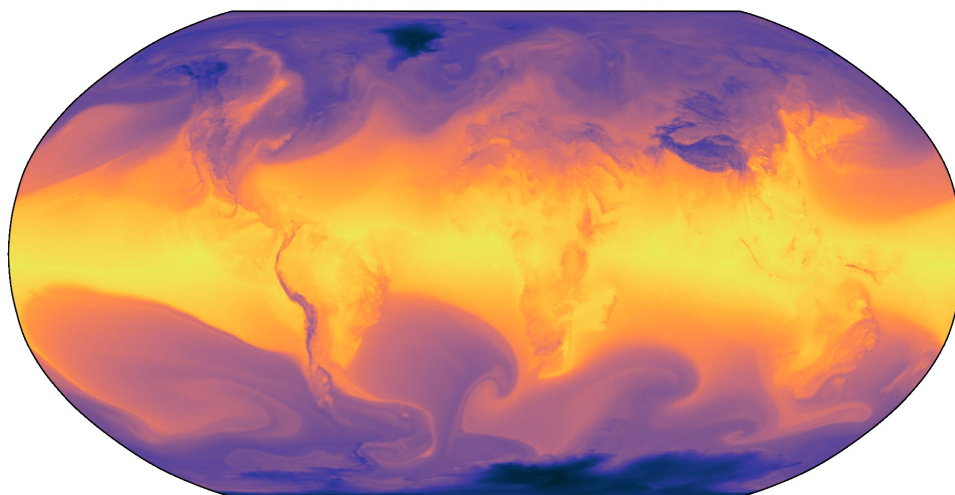


Figure 1: Surface temperature simulated with the primitive equation model in SpeedyWeather.jl. (Figure will be updated)

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. Single-precision floating-point numbers (Float32) are the default as adopted by other modelling efforts (Nakano et al., 2018; Váňa et al., 2017), but Float64 and other custom number formats can be used with a single code basis (M. Klöwer et al., 2020;

76 [Milan Klöwer et al., 2022](#)). Julia will compile to the choice of the number format, the grid,
77 and other model components just-in-time. A simple parallelization across vertical layers is
78 supported by Julia's multithreading. Output is stored as NetCDF files using [NCDatasets.jl](#).

79 Statement of need

80 SpeedyWeather.jl is a fresh approach to atmospheric models that have been very influential in
81 many areas of scientific and high-performance computing as well as climate change mitigation
82 and adaptation. Most weather, ocean and climate models are written in Fortran and have
83 been developed over decades. From this tradition follows a specific programming style and
84 associated user interface. SpeedyWeather.jl aims to overcome the constraints of traditional
85 Fortran-based models. The modern trend sees simulations in Fortran and data analysis in
86 Python, making virtually impossible to interact with various model components directly. In
87 SpeedyWeather.jl, interfaces to the model components are exposed to the user. Furthermore,
88 data-driven climate modelling ([Rasp et al., 2018](#); [Schneider et al., 2023](#)), which replaces
89 existing model components with machine learning, is more difficult in Fortran due to the lack
90 of established machine learning frameworks ([Meyer et al., 2022](#)). In Julia, Flux.jl is available
91 for machine learning ([Innes et al., 2019](#)) as well as automatic differentiation with Enzyme
92 ([Moses & Churavy, 2020](#)), which calculates gradients, necessary to optimize network weights
93 or parameters during training.

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

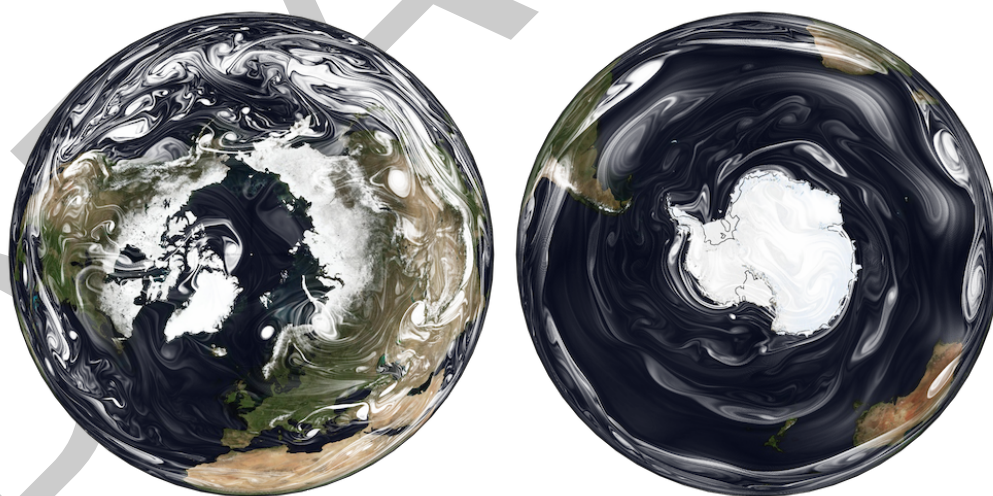


Figure 2: Relative vorticity simulated with the shallow water model in SpeedyWeather.jl. The simulation used a spectral resolution of T1023 (about 20 km) and Float32 arithmetic on an octahedral Clenshaw-Curtis grid ([Hotta & Ujiie, 2018](#)). Relative vorticity is visualized with Matplotlib ([Hunter,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

98 Acknowledgements

99 We acknowledge contributions from Mosè Giordano, Valentin Churavy, and Pietro Monticone
100 who have also committed to the SpeedyWeather.jl repository, and the wider Julia community
101 for help and support. MK acknowledges funding from the National Science Foundation (Chris

102 please add). MK and TK acknowledge funding from the European Research Council under the
 103 European Union's Horizon 2020 research and innovation programme for the ITHACA grant (no.
 104 741112). NCC acknowledges support by the Australian Research Council DECRA Fellowship
 105 DE210100749.

106 References

- 107 Amezcua, J., Kalnay, E., & Williams, P. D. (2011). The Effects of the RAW Filter on the
 108 Climatology and Forecast Skill of the SPEEDY Model. *Monthly Weather Review*, 139(2),
 109 608–619. <https://doi.org/10.1175/2010MWR3530.1>
- 110 Bezanson, Jeff., Edelman, Alan., Karpinski, Stefan., & Shah, V. B. (2017). Julia: A Fresh
 111 Approach to Numerical Computing. *SIAM Review*, 59(1), 65–98. [https://doi.org/10.1137/](https://doi.org/10.1137/141000671)
 112 [141000671](https://doi.org/10.1137/141000671)
- 113 Bourke, W. (1972). An Efficient, One-Level, Primitive-Equation Spectral Model. *Monthly*
 114 *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)
 115 [3C0683:AEOPSM%3E2.3.CO;2](https://doi.org/10.1175/1520-0493(1972)100%3C0683:AEOPSM%3E2.3.CO;2)
- 116 Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., & Brown, B. P. (2020). Dedalus:
 117 A flexible framework for numerical simulations with spectral methods. *Physical Review*
 118 *Research*, 2(2), 023068. <https://doi.org/10.1103/PhysRevResearch.2.023068>
- 119 Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &
 120 Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and
 121 Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, 622(2), 759.
 122 <https://doi.org/10.1086/427976>
- 123 Hoskins, B. J., & Simmons, A. J. (1975). A multi-layer spectral model and the semi-implicit
 124 method. *Quarterly Journal of the Royal Meteorological Society*, 101(429), 637–655.
 125 <https://doi.org/10.1002/qj.49710142918>
- 126 Hotta, D., & Ujiie, M. (2018). A nestable, multigrid-friendly grid on a sphere for global
 127 spectral models based on Clenshaw–Curtis quadrature. *Quarterly Journal of the Royal*
 128 *Meteorological Society*, 144(714), 1382–1397. <https://doi.org/10.1002/qj.3282>
- 129 Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &*
 130 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 131 Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W.
 132 (2019). A Differentiable Programming System to Bridge Machine Learning and Scientific
 133 Computing (No. arXiv:1907.07587). arXiv. <https://doi.org/10.48550/arXiv.1907.07587>
- 134 Klöwer, M., Düben, P. D., & Palmer, T. N. (2020). Number formats, error mitigation, and
 135 scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow
 136 water model. *Journal of Advances in Modeling Earth Systems*, 12(10), e2020MS002246.
 137 <https://doi.org/https://doi.org/10.1029/2020MS002246>
- 138 Klöwer, Milan, Hatfield, S., Croci, M., Düben, P. D., & Palmer, T. N. (2022). Fluid
 139 Simulations Accelerated With 16 Bits: Approaching 4x Speedup on A64FX by Squeezing
 140 ShallowWaters.jl Into Float16. *Journal of Advances in Modeling Earth Systems*, 14(2),
 141 e2021MS002684. <https://doi.org/10.1029/2021MS002684>
- 142 Kucharski, F., Molteni, F., King, M. P., Farneti, R., Kang, I.-S., & Feudale, L. (2013). On
 143 the Need of Intermediate Complexity General Circulation Models: A “SPEEDY” Example.
 144 *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)
 145 [BAMS-D-11-00238.1](https://doi.org/10.1175/BAMS-D-11-00238.1)
- 146 Malardel, S., Wedi, N., Deconinck, N., Diamantakis, M., Kuehnlein, C., Mozdzyński, G.,
 147 Hamrud, M., & Smolarkiewicz, P. (2016). A new grid for the IFS. In *ECMWF Newsletter*.

- 148 <https://www.ecmwf.int/node/15041>.
- 149 Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*.
150 <https://scitools.org.uk/cartopy>
- 151 Meyer, D., Grimmond, S., Dueben, P., Hogan, R., & Reeuwijk, M. van. (2022). Machine
152 learning emulation of urban land surface processes. *Journal of Advances in Modeling Earth*
153 *Systems*, 14(3). <https://doi.org/10.1029/2021ms002744>
- 154 Molteni, F. (2003). Atmospheric simulations using a GCM with simplified physical param-
155 etrizations. I: Model climatology and variability in multi-decadal experiments. *Climate*
156 *Dynamics*, 20(2), 175–191. <https://doi.org/10.1007/s00382-002-0268-2>
- 157 Moses, W., & Churavy, V. (2020). Instead of Rewriting Foreign Code for Machine Learning,
158 Automatically Synthesize Fast Gradients. *Advances in Neural Information Processing*
159 *Systems*, 33, 12472–12485.
- 160 Nakano, M., Yashiro, H., Kodama, C., & Tomita, H. (2018). Single Precision in the Dynamical
161 Core of a Nonhydrostatic Global Atmospheric Model: Evaluation Using a Baroclinic
162 Wave Test Case. *Monthly Weather Review*, 146(2), 409–416. <https://doi.org/10.1175/MWR-D-17-0257.1>
- 164 Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza,
165 A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and friendly
166 geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, 5(53), 2018.
167 <https://doi.org/10.21105/joss.02018>
- 168 Rasp, S., Pritchard, M. S., & Gentile, P. (2018). Deep learning to represent subgrid processes
169 in climate models. *Proceedings of the National Academy of Sciences*, 115(39), 9684–9689.
- 170 Rose, B. E. J. (2018). CLIMLAB: A Python toolkit for interactive, process-oriented climate
171 modeling. *Journal of Open Source Software*, 3(24), 659. <https://doi.org/10.21105/joss.00659>
- 172
- 173 Schneider, T., Behera, S., Boccaletti, G., Deser, C., Emanuel, K., Ferrari, R., Leung, L. R., Lin,
174 N., Müller, T., Navarra, A., Ndiaye, O., Stuart, A., Tribbia, J., & Yamagata, T. (2023).
175 Harnessing AI and computing to advance climate modelling and prediction. *Nature Climate*
176 *Change*, 13(9), 887–889. <https://doi.org/10.1038/s41558-023-01769-3>
- 177 Simmons, A. J., & Burridge, D. M. (1981). An Energy and Angular-Momentum Conserving Ver-
178 tical Finite-Difference Scheme and Hybrid Vertical Coordinates. *Monthly Weather Review*,
179 109(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)
- 180
- 181 Simmons, A. J., Hoskins, B. J., & Burridge, D. M. (1978). Stability of the Semi-Implicit
182 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)
- 183
- 184 Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., & Carver, G. (2017).
185 Single Precision in Weather Forecasting Models: An Evaluation with the IFS. *Monthly*
186 *Weather Review*, 145(2), 495–502. <https://doi.org/10.1175/MWR-D-16-0228.1>
- 187 Williams, P. D. (2011). The RAW Filter: An Improvement to the Robert–Asselin Filter
188 in Semi-Implicit Integrations. *Monthly Weather Review*, 139(6), 1996–2007. <https://doi.org/10.1175/2010MWR3601.1>
- 189