

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Published: unpublished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 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 their 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. At the same time, sensible default arguments enable even inexperienced

44 users to run simulations in just a few lines of code.

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

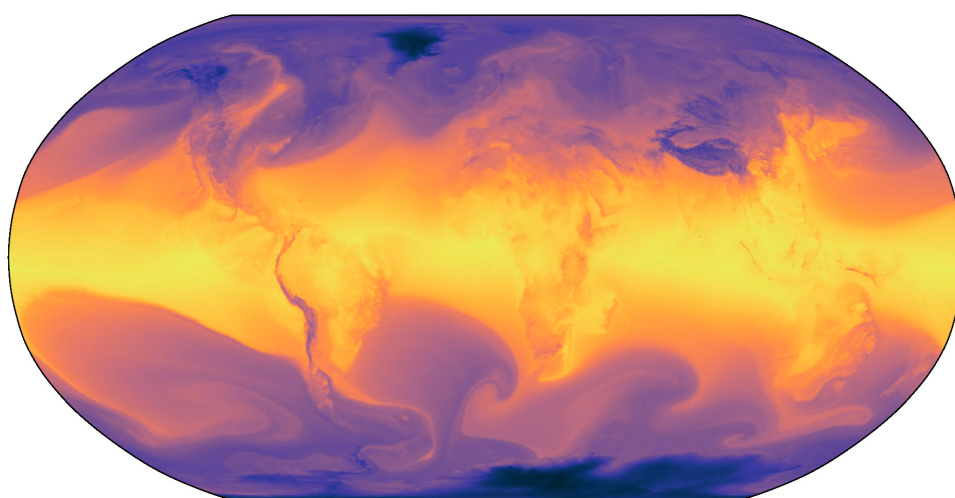


Figure 1: Surface temperature simulated with the primitive equation model in SpeedyWeather.jl. (Figure will be updated)

56 The dynamical core of SpeedyWeather.jl uses established numerics (Bourke, 1972; Hoskins
57 & Simmons, 1975; Simmons et al., 1978; Simmons & Burridge, 1981), widely adopted in
58 numerical weather prediction. It is based on the spherical harmonic transform with a leapfrog-
59 based semi-implicit time integration (Hoskins & Simmons, 1975) and a Robert-Asselin-Williams
60 filter (Amezcuca et al., 2011; Williams, 2011). The spherical harmonic transform is grid-flexible.
61 Any iso-latitude ring-based grid can be used and new grids can be externally defined and
62 passed in as an argument. Many grids are already implemented: the conventional Gaussian
63 grid, a regular longitude-latitude grid, the octahedral Gaussian grid (Malardel et al., 2016), the
64 octahedral Clenshaw-Curtis grid (Hotta & Ujiie, 2018), and the HEALPix grid (Górski et al.,
65 2005). Both SpeedyWeather.jl and its spherical harmonic transform SpeedyTransforms are also
66 number format-flexible. 32-bit single-precision floating-point numbers (Float32) are the default
67 as adopted by other modelling efforts (Nakano et al., 2018; Váňa et al., 2017), but Float64
68 and other custom number formats can be used with a single code basis (M. Klöwer et al., 2020;
69 Milan Klöwer et al., 2022). Julia will compile to the choice of the number format, the grid,
70 and and other model components just-in-time. A simple parallelisation across vertical layers is
71 supported with Julia's multi threading. Output is stored as NetCDF files using NCDatasets.jl.

72 Statement of need

73 SpeedyWeather.jl is a fresh approach to atmospheric models that have been very influential in
74 many areas of scientific and high-performance computing as well as climate change mitigation
75 and adaptation. Most weather, ocean and climate models are written in Fortran and have been
76 developed over decades. From this tradition follows a specific programming style and associated
77 user interface. Running a simulation in Fortran and analysing the data in Python makes it
78 virtually impossible to interact with various model components interactively. Furthermore,
79 data-driven climate modelling (Rasp et al., 2018; Schneider et al., 2023), which replaces
80 existing model components with machine learning is difficult due to the lack of established
81 deep learning frameworks in Fortran (Innes et al., 2019). Let alone online learning, which
82 trains a neural network-based component together with the rest of the model, accounting for
83 interactions between components. Gradients, necessary to optimize training, can be computed
84 with automatic differentiation (Moses & Churavy, 2020), but only if differentiable functions in
85 a coherent language framework are provided.

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

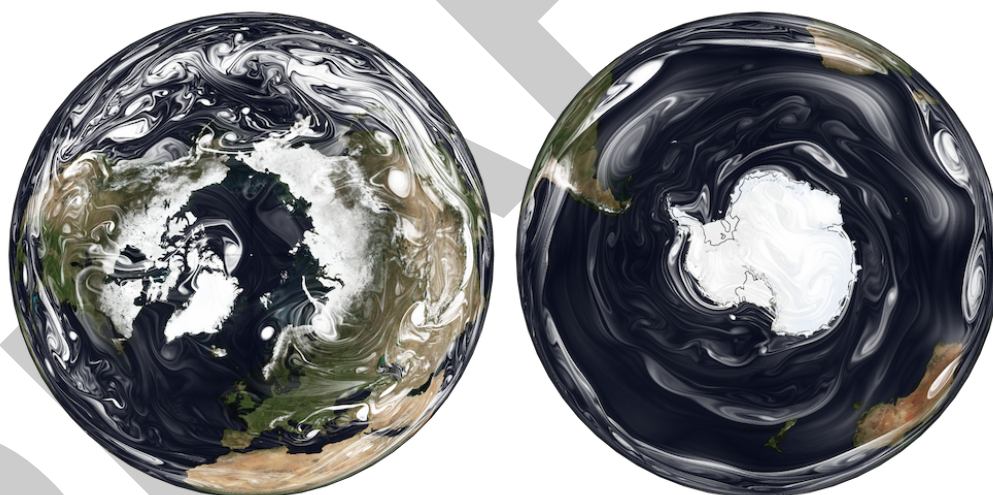


Figure 2: Relative vorticity simulated with the shallow water model in SpeedyWeather.jl. The simulation used a spectral resolution of T1023 (about 20 km) and Float32 arithmetic on an octahedral Clenshaw-Curtis grid (Hotta & Ujiie, 2018). Relative vorticity is visualised with Matplotlib (Hunter, 2007)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

90 Acknowledgements

91 We acknowledge contributions from Mosè Giordano, Valentin Churavy, and Pietro Monticone
92 who have also committed to the SpeedyWeather.jl repository, and the wider Julia community
93 for help and support. MK acknowledges funding from the National Science Foundation (Chris
94 please add). MK and TK acknowledge funding from the European Research Council under the
95 European Union's Horizon 2020 research and innovation programme for the ITHACA grant (no.
96 741112). NCC acknowledges support by the Australian Research Council DECRA Fellowship
97 DE210100749.

References

- 98
- 99 Amezcua, J., Kalnay, E., & Williams, P. D. (2011). The Effects of the RAW Filter on the
100 Climatology and Forecast Skill of the SPEEDY Model. *Monthly Weather Review*, 139(2),
101 608–619. <https://doi.org/10.1175/2010MWR3530.1>
- 102 Bezanson, Jeff., Edelman, Alan., Karpinski, Stefan., & Shah, V. B. (2017). Julia: A Fresh
103 Approach to Numerical Computing. *SIAM Review*, 59(1), 65–98. [https://doi.org/10.1137/](https://doi.org/10.1137/141000671)
104 [141000671](https://doi.org/10.1137/141000671)
- 105 Bourke, W. (1972). An Efficient, One-Level, Primitive-Equation Spectral Model. *Monthly*
106 *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)
107 [3C0683:AEOPSM%3E2.3.CO;2](https://doi.org/10.1175/1520-0493(1972)100%3C0683:AEOPSM%3E2.3.CO;2)
- 108 Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., & Brown, B. P. (2020). Dedalus:
109 A flexible framework for numerical simulations with spectral methods. *Physical Review*
110 *Research*, 2(2), 023068. <https://doi.org/10.1103/PhysRevResearch.2.023068>
- 111 Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &
112 Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and
113 Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, 622(2), 759.
114 <https://doi.org/10.1086/427976>
- 115 Hoskins, B. J., & Simmons, A. J. (1975). A multi-layer spectral model and the semi-implicit
116 method. *Quarterly Journal of the Royal Meteorological Society*, 101(429), 637–655.
117 <https://doi.org/10.1002/qj.49710142918>
- 118 Hotta, D., & Ujiie, M. (2018). A nestable, multigrid-friendly grid on a sphere for global
119 spectral models based on Clenshaw–Curtis quadrature. *Quarterly Journal of the Royal*
120 *Meteorological Society*, 144(714), 1382–1397. <https://doi.org/10.1002/qj.3282>
- 121 Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &*
122 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 123 Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W.
124 (2019). *A Differentiable Programming System to Bridge Machine Learning and Scientific*
125 *Computing* (No. arXiv:1907.07587). arXiv. <https://doi.org/10.48550/arXiv.1907.07587>
- 126 Klöwer, M., Düben, P. D., & Palmer, T. N. (2020). Number formats, error mitigation, and
127 scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow
128 water model. *Journal of Advances in Modeling Earth Systems*, 12(10), e2020MS002246.
129 <https://doi.org/10.1029/2020MS002246>
- 130 Klöwer, Milan, Hatfield, S., Croci, M., Düben, P. D., & Palmer, T. N. (2022). Fluid
131 Simulations Accelerated With 16 Bits: Approaching 4x Speedup on A64FX by Squeezing
132 ShallowWaters.jl Into Float16. *Journal of Advances in Modeling Earth Systems*, 14(2),
133 e2021MS002684. <https://doi.org/10.1029/2021MS002684>
- 134 Kucharski, F., Molteni, F., King, M. P., Farneti, R., Kang, I.-S., & Feudale, L. (2013). On
135 the Need of Intermediate Complexity General Circulation Models: A “SPEEDY” Example.
136 *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)
137 [BAMS-D-11-00238.1](https://doi.org/10.1175/BAMS-D-11-00238.1)
- 138 Malardel, S., Wedi, N., Deconinck, N., Diamantakis, M., Kuehnlein, C., Mozdynski, G.,
139 Hamrud, M., & Smolarkiewicz, P. (2016). A new grid for the IFS. In *ECMWF Newsletter*.
140 <https://www.ecmwf.int/node/15041>.
- 141 Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*.
142 <https://scitools.org.uk/cartopy>

- 143 Molteni, F. (2003). Atmospheric simulations using a GCM with simplified physical param-
 144 etrizations. I: Model climatology and variability in multi-decadal experiments. *Climate*
 145 *Dynamics*, 20(2), 175–191. <https://doi.org/10.1007/s00382-002-0268-2>
- 146 Moses, W., & Churavy, V. (2020). Instead of Rewriting Foreign Code for Machine Learning,
 147 Automatically Synthesize Fast Gradients. *Advances in Neural Information Processing*
 148 *Systems*, 33, 12472–12485.
- 149 Nakano, M., Yashiro, H., Kodama, C., & Tomita, H. (2018). Single Precision in the Dynamical
 150 Core of a Nonhydrostatic Global Atmospheric Model: Evaluation Using a Baroclinic
 151 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)
 152 [MWR-D-17-0257.1](https://doi.org/10.1175/MWR-D-17-0257.1)
- 153 Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza,
 154 A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and friendly
 155 geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, 5(53), 2018.
 156 <https://doi.org/10.21105/joss.02018>
- 157 Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes
 158 in climate models. *Proceedings of the National Academy of Sciences*, 115(39), 9684–9689.
- 159 Schneider, T., Behera, S., Boccaletti, G., Deser, C., Emanuel, K., Ferrari, R., Leung, L. R., Lin,
 160 N., Müller, T., Navarra, A., Ndiaye, O., Stuart, A., Tribbia, J., & Yamagata, T. (2023).
 161 Harnessing AI and computing to advance climate modelling and prediction. *Nature Climate*
 162 *Change*, 13(9), 887–889. <https://doi.org/10.1038/s41558-023-01769-3>
- 163 Simmons, A. J., & Burridge, D. M. (1981). An Energy and Angular-Momentum Conserving Ver-
 164 tical Finite-Difference Scheme and Hybrid Vertical Coordinates. *Monthly Weather Review*,
 165 109(4), 758–766. [https://doi.org/10.1175/1520-0493\(1981\)109%3C0758:AEAAMC%3E2.](https://doi.org/10.1175/1520-0493(1981)109%3C0758:AEAAMC%3E2.0.CO;2)
 166 [0.CO;2](https://doi.org/10.1175/1520-0493(1981)109%3C0758:AEAAMC%3E2.0.CO;2)
- 167 Simmons, A. J., Hoskins, B. J., & Burridge, D. M. (1978). Stability of the Semi-Implicit
 168 Method of Time Integration. *Monthly Weather Review*, 106(3), 405–412. [https://doi.org/](https://doi.org/10.1175/1520-0493(1978)106%3C0405:SOTSIM%3E2.0.CO;2)
 169 [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)
- 170 Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., & Carver, G. (2017).
 171 Single Precision in Weather Forecasting Models: An Evaluation with the IFS. *Monthly*
 172 *Weather Review*, 145(2), 495–502. <https://doi.org/10.1175/MWR-D-16-0228.1>
- 173 Williams, P. D. (2011). The RAW Filter: An Improvement to the Robert–Asselin Filter
 174 in Semi-Implicit Integrations. *Monthly Weather Review*, 139(6), 1996–2007. <https://doi.org/10.1175/2010MWR3601.1>
 175