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

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** [6], **Tom Kimpson** [2,8], **Navid C Constantinou** [9], and **Chris Hill**[1]

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** University of Melbourne, Australia **9** Australian National University, Canberra, Australia ¶ Corresponding author

Summary

SpeedyWeather.jl is a library to simulate and analyze the global atmospheric circulation on the sphere. It implements several 2D and 3D models which solve different sets of equations:

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

The primitive equation model in SpeedyWeather.jl is an atmospheric general circulation model (Kucharski et al., 2013) with simple parameterizations for unresolved physical processes including precipitation or boundary layer mixing. 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). However, all models here are written in a modular way to make its components easily extensible. For example, a new parameterization can be externally defined and passed as an argument to the model constructor. Operators used inside SpeedyWeather.jl are exposed to the user, facilitating analysis of the simulation data. SpeedyWeather.jl is therefore, beyond its main purpose of simulating atmospheric motion, also a library for the analysis of gridded data on the sphere. Running and analyzing 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. A model is constructed 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 constructed 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 some parameters 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 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

43 same time, defaults, set to well-established test cases, enable even inexperienced users to run
44 simulations in just a few lines of code.

45 SpeedyWeather.jl relies on Julia's multiple dispatch programming paradigm (Bezanson et al.,
46 2017) to be extensible with new components including parameterizations, forcings, drag, or
47 even the grid. All such supported model components define an abstract type that can be
48 subtyped to introduce, for example, a new parameterization. To define precipitation due to
49 the physical process of large-scale condensation, one would define `MyCondensation` as a new
50 subtype of `AbstractCondensation`. One then only needs to extend the `initialize!` and
51 `condensation!` functions for this new type. Passing on `condensation = MyCondensation()`
52 to the model constructor then implements this new model component without the need to
53 branch off or overwrite existing model components. Conceptually similar scientific modelling
54 paradigms have been very successful in the Python-based generic partial differential equation
55 solver Dedalus (Burns et al., 2020), the process-oriented climate model CLIMLAB (Rose,
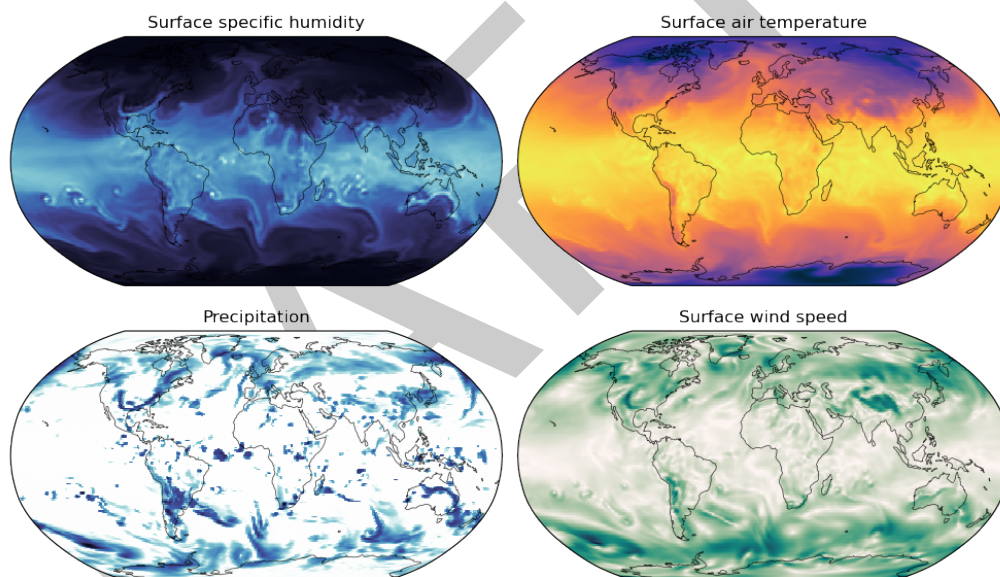56 2018), and the Julia ocean model Oceananigans.jl (Ramadhan et al., 2020).

Figure 1: Surface humidity, air temperature, wind speed and precipitation simulated with the primitive equation model in SpeedyWeather.jl. Spectral resolution is T127 (about 100km) on an octahedral Gaussian grid (Malardel et al., 2016) with simple physics to represent unresolved processes such as surface fluxes including evaporation, and precipitation due to large-scale condensation and convection.

57 The dynamical core of SpeedyWeather.jl uses established numerics (Bourke, 1972; Hoskins
58 & Simmons, 1975; Simmons et al., 1978; Simmons & Burridge, 1981), widely adopted in
59 numerical weather prediction. It is based on the spherical harmonic transform (Reinecke &
60 Seljebotn, 2013; Stompor, 2011) with a leapfrog-based semi-implicit time integration (Hoskins
61 & Simmons, 1975) and a Robert-Asselin-Williams filter (Amezcua et al., 2011; Williams, 2011).
62 The spherical harmonic transform is grid-flexible (Willmert, 2020). Any iso-latitude ring-based
63 grid can be used and new grids can be externally defined and passed in as an argument. Many
64 grids are already implemented: the conventional Gaussian grid, a regular longitude-latitude
65 grid, the octahedral Gaussian grid (Malardel et al., 2016), the octahedral Clenshaw-Curtis grid
66 (Hotta & Ujiie, 2018), and the HEALPix grid (Górski et al., 2005). Both SpeedyWeather.jl and
67 its spherical harmonic transform are also number format-flexible. Single-precision floating-point
68 numbers (Float32) are the default as adopted by other modelling efforts (Nakano et al., 2018;
69 Váňa et al., 2017), but Float64 and other custom number formats can be used with a single
70 code basis (M. Klöwer et al., 2020; Milan Klöwer et al., 2022). Julia will compile to the

⁷¹ choice of number format, the grid, and and other model components just-in-time. A simple
⁷² parallelization across vertical layers is supported by Julia's multithreading.

⁷³ SpeedyWeather.jl internally uses three sub-modules `RingGrids`, `LowerTriangularMatrices`,
⁷⁴ and `SpeedyTransforms`. `RingGrids` is a module that discretizes the sphere on iso-latitude
⁷⁵ rings and implements interpolations between various such grids. `LowerTriangularMatrices`
⁷⁶ facilitates the implementation of the spherical harmonics by organizing their coefficients in a
⁷⁷ lower triangular matrix representation. `SpeedyTransforms` implements the spectral transform
⁷⁸ between the grid-point space as defined by `RingGrids` and the spectral space defined in
⁷⁹ `LowerTriangularMatrices`. These three modules are independently usable and therefore
⁸⁰ support SpeedyWeather's library-like user interface. Output is stored as NetCDF files using
⁸¹ NCDatasets.jl.

Statement of need

⁸³ SpeedyWeather.jl is a fresh approach to atmospheric models that have been very influential in
⁸⁴ many areas of scientific and high-performance computing as well as climate change mitigation
⁸⁵ and adaptation. Most weather, ocean and climate models are written in Fortran and have
⁸⁶ been developed over decades. From this tradition follows a specific programming style and
⁸⁷ associated user interface. SpeedyWeather.jl aims to overcome the constraints of traditional
⁸⁸ Fortran-based models. The modern trend sees simulations in Fortran and data analysis in
⁸⁹ Python, making it virtually impossible to interact with various model components directly. In
⁹⁰ SpeedyWeather.jl, interfaces to the model components are exposed to the user. Furthermore,
⁹¹ data-driven climate modelling (Rasp et al., 2018; Schneider et al., 2023), which replaces
⁹² existing model components with machine learning, is more difficult in Fortran due to the lack
⁹³ of established machine learning frameworks (Meyer et al., 2022). In Julia, Flux.jl is available
⁹⁴ for machine learning (Innes et al., 2019) as well as automatic differentiation with Enzyme
⁹⁵ (Moses & Churavy, 2020), which calculates gradients, necessary to optimize network weights
⁹⁶ or parameters during training.

⁹⁷ With SpeedyWeather.jl we hope to provide a platform for data-driven atmospheric modelling
⁹⁸ and in general an interactive model that makes difficult problems easy to simulate. Climate
⁹⁹ models that are user-friendly, trainable, but also easily extensible will suddenly make many
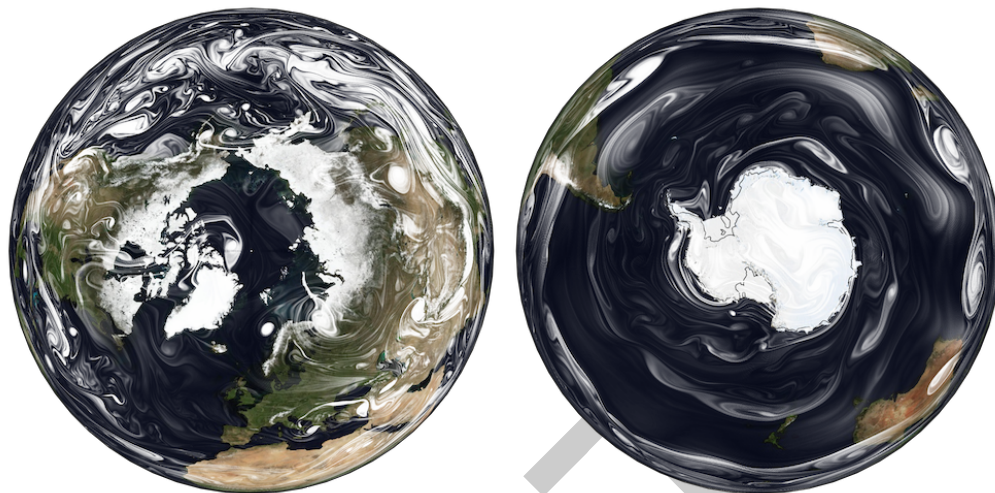¹⁰⁰ complex research ideas possible.

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.

Acknowledgements

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

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

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

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

134 Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W.
135 (2019). *A Differentiable Programming System to Bridge Machine Learning and Scientific
136 Computing* (No. arXiv:1907.07587). arXiv. https://doi.org/10.48550/arXiv.1907.07587

137 Klöwer, M., Düben, P. D., & Palmer, T. N. (2020). Number formats, error mitigation, and
138 scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow
139 water model. *Journal of Advances in Modeling Earth Systems*, *12*(10), e2020MS002246.
140 https://doi.org/https://doi.org/10.1029/2020MS002246

141 Klöwer, Milan, Hatfield, S., Croci, M., Düben, P. D., & Palmer, T. N. (2022). Fluid
142 Simulations Accelerated With 16 Bits: Approaching 4x Speedup on A64FX by Squeezing
143 ShallowWaters.jl Into Float16. *Journal of Advances in Modeling Earth Systems*, *14*(2),
144 e2021MS002684. https://doi.org/10.1029/2021MS002684

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

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

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

154 Meyer, D., Grimmond, S., Dueben, P., Hogan, R., & Reeuwijk, M. van. (2022). Machine
155 learning emulation of urban land surface processes. *Journal of Advances in Modeling Earth
156 Systems*, *14*(3). https://doi.org/10.1029/2021ms002744

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

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

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

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

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

173 Reinecke, M., & Seljebotn, D. S. (2013). Libsharp - spherical harmonic transforms revisited.
174 *Astronomy and Astrophysics*, *554*, A112. https://doi.org/10.1051/0004-6361/201321494

175 Rose, B. E. j. (2018). CLIMLAB: A Python toolkit for interactive, process-oriented climate
176 modeling. *Journal of Open Source Software*, *3*(24), 659. https://doi.org/10.21105/joss.
177 00659

178 Schneider, T., Behera, S., Boccaletti, G., Deser, C., Emanuel, K., Ferrari, R., Leung, L. R., Lin,
179 N., Müller, T., Navarra, A., Ndiaye, O., Stuart, A., Tribbia, J., & Yamagata, T. (2023).
180 Harnessing AI and computing to advance climate modelling and prediction. *Nature Climate*
181 *Change*, *13*(9), 887–889. https://doi.org/10.1038/s41558-023-01769-3

182 Simmons, A. J., & Burridge, D. M. (1981). An Energy and Angular-Momentum Conserving Ver-
183 tical Finite-Difference Scheme and Hybrid Vertical Coordinates. *Monthly Weather Review*,
184 *109*(4), 758–766. https://doi.org/10.1175/1520-0493(1981)109%3C0758:AEAAMC%3E2.
185 0.CO;2

186 Simmons, A. J., Hoskins, B. J., & Burridge, D. M. (1978). Stability of the Semi-Implicit
187 Method of Time Integration. *Monthly Weather Review*, *106*(3), 405–412. https://doi.org/
188 10.1175/1520-0493(1978)106%3C0405:SOTSIM%3E2.0.CO;2

189 Stompor, R. (2011). *S2HAT: Scalable Spherical Harmonic Transform Library*. Astrophysics
190 Source Code Library, record ascl:1110.013.

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

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

197 Willmert, J. (2020). *Blog series: Notes on calculating the spherical harmonics*. https:
198 //justinwillmert.com/articles/2020/notes-on-calculating-the-spherical-harmonics/.