

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

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, Tom Kimpson^{2,8}, Navid C Constantinou⁸, 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 The University of Melbourne, Parkville, VIC, 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Published: unpublished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 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 ([Figure 3](#)).

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 ([Mazlami et al., 2017](#)), controlling most of the model's functionality through arguments of a single function or through parameter files (often called namelists in Fortran), 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

43 interactivity of a simulation and its various model components far beyond the options provided
44 in a monolithic interface. At the same time, defaults, set to well-established test cases, enable
45 even inexperienced users to run simulations in just a few lines of code.

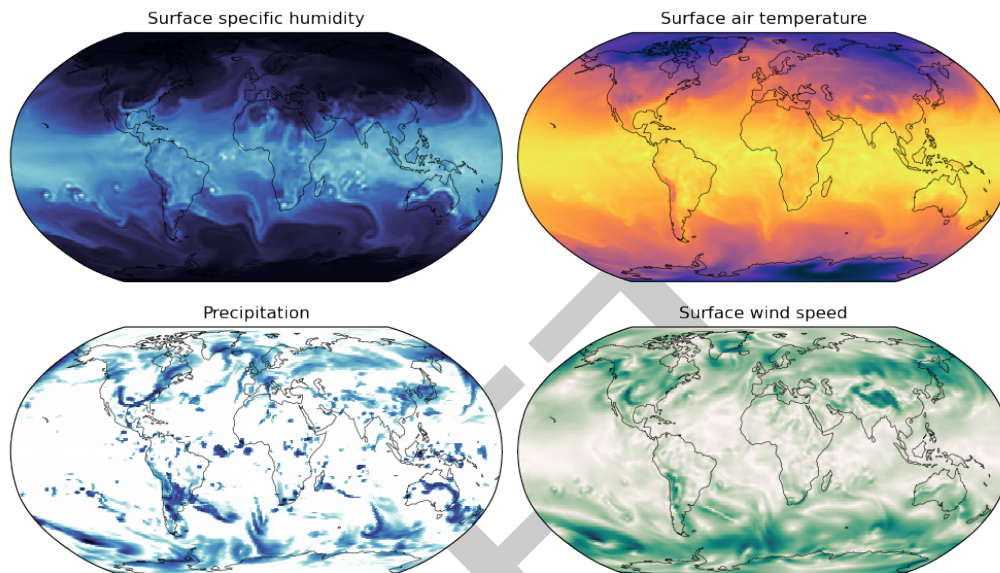


Figure 1: Surface humidity, air temperature, wind speed and precipitation simulated with the primitive equation model in SpeedyWeather.jl. Spectral resolution is T127 (about 100km) on an octahedral Gaussian grid (Malardel et al., 2016) with simple physics to represent unresolved processes such as surface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

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

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

71 [Vána et al., 2017](#)), but Float64 and other custom number formats can be used with a single
72 code basis ([Klöwer et al., 2020](#); [Klöwer et al., 2022](#)). Julia will compile to the choice of number
73 format, the grid, and and other model components just-in-time. A simple parallelization (across
74 vertical layers for the dynamical core, across horizontal grid points for the parameterizations)
75 is supported by Julia's multithreading. No distributed-memory parallelization is currently
76 supported, GPU support is planned.

77 SpeedyWeather.jl internally uses three sub-modules RingGrids, LowerTriangularMatrices,
78 and SpeedyTransforms. RingGrids is a module that discretizes the sphere on iso-latitude
79 rings and implements interpolations between various such grids. LowerTriangularMatrices
80 facilitates the implementation of the spherical harmonics by organizing their coefficients in a
81 lower triangular matrix representation. SpeedyTransforms implements the spectral transform
82 between the grid-point space as defined by RingGrids and the spectral space defined in
83 LowerTriangularMatrices. These three modules are independently usable and therefore
84 support SpeedyWeather's library-like user interface. Output is stored as NetCDF files using
85 NCDatasets.jl([Barth, 2023](#)).

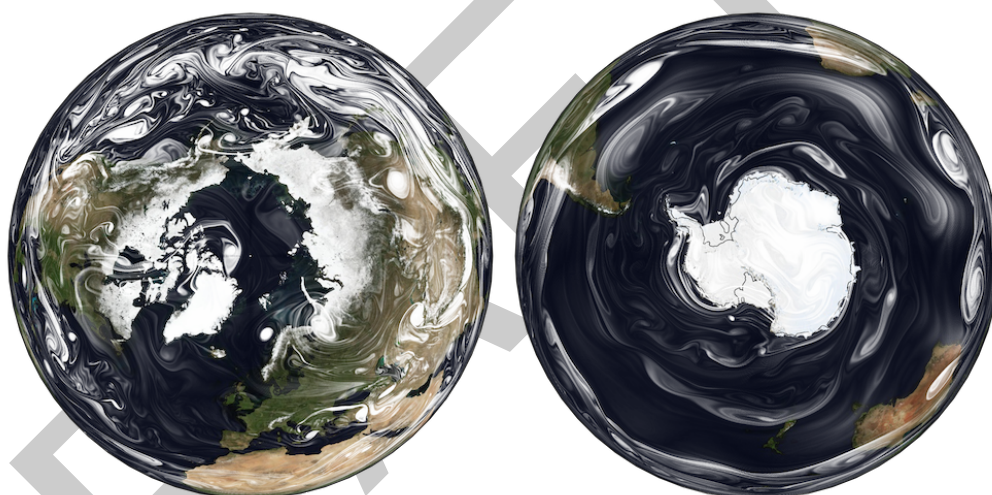


Figure 2: Relative vorticity simulated with the shallow water model in SpeedyWeather.jl. The simulation used a spectral resolution of T1023 (about 20 km) and Float32 arithmetic on an octahedral Clenshaw-Curtis grid ([Hotta & Ujiie, 2018](#)). Relative vorticity is visualized with Matplotlib ([Hunter,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

86 Statement of need

87 SpeedyWeather.jl is a fresh approach to atmospheric models that have been very influential in
88 many areas of scientific and high-performance computing as well as climate change mitigation
89 and adaptation. Most weather, ocean and climate models are written in Fortran (e.g. ICON
90 ([Giorgetta et al., 2018](#)), CESM ([Hurrell et al., 2013](#)), MITgcm ([Marshall et al., 1997](#)), NEMO
91 ([Madec et al., 2017](#))) and have been developed over decades. From this tradition follows a
92 specific programming style and associated user interface. SpeedyWeather.jl aims to overcome
93 the constraints of traditional Fortran-based models. The modern trend sees simulations in
94 Fortran and data analysis in Python (e.g. NumPy ([Harris et al., 2020](#)), Xarray ([Hoyer &
95 Hamman, 2017](#)), Dask ([Dask Development Team, 2016](#)), Matplotlib ([Hunter, 2007](#))), making
96 it virtually impossible to interact with various model components directly. In SpeedyWeather.jl,
97 interfaces to the model components are exposed to the user. Furthermore, data-driven climate
98 modelling ([Rasp et al., 2018](#); [Schneider et al., 2023](#)), which replaces existing model components
99 with machine learning, is more difficult in Fortran due to the lack of established machine

100 learning frameworks (Meyer et al., 2022). In Julia, Flux.jl (Innes et al., 2019) is available for
101 machine learning as well as automatic differentiation with Enzyme (Moses & Churavy, 2020)
102 for gradients-based optimization.

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

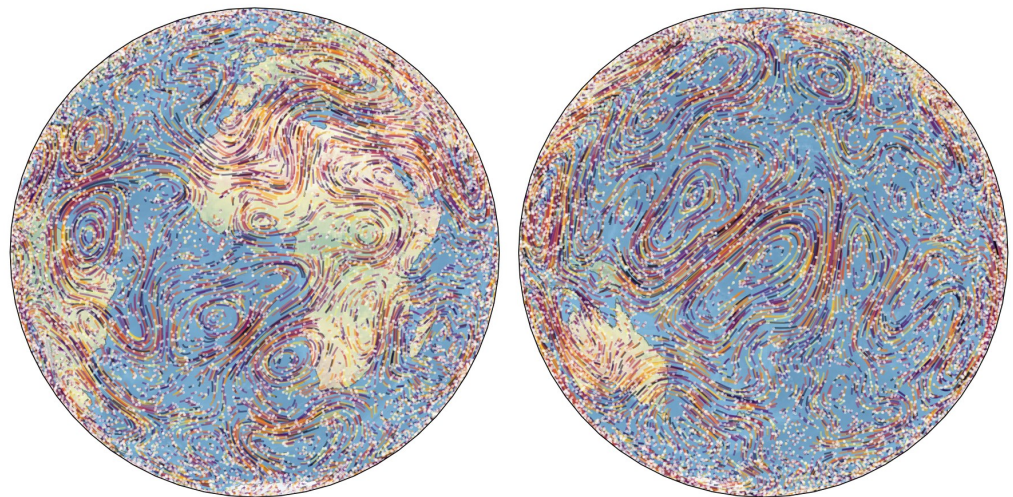


Figure 3: Particle trajectories advected in the barotropic vorticity model. The barotropic vorticity equations were stochastically stirred at wave numbers 8 to 40 for homogeneous turbulence on the sphere. The simulation was computed at T340 (about 40km global resolution). Visualized are 20,000 particles (white dots) with trajectories (colored randomly) for the previous 6 hours.

Acknowledgements

107 We acknowledge contributions from David Meyer, Mosè Giordano, Valentin Churavy, and
108 Pietro Monticone who have also committed to the SpeedyWeather.jl repository, and the wider
109 Julia community for help and support. MK acknowledges funding from the National Science
110 Foundation. MK and TK acknowledge funding from the European Research Council under the
111 European Union's Horizon 2020 research and innovation programme for the ITHACA grant (no.
112 741112). NCC acknowledges support by the Australian Research Council DECRA Fellowship
113 DE210100749.
114

References

- 115
116 Amezcua, J., Kalnay, E., & Williams, P. D. (2011). The Effects of the RAW Filter on the
117 Climatology and Forecast Skill of the SPEEDY Model. *Monthly Weather Review*, 139(2),
118 608–619. <https://doi.org/10.1175/2010MWR3530.1>
119 Barth, A. (2023). NCDatasets: A julia package for manipulating netCDF data sets. In *GitHub*
120 *repository*. <https://github.com/Alexander-Barth/NCDatasets.jl>; GitHub.
121 Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to
122 Numerical Computing. *SIAM Review*, 59(1), 65–98. <https://doi.org/10.1137/141000671>
123 Bourke, W. (1972). An Efficient, One-Level, Primitive-Equation Spectral Model. *Monthly*
124 *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)
125 [3C0683:AEOPSM%3E2.3.CO;2](https://doi.org/10.1175/1520-0493(1972)100%3C0683:AEOPSM%3E2.3.CO;2)

- 126 Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., & Brown, B. P. (2020). Dedalus:
127 A flexible framework for numerical simulations with spectral methods. *Physical Review*
128 *Research*, 2(2), 023068. <https://doi.org/10.1103/PhysRevResearch.2.023068>
- 129 Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. <http://dask.pydata.org>
- 131 Giorgetta, M. A., Brokopf, R., Crueger, T., Esch, M., Fiedler, S., Helmert, J., Hohenegger,
132 C., Kornblueh, L., Köhler, M., Manzini, E., Mauritsen, T., Nam, C., Raddatz, T., Rast,
133 S., Reinert, D., Sakradzija, M., Schmidt, H., Schneck, R., Schnur, R., ... Stevens, B.
134 (2018). ICON-a, the atmosphere component of the ICON earth system model: I. Model
135 description. *Journal of Advances in Modeling Earth Systems*, 10(7), 1613–1637. <https://doi.org/10.1029/2017MS001242>
- 137 Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &
138 Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and
139 Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, 622(2), 759.
140 <https://doi.org/10.1086/427976>
- 141 Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau,
142 D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van
143 Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ...
144 Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362.
145 <https://doi.org/10.1038/s41586-020-2649-2>
- 146 Hoskins, B. J., & Simmons, A. J. (1975). A multi-layer spectral model and the semi-implicit
147 method. *Quarterly Journal of the Royal Meteorological Society*, 101(429), 637–655.
148 <https://doi.org/10.1002/qj.49710142918>
- 149 Hotta, D., & Ujiie, M. (2018). A nestable, multigrid-friendly grid on a sphere for global
150 spectral models based on Clenshaw–Curtis quadrature. *Quarterly Journal of the Royal*
151 *Meteorological Society*, 144(714), 1382–1397. <https://doi.org/10.1002/qj.3282>
- 152 Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal*
153 *of Open Research Software*, 5(1). <https://doi.org/10.5334/jors.148>
- 154 Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &*
155 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 156 Hurrell, J. W., Holland, M. M., Gent, P. R., Ghan, S., Kay, J. E., Kushner, P. J., Lamarque,
157 J.-F., Large, W. G., Lawrence, D., Lindsay, K., Lipscomb, W. H., Long, M. C., Mahowald,
158 N., Marsh, D. R., Neale, R. B., Rasch, P., Vavrus, S., Vertenstein, M., Bader, D., ...
159 Marshall, S. (2013). The community earth system model: A framework for collaborative
160 research. *Bulletin of the American Meteorological Society*, 94(9), 1339–1360. <https://doi.org/10.1175/BAMS-D-12-00121.1>
- 162 Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W.
163 (2019). *A Differentiable Programming System to Bridge Machine Learning and Scientific*
164 *Computing* (No. arXiv:1907.07587). arXiv. <https://doi.org/10.48550/arXiv.1907.07587>
- 165 Klöwer, M., Düben, P. D., & Palmer, T. N. (2020). Number formats, error mitigation, and
166 scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow
167 water model. *Journal of Advances in Modeling Earth Systems*, 12(10), e2020MS002246.
168 <https://doi.org/10.1029/2020MS002246>
- 169 Klöwer, M., Hatfield, S., Croci, M., Düben, P. D., & Palmer, T. N. (2022). Fluid Sim-
170 ulations Accelerated With 16 Bits: Approaching 4x Speedup on A64FX by Squeezing
171 ShallowWaters.jl Into Float16. *Journal of Advances in Modeling Earth Systems*, 14(2),
172 e2021MS002684. <https://doi.org/10.1029/2021MS002684>

- 173 Kucharski, F., Molteni, F., King, M. P., Farneti, R., Kang, I.-S., & Feudale, L. (2013). On
174 the Need of Intermediate Complexity General Circulation Models: A “SPEEDY” Example.
175 *Bulletin of the American Meteorological Society*, 94(1), 25–30. [https://doi.org/10.1175/
176 BAMS-D-11-00238.1](https://doi.org/10.1175/BAMS-D-11-00238.1)
- 177 Madec, G., Bourdallé-Badie, R., Bouttier, P.-A., Bricaud, C., Bruciaferri, D., Calvert, D.,
178 Chanut, J., Clementi, E., Coward, A., Delrosso, D., Ethé, C., Flavoni, S., Graham, T.,
179 Harle, J., Iovino, D., Lea, D., Lévy, C., Lovato, T., Martin, N., ... Vancoppenolle, M. (2017).
180 *NEMO ocean engine*. <https://doi.org/10.5281/zenodo.3248739>
- 181 Malardel, S., Wedi, N., Deconinck, N., Diamantakis, M., Kuehnlein, C., Mozdzyński, G.,
182 Hamrud, M., & Smolarkiewicz, P. (2016). A new grid for the IFS. In *ECMWF Newsletter*.
183 <https://www.ecmwf.int/node/15041>. <https://doi.org/10.21957/zwdu9u5i>
- 184 Marshall, J., Adcroft, A., Hill, C., Perelman, L., & Heisey, C. (1997). A finite-volume,
185 incompressible navier stokes model for studies of the ocean on parallel computers. *Journal*
186 *of Geophysical Research: Oceans*, 102(C3), 5753–5766. <https://doi.org/10.1029/96JC02775>
- 188 Mazlami, G., Cito, J., & Leitner, P. (2017). Extraction of microservices from monolithic
189 software architectures. *2017 IEEE International Conference on Web Services (ICWS)*,
190 524–531. <https://doi.org/10.1109/ICWS.2017.61>
- 191 Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a matplotlib interface*.
192 <https://scitools.org.uk/cartopy>
- 193 Meyer, D., Grimmond, S., Dueben, P., Hogan, R., & Reeuwijk, M. van. (2022). Machine
194 learning emulation of urban land surface processes. *Journal of Advances in Modeling Earth*
195 *Systems*, 14(3). <https://doi.org/10.1029/2021ms002744>
- 196 Molteni, F. (2003). Atmospheric simulations using a GCM with simplified physical param-
197 etrizations. I: Model climatology and variability in multi-decadal experiments. *Climate*
198 *Dynamics*, 20(2), 175–191. <https://doi.org/10.1007/s00382-002-0268-2>
- 199 Moses, W., & Churavy, V. (2020). Instead of Rewriting Foreign Code for Machine Learning,
200 Automatically Synthesize Fast Gradients. *Advances in Neural Information Processing*
201 *Systems*, 33, 12472–12485. <https://doi.org/10.48550/arXiv.2010.01709>
- 202 Nakano, M., Yashiro, H., Kodama, C., & Tomita, H. (2018). Single Precision in the Dynamical
203 Core of a Nonhydrostatic Global Atmospheric Model: Evaluation Using a Baroclinic
204 Wave Test Case. *Monthly Weather Review*, 146(2), 409–416. [https://doi.org/10.1175/
205 MWR-D-17-0257.1](https://doi.org/10.1175/MWR-D-17-0257.1)
- 206 Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza,
207 A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and friendly
208 geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, 5(53), 2018.
209 <https://doi.org/10.21105/joss.02018>
- 210 Rasp, S., Pritchard, M. S., & Gentile, P. (2018). Deep learning to represent subgrid processes
211 in climate models. *Proceedings of the National Academy of Sciences*, 115(39), 9684–9689.
212 <https://doi.org/10.1073/pnas.1810286115>
- 213 Reinecke, M., & Seljebotn, D. S. (2013). Libsharp - spherical harmonic transforms revisited.
214 *Astronomy and Astrophysics*, 554, A112. <https://doi.org/10.1051/0004-6361/201321494>
- 215 Rose, B. E. J. (2018). CLIMLAB: A Python toolkit for interactive, process-oriented climate
216 modeling. *Journal of Open Source Software*, 3(24), 659. [https://doi.org/10.21105/joss.
217 00659](https://doi.org/10.21105/joss.00659)
- 218 Schneider, T., Behera, S., Boccaletti, G., Deser, C., Emanuel, K., Ferrari, R., Leung, L. R., Lin,
219 N., Müller, T., Navarra, A., Ndiaye, O., Stuart, A., Tribbia, J., & Yamagata, T. (2023).

- 220 Harnessing AI and computing to advance climate modelling and prediction. *Nature Climate*
221 *Change*, 13(9), 887–889. <https://doi.org/10.1038/s41558-023-01769-3>
- 222 Simmons, A. J., & Burridge, D. M. (1981). An Energy and Angular-Momentum Conserving Ver-
223 tical Finite-Difference Scheme and Hybrid Vertical Coordinates. *Monthly Weather Review*,
224 109(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)
- 225
- 226 Simmons, A. J., Hoskins, B. J., & Burridge, D. M. (1978). Stability of the Semi-Implicit
227 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)
- 228
- 229 Stompor, R. (2011). *S2HAT: Scalable Spherical Harmonic Transform Library*. <https://ascl.net/1110.013>.
- 230
- 231 Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., & Carver, G. (2017).
232 Single Precision in Weather Forecasting Models: An Evaluation with the IFS. *Monthly*
233 *Weather Review*, 145(2), 495–502. <https://doi.org/10.1175/MWR-D-16-0228.1>
- 234 Williams, P. D. (2011). The RAW Filter: An Improvement to the Robert–Asselin Filter
235 in Semi-Implicit Integrations. *Monthly Weather Review*, 139(6), 1996–2007. <https://doi.org/10.1175/2010MWR3601.1>
- 236
- 237 Willmert, J. (2020). *Blog series: Notes on calculating the spherical harmon-*
238 *ics*. [http://web.archive.org/web/*/https://justinwillmert.com/articles/2020/](http://web.archive.org/web/*/https://justinwillmert.com/articles/2020/notes-on-calculating-the-spherical-harmonics/)
239 [notes-on-calculating-the-spherical-harmonics/](https://justinwillmert.com/articles/2020/notes-on-calculating-the-spherical-harmonics/).