




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

Milan Klöwer<sup>1,2¶</sup>, Maximilian Gelbrecht<sup>3,4</sup>, Daisuke Hotta<sup>5,6</sup>, Justin Willmert<sup>7</sup>, Simone Silvestri<sup>1</sup>, Gregory L Wagner<sup>1</sup>, Alistair White<sup>3,4</sup>, Sam Hatfield<sup>6</sup>, Tom Kimpson<sup>2,8</sup>, Navid C Constantinou<sup>8,9</sup>, and Chris Hill<sup>1</sup>

<sup>1</sup> Massachusetts Institute of Technology, Cambridge, MA, USA <sup>2</sup> University of Oxford, UK <sup>3</sup> Technical University of Munich, Germany <sup>4</sup> Potsdam Institute for Climate Impact Research, Germany <sup>5</sup> Japan Meteorological Agency, Tsukuba, Japan <sup>6</sup> European Centre for Medium-Range Weather Forecasts, Reading, UK <sup>7</sup> University of Minnesota, Minneapolis, MN, USA <sup>8</sup> University of Melbourne, Parkville, VIC, Australia <sup>9</sup> ARC Centre of Excellence for the Weather of the 21st Century, 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Published: unpublished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](https://creativecommons.org/licenses/by/4.0/)).

## Summary

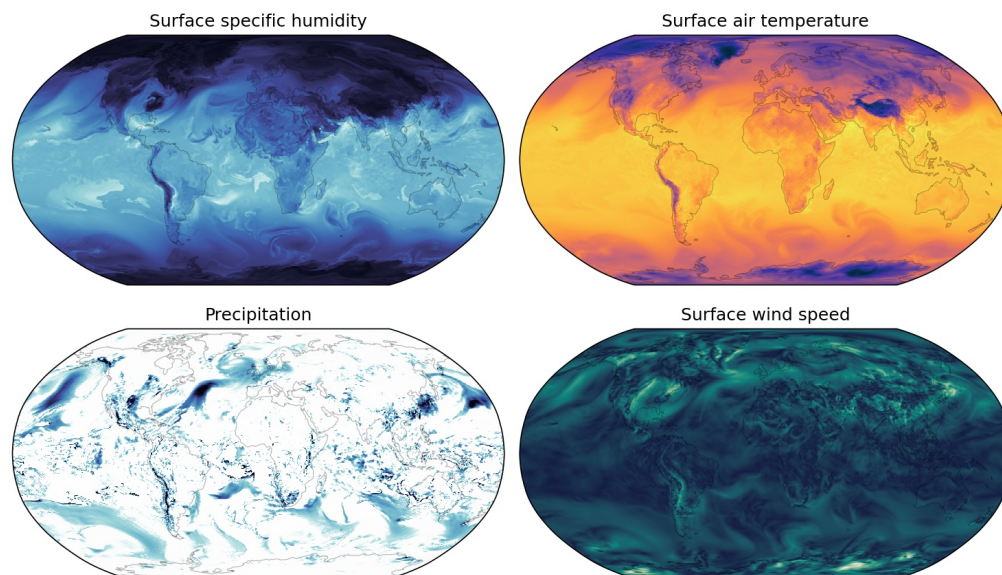
SpeedyWeather.jl is a library to simulate and analyze the global atmospheric circulation on the sphere. It implements several 2D and 3D models which solve different sets of equations:

- 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

43 read-evaluate-print loop (REPL) or in a Jupyter or Pluto notebook. We thereby achieve an  
44 interactivity of a simulation and its various model components far beyond the options provided  
45 in a monolithic interface. At the same time, defaults, set to well-established test cases, enable  
46 even inexperienced users to run simulations in just a few lines of code.



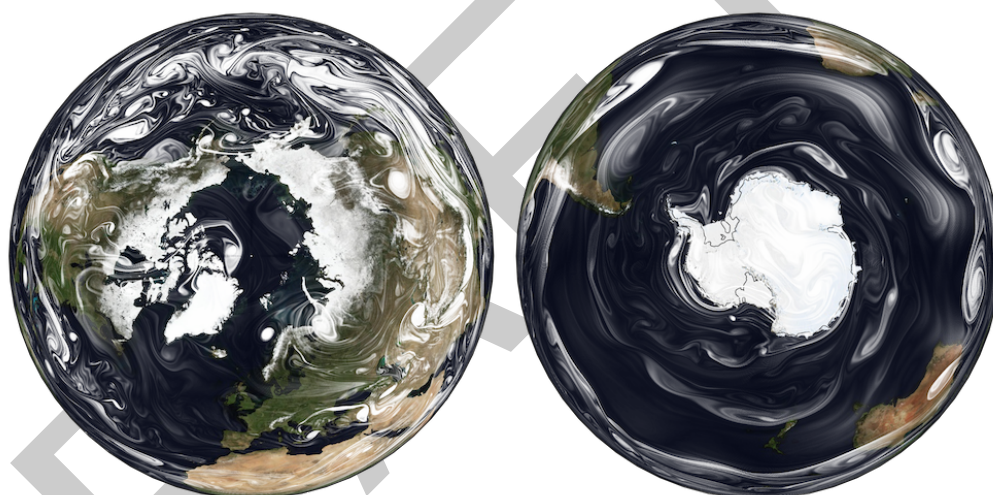
**Figure 1:** Surface humidity, air temperature, wind speed and precipitation simulated with the primitive equation model in SpeedyWeather.jl. Spectral resolution is T340 (about 40km) on an octahedral Gaussian grid (Malardel et al., 2016) with simple physics to represent unresolved processes such as surface fluxes including evaporation, and precipitation due to large-scale condensation and convection.

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

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

71 numbers (Float32) are the default as adopted by other modelling efforts (Nakano et al., 2018;  
72 Váňa et al., 2017), but Float64 and other custom number formats can be used with a single  
73 code basis (Klöwer et al., 2020; Klöwer et al., 2022). Julia will compile to the choice of number  
74 format, the grid, and other model components just-in-time. A simple parallelization (across  
75 vertical layers for the dynamical core, across horizontal grid points for the parameterizations)  
76 is supported by Julia's multithreading. No distributed-memory parallelization is currently  
77 supported, GPU support is planned.

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

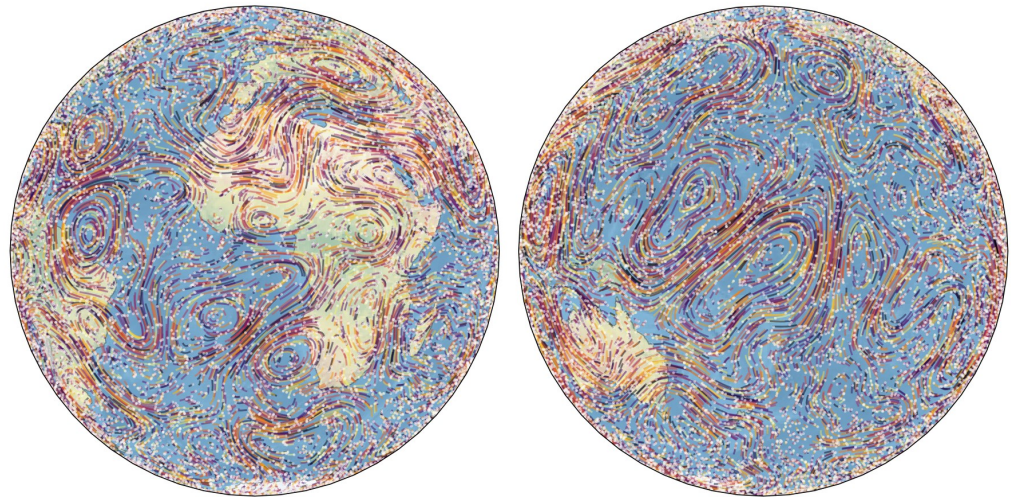
## 86 Statement of need

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

We acknowledge contributions from David Meyer, Mosè Giordano, Valentin Churavy, and Pietro Monticone who have also committed to the SpeedyWeather.jl repository, and the wider Julia community for help and support. MK acknowledges funding from the National Science Foundation. MK and TK acknowledge funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme for the ITHACA grant (no. 741112). NCC acknowledges support by the Australian Research Council under DECRA Fellowship DE210100749 and the Center of Excellence for the Weather of the 21st Century CE230100012.

## References

- Amezcuca, 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>
- Barth, A. (2023). NCDatasets: A Julia package for manipulating netCDF data sets. In *GitHub repository*. <https://github.com/Alexander-Barth/NCDatasets.jl>; GitHub.
- Bezanson, J., Edelman, A., Karpinski, S., & 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*

- 125 *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)  
126 [3C0683:AEOPSM%3E2.3.CO;2](https://doi.org/10.1175/1520-0493(1972)100%3C0683:AEOPSM%3E2.3.CO;2)
- 127 Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., & Brown, B. P. (2020). Dedalus: A  
128 Flexible Framework for Numerical Simulations with Spectral Methods. *Physical Review*  
129 *Research*, 2(2), 023068. <https://doi.org/10.1103/PhysRevResearch.2.023068>
- 130 Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. <http://dask.pydata.org>  
131
- 132 Giorgetta, M. A., Brokopf, R., Crueger, T., Esch, M., Fiedler, S., Helmert, J., Hohenegger,  
133 C., Kornblueh, L., Köhler, M., Manzini, E., Mauritsen, T., Nam, C., Raddatz, T., Rast,  
134 S., Reinert, D., Sakradzija, M., Schmidt, H., Schneck, R., Schnur, R., ... Stevens, B.  
135 (2018). ICON-A, the Atmosphere Component of the ICON Earth System Model: I.  
136 Model Description. *Journal of Advances in Modeling Earth Systems*, 10(7), 1613–1637.  
137 <https://doi.org/10.1029/2017MS001242>
- 138 Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &  
139 Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and  
140 Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, 622(2), 759.  
141 <https://doi.org/10.1086/427976>
- 142 Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau,  
143 D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van  
144 Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ...  
145 Oliphant, T. E. (2020). Array Programming with NumPy. *Nature*, 585(7825), 357–362.  
146 <https://doi.org/10.1038/s41586-020-2649-2>
- 147 Hoskins, B. J., & Simmons, A. J. (1975). A Multi-Layer Spectral Model and the Semi-Implicit  
148 Method. *Quarterly Journal of the Royal Meteorological Society*, 101(429), 637–655.  
149 <https://doi.org/10.1002/qj.49710142918>
- 150 Hotta, D., & Ujiie, M. (2018). A Nestable, Multigrid-Friendly Grid on a Sphere for Global  
151 Spectral Models Based on Clenshaw–Curtis Quadrature. *Quarterly Journal of the Royal*  
152 *Meteorological Society*, 144(714), 1382–1397. <https://doi.org/10.1002/qj.3282>
- 153 Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal*  
154 *of Open Research Software*, 5(1). <https://doi.org/10.5334/jors.148>
- 155 Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &*  
156 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 157 Hurrell, J. W., Holland, M. M., Gent, P. R., Ghan, S., Kay, J. E., Kushner, P. J., Lamarque,  
158 J.-F., Large, W. G., Lawrence, D., Lindsay, K., Lipscomb, W. H., Long, M. C., Mahowald,  
159 N., Marsh, D. R., Neale, R. B., Rasch, P., Vavrus, S., Vertenstein, M., Bader, D., ...  
160 Marshall, S. (2013). The Community Earth System Model: A Framework for Collaborative  
161 Research. *Bulletin of the American Meteorological Society*, 94(9), 1339–1360. <https://doi.org/10.1175/BAMS-D-12-00121.1>  
162
- 163 Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W.  
164 (2019). *A Differentiable Programming System to Bridge Machine Learning and Scientific*  
165 *Computing* (No. arXiv:1907.07587). arXiv. <https://doi.org/10.48550/arXiv.1907.07587>
- 166 Klöwer, M., Düben, P. D., & Palmer, T. N. (2020). Number Formats, Error Mitigation, and  
167 Scope for 16-bit Arithmetics in Weather and Climate Modeling Analyzed With a Shallow  
168 Water Model. *Journal of Advances in Modeling Earth Systems*, 12(10), e2020MS002246.  
169 <https://doi.org/10.1029/2020MS002246>
- 170 Klöwer, M., Hatfield, S., Croci, M., Düben, P. D., & Palmer, T. N. (2022). Fluid Sim-  
171 ulations Accelerated With 16 Bits: Approaching 4x Speedup on A64FX by Squeezing  
172 ShallowWaters.jl Into Float16. *Journal of Advances in Modeling Earth Systems*, 14(2),

- e2021MS002684. <https://doi.org/10.1029/2021MS002684>
- Kucharski, F., Molteni, F., King, M. P., Farneti, R., Kang, I.-S., & Feudale, L. (2013). On the Need of Intermediate Complexity General Circulation Models: A “SPEEDY” Example. *Bulletin of the American Meteorological Society*, 94(1), 25–30. <https://doi.org/10.1175/BAMS-D-11-00238.1>
- Madec, G., Bourdallé-Badie, R., Bouttier, P.-A., Bricaud, C., Bruciaferri, D., Calvert, D., Chanut, J., Clementi, E., Coward, A., Delrosso, D., Ethé, C., Flavoni, S., Graham, T., Harle, J., Iovino, D., Lea, D., Lévy, C., Lovato, T., Martin, N., ... Vancoppenolle, M. (2017). *NEMO ocean engine*. <https://doi.org/10.5281/zenodo.3248739>
- Malardel, S., Wedi, N., Deconinck, N., Diamantakis, M., Kuehnlein, C., Mozdzyński, G., Hamrud, M., & Smolarkiewicz, P. (2016). A New Grid for the IFS. In *ECMWF Newsletter*. <https://www.ecmwf.int/node/15041>. <https://doi.org/10.21957/zwdu9u5i>
- Marshall, J., Adcroft, A., Hill, C., Perelman, L., & Heisey, C. (1997). A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research: Oceans*, 102(C3), 5753–5766. <https://doi.org/10.1029/96JC02775>
- Mazlami, G., Cito, J., & Leitner, P. (2017). Extraction of Microservices from Monolithic Software Architectures. *2017 IEEE International Conference on Web Services (ICWS)*, 524–531. <https://doi.org/10.1109/ICWS.2017.61>
- Met Office. (2010 - 2015). *Cartopy: A cartographic python library with a Matplotlib interface*. <https://scitools.org.uk/cartopy>
- Meyer, D., Grimmond, S., Dueben, P., Hogan, R., & Reeuwijk, M. van. (2022). Machine learning emulation of urban land surface processes. *Journal of Advances in Modeling Earth Systems*, 14(3). <https://doi.org/10.1029/2021ms002744>
- Molteni, F. (2003). Atmospheric Simulations Using a GCM with Simplified Physical Parameterizations. I: Model Climatology and Variability in Multi-Decadal Experiments. *Climate Dynamics*, 20(2), 175–191. <https://doi.org/10.1007/s00382-002-0268-2>
- Moses, W., & Churavy, V. (2020). Instead of Rewriting Foreign Code for Machine Learning, Automatically Synthesize Fast Gradients. *Advances in Neural Information Processing Systems*, 33, 12472–12485. <https://doi.org/10.48550/arXiv.2010.01709>
- Nakano, M., Yashiro, H., Kodama, C., & Tomita, H. (2018). Single Precision in the Dynamical Core of a Nonhydrostatic Global Atmospheric Model: Evaluation Using a Baroclinic Wave Test Case. *Monthly Weather Review*, 146(2), 409–416. <https://doi.org/10.1175/MWR-D-17-0257.1>
- Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza, A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and Friendly Geophysical Fluid Dynamics on GPUs. *Journal of Open Source Software*, 5(53), 2018. <https://doi.org/10.21105/joss.02018>
- Rasp, S., Pritchard, M. S., & Gentile, P. (2018). Deep Learning to Represent Subgrid Processes in Climate Models. *Proceedings of the National Academy of Sciences*, 115(39), 9684–9689. <https://doi.org/10.1073/pnas.1810286115>
- Reinecke, M., & Seljebotn, D. S. (2013). Libsharp - spherical harmonic transforms revisited. *Astronomy and Astrophysics*, 554, A112. <https://doi.org/10.1051/0004-6361/201321494>
- Rose, B. E. J. (2018). CLIMLAB: A Python Toolkit for Interactive, Process-Oriented Climate Modeling. *Journal of Open Source Software*, 3(24), 659. <https://doi.org/10.21105/joss.00659>
- Schneider, T., Behera, S., Boccaletti, G., Deser, C., Emanuel, K., Ferrari, R., Leung, L. R., Lin, 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*  
221 *Climate 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)
- 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)
- 229 Stompor, R. (2011). *S2HAT: Scalable Spherical Harmonic Transform Library*. <https://ascl.net/1110.013>.
- 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>
- 237 Willmert, J. (2020). *Blog Series: Notes on calculating the spherical harmonics*. <https://justinwillmert.com/articles/2020/notes-on-calculating-the-spherical-harmonics/>.
- 238