

Review of egusphere-2025-6266: *JCM v1.0: A Differentiable, Intermediate-Complexity Atmospheric Model*

By Ellen H. Davenport, J. Varan Madan, Rebecca Gjini, Jared Brzenski, Nick Ho, Tien-Yiao Hsu, Yueshan Liang, Zhixing Liu, Veeramakali Manivannan, Eric Pham, Rohith Vutukuru, Andrew I. L. Williams, Zhiqi Yang, Rose Yu, Nicholas J. Lutsko, Stephan Hoyer, and Duncan Watson-Parris

Verdict: **Minor Revision**

This is an excellent paper which describes the development of a new global atmospheric circulation model, JCM. This model is novel for being written entirely in the JAX programming framework, which follows the new style of "tensor programming". This framework provides performance portability, with JAX running smoothly on not just traditional hardware but also accelerators, and intrinsic differentiability. The paper outlines the design of JCM, making reference and comparisons to the classic SPEEDY model from which the physical parametrizations are derived, gives some brief benchmarking numbers on a laptop, and concludes with a few simple examples of the applications of differentiability.

The paper was enjoyable to read, and JCM itself is a very impressive achievement. Congratulations to the authors. I would be happy to accept the paper subject to some minor suggestions below.

Minor Comments

- General: given the similarities and the shared foundation of SPEEDY, I think it would be good to cite the SpeedyWeather.jl project somewhere (I am not an author of this model incidentally).
- Section 2.1, Dynamical Core: could you say a word about the limits of scalability of the JCM dycore? What determines the limit of T425? Is it the memory of the current generation of accelerators? Would it be feasible to do domain decomposition like traditional models, in order to run on distributed memory clusters, and therefore access higher resolutions? Or was it decided a priori that JCM will not target those resolutions for simplicity?
- Line 112: typo - "parametrization" or "parameterization".
- Section 3, Model Interface (API): nothing wrong with code listings, but it's neater to treat them like any other figure and make an explicit reference to them in the text, e.g. "Figure X gives an example usage of JCM".
- Section 5, Performance: I would recommend presenting all computational speed results in common units of e.g. "simulated years (or days) per day" (SYPD/SDPD). This gives a nice intuitive number which summarises what the user cares about.
- Figure 5: the presentation of the benchmark here is a little unusual. The graph shows elapsed time vs. simulation time. I would imagine that all of these lines are in fact straight when plotted on a linear scale, indicating that the wall time per step is constant, assuming each step is identical (e.g. no I/O on every n th step). Then, why bother with a graph at all? All that matters is the gradient - the speed of the simulation in simulated-things-per-actual-thing, which can just be presented in a table. I would recommend that approach.
- Figure 6: could you mention in the caption what hardware you used?
- Line 241: I would be a bit more careful in wording here. ECMWF's IFS has hand-coded tangent-linear and adjoint for its entire atmospheric model, and has maintained them for 25 years now. Maintaining these parallel codes is of course a substantial burden, but far from "intractable".
- Section 6.1, Automatic-differentiation (VJP and JVP): personally I would find this explanation much easier with a few equations. The verbal description left a few unanswered questions for me. For example, what is the mathematical difference between VJP and JVP? If I'm understanding the text correctly, one is simply the transpose of the other. Also, when you say such and such is "computationally efficient" under certain conditions, what is meant exactly?
- Section 6.2, Calibration: could you be specific about what "statistics" you compute in this example?
- Line 311-: again, this code listing would be better as a figure, though this one at least has a reference in the text. Why does the line number start at 13 by the way?