We would like to thank the reviewers for careful consideration of our work and for providing thoughtful feedback that has improved the quality of the manuscript in many ways, including the documentation, the detail and value of the examples, and the clarity of the discussion about automatic differentiation. We have numbered the comments from each reviewer below. Responses are in blue with quoted text from the new tracked-changes manuscript in *italics* with associated line numbers.

Milan Klöwer

The authors present JCM a JAX-based general circulation model whereby the authors contributions is to have translated existing parameterizations from Fortran SPEEDY into JAX so they can be used together with the existing dynamical core Dinosaur (which was developed for NeuralGCM). I very much agree with the overall narrative of the paper: Modern GCMs written in easily accessible and productive languages (such as Python) and for modern hardware (GPU, TPU) with modern features such as differentiability can lower the barrier for climate modelling and accelerate climate research. In that sense, the paper is timely, important and bridges existing model (Dinosaur and Fortran SPEEDY's parameterizations) for a useful and user-friendly model.

I do recommend this paper for publication after following points have been addressed

Major points:

1. Similar efforts by other authors: The introduction currently misses many modelling efforts pursued by other teams that achieve similar goals: On the Python side there is the ocean model Veros (only mentioned in the conclusions) which is also JAX-based and which has been tested on multiple GPUs too, and I believe they also have been working with JAX's AD capabilities too. Please cite them accordingly, otherwise the introduction reads as if JCM is the first JAX-based Earth-system model component that as GPU/TPU and differentiability in mind. On the Julia side there are also many efforts, primarily Oceananigans (e.g. https://arxiv.org/abs/2502.14148) and SpeedyWeather (https://doi.org/10.21105/joss.06323) with the latter being the closest to your efforts. AD for both are documented in https://doi.org/10.22541/essoar.176314951.18114616/v1. Otherwise the reader would get an incomplete summary on ongoing modelling efforts and the literature published on them.

   SpeedyWeather is an important and very relevant precursor to this work, and we have added a citation in the introduction, which was inadvertently omitted. We have also added a citation for (Wagner et al., 2025), which is a precursor for the open-source and GPU-accelerated nature of our work. Lastly, we have added additional mention of Veros and DifferLand in the introduction (previously only mentioned in the conclusion). We have not added a citation for (Moses et al., 2025) as it is a concurrent preprint which was submitted very shortly before this manuscript.

   Updated text (lines 67–69): *"This work builds upon a growing movement to generate modular, differentiable Earth system components in modern programming languages with GPU acceleration (Klöwer et al., 2024; Häfner et al., 2018; Fang et al., 2024; Wagner et al., 2025)."*

2. Spectral ringing (Fig 2a, b, Fig. 4a-c). Evident from the figures is a strong spectral ringing present in JCM simulations. They do seem to be constant in time such that I would conclude

they do not necessarily present a dynamic (e.g. CFL) stability issue in, e.g., strong winds. But they may originate as a consequence of orography. While you could use a super smooth orography this also may show deficiencies in your dynamical core. Some spectral ringing isn't strictly problematic as it's the equivalent of step-wise changes of smooth functions in grid-point models but given that this particularly projects onto precipitation your model clearly seems to react to "fake mountains" somehow. I suggest to include a discussion on this and test and report on the associated stability. A little bit of ringing is not a problem but a lot of ringing can easily make the model unusable through numerical instability.

We agree this warrants investigation and have been looking into it. Based on our testing, we believe that it stems from two things: (1) ringing in low-resolution spectral models is a known phenomenon (there is some ringing seen in the results from speedy.f90 in Fig.5), and (2) additional ringing induced by the dynamical core. We will be looking into the role of the dynamical core in this phenomenon more in future work. As of now we have run several "stress" tests of the current configuration (e.g., running for 100 years) in order to verify that the signal is not growing/leading to instability. In these tests we do not see signs that the ringing grows over time, nor do we see any induced instability. We are able to take the gradients in a variety of applications, including over these long runs. We have added a statement about the expected stability of the model to the text, and we do not believe that these artifacts lead to issues when actually using the model.

Updated Text (lines 215–219): *"Spectral ringing in low-resolution models is a common phenomenon (Durran, 1999). Artifacts of this type of ringing can be seen in Figure 5 in the results from both models. However, the ringing in JCM is stronger than in speedy.f90. We have investigated whether this difference affects model stability, and we have found no indication that it grows over time in these low-resolution configurations (T31-T85). This includes running the model for 100-year runs and taking gradients through those long integrations."*

Kudos to your efforts and many thanks for writing them up.

Minor points:

1. l30: Many/most operational NWP models use a hand-written adjoint? It's still tedious, error-prone and can massively slow down code development or require additional developers to maintain so your point still stands.

   The text has been updated to be more accurate.

   Updated Text (lines 29–30): *"The analytical gradients of these functions are often unknown or are challenging to derive by hand."*

2. l48: I guess you argue that the gained stability in hybrid models over MLWP is because NeuralGCM can be integrated for decades while e.g. GraphCast/GenCast can't? I don't disagree with this but it's still an apple to pears comparison as these models use vastly different ML architectures too. A fair comparison was done in https://doi.org/10.1029/2025MS004969

   We have included a mention of this study (which is a nice demonstration of the point) in

addition to our mention of NeuralGCM.

Updated Text (lines 47–49): *"This can significantly improve the stability and accuracy of the resulting hybrid models. These benefits are exemplified by NeuralGCM, a hybrid atmospheric model that replaces the full physics-suite with a neural network (Kochkov et al., 2024) and by the recent study of Gelbrecht et al. (2025)."*

3. l54: ICON won the Gordon Bell prize with a heterogeneous Fortran-CPU-GPU computing so not impossible, but given they required a whole team ended up with less than 50% of lines of code actually describing algorithms (most is OpenMP/MPI/CUDA boilerplate instructions for the compiler, wrapped around Fortran loops) and sold this positively as "separation of concerns" you have a very good argument for "extremely costly" and likely also unwieldy and killing a lot of user-friendliness.

   Thanks for this example. It is a good demonstration of what we were hoping to convey. We have removed the phrase "impossible" to be more accurate.

4. l81: geopotential is not a prognostic variable in the hydrostatic equations? Rather diagnosed from surface geopotential (gravity*orography) and the locally vertical virtual temperature profile?

   Yes, thank you for pointing this out. It is a diagnostic and not prognostic variable in SPEEDY. The description has been fixed.

   Updated Text (lines 83–84): *"The dycore's prognostic variables include horizontal wind (vorticity and divergence), temperature (anomaly from a reference temperature), and log surface pressure."*

5. l84: T? are the *spectral* resolutions (or truncation) which can be paired with different *grid* resolutions, presumably the full Gaussian grid with N latitudes yielding linear, quadratic or cubic truncation. Fortran SPEEDY uses T31 with a 96x48 full Gaussian grid which is a quadratic truncation. Your sentences here could be clearer on this?

   Clarifying text has been added.

   Updated Text (lines 85–89): *"Dinosaur supports a range of horizontal grid resolutions; these are labeled by their spectral truncation, the maximum wavenumber of spherical harmonic they can resolve. Dinosaur's grids use a quadratic truncation, meaning there are 3 grid points per wavelength for the highest resolved wavenumber.*

   Supported spectral truncations range from T21 (a 64x32 grid) to T425 (a 1280x640 grid), with JCM's default being T31 (96x48, roughly 4°×4° grid cell size at the equator)."

6. l95: Are these the same or different from Dinosaur or Fortran SPEEDY?

   Exponential filters are no longer applied, and the only filtering is that associated with horizontal diffusion. This has been updated in the text. The horizontal diffusion filtering is different from what is implemented in speedy.f90 and is described in more detail in the Model Validation section.

Updated Text (lines 102–104): *"After computation of parameterized tendencies at each time step, horizontal diffusion of the prognostic variables is computed and the surface pressure field is corrected to maintain constant global mean surface pressure (mass conservation)."*

7. l100: The Fortran 90 version was written by Sam Hatfield, maybe give him credit?

   The Fortran 90 version by Sam Hatfield is cited on line 177 of the original manuscript.

8. l126: Fortran SPEEDY has no daily cycle, have you changed that?

   We have not changed this, and it is now mentioned in the manuscript.

   Updated Text (lines 119–120): *"Neither SPEEDY nor JCM v1.1 implements a daily cycle."*

9. l148: The time scales (or equivalently a normalization of the diffusion operator) should also depend on the horizontal resolution, is this currently done?

   Yes—this is already accounted for. The normalization uses the maximum Laplacian eigenvalue (i.e., the grid scale), so the specified timescale applies at the smallest resolved scale and automatically adjusts with horizontal resolution.

10. l152: These appear in the code but aren't documented in Fortran SPEEDY, I also don't know why there were introduced. SpeedyWeather left them out.

    Yes, we are in the same situation and have left these out for now. We are going to investigate whether it is worth reintroducing.

11. l172: Why is 'forcing' and option of model.run and not of the model constructor 'Model'?

    The intent here is that a Model's geometry (resolution, orography, and time step) will be fixed at construction, but the forcing, much like initial conditions, is a natural thing to vary between different runs of the same Model. This could be in the context of studying the dependence of a model run on different surface boundary conditions, or in the context of e.g a set of consecutive 1-year runs forced by a multi-year time series (where a different chunk of the time series would be the forcing for each run). This is especially useful if we want to optimize over surface forcing conditions (like in variational DA), because it prevents recompilation of the model run function.

12. l182: Most of the forcing (in terms of energy) comes from solar radiation?

    Sorry for confusion, this was a mis-statement. The forcing is monthly climatology obtained from ERA-Interim and includes several variables which are now listed in the text.

    Updated Text (lines 178–180): *"The models are forced by a monthly climatology of sea-surface temperature, surface albedo, land surface temperature, soil moisture, sea ice concentration, and snow cover along with realistic orography."*

13. l184: I thought a 30min time step are default?

30 min time step is the default in the JCM, and 40 min is the default in speedy.f90. We had to pick one of the defaults in order to make the test consistent between models, and we arbitrarily decided to use the speedy.f90 default.

14. l193: What is this wind drag? Only in the PBL? Why did you leave it out?

    There appears to be a linear wind drag applied to the zonal mean vorticity and divergence in the stratosphere (speedy.f90, time_stepping.f90:79-83). It is not documented in SPEEDY or speedy.f90 (similar to the orographic corrections), so we have initially left it out of our implementation. Similar to the orographic corrections, we are investigating the necessity of this term in the JCM if the priority is to bring it as close as possible to the Fortran implementation.

    Updated Text (lines 193–196): *"Although the general formulation of the spectral dynamical cores is the same, they are not identical and the additional choices to use Runge-Kutta and explicit horizontal diffusion in the JCM lead to differing results. Speedy.f90 also implements additional linear drag on the mean vorticity and divergence in the stratosphere (this drag is undocumented and thus currently excluded from the JCM)."*

15. l207: Is this because you apply a much stronger diffusion? implicit 4th order Laplacian vs explicit 2nd order (or even 1st for divergence)?

    We apply explicit Laplacian diffusion: 2nd-order (Laplacian) for divergence, and 4th-order (Laplacian squared) for vorticity, temperature, and humidity. Lower-order diffusion acts more strongly on large scales, while higher-order diffusion preferentially damps small scales. Although our explicit scheme is inherently less diffusive per timestep than the implicit scheme in SPEEDY, some fields in JCM appear more diffuse because our 2nd-order and 4th-order diffusion target larger scales than SPEEDY's effective 8th-order hyperdiffusion. Speedy.f90 implements leap-frog + implicit diffusion + RAW filter and JCM implements Runge-Kutta + explicit diffusion. We've added a note in the text about the larger scales involved (the time-stepping is discussed in the section on Model Parameterization).

    Updated Text (lines 212–213): *"precipitation due to convection is more concentrated in speedy.f90 and more diffuse in JCM, likely due to the larger spatial scales of the horizontal diffusion."*

16. l228: Why only sometimes? Otherwise you are violating CFL?

    Updated Text (lines 238–239): *"It should be noted that in practice, the runtime for the higher resolutions may pick up another factor of 2 or 3 relative to lower resolutions due to reductions to the model timestep that are eventually needed for stability."*

17. l229: Do you mean undersaturation of GPU/TPU threads at lower resolution?

    Yes, though this section has now been reworked to present the data differently (and new performance data has been computed). The difference between the observed scaling and theoretical quadratic scaling no longer seems apparent/notable so we're no longer making this claim.

18. l232: Give these timings in simulated year per day (SYPD) as it's the more common unit for performance?

    Original Text: "Figure 5 shows the total time to run a default configuration of JCM at T85 (approx. 150km resolution at the Equator) using either a laptop CPU, laptop GPU (NVIDIA RTX 4050), or Google Cloud TPU. The behavior in the figure has been our typical experience: runtime on GPUs and TPUs is at least an order of magnitude faster than on CPUs (except for Apple Silicon chips), with parallelization (sharding the model state arrays so they can be distributed among multiple devices) giving a significant speedup to the TPU runs on Google Cloud. For this set of parameters, the laptop GPU was faster than the cloud TPUs, but initial testing indicates that the TPU (like any high-performance cloud platform) will scale better for higher resolutions than the laptop GPU with its limited VRAM. The platforms used here do not all have the same number of cores or clock speeds; the goal is to illustrate typical runtime on common devices.

    As another point of comparison, JCM on the RTX 4050 laptop GPU was about 40% faster per year of model time than speedy.f90 on the same laptop's Intel i9 13th Gen CPU.

    [...]

    With regards to the model run configuration itself, the number of operations for a model run typically increases with the square of the spectral truncation, with an additional factor for reductions to the model timestep (sometimes needed for stability at a higher resolution). However, runtime is observed to scale better than the number of operations, presumably due to parallelization. Figure 6 shows how model runtime scales with the spatial resolution of the model grid, using as a benchmark a one year simulation with default 30 minute time step, run on a NVIDIA RTX 4050 laptop GPU. At T31 resolution, one year of model time takes only a minute and a half, and even T119 can be run for a year in less than 10 minutes."

    Updated Text (lines 223–240): *"Figure 6 shows the simulation speed, in units of simulated years per day (SYPD), of running JCM at various resolutions and on various device types. The behavior in the figure has been our typical experience: runtime on GPUs and TPUs is significantly faster than on CPUs, with parallelization (sharding the model state arrays so they can be distributed among multiple devices) giving an additional significant speedup to TPU runs on Google Cloud (parallelization was not used for the results in Figure 6). The dashed line in the figure represents the theoretical quadratic scaling of number of operations (and memory requirement) with spectral truncation.*

    [...]

    It should be noted that in practice, the runtime for the higher resolutions may pick up another factor of 2 or 3 relative to lower resolutions due to reductions to the model timestep that are eventually needed for stability. For the results shown in Figure 6, the timestep was left as the default 30 minutes for all resolutions."

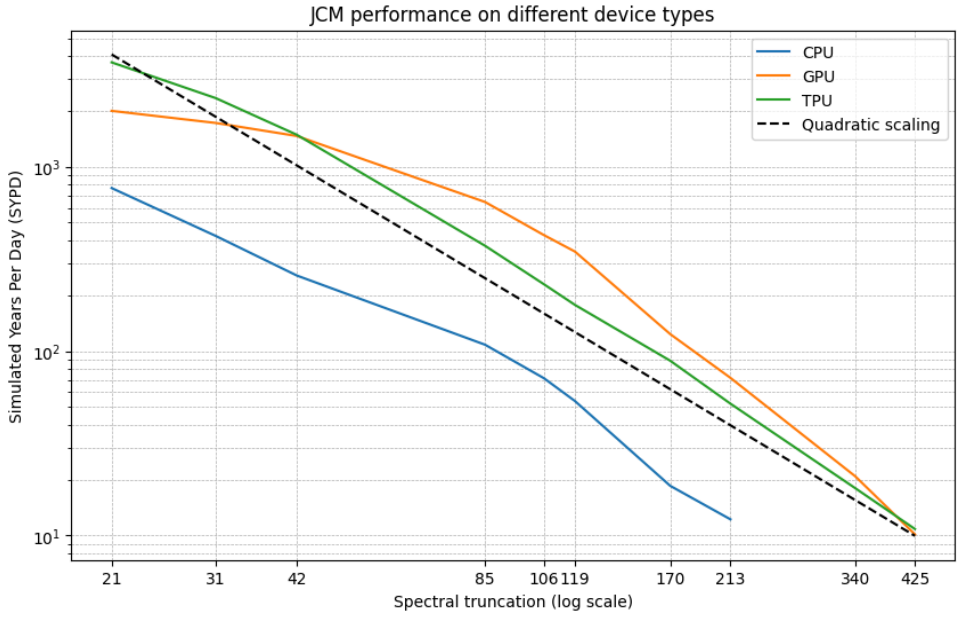
19. Fig. 5: If you put simulation period on the xaxis the this graph actual measures the overhead of initializating/compiling/precomputing (depends on what happens when you hit model.run) rather than the actual performance after that? Given you use a log yaxis that information should be in the vertical offset of the lines in the limit? I believe this is the actually interesting performance metric. The initial overhead may depend a lot on what's done at construction via 'Model' and what's done at initialization 'model.run'? Which is still useful information to be added! E.g. can a user run many performant 10-day simulations starting from varying initial conditions? Atm there seems to be quite an overhead for doing that!

Good questions–the plot now shows SYPD, excluding compilation time, which hopefully avoids some of this confusion. We do have a way of avoiding re-compilation for varying initial conditions and forcings, and have made that explicit in the manuscript.

Added text (lines 230–234): *"Before a run is executed, there is a small amount of compilation time (10-15 seconds for typical runs at default resolution) which varies little with resolution and run length. Advanced users conducting many short runs with varying initial conditions or forcings should use jcm.Model.run_from_state, a slight variant of jcm.Model.run which after the first run with a given resolution, timestep, save interval and total time requires no compilation time for any subsequent runs with the same values for those parameters. We encourage users to read the JCM documentation for details."*

20. Fig. 6: Y axis in SYPD? Add CPU, GPU, TPU performance to inform about where low/high resolutions are faster on CPU vs GPU/TPU?

    Done! Figure 6 has been replaced with a plot showing SYPD over a range of resolutions on a CPU, GPU, and TPU:



21. l241: See also comment on NWP adjoints above.

    This comment is addressed in response to Reviewer 1 (question 8), the updated text is pasted below.

    Updated Text (lines 250–252): *"Given the complexity of state-of-the-art Fortran models, this manual generation of gradients is expensive to implement and maintain. Thus, in most cases, gradient information is inaccessible or presents a substantial burden to model developers."*

22. l266: I agree with you that such a simple example is the first evidence that the method works but maybe note that for your example only the sign of the gradient has to be correct for the

optimization to work?

The simple example has been upgraded to a two-parameter example in response to Reviewer 2 (question 6). While it is still not realistically complex compared to cutting-edge parameter estimation challenges (so we have left our statement about simplicity), success of the two-parameter example depends on both the sign and magnitude of the gradients.

23. Fig 8/9: I'm missing a very brief discussion on the results of the sensitivity analysis. I agree that a proper analysis is beyond the scope of this paper but a) have you checked that the gradients are correct with finite differences? and b) it looks like there's some dynamically consistent response in the system that could be briefly explained, e.g. is the wind geostrophically adjusting to a higher geopotential due to higher SST? This would also provide more evidence on the correctness.

    We have tested gradients of the parameterizations using JAX built in verification methods, which compare the automatically derived values to finite differences. These tests are a part of our unit testing suite. We have noted this in the text.

    Updated Text (lines 244–246): *"The JAX library provides verification functions to test model gradients against finite difference estimates, which are included in the unit testing suite for JCM."*

    We have also added a brief explanation of the dynamical response seen in the sensitivities section.

    Updated Text (lines 324–329): *"The localized warm SST anomaly centered near the equator drives a rapid, large-scale dynamical response in the upper-level winds, with both meridional and zonal wind perturbations propagating globally within 12–36 hours. The alternating meridional wind response straddling the equator resembles Rossby wave dispersion from an equatorial heat source (Matsuno, (1966), Gill (1980)). The TOA outgoing longwave radiation response remains local to the SST forcing throughout the 36-hour period, indicating that while the dynamical adjustment is fast and far-reaching, the thermodynamic/radiative response is still confined near the heating source on this timescale."*