Response to Anonymous Referee 1

Langwen Huang and David Topping

Dear colleague. Thank you for taking the time to provide a review of our submitted manuscript, submitted on 29 November 2020. We are pleased your review is supportive of our work and we are of course happy to response to points raised and revise the manuscript accordingly. Please find our responses with any suggested changes in the manuscript below.

General comment 1): The authors mention that sometimes the time-step length has to be reduced in order to preserve the positive sign of the chemical compounds. I'm wondering how often does that happen and how big of a potential bottleneck could this become. Are there other algorithms that could be implemented to deal with this problem that would not require decreasing the time step?

Response: Negative values occurs when a inaccurate Jacobian is evaluated in an implicit ODE solver at a state with 0 concentrations but nonzero tendencies. In practice, this never happens unless the Jacobian or its preconditioner is incorrect or inaccurate. As a result, JlBox almost never shortens the time-step to conserve positivity. However, implicit ODE solvers do decrease the time steps when strong stiffness is detected (e.g. Newton step takes too many iterations, or predicted error is greater than expected). This occurs in some initial conditions. We believe that it is the nature of the ODE system and one cannot do much better compared to current state.

General comment 2): I was wondering if the performance discussion could be extended a little. Table B2 provides simulation times for selected 4 modeling cases. As mentioned before, I'm not an atmospheric chemist, but I was wondering if a plot with the number of chemical compounds, reaction pathways or size bins on one axis and elapsed time on another for the different box models would be useful. This would show how the computational cost scales with the problem complexity for both Julia and Python implementations. I was also wondering, for the large scale simulations performed on the cluster would it be possible to show some scaling plots with the increasing number of processors? Additionally, the authors briefly mention the potential to extend the JlBox to GPU accelerators. What would be the expected performance gain over the current simulations. Are there any other models similar to JlBox that are already running on GPUs?

Response: Regarding the first point, we also agree it is a good idea to extend the number of cases used to demonstrate performance scaling. In the new manuscript we have now included additional use cases in Figure B1 that plots the simulation time as a function of number of size bins with different initial conditions (parent VOC and single/mixed option). Whilst each VOC will have varying impacts on the time-to-solution by virtue of the stiffness of the problem and parameters that dictate partitioning, the plot demonstrates that the running time of JlBox with a sparse Jacobian scales roughly linearly with number of size bins. This enables JlBox to perform simulations with higher complexity than was ever possible using PyBox.

Regarding the second point, currently JlBox does not explicitly use multiple cores. While the ODE solver TRBDF2 and CVODE_BDF do utilise multiple cores, the parallelised component is not a dominant contributor to the total simulation time,

making the scaling test easily hitting upper bound of Amdahl's law. In practice, comparative speedups of 200% were only observed in simulations based on the full MCM mechanism that utilised a sparse Jacobian. However, the CPU consumption never exceed 400% in those scenarios. As a result, we did not provide a separate investigation into multi-core use at this stage and allocated 4 cores in all benchmarks.

For the third point, the advantage of running JlBox on a GPU is that the GPU has higher memory bandwidth allowing faster sparse matrix operations. It also has higher floating point computation throughput which is beneficial when inverting dense Jacobian matrices and computing right-hand-side functions. There are efforts to port atmospheric models into GPUs like Linford et al. (2010); Sun et al. (2018); Alvanos and Christoudias (2017). However, these efforts only focus on gas-phase kinetics and small mechanisms designed specifically for use in global chemistry models. As far as we know, there is no other models similar to JlBox that focus on multi-phase and large mechanisms. For sure we agree this is a very interesting area and we should further explore opportunities for parallelisation in the future.

Minor comments

Minor comment 1): line 38 - Is the text in italics a quote from somewhere? If yes, could you provide thereference? If not, why is it highlighted?

Response: Yes, our apologies. This text is taken from the Julia documentation. In the new manuscript we will add the following reference: (Julia Documentation: https://julia-doc.readthedocs.io/en/latest/manual/introduction/)

Minor comment 2): ∼line 56 - Add section 5 into the list of paper sections described here.

Response: Thank you for identifying this. We have now added the following text: In section 5 we discuss the relative merits of JlBox in comparison with other models whilst presenting a narrative on required future developments.

Minor comment 3): line 109 - JlBox is written

Response: We have made sure all instances of JlBox are now consistent.

Minor comment 4): line 249 - previously different fonts were used for package names such as Differen-tialEquations.j

Response: Apologies, we have now changed the formatting to `DifferentialEquations.jl` to be consistent with other references to Julia packages.

Minor comment 5): - line 262 - I would cross out "simply

Response: This has now been deleted.

Minor comment 6): - subsection 4.1 - Maybe the header could be "Validation against existing box-models" the repeated model seems off

Response: This has now been changed.

Minor comment 7): - Table 3 - Python library instead of python library, UManSysprop instead of Umansysprop, Numba should be capitalized too

Response: These have now been corrected.

Comments after cloning from the GitHub repo: I tried compiling and running the project following instructions from GitHub. I ran into several small problems, but in general I was able to execute the tests and example simulations. I'm listing the problems I had below as feedback, but it does not concern the manuscript itself.When running the tests I got an error in the Gas Phase Sparse with Matrix-free operator test complaining about UndefVarError: AnalyticalJacVecOperator not defined. I manually updated DiffEqOperators which solved the problem. After that, all the tests passed.When running one of the example simulations I got some additional complaints about packages not being installed. I manually added them via Julia package manager following what the error messages were suggesting. In general it seems that the build and test stage did not set up correctly all the dependencies for me but I was able to easily resolve that.After that I started getting errors about ../data/.txt files being missing. The way I understood the "Get Started" section, it suggests executing the simulations from the JlBox folder by include("example/Simulation_*.jl"). But the simulations themselves look for the .txt files in ../data/ folder. The correct way for me to execute the simulations without changing the files was to include("Simulation_*.jl") inside the example folder inside Julia REPL. It might be worth it to update the "Get Started" section on GitHub to clarify that. Everything worked otherwise. As future work it would be great to include some example plotting scripts in Julia within the GitHub repo and to add a binder setup. This would allow the new users to run and plot the model from the web browser and showcase even better the strength of using Julia where both the high-performance computing and the analysis can be done in one programming language. It would also help a lot to get the new users up to speed in running and visualizing the simulations on their own. I also think that package naming conventions in Julia suggest not having Julia in their name and instead ask for the package name to finish with.jlextension (https://julialang.github.io/Pkg.jl/v1/creating-packages/#Package-naming-guidelines). Might be too late to suggest changing the package name now, but I thought I should leave it as a comment*

Response: We apologize for the inconvenience of using JlBox following outdated documentation. The documentation has subsequently been updated. All examples have been fixed so that they can find the correct data path independent of the working directory. We also appreciate the idea of using Binder, so we have setup a binder link in the repository as well as providing guidelines to build docker images. We have also updated the Zenodo archive snapshots of the repository as a result.

For the naming of JlBox, it is a bit unfortunate that this name was conceived as a successor of PyBox at summer of 2018 when the naming convention was less clear.

References

Alvanos, M. and Christoudias, T.: GPU-accelerated atmospheric chemical kinetics in the ECHAM/MESSy (EMAC) Earth system model (version 2.52), Geoscientific Model Development, 10, 3679–3693, 2017.

Linford, J. C., Michalakes, J., Vachharajani, M., and Sandu, A.: Automatic generation of multicore chemical kernels, IEEE Transactions on Parallel and Distributed Systems, 22, 119–131, 2010.

Sun, J., Fu, J. S., Drake, J. B., Zhu, Q., Haidar, A., Gates, M., Tomov, S., and Dongarra, J.: Computational benefit of GPU optimization for the atmospheric chemistry modeling, Journal of Advances in Modeling Earth Systems, 10, 1952–1969, 2018.