Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-26-AC1, 2020

Geoscientific Model Development Discussions
Open Access
EGU

Interactive comment on "Rapid development of fast and flexible environmental models: The Mobius framework v1.0" *by* Magnus Dahler Norling et al.

Magnus Dahler Norling et al.

magnus.dahler.norling@gmail.com

Received and published: 19 November 2020

We thank the reviewer for taking their time to do the review.

As a reader interested in general ode models paragraph 280 is not clear. If the goal is to compare manual vs autocalibration of model parameters the starting points should be independent (perhaps random in feasible ranges?). Why do the authors use the results of manual optimization to start the auto-calibration?

The goal was not to compare manual to autocalibration, but to show one possible workflow when calibrating the model and iterating on model structures (i.e. use autocalibra-

Printer-friendly version

Discussion paper

tion to "fine-tune" a manual calibration). We are not the authors of the autocalibration algorithms, and so the point is not to show off how good these are at reaching good solutions from random starting points. We are only showing one possible way these python libraries can be used with models built in Mobius. Feasibility of autocalibration would also be highly dependent on how complex the model structure is. This could be a selling point for a given model (it is easy to auto-calibrate), but the model we present is just there to illustrate how one can use the framework, and this model itself is not the main point of the paper.

If the authors foresee that the changes in code would break the experiments and files referenced in the paper they can include the commit number (or date) that preserves the experiments since the repository seems to be actively developed

Future edits to the github repository should not break the experiments described in the paper. However, we also made an archived version of the repository corresponding to this paper, which is linked to in section 5 (line 376).

The benchmark of runtime experiments can use a better description. Paragraph 335 mentions: "Results of the benchmarking show that Mobius models have a slight performance loss compared to hard-coded C++ models but run several orders of magnitude faster than hard-coded Python models (Table 3)". If the source files for these benchmarks are also included in the github repository, please reference them in the paper.

The authors should give a simple description of the hardware they are using to run tests. Just the manufacturer, number of cores, and frequency of the CPU is enough. This will ensure the timings have enough context.

The source files for the benchmarking are referenced in the paper, line 335 "Code used in these experiments can be found in the "Evaluation" subfolder of the Mobius repository". Description of the hardware used in benchmarking can be provided in the revised manuscript. Note that we only report the ratio between the run times of

different implementations. The run times themselves, are highly hardware dependent, while the ratios are fairly stable across several different machines we tried. What we want to show is how fast a framework-implemented model runs compared to other implementation options.

The number of cores is not going to be that relevant since a single model run is single threaded (while you can parallellize if you want to run the model many times such as in MCMC).

The timing experiments are average model evaluation runs. If possible (at least for the Mobius model using the python interface), It would be valuable to report optimization times in a separate table.

I'm not sure what precisely is asked for here. Optimization times are highly variable, depending on algorithm, the model used, input data etc. These together determine how many model evaluations are needed to reach convergence. The number of evaluations to reach convergence are independent of the model implementation (as long as the implementations give identical results). The model implementation only determines how fast each model evaluation is, and so that is what we focused on in the benchmarking. In other words the time for the optimization is roughly N*t, where N is the number of evaluations and t is the time for a single evaluation (the total time can be reduced if the algorithm allows paralellisation, and t can be variable due to different convergence speed of ODE solvers depending on parameter values). The number N is not controlled by the model implementation, but by the optimization algorithm (and also the specific use case). Since the concern of this paper is the model implementation (framework), we focus on the single-run time in the benchmarking.

Nevertheless, we did provide the timing for one particular optimization run on line 287 (using one specific algorithm, model setup and machine) just to give a general idea that using autocalibration is feasible in fast-iteration model building.

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-26,

2020.