Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-3-RC2, 2018

Geoscientific Model Development Discussions
Open Access
EGU

Interactive comment on "Veros v0.1 – a Fast and Versatile Ocean Simulator in Pure Python" *by* Dion Häfner et al.

Anonymous Referee #2

Received and published: 21 June 2018

The paper presents a purely python based ocean model (VEROS), which is translated from the the pyOM2 model with a fortran backbone. The author addresses many of the implementation obstacles that had to be solved in python to obtain a python based ocean model with a descent performance. In general the author did a good job in presenting their way of translating and implementing VEROS. To my knowledge, is the introduction of a purely python based ocean model, mainly for teaching purposes and as an easy entry for beginners as well as a fast testing tool for new ideas, a novel effort in the ocean model community. I would therefor recommend that the paper is accepted after some minor revision.

General Comments: -Introduction,line 6: ...using the low level programming language fortran... I think it should be mentioned that fortran is only one of the programming

Printer-friendly version

Discussion paper

languages used in the ocean model community, not THE language. There are plenty of models that are also written in C or C++.

-Introduction, line 8-10: ...violates a core principle of science: reproducibility... I don't see how complexity violates reproducibility. Its the job of the programmer and the development community to do implementation step by by step and that new implemented features are carefully tested, but this inherits any kind of programming effort independent of the used language.

-Introduction, line 11-12: ...designed with the explicit goal to improve code structure and readability... Any code in any language can be structured and commented that "anybody" can read and understand it, but only when the responsible programmer cares about. The difference in python to all other languages I know is, that the structure of the code (line intents, tabs, spaces...) is a necessary part of the syntax of python, which forces the programmer to structure its code to a certain extend.

-Introduction, line 16-17: ...a substantial amount of ... projects is devoted to understanding, writing, and debugging legacy Fortran code... Understanding writing and debugging is part of any kind of model programming effort, irrespective of the used language, that is a burden one always has to deal with. I think the big advantage of the high level programming language python is, that its first unless like MATLAB fully open source, so there are no nasty licensing issues to address for any package and second that the running of the code and the visualization of any kind of model variable can be done theoretically together. This would make it much easier, especially for beginners, to understand what is going on in the model and speed up any debugging work-flow considerably. Low level programming languages only allow limited output to the screen/log-file or need own complicated output routines to write out more complex variables which are visualized with something afterwards which makes it often time and resources consuming to find the origin of bugs.

- 2.1 From Fortran to naive python, line 28: ...arbitrary indexing in Fortran... I'm not

sure what the author means here with the term “arbitrary indexing”. Also indexing in Fortran is anything else than arbitrary.

- 2.3.3 Multi-threaded I/O with Compression, line 27: . . .ranging from Gigabytes to Petabytes... I haven’t met yet any model application where single output files in the size of Petabyte where written. Did the author meant Terabyte ?

- 2.3.3 Multi-threaded I/O with Compression, line 29-30: Since writing output becomes more and more to a critical bottleneck especially, for large model configurations, it would be nice if the author could describe a bit more in detail how writing the output is organized in VEROS especially with respect to the separated threads. How is prevented that the data are overwritten when the model runs further, while one thread is writing out ?, Are the output data duplicated for writing the output?, Does it affect the RAM demand of the model? . . .

-2.3.5 Modular Diagnostic Interface: How does VEROS structure the output, does it follow the CMIP protocol, one file for one variable or it combine several variables into a file?

-4.1. Modified Geometry with flexible Resolution: The author should mention what he used as forcing to obtain these results

It would be nice if the author could also make some statements about the memory (RAM) demand between VEROS and pyOM2, when running the same configuration. Are they the same?, Are there differences in the size of the model configuration that VEROS can handle compared to pyOM2...

Technical Comments: - page1, line15: ...to further advance our. . .

- page3, line26 (same page8, line9): 2.1 From Fortran to naïive Python

- page9, line24: ...and the implementation of

- page16, line21: ...(Chelton and coauthors et al. ,1998)

- page16, line25: ...(Chelton and coauthors et al. ,1998)

- page19, line 34: . . . Last Glacial Meaximum. . .

- page 21,line 4: ...PopularitYy...

Please also note the supplement to this comment:
https://www.geosci-model-dev-discuss.net/gmd-2018-3/gmd-2018-3-RC2-supplement.pdf

---

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-3, 2018.