Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-3-AC1, 2018

Geoscientific Model Development Discussions

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

Dion Häfner et al.

mail@dionhaefner.de

Received and published: 21 April 2018

Dear referee,

thank you very much for your detailed review. We shall address the issues you raised point-by-point below. A 'latexdiff' of the revised manuscript is attached as a supplement to this response.

Please let us know should you have further questions.

On behalf of the authors,

Dion Häfner

1 General comments

> The manuscript is introducing a new generation of ocean circulation model coded in Python. If it is the first Python code for those applications, it would be intersting to mention it. If not, it would be relevant to introduce other similar applications.

To our knowledge, Veros is the first serious approach to a pure Python ocean model. There are several projects that provide a Pythonic front-end to some wrapped Fortran code, such as CliMT [1], OOFε [2], or Veros' parent project, PyOM [3], but none that go all the way. This may be connected to the fact that there is no obvious "right" way to parallelize Python / NumPy code. I added a paragraph to the introduction.

> Veros is based on a wide range of Python libraries. As this code will be used, for example, for educational purpose, can the authors detail the code management plan considering potential near future compatibility issues between those libraries ?

In an ecosystem as volatile as the scientific Python stack, proper dependency management is indeed important. So far we have only experienced minor compatibility issues, which were easily fixed by requiring certain minimum versions of dependencies in 'setup.py'. We thus chose not to discuss this in the manuscript. In case dependency issues should become a major concern in the future, we would probably resort to a package manager like 'conda', and supply an official 'conda' recipe for Veros (so people could use 'conda install veros' to get a working copy of all dependencies and the Veros code).

> The Veros code is developed for single-node computation. Can the authors

> discuss the potential near future extent of the code for parallel computing (several nodes) and then more expensive applications ?

Implementing distributed memory support in Bohrium won't be a trivial task, but it could be done (see below). Apart from that, there are other libraries we could leverage for multi-node support, such as 'dask.array' in conjunction with 'dask.distributed'. However, distributed simulation in Python is not very mature, and we don't know yet how any given solution will perform in practice. We thus don't feel comfortable to speculate too much about a possible time frame.

> Despite the description of advantages of Python language, it seems that the code is mainly designed for educationnal purpose. Could the authors confirm this or detail those advantages in the manuscript ?

Yes and no. Veros is built for getting rapid insights into how the ocean works. This is of course very useful for students, but there are still many fundamental open questions regarding ocean mechanics that are to be addressed on a research level, and where traditional models might be too inflexible. I added a sentence to the introduction.

> A last general comment is referring to the experiment. The choice of the model grid is surprising. It is not self explanatory why the straight meridional line is used in the Atlantic.

The transition to section 4.1 was indeed a bit harsh. I have added a paragraph on our motivation.

> It seems that coastline modification is manually modified outside Veros code and, then, it does not show Veros extended functionnality as mentionned in section 4 (p. 14 Âń uncomplicated ways to modify the coastline Âż)

The sentence you quoted was indeed poorly worded. I have added a paragraph. Keep in mind that we do *not* modify the input data sets for bathymetry, forcing fields, and intial conditions. All we supply is a binary mask image, and Veros does the rest (including interpolating everything to the chosen, flexible, resolution).

2 Specific comments

> Page 1 :
> line 5 : Please add Âń global Âż before Âń ocean model Âż as using a coarse resolution (1° x 1°), 15 vertical levels over a global ocean are needed to reach one millions points.

Agreed.

> Page 2 :
> lines 5-10 : The authors suggest that there are more possible errors in Fortran programming. However, this is related to the technical rigour of the developper/user and of the strict application of good practices and Fortran norms as it should for developping a Python code. Please consider rephrasing thoses lines.

I agree that we did not convey our point very well here. It is probably true that an expert Fortran programmer lets *fewer* bugs slip into production code than an expert Python programmer, if only due to the strict type checking provided by the Fortran compiler. However, there are several things to consider:

- People implementing new features such as a novel parameterization are, in our experience, often *not* expert Fortran programmers, but grad students or postdocs more

interested in Physics than software engineering. - While it is in principle possible to write clear Fortran code with meaningful abstractions that may be just as readable as a high-level implementation, the reality is often different. Popular ocean models such as MOM [4] or POP2 [5] feature subroutines that are hundreds to thousands of lines long, and both models rely on more obscure Fortran features such as 'COMMON' blocks, which makes it hard to keep track of variable scopes for inexperienced programmers. This is not necessarily due to flaws in Fortran's core design, but we do consider the established idiomatic style of a community to be tightly bound to the language used.

I have re-worded this section a bit.

> Page 4 :
> ligne 8 : Why don't use Python 3.x which is now mature and ready to use ?

The first prototype of Veros was written in Python 2.7 to work around some bugs in Bohrium at the time. Those issues have since been resolved, and we fully support both Python 2.7 and Python 3.x. I have removed all Python 2 references from the manuscript, as it was not really relevant.

> Page 6 :
> paragraph 2.3 : using good practice, Fortran code could also be elegant and easily readable... Keep in mind that Fortran means FORmula TRANslator ! Please consider to be more factual in your remarks

While Fortran might indeed be a good choice to translate formulae, this section specifically deals with the ecosystem around the numerical core of a simulation project: third-party library integration, dynamic switching between modules during run time, modern productivity and QA tools, modularity and object-oriented programming. Since Fortran 90 does not support classes, has very few users outside of academia, and is entirely

static, I think it is safe to say that Python allows for some elegant implementations that are infeasible or outright impossible in Fortran 90.

> Page 7 :
> line 11-16 : Would it mean that user should use the appropriate algebra library depending on the size of the problem ? Numpy, PETSc, CUSP ? Or Veros chooses automatically the best one like in section 2.3.4 ?

Whether to use NumPY, PETSc, or CUSP depends more on the available hardware and software architecture than the size of the problem, so it would make sense to pick the most appropriate one automatically as it is done with the tridiagonal solver. But since we don't know the performance characteristics of those libraries yet, we can only speculate at this point.

> line 17 : The authors refer to a following section. Could you consider to give more details to improve the readability ?

I made this section a bit more descriptive.

> lines 29-30 : Indeed, I/O management is a main issue in many codes for now and in the future. Give more details on this output strategy.

Done.

> Page 8 :
> Line 1-5 : Is there a loss in accuracy with compress and decompress processes ?

Both compression algorithms (‘zlib‘ and ‘gzip‘) are lossless.

> Page 9 :
> Line 15 : Âń certain tolerance Âż : Please, could you be more explicit on this point ?

I have expanded the section.

> Page 10 :
> Line 28 : Which variables do you consider for those relative errors ?

Thank you for catching this mistake; that information was indeed missing. It was the long-term average of barotropic stream function and zonally averaged temperature.

> Page 11 :
> Line 13 : It sounds that a main drawback is that Borhium an only be used in a single computational node. Have you any idea the schedule for parallelized implementation of Bohrium ?

“Automagical” distributed computing (in the sense that Bohrium automatically distributes computations between multiple compute nodes) is a hot, ongoing research problem. The developers of Bohrium have made some advances towards this [6], but a lot of work is still to be done.

However, it should be fairly straightforward to implement an abstraction to support distributed architectures at a user level, similar to the explicit style of MPI. In that case, we would have to take a step back in Veros and re-introduce some explicit parallelization logic, which is something we want to avoid as much as possible. If we can find a clean, straightforward way to support multi-node architectures without sacrificing too much flexibility, this should be achievable after a few months of work.

> Page 12 :
> Figure 1.
> - Bh CPU and Bh GPU curves can not be distinguished.

Architecture 1 does not include a GPU benchmark, so the Bh GPU curve is not present in this panel. While Bh CPU and Bh GPU performance are mostly identical on architecture 2, they differ for large problem sizes, which is the only intended take-away message of the GPU curve.

> - I do not clearly understand what does the Âń line fit Âż for the MPI curve ?

This annotation indicates that *all* solid lines are line fits (in the sense of ‘y = mx + b‘) to the available data to highlight that the performance characteristics scale as expected. I updated the figure caption to make this clearer.

3 Technical corrections

You will find most of your comments reflected in the changed manuscript.

> line 11 : it seems the character – in reference Jones et al is not necessary. (Idem page 20 line16)

By using this format, we are following the official recommendations on how to cite SciPy [7].

4 References

[1] Monteiro, Joy Merwin, and Rodrigo Caballero. "The climate modelling toolkit." Proceedings of the 15th Python in Science Conference. 2016.

[2] Marta-Almeida, Martinho, et al. "OOFε: a python engine for automating regional and coastal ocean forecasts." Environmental modelling software 26.5 (2011): 680-682.

[3] Eden, Carsten. "Closing the energy cycle in an ocean model." Ocean Modelling 101 (2016): 30-42.

[4] Pacanowski, Ronald C., K. Dixon, and Anthony Rosati. "The GFDL modular ocean model users guide." GFDL Ocean Group Tech. Rep 2 (1991): 142.

[5] Smith, Rick, and Peter Gent. "Reference manual for the parallel ocean program (POP)." Los Alamos unclassified report LA-UR-02-2484 (2002).

[6] Kristensen, Mads Ruben Burgdorff, and Brian Vinter. "Numerical python for scalable architectures." Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model. ACM, 2010.

[7] https://www.scipy.org/citing.html

Please also note the supplement to this comment:
https://www.geosci-model-dev-discuss.net/gmd-2018-3/gmd-2018-3-AC1-supplement.pdf

---

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