Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2019-336-AC1, 2020
© Author(s) 2020. This work is distributed under
the Creative Commons Attribution 4.0 License.

Geoscientific
Model Development
Discussions
Open Access

Interactive comment on "Simple algorithms to compute meridional overturning and barotropic streamfunction on unstructured meshes" *by* Dmitry Sidorenko et al.

Dmitry Sidorenko et al.

dmitry.sidorenko@awi.de

Received and published: 9 June 2020

[gmd, manuscript]copernicus

We thank Mark R. Petersen for reading our manuscript and for his constructive suggestions. Please find our answers below.

The Authors

This paper provides a clear explanation on how to compute the meridional overturning and barotropic streamfunction on unstructured meshes. It is well-written and

clearly communicates the motivation, the mesh structure, and mathematics of the computation, and the sensitivity of the parameters. I have not seen these details discussed in other publications, so it will be a useful reference to ocean modelers reproducing this calculation. In MPAS-Ocean, we compute the meridional overturning circulation (MOC) streamfunction by integrating the vertical velocity in latitude bins, level-wise, as described in this paper as Method A. I adapted this method from our previous model, POP, which is a structured quadrilateral mesh but usually ran on a tri-pole grid, so the generality of latitude binning was required for that model as well. We have the ability to run the MOC calculation as an in-situ calculation during the simulation: https://github.com/MPAS-Dev/MPAS-Model/blob/master/src/core_ocean/analysis_members/mpas_ocn_moc_streamfunction.F#L518 or as a post-processing calculation from the time-averaged velocity fields: https://github.com/MPAS-Dev/MPAS-Analysis/blob/develop/mpas_analysis/ocean/streamfunction_moc.py#L436
We describe the philosophy of the in-situ computation here: Woodring, J., Petersen, M., Schmeisser, A., Patchett, J., Ahrens, J., Hagen, H., In Situ Eddy Analysis in a High-Resolution Ocean Climate Model, IEEE Transactions on Visualization and Computer Graphics, vol.22, no.1, pp.857-866, Jan. 31 2016 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7192723 If you feel it is relevant, you are free to add any of that background to your paper.

We thank the reviewer for informing us about similar techniques being used in MPAS-Ocean and in POP. At the time of writing we were aware only of the cubed-sphere configuration of MITgcm using a similar strategy of transport computation (Adcroft et al., 2004). In this paper we aim at documenting these techniques for unstructured meshes in order to facilitate the use of unstructured-mesh models by a broader community. We corrected the introduction mentioning MPAS-Ocean and POP and also cite Woodring et al. 2016 on in situ computation.

We have found that the python post-processing method is fine at lower resolutions, but at high resolutions (1 million horizontal cells) it is extremely slow or fails, and we rely on our in-situ calculation. It would be good to hear in section 5 about any performance limitations in your calculation. On page 9 line 20, are these python routines just the plotting part, or do they reproduce the functionality of the Fortran and c code? If the later, did you run into performance bottlenecks, and how did you run python at high resolution – some combination of xarray and dask?

Diagnostic of MOC in z coordinate is fully done in python (method A, by integrating the vertical velocity in latitude bins, level-wise). The dask and xarray are used to read a 3D field if the mean over some time interval is required. The MOC calculation requires that the data that are located in memory. The reason is that we did not manage to make dask and xarray working efficiently with selecting "arbitrary" indices, which is required for computing vertical flux contribution from a cell and selecting cells that belong to a certain bin. Putting unused values to np.nan, when possible, instead of selecting by index, causes a good performance gain. One should have, of course, a sufficient amount of memory on the post processing machine. Our experience shows that 200G is enough to compute MOC for a mesh with 20M surface vertices. For a mesh with 1.3M surface vertices (2.6M surface cells) and 49 levels it takes about 7 seconds to compute a global MOC when using 91 latitudinal bins. For the largest mesh we have (23M vertices), and 80 levels it takes about 7 minutes. The python function can be found here: https://github.com/FESOM/pyfesom2/blob/7609c4ec54c13bcdbc1d15abc5da14bc40b9ec63/pyfesom2/diagnostics.py#L348
Computation of MOC in density coordinate can be also done in python in the same manner as methods A or B. For this, the meridional velocity (for method A) or horizontal divergence (for method B) need to be remapped conservatively onto density bins. This, however, can be done only in situ and results in 25% slow down of the code if 80 density classes are used. Method B can be then

used straightforward in python. For method A we need first diagnose the diapycnal velocity from horizontal divergence. This is also done in python using xarray and dask.
Computation of the barotropic streamfunction is currently also made offline and we confirm that it is slow. Hence we plan to implement the in situ computation of the barotropic streamfunction.
As suggested by the reviewer we included this discussion into the chapter 5 of the revised paper.

Your equations are missing numbers.

Fixed!

I find the notation of the streamfunction definition confusing (page 4, top). I prefer to see the $d\theta$ and dz at the very end, to indicate what is included in the integral. The first one, in particular, could be confused with the multiplication of two integrations, rather than one case of a double integral. Another way to clarify is to write w and v as: $w(x, \theta', z)$ and $v(x, \theta, z')$ and integrate in θ' on the first equation and z' on the second, to explicitly state that the primes are dummy integration variables, not to be confused with the upper bound and independent variable on the left side. For the new reader, it would be useful to review the definition of a streamfunction at the beginning of section 3. That is, if we have zonally-integrated velocities W and V, then for an incompressible fluid we can define a streamfunction Ψ as $d\Psi/d\theta = W(1)$ $d\Psi/dz = -V(2)$ and refer to some fluid dynamics textbook – my favorite is Kundu, https://www.amazon.com/Fluid-Mechanics-Pijush-K-Kundu/dp/0123821002 (I forget which line is negative, I always have to work through it. I assume one of your streamfunctions has a negative sign absorbed in a coordinate or variable definition, but that would be good to note as well). From my equations above, it is clear that one may compute Ψ by either integrating (1) in θ, or integrating (2) in z. This is identical an-

alytically, but leads to different implementations and numerical results, as you describe.

Thank you, we modified the equations according to reviewer's suggestion and revised the definition of the streamfunction in section 3.

Please state the boundary conditions for the two equations, top of page 4. For Method A and a global MOC it is easy – just set ψ to zero at the Southern boundary. But for regional MOCs, like the Atlantic, we use the second equation with v along a connected set of edges nearθ_0 (33S, say) and integrate from the bottom up. That is our southern boundary condition $\Psi(z, theta_0)$ to then integrate northward with the first equation. For Method B, the stream function is just zero at the top or bottom.

Done!

Although it is implied, it would be useful to state in each figure caption whether you are using Method A or B.

We modified the figure captions in the revised paper as suggested by the reviewer.

Grammatical comments, Please change MPAS-o to MPAS-Ocean throughout...

fixed! we have also corrected for all other grammar comments listed by the reviewer!