Geoscientific Model Development Discussions

Geosci. Model Dev. Discuss., referee comment RC6
https://doi.org/10.5194/gmd-2021-367-RC6, 2022

Comment on gmd-2021-367

Anonymous Referee #6

Referee comment on "Towards automatic finite-element methods for geodynamics via Firedrake" by D. Rhodri Davies et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2021-367-RC6, 2022

I agree with the comments RC4 (by Wolfgang Bangerth) and RC5 that the overall purpose of the paper needs to be clarified.

In general the authors should try to answer the following question: **What can we do with this code now that was impossible with other approaches before?**

I'll try to provide some suggestions for the authors in order to answer this questions and improve the manuscript to a point where I can recommend it for publication.

Major remarks:

The authors actually give some suggestions themselves in Section 8 of the paper which have not been followed up but would have made the manuscript
a clear candidate for publication. Furhtermore, I would suggest to restructure the paper such that we have a section presenting the benchmark implementations using UFL and a part discussing improvements over other existing packages and approaches.

Benchmark cases implemented using UFL: It would be a valuable contribution to provide a ready-to-use collection of benchmark cases (as presented in the paper) described in UFL to be used with multiple packages that can make use of UFL. An example here is **dolfin_dg** (https://bitbucket.org/nate-sime/dolfin_dg/), which provides UFL forms for discontinuous Galerkin discretizations for compressible flow and can be used by Fenics, Firedrake, or even DUNE-FEM. Right now it is not entirely clear whether the presented examples could be used with Fenics or TerraFERMA. It would also be in line with the much advertized separation of concerns.

Improvements over other existing approaches: In the second part of the paper I suggest to better highlight the strength of using the benchmark cases with Firedrake over other available packages. For example:

- Could we easily define our own preconditioner, for example, making use of UFL again? And if so, an example should be added.
- The presented finite element discretization (Q2-Q1 on Cartesian grids) is standard. How easy and feasible is it to use other (maybe more appropriate) discretiztions or other grid element types?
- Could we easily do a space-time discretization since UFL should make it actually very easy to write this down?
- As stated by the authors, "the automated approach underpinning Firedrake has the potential to revolutionize the use of adjoints and other inverse schemes in geodynamics". This would have been a clear major improvement over the existing state of the art. Could this at least be demonstrated for a simplified example? It would clearly help the community to have a starting point in this direction.
- Scalability: Would it be possible to also see at least one strong scaling result for one of the presented cases?

Limitations: What are the limitations of the presented approach? Existing approaches already cover a wide range of features.For example, could this code be used to replicate the results presented in "Large-scale adaptive mantle convection simulation" by Burstedde, Stadler et al. (https://doi.org/10.1093/GJI%2FGGS070) at the same grid resolution?

Besides that, the length of the paper in it's current form is ok, since for this journal the resulting published paper will be much shorter due to the two-column format used. I'm just wondering how the code representation would look like in that format. And then, addressing my comments from above will probably mean to shorten the current presentation of the test cases. In addition, I have a few minor remarks.

Minor remarks:

- Throughout the paper: Crank-Nicholson should be **Crank-Nicolson**.
- Line 418: The citation of Homolya et al. is in my opinion wrong here, because the cited paper in the end only shows compilation times of the TSFC but does not make any statement about the efficiency of the produced code.