Review of "*icepack*: a new glacier flow modeling package in Python, version 1.0" by D. Shapero, J. Badgeley, A. Hoffmann and I. Joughin

February 18, 2021

In this manuscript, the authors introduce a new land-ice modeling software package known as *icepack*. *icepack* is written in Python on top of the Firedrake library, which uses the domain-specific Unified Form Language (UFL) and provides a high-level symbolic description of the problem to facilitate the add of new physics and/or equations. The intended user base of *icepack* is the glaciological community, in particular, glaciologists who may not have an extensive background/training in computational science. The idea is to provide a code that would be easy to use/develop by this class of prospective users. Following a description of the code, some numerical examples are presented to demonstrate the method's capabilities and accuracy.

The manuscript in question is well-written and interesting to read. Addressing usability is noteworthy and something that not enough authors in glaciology/climate science address. I personally have some qualms with some of the philosophy described by the authors, namely I worry about folks who are not familiar with numerical methods developing an application code in which a lot of what is under the code is hidden from them in some sense. In an ideal world, one would have glaciologists working with computational scientists to help them pick the right solvers, discretizations, etc., for their problem. The authors are correct that some solver options are for optimizing performance, which is secondary to getting the code/model running; but there are also solver/algorithm choices that depend very much on the physics (e.g., CG is only valid for symmetric problems) - is *icepack* designed so as to prevent the naive user from inadvertently using the incorrect default setting for their problem? I will assume it is to the extent it can be, and that the authors' argument is that the default hidden settings are likely to do less damage than some arbitrary settings a user might put in his/her input file without knowing what they are doing. Also, I realize that the reality is that many glaciologists do not have strong ties to computational scientists, and still wish to make progress in numerical modeling of land-ice; therefore, I will not focus too much on much "philosophical" perspective described above.

Another qualm I have about the paper has to do with performance - I am skeptical whether *icepack* can really be performant if one tries to run it on continental scale problems, and it is not clear to me if the code is even parallel. I think the intention may be to use *icepack* as more of a sandbox for prototyping small problems (similar to FeniCS), in which case, this is not a huge deal.

Despite the above concerns, I like this paper and see it published in GMD. I like in particular the idea of describing all the equations using the variational principle/action functional and having everything else propagate from there - not enough people do this. I do, however, feel that there is a lot of missing information in various parts of the paper, which should be filled in in the revision prior to the paper being suitable for publication. Please see below my enumerated list of questions/comments to address in the revision.

1. The authors suggest that C++ makes it inherently difficult to add new physics/PDEs (e.g., on p. 10

and p. 20), which I somewhat disagree with. One advancement that can make a C++-based code easy to add to is Automatic Differentiation (AD) - with AD, one can effectively code the weak form of the residual within a C++ code and AD will handle the rest, making it very easy to add new physics. An example is the Albany/Land Ice model (previously known as Albany/FELIX) of Tezaur et al. (`https://doi.org/10.5194/gmd-8-1197-2015`). I think it would be worth mentioning that there have been efforts like Albany/Land Ice out there to make C++-based codes more accessible to users of varied backgrounds. I agree that even an "easy-to-use" C++ code will be more difficult and more intimidating than a Python code, so I am not trying to minimize the authors' efforts at all.

2. Can the authors comment on the overhead of the symbolic descriptions/manipulations done by their framework? This sounds potentially like it would be very expensive. How does the cost compare to automatic differentiation, for example? A broader question is: is computational performance/cost a concern for users of *icepack*, or is it intended to be a "sandbox" in which performance is secondary to being able to code up something "quick-and-dirty" for initial prototyping?

3. I was a little bit confused about the reference of the FO Stokes-based model in this paper as a "hybrid model". I see that it is hybrid in the sense that you have a different discretization in the horizontal and vertical direction, but there are also hybrid ice models that use different PDEs in different domains, e.g., the ISCAL model of Ahlkrona et al. (`https://doi.org/10.1016/j.quascirev.2016.01.032`). Is the term "hybrid model" a common name for the approach in Section 2.2.3? Perhaps it is and I am not aware of it.

4. Does the hybrid model described in the paper have the same applicability as say the First Order Stokes model? Can it be used for both Greenland and Antarctica at continental scales?

5. Section 2.2.1: in my opinion, the authors do not provide sufficient justification for the penalty term, equation (7). They describe this as something that is added to smooth over artifacts - this would be needed based on the discretization, which there is little discussion of. The authors should state what order finite elements they are using - I presume they are linear, and that this is why the stabilization is needed? Why is stabilization needed only for the SIA? I think these things should be made clear.

6. Section 2.2.2: there is some imprecision here in equation (13) - you have not defined anywhere that Γ is the boundary, and which boundary you are referring to. One can figure it out, but it is not precise. Ω is not defined either though one will assume invariably that this is an open bounded domain in 2D or 3D depending on which approximation one is looking at.

7. The boundary conditions are not discussed very rigorously systematically - the authors seem to sprinkle in some boundary conditions here and there. I think the boundary conditions need to be given for each of the models at the time the models are presented - boundary conditions are needed to complete the definition of each models.

8. Certain terms in the equations I do not believe are defined anywhere, for instance, in equation (10), there is no expression given for the strains $\dot{\epsilon}(u)$. This is one of the things I cam across that need to be made more precise.

9. Section 2.2.3: the authors comment that higher degree polynomials can be used in the vertical layer in the hybrid approach. What order is typically used?

10. Section 2.2.3: this might be a naive question, but does Glen's flow law come into the hybrid model? I was expecting to see it there, but maybe I'm missing something.

11. P. 10: the discussion here about substituting model components suggests it may be possible to use different models in different regions and couple them (a la the ISCAL method). Is this possible, or something that the authors are thinking to add to their model/code?

12. Section 2.4: This section is very incomplete. You need to give the enthalpy equation and given the Glen's law expression as well, since it is mentioned.

13. In my opinion, there is not enough discussion of the thickness equation (Section 2.1) and how it is discretized. In typical ice sheet models, this equation is used to change the ice extent - one meshes up a region of "potential" ice, and then uses the thickness to dynamically determine a mask for ice-covered regions. Do you do something like this in your model? It should be discussed for completeness. I think you maybe start to do this in Section 4.1, but it is very confusing and hard to make the connection.

14. What sort of meshes do you use in your model (in the horizontal dimension, for the hybrid one)? Structured/unstructured? Hex/tet (quad/tri)?

15. Section 3: It is not clear from the description what the inverse problem you are describing is for. Is it to obtain parameters in the model like the basal friction using observational data of e.g. surface velocity? There really needs to be more discussion here, and I personally would like to see a mathematical statement of a representative inverse problem you are solving. It would be worth citing the work Perego et al. on optimization-based inversion, if what you are doing is similar: `https://doi.org/10.1002/2014JF003181`. BTW, the basal friction has not been defined, yet it is discussed - it needs to be defined earlier, when talking about boundary conditions (which needs to be added).

16. Section 4.1: I find this section confusing. I assume you are talking about discretizing the thickness equation here - that should be made clear. I disagree with several statements in this section as well. "The simplest explicit timestepping schemes are unstable with CG finite elements" - if you are talking about CFL stability, this is not true. You need to satisfy a CFL condition which could give rise to very small time-steps but you can get the scheme to be stable. I'm also confused about the notion of SUPG as a time-stepping scheme - I think of SUPG as a finite element approach to deal with advection-dominated flow problems, for example, that does not have anything to do with time-stepping. Maybe you are referring to upwinding? In any case, I think SUPG has nothing to do with forward Euler, so the discussion about forward Euler requiring parameters is erroneous. Additionally, I don't understand the comment about implicit Euler smoothing out sharp discontinuities... I believe explicit and implicit Euler have effectively the same diffusion and dispersion properties, so there should not really be a difference between the schemes. Did the authors verify their time-stepper on a manufactured problem to ensure that it was implemented correctly?

17. Section 4.2: there is an approach discussed in Tezaur et al. (`https://doi.org/10.5194/gmd-8-1197-2015`) for dealing with bad initial conditions in a Newton solver that relies on homotopy continuation that would be worth citing. It is an alternate to the approach you describe that lets you get away with not doing a line search for Newton. By the way, it should be no surprise that Newton is not converging

without a line search - in general Newton is not guaranteed to converge from an arbitrary initial guess without the line search.

18. Section 4.3, lines 411-412: there are actually ways to construct weighted norms to deal with the issue of DOFs having different orders of magnitude for the purpose of convergence.

19. Section 4.4: Again, it is not clear to me what is your inverse problem. You need to state this explicitly so it is clear.

20. Section 4.5.1: you talk about problems defined on "extruded geometries" - do you ever use non-extruded geometries? It has been shown in various references that there can be numerical problems for land-ice solvers that do not use extruded geometries, e.g. Tezaur et al.

21. Section 4.5.2: I think this section needs to be made earlier, and other BCs need to be added to that discussion.

22. Section 4.6: The authors mention running their code on 1 core. Is the code parallel - can it be run on multiple cores? Are there any hope for performance portability of the code to take advantage of emerging HPC architectures, e.g., GPUs?

23. Section 4.6: Can you please clarify what you mean by the following statement? "Large problems, such as continental-scale modeling, will require more sophisticated and possibly problem-specific approaches". I'm wondering in particular about the problem-specific approach part. There are models like first order Stokes that can be used at the continental scale and they are not really problem specific.

24. p. 20: I don't understand why you need to create an analytical expression of (32) using special functions. Is this something specific to your framework, which requires expressions in a certain form for the symbolic representation?

25. It's great that you have executable documentation in something easy-to-use such as Jupyter notebooks! (no need to address this comment)

26. p. 29: are you considering putting in the first-order Stokes//Blatter Pattyn model into your code framework?

27. The methods do not discuss their code development/testing stance on *icepack*. How do users contribute to the code - through pull requests? Is there regression/performance testing? Continuous integration testing? These are all really important, especially if you have non-experts contributing to the code!

Minor comments:

- "UFL" is not defined in the abstract.
- p. 4, line 90: A is also called the "flow factor". I would mention here that it is usually a function of the temperature, which comes from a different equation.
- p. 5, line 125: should be "checking", not "check".

- p. 5, line 126: I think it should be “Bueler” not “Beuler”, if I’m thinking of the right person.
- p. 5, line 147: change “we verified the correctness of the ice shelf model” to “we verified the correctness of our implementation of the ice shelf model”.
- p. 10, line 270: change “we’ll” to “we will”.