Geoscientific Model Development Discussions

Geosci. Model Dev. Discuss., referee comment RC4
https://doi.org/10.5194/gmd-2021-367-RC4, 2022

Comment on gmd-2021-367

Wolfgang Bangerth (Referee)

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-RC4, 2022

I've read the paper about using Firedrake for mantle convection with great interest. It is clear that Firedrake is an excellent system for writing such simulators in quite a small amount of code, something I think is just really impressive!

That said, there are two big issues I would like to raise (and on which I will then let the editor make a decision):
* What is it actually that is new/what is the *message* of the paper?
* The paper is quite long.

Let me take these on in turn:

1/ It hasn't quite become clear to me what the *message* of the paper is or should be. Most of the paper is devoted to things that are quite standard for those who have used the finite element method, and to showing numerical results. But neither of that is new: We know how the finite element method works. And how we turn the weak form into a linear system, and one would expect that the authors have checked that their code produces correct answers -- the fact that it does could have been stated in a couple of sentences of the form "We have verified that our code produces correct answers for benchmarks X, Y, and Z, to within the errors that have previously been reported when using other codes (ref 1, ref 2, ...)". We don't need to see many pages of tables and figures to believe this; this is particularly true because these figures do not actually compare against other codes, and so all we can infer from them is that the code is correct, but not whether it is better or worse in any regard than existing codes.

So, then, one is left with the question what actually *is* the message of the paper, seeing that the authors do not actually use the space well to make their point. I *suspect* that the authors think that the message should be "Using Firedrake, we can write these codes in 200 lines that using other software systems takes 2000 or 20,000". If this were the message, I think it would make a fine paper, though then I would try to remove much of the extraneous stuff mentioned above (see also my point 2 below).

I would, however, like to point out that that by itself is not as impressive as it may sound like. Being able to do what others have been able to do for a long time is not actually particularly interesting, even if the new system can do it quicker and with less code. The question that *should* have been answered is what one can do with this system that others cannot do -- what does Firedrake make possible that would otherwise not have been possible (including because it would take too much work in other systems)? Unfortunately, the paper does not answer this question, and I'm not sure whether there really *is* a good answer. The fundamental issue is that systems like Firedrake are really good at *making simple things easy*; the paper illustrates this: all of the codes shown are very nice and concise, but they fundamentally all solve very simple problems that the community has been able to solve for a long time. But the other codes that are out there can *also* solve much more complicated things, with moving boundaries, much more complex material laws, adaptive meshes, adaptive time step choices, and particles. It is not a stretch to speculate that doing all of these things would not actually be much easier with Firedrake either -- the majority of the code in ASPECT, for example, is after all not in the description of the finite element shape functions, or the description of linear solvers, but in material models, postprocessors, time step control, dealing with how compositional fields affect this that and the other, etc. It wouldn't surprise me if that could be done with 5x less code in Firedrake than in C++, but that would still take several 10,000 lines of code. (I will add as a sidenote that the model of writing many small codes for each benchmark is also not sustainable in that it encourages the fragmentation mentioned early on in the paper where everyone has their own variation of the code for a specific problem; codes like ASPECT spend many thousands of lines of code on run-time parameter handling precisely so that we can have a single code base and only need to exchange input files.)

In other words, I would appreciate if the authors could sharpen their message: Being able to do what others have already done, and illustrating this with a long list of examples is just not very interesting in itself.

2/ Space: The paper is quite long, at 50 pages. That's because in many regards it reads like a *manual*: It is addressed to people who don't already have the finite element background to understand the basics

that underly the codes, and to people who need to be convinced with a long list of detailed examples that the codes produce the correct answer. But the audience for a manual is different from the audience of a research paper. The authors might want to think about who they are writing this paper for, and then think about every section and every figure, and re-evaluate whether that section or figure is necessary for that audience. For example, I thought that many of the figures carry no more meaning than one could equally express with the sentence "We have verified that our simulations converge to the same values as reported in X and Y (1997)". It would not greatly surprise me if the paper could be shrunk to 30 or fewer pages without actually losing any of the message.

Other comments -- only a few, the paper is actually really well written:

l. 206: This is a rather unusual formulation, with not integrating by parts the 'grad p' term, but integrating by parts the 'div u' term. The only comparable formulation I know of is what is done when comparing the primal and dual mixed formulations of the mixed Laplace equation -- there it makes sense based on whether one wants the velocity or the pressure to be in a space that requires derivatives. But for the Stokes equations, the velocity *always* needs derivatives because the viscous friction term already has derivatives on it. The usual approach is to get the derivative off the pressure and let it be in L2, but the authors here gratuitously require the pressure to be in H1. This seems unnecessary and at least requires explanation.

l. 277: Choosing C_ip=100 seems like a sledgehammer. It will for sure affect the condition number of the matrix. Is there not a smarter approach to choosing it.

l. 284: Choosing Crank-Nicolson isn't stupid, but it is also not particularly good choice. Why not use something a bit more accurate? Or, if you want to use something simple for expository reasons, at least say so. Separately, all codes use a fixed time step size; this too is not sufficient in practice.

l. 323: For more complex rheologies -- specifically if the viscosity also depends on the pressure -- the Newton matrix will have additions also in other blocks. But there are other complications one has to address for nonlinear rheologies as well; a straight up Newton scheme does not always work for strain-weakening rheologies as are common in geodynamics (see Fraters et al., GJI, 2019). In fact, this falls in the category of things that just require a decent amount of code: One has to choose a Newton matrix that is *not* just the derivative of the

residual, and that has to be programmed because the auto-differentiation of the residual isn't going to produce the matrix.

p. 13: For this and all other listings: Align comments vertically to make the code easier to read. If you can align a few '=' signs, then that's worth doing as well.

l. 497: "of of" -> "of"

l. 683, and the following paragraph about solver scaling: This does not actually sound true. It is simply a deficiency of the preconditioner the authors choose. There are plenty of papers that show that with good preconditioners, one can achieve almost perfect weak scaling, and in particular achieve a more or less constant number of linear iterations regardless of problem size. The statement as given is a cop-out :-)

Fig. 10: Please use a logarithmic x-axis to make these graphs easier to read.

Also Fig. 10: It would have been nice if there was a comparison to how long these sorts of computations take with other software systems. For example, how long would ten time steps with ASPECT take? This goes in the same direction as the "mission" of the paper I mentioned above. Just knowing *that* your code can solve a problem others have been able to solve for a long time is not so interesting. If you can do it *as fast as others* with 1/100th the lines of code they need, then that *is* interesting.

Section 7 (l. 720): It doesn't sound quite right to call this section "realistic applications". The application uses an incompressible mantle, a linear temperature and depth dependence for the viscosity, and has no special provisions to deal with the crust any different than the rest of the mantle. This is neither realistic, nor new. "Realistic" models have been far more complex than this for a very long time already.

l. 733: Are T=0 and T=1 mixed up here?