Review of **Implementing nonlinear viscoplasticity in ASPECT: benchmarking and applications to 3D subduction modeling** by Glerum, Thieulot, Fraters, Blom and Spakman

Boris Kaus, Mainz

This is a well-written manuscript in which the authors document their efforts to incorporated plasticity in the open-source ASPECT code, and demonstrate with a number of benchmarks, that the code gives similar results to that of other codes in our community. It also includes a new science application on 3D subduction with an adjacent plate and a few demonstrations of subduction models with more complex (nonlinear and compressible) rheologies. The paper is likely to be useful for others in the community and deserves to be published in SE.
Yet, I do have a few of minor to moderate issues, which I believe are important to address or at least discuss in more details before publication, as it will clarify the paper. All these can likely be incorporated without too much additional effort.

1. Plasticity implementation
The manuscript mainly deals with several benchmarks that demonstrate that the plasticity implementation gives similar results as that of other geodynamic codes. Yet, the implementation of plasticity itself is not described in very much detail, which is why the title of the paper is somewhat misleading (maybe drop “implementing” from the title. The plasticity implementation itself is also rather primitive (using only Picard iterations, for example), and some points deserve more discussion:
If you use a viscoplastic rheology, for example, you need an initial guess, which is usually done by taking a viscous-only step. You seem to use a user-specifiable constant viscosity for this, which can change from field to field. This can work (if you tune parameters just about right), but in my experience, viscoplastic models are extremely sensitive to this choice; if you use a too large value of viscosity, it won’t converge to a physically sensitive solution because the initial guess for pressure (which is approximately equal to 4*EII*eta where EII is the strain rate and eta the viscosity) may be massively larger than what is physically meaningful. A background strain rate of 1e-15 1/s and initial viscosity of 1e24 Pas, for example, would result in a dynamic pressure of 4 GPa. Yet, in the setup of Figure 4, the lithostatic pressure at the base of the model should be on the order of 270 MPa, and for a frictional material (with friction angle 30 degree) the maximum pressure should be no more than about twice lithostatic (see Petrini & Podlachikov, 2001, among others). Thus, the dynamic pressure in this case is significantly larger than the physically admissible pressure, which can cause problems with convergence of the viscoplastic solution. So even while running the same model setup with ASPECT, different users could end up with totally different results simply because they had a different choice of initial viscosities. Since you implement this in a community code, it is important that you take care that users with less background in computational geodynamics don’t produce physically meaningless results (I realize that this cannot be fully excluded, but you can at least try to minimize the chance for this to happen).

One way to do this different is to incrementally increase the boundary velocity (as described in Kaus 2010), during the first timestep. In practice, I found this is non-trivial to implement for more complicated model setups; moreover, it requires a large amount of initial iterations. A technique that is easier and more general (and which I have used since), is to

compute the admissible upper and lower bounds of pressure (that are usually a function of the lithostatic pressure), under the assumption of a homogeneous setup for a frictional material under compression & extension. The derivation of the admissible pressures is given in (Petrini & Podladchikov 2000) for a case with zero cohesion - you can easily extend that to a case with cohesion. During the first iteration step, or the first timestep, a pressure cutoff is applied within the yield function routine, which will ultimately limit the viscosities that the plasticity algorithm gives to reasonable. During subsequent iterations and subsequent timestep, this pressure cutoff is no longer applied. An advantage of this method is that it takes away 'user-tuning' of the initial viscosities. It is implemented in both MVEP2 and LaMEM (both available as open-source on bitbucket).
If you don't want to implement this in ASPECT, I would at least appreciate a longer discussion on the choice of reference viscosities and/or background strain rate and how new (or less experienced) users can detect non-sense results.

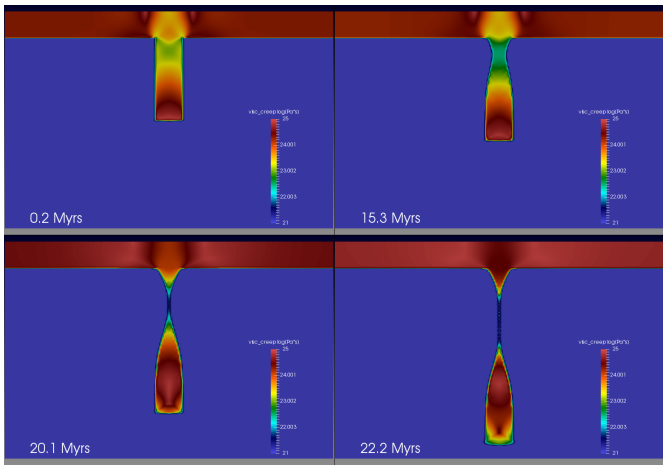
2. Appendix A/B: Subduction benchmark:
The benchmark setup you discuss has a 90 degree 'notch' and is the one which gave the worst results in the Schmeling benchmark. This may well be related to the 90 degree initial subduction angle which is very far removed from the angle the slab wants to make. A much better setup is case 3 considered in that paper, for which we also have laboratory experiments and for which the various codes had better agreement. It would be very interesting to see the effect of the viscosity averaging methods for this setup as well (I expect the discrepancies between the models to be much less severe). Can you add that?

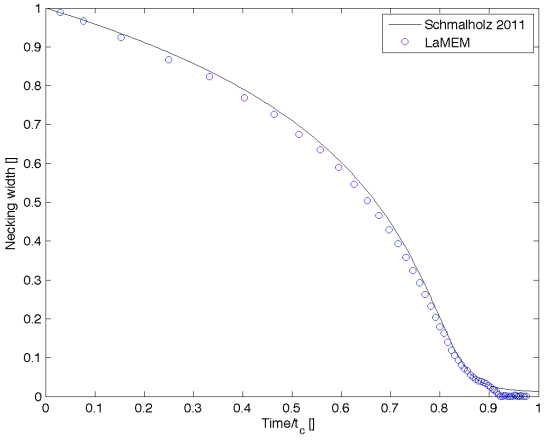
3. 3D Viscoplastic models
Model setups 1 and 2 show that ASPECT can handle more complicated setups. Yet, from a science point of view, the initial geometry of the two cases is so different that it is difficult to discern what the effect of the adjacent plate is. You cite the paper of Schellart and Moresi (2013) in stating that the adjacent plate does not affect the geometry of the subducting plates. Yet on page 23 you also state that your rheology differs from their model. To make your conclusions more robust, it would be good if you can add at least one additional simulation with an adjacent plate for say the simple model setup (model 1). This would support your conclusions that the differences between models 1 and 2 are mainly caused by rheology and not by the adjacent plate.
Along similar lines, you mention that the plate viscosity of model 2 is an order of magnitude larger than that in model 1. From Fig. 19, it seems that the asthenosphere also has a lower viscosity, such that the overall slab-mantle viscosity contrast increases. I agree with you that a systematic study of these differences is probably beyond the current paper (even more since each 3D simulation takes several weeks). Yet, what would be interesting is to better understand whether the same effects are observed with 2D simulations as well, using exactly same setup but without adjacent plates. That should be computationally much faster and will at least give the reader some insights into how important 3D effects are for these kinds of subduction scenarios. It would be great

4. Required wall-clock time and Picard vs. Newton iterations
I really appreciate the honesty of the authors by reporting actual wall-clock times of the simulations, which gives interested readers a feeling for the computational costs involved in running ASPECT simulations. To be honest, the results left me a bit shocked. If a 3D free subduction simulation takes up to 6 weeks on 260 processors, with a maximum equivalent resolution of 640x128x128 elements, it essentially implies that it is nearly impossible to perform systematic science with this code (think about the carbon footprint

that this simulation has...). It is ofcourse difficult to make a one to one comparison to other codes (also since you did not report the number of timesteps in the tables - would be great if you can add this information). Yet, I'm routinely running 3D models with about the same maximum resolution (but without AMR) with LaMEM in a day or so on 64-128 cores. This thus at least suggests that there are massive differences between different codes and in terms of the science per CPU-hour such difference matter.

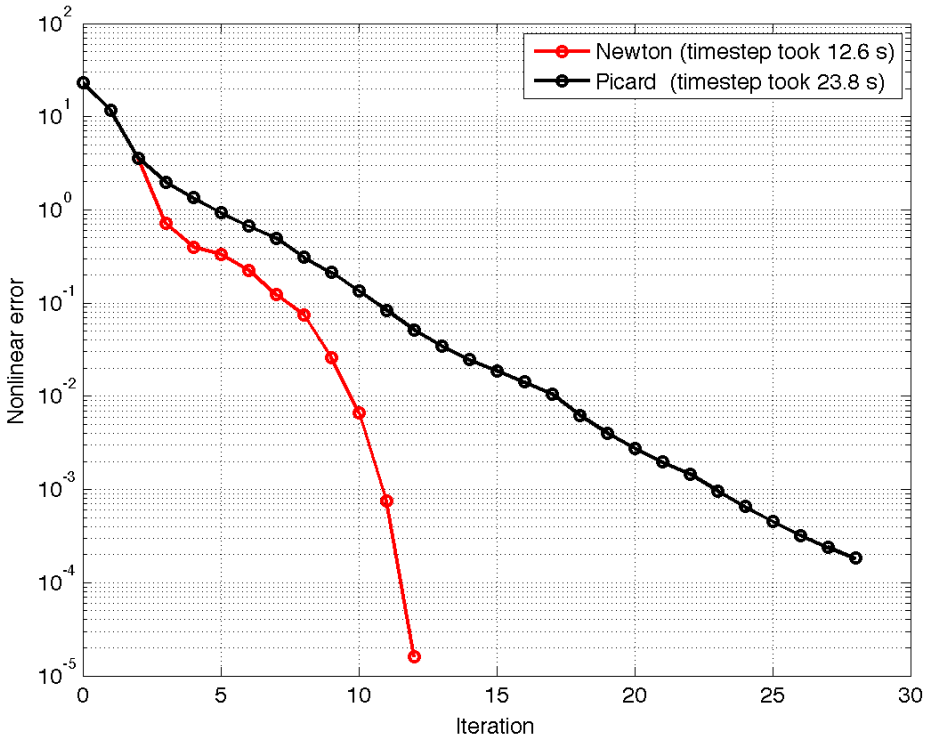
To get a somewhat better estimate of how timings differ and what the importance of Newton iterations is in this, I redid the 2D slab detachment benchmark (section 3.4) with LaMEM using your largest resolution (256x256 elements) on 8 cores. As LaMEM is currently a 3D-only code, the LaMEM simulation was done with 2 elements in the y-direction and employed 3 rather than 2 velocity components per node (such that the total degrees of freedom of the 256x256x2 simulations were with 590'848 slightly larger than the 456'400 DOFs used in the ASPECT simulation). In the LaMEM simulation, I reconstruct the slab thickness from the interpolated phase proportions on the staggered grid. This is likely to be slightly less precise than the marker-line approach used in your manuscript. Nevertheless, results are quite comparable to the Schmalholz solution:





Rather than taking 28 cores and 16 hours to compute the full simulation, the 295 timesteps of the LaMEM simulation were computed on 8 cores and took about 62 minutes. LaMEM thus seems to require (28*16*60)/(8*62)~55 times less CPU-time than ASPECT (assuming ASPECT employed a similar number of timesteps).

Part of this discrepancy may be caused by LaMEM employing a Newton nonlinear solver, rather than a Picard iterations. To understand how much this accounts for, I reran the simulation with Picard-only iterations and show the convergence behavior of timestep 10:



This demonstrates that Newton iterations are (in this case and for this timestep) around a factor 2 faster (note that we start every timestep with picard iterations before switching to the Newton solver). Evidently, for a tighter tolerance criteria the discrepancy between the two method is larger whereas it is less for a more relaxed nonlinear solver tolerance. Overall, the full simulation with Picard-only took around 153 minutes. So, whereas Newton can explain part of the discrepancy between the required wall-clock time for a full simulation, significant differences remain.

The input files for this setup, together with plotting routines, analysis tools and logfiles of the two simulations described above, are uploaded to the LaMEM repository under /input_models/DetachmentBenchmark.

It would be interesting to see how future optimizations of ASPECT (and of LaMEM) will reduce these timings and how this is in other geodynamic codes. The time-to-do-science is an important factor as well in computational geodynamics, that is unfortunately rarely documented for realistic cases (an exception being Pourhiet et al. 2017 for a different plasticity setup). I thus appreciate reporting these numbers – it would be great if you can report on how the latest ASPECT release affected the timings.

Minor points:

p2. l14. It's LaMEM and the reference is wrong (should be Kaus et al., 2016)
p3. l23. Please clarify whether you employed tracers here or not
p4. l6/7: Please clarify whether these are the PETSc SNES options.
p5. l7: Do you also use a zero initial guess for pressure or a lithostatic value?
p5. eq(9): As far as I am aware most geodynamic codes employ the same yield-criteria in 2D as in 3D (so eq.8). That has the advantage that if you do pseudo-2D calculations with the 3D code (using say 1 element in the 3rd dimension) you

retrieve the 2D formulation. In your case, for typical values of the friction angle (30 degree), the 3D formulation deviates a few percent from the 2D one.

p10. l2: Different than in Kaus (2010), you don't apply strain weakening in your setup. You do mention that later, but a comment at this stage would clarify things already. I would also appreciate a brief discussion on the choice of \mu_{init} on the model results.

p12: You performed these simulations without AMR. What is the effect of using AMR on the shear band angles (if any)?

p17: Fig 13: Did you mix the labelling of the x and y-axes of the figure?

p17. l 4: The "first" three benchmarks (as the detachment benchmark is not plastic)

p24, l21: You mention a benchmark of ASPECT that employs a different viscoplastic formulation. Can you explain better what the difference is? Do they not use a similar yield stress formulation and plastic viscosity?

p25, l4/5: The first one to show the effect of a nonzero dilation angle in the geodynamics community was, as far as I am aware, the paper of Gerya & Yuen (2007) - see their figure 7.

p25, l14/15: In my experience, adding elasticity significantly improves the convergence behavior of simulations with plastic failure (even though it does not solve all issues), and because of that it is worthwhile to incorporate. You are welcome to try MVEP2 or LaMEM to verify this.

p25, l16/17: Newton iterations are crucial for fast convergence - you can add LaMEM and TerraFERMA to the list here. Yet, a pure viscoplastic rheology remains difficult to impossible to converge (as explained by Spiegelman et al., 2016).

p26, sect. 6: Can you attach all scripts used to generate the benchmarks and figures to this paper, together with detailed instructions in the exact version of ASPECT you used to create the models? It seems likely that future code changes may give slighty different results; this way the interested reader has a reference point to reproduce your results.

p27, l7: Why is the infinite norm computationally more expensive? Is that because you effectively end up with larger jumps in viscosity between adjacent elements, and you use iterative rather than direct solvers?

p39, Table8: In model 1, I am a bit puzzled about the relationship between the B-parameter and the initial viscosity. These models are linearly viscous (apart from the crust), so why is \mu_{init} not simply 1/(2*B) as suggested by eq. 6?

Additional references:

Petrini, K. & Podladchikov, Y., 2000. Lithospheric pressure-depth relationship in compressive regions of thickned crust. *Journal of Metamorphic Geology*, 18(1), pp.67–78.

Le Pourhiet L, May DA, Huille L, et al (2017) A genetic link between transform and hyper-extended margins. EPSL 465:184–192. doi: 10.1016/j.epsl.2017.02.043