Solid Earth Discuss., 6, C969–C975, 2014
www.solid-earth-discuss.net/6/C969/2014/

Solid Earth
Discussions
Open Access

Interactive comment on "ELEFANT: a user-friendly multipurpose geodynamics code" *by* C. Thieulot

B.J.P. Kaus (Referee)

kaus@uni-mainz.de

Received and published: 18 September 2014

Review of Thieulot

Remarks to editor (not shown to authors) ==========

Comments to author(s) ==========

A new code, ELEFANT, is described that can be used to model lithospheric scale problems and the code is tested versus a large number of existing benchmarks. Whereas the manuscript is well written, it is very long with an large number of figures.

At the same time, the code itself uses standard techniques that has been in employed in the geodynamic community for at least two decades (the code SOPALE by Fullsack is based on exactly the same principle; SULEC, MILAMIN_VEP, GALE and many others are very similar), and that are known to be problematic. Moreover, the author itself has

already published a paper on a code based on the same principles (but with a different name).

As such this manuscript reads more like a user guide than a scientific paper, even though it is nice to have a collection of benchmarks in one place (more about that below). Whereas, I am generally in favor of publishing technical papers on topics related to computational geodynamics, I do think that one should have something new to tell in order to publish a scientific paper. Rewriting your code in a different computer language is a lot of work but if you do the same as what others did already, what is the point of publishing it again? If the whole community would start to do this, we would be flooded with similar papers but make little overall progress. To be fair, you do include a few things that are new, for example a complex salt-tectonics simulation, a generalization of averaging methods, and the computation of lithostatic pressure using a FE method.

I will leave it up to the editor to decide whether it is worthwhile publishing or not. In the case he believes it is fine, I have a number of issues that I believe would help to improve the paper if you can address them. I have added many questions in an uploaded annotated version of your MS, and list some of the major ones here. It would be great if you can address those.

1) It would be good to be more upfront about the well-known shortcomings of this kind of code, such that future generations of students can spend their time on solving those issues rather than on repeating them. In particular:

- Q1P0 elements work ok if combined with direct solvers, but since they are not LBB stable they do not work well (or nearly don't work at all) if combined with iterative solvers. The reason is that unstable elements require a larger number of iterations the larger the resolution is. A fix is to use stable elements (e.g. Q2Pdisc), which are however computationally significantly more expensive .

- As you mention in the text 100^3 is about the maximum resolution you can do in

3D with direct solvers, due to memory restrictions. Even if you use a computer with 10'000 processors you won't be able to go beyond this. Whereas I agree that there are a lot of interesting problems that can be solved with 100^3, many other problems will require higher resolutions. The only way to achieve that is to use multigrid solvers. That requires quite a different solution strategy (papers by May and Moresi, 2008, among others, discusses this for decoupled solution methods, and for example Taras Gerya has had some success with a fully coupled MG scheme in his code I3ELVIS)

- The penalty based method you use does not work or works poorly in combination with iterative solvers as the number of iterations depend on the condition number. Rewriting the code to handle these method is a non-trivial task as it requires the matrixes to be present in block-format.

- MPI-parallelization is in my experience difficult to do in hindsight and should be done from the start. OpenMP parallelization can be done reasonably easily but limits your code to32 or 64 cores at most.

- Picard iterations is indeed what the majority of the geodynamics community uses, but as you point out, in some cases plastic localization requires a massive amount of iterations. The computational engineering community have proposed workarounds for this nearly 30 years ago by using Newton-like methods. Yet, in our community I am only aware of two codes that use this (SLIM3D by Anton Popov) and pTatin by Dave May and coworkers. The potential payoffs of using this are enormous, but more work is required to make this work in combination with marker-and-cell approaches.

You can include an additional section in your paper describing this.

2) Benchmarks: Most benchmarks you show are fine, but the drawback is that they are for relatively simple systems and almost never test the full viscoplastic thermomechanical code (with plasticity, free surface etc. etc.). That is a potential shortcoming of most benchmark studies in geodynamics. You do show a nice example of salt tectonics coupled to lithosphere deformation, but that is a new result and was not done by others

before. It would be nice if you can go one step further than existing benchmarks by reproducing a complex setup (with thermo-mechanical interactions, plasticity, surface processes etc. etc.). A good starting point for that would be the textbook of Taras Gerya; you could, for example, reproduce his collision and slab break off example. As the full code is distributed together with the textbook you can use exactly the same input geometry and material parameters. A comparison of your and Taras's results would be pretty convincing and go well beyond existing benchmarks.

3) Solver performance: - You give a few examples on how your solver scale w.r.t. number of processors (strong scaling). Yet what is important for many users is how much time and memory is required for a given model resolution (or in other words: what can I run on my workstation?). You mention in the conclusion section that it is prohibitive to go beyond 100^3 resolutions (I agree with that), but you don't give scaling results on time and memory required for a given number of nodes. It would be great if you can add this (for both 2D and 3D as scaling differs in the two cases).

- It would be interesting if you can also show weak scaling results (keep resolution per core the same, but increase number of cores) for both the MUMPS and the iterative solver.

- You state that you also have an algebraic multigrid solver implemented. This solver should ideally have a nearly linear solution time for larger resolutions. Is that what you observe? Would be great to show this.

Minor things:

Introduction, p. 1952 around line 10: For the sake of completion you might want to mention the finite-difference/spectral method as well at this stage. In Kaus and Podladchikov (2001), GRL, we were able to do 513x128x128 resolution simulations (on a single CPU) which still beats many 3D simulations performed nowadays (but with the restriction that it only works for systems that are isoviscous or have a depth-dependent viscosity) Schmalholz et al. (GJI, 2001) showed how to expand this technique to vari-

able viscosity cases and Kaus et al. (2004) and Kaus (2005, PhD thesis) showed how to do this for viscoelastoplastic materials (with Mohr Coulomb plasticity). It remains an interesting and memory-efficient method with the main disadvantage that it requires periodic boundary conditions.

equation section 2. - there are a number of inconsistencies with stress definition (see annotated MS) - Adiabatic heating should potentially be included if doing convection studies

Section 3.9: Please explain how you find the local coordinates of your markers in a deformed mesh (Moresi 2003 had a discussion about it).

Figure 4: It is unclear whether figure 4 shows the analytical (true) solution or the numerically integrated one with your code. It would be nice if you should show a comparison of how good/bad the various schemes are working for this test.

Section 3.9: It is unclear whether you do a Runga-Kutta metjhod in SPACE (requiring you to solve the Stokes system of equations once, but interpolate velocity 4 times from mesh to particles for the 4th order RK method), or whether you also perform a Runga-Kutta in time (which would require you to solve the system of equations 4 times per timestep for the RK4 method). If you do the first method, it's pretty similar to what many do but it is only first order accurate in time (in a recent paper by Keller et al. GJI 2013 it was found to be insufficiently accurate for a two-phase system). Please clarify what you do.

Section 3.10: How do you determine the history properties and the phase that injected particles receive? Using the closest particle, or the average of the surrounding particles?

Section 3.13: You don't describe whether you iterate for plasticity on the markers (and project it back to the mesh after every iteration step), or on the integration points. In MILAMIN_VEP I did it on the integration points, but others do everything on markers

and interpolate the resulting effective viscosity back to integration points. It can make quite a bit of difference, and it is unclear which one is better or why. It would therefore be nice if you describe what you do. Related questions: - In the case of Drucker-Prager with no zero friction angle, do you use the smoothened pressure to compute the yield stress or the direct elemental pressure (unsmoothened)? - If you start a typical viscoplastic simulation (such as shown on figure 10), you are typically massively above the yield stress during the first iterations. How do you deal with that and ensure convergence?

Section 4.1.1: please give more details on stopping criteria used for the nonlinear iterations and explain whether gravity is activated or not.

Section 4.1.3: Do you keep the grid fixed in all modelled cases and show the results after 1000 iterations? That is slightly different than the Lemiale and Kaus papers where timestepping was performed. Can you explain the convergence numbers in each case after 1000 iterations (I suppose that they are not converged and that they are different depending on resolution).

Equation 52: Which strainrate component is used here - second invariant or individual components?

Section 4.5: you use the most difficult benchmark of the Schmeling et al. (2008) paper - If you do case 3 instead of that paper, results converge much better.

Appendix C: How do you input a geometrically complex 3D setup to the code?

References: Kaus, B., 2010. Factors that control the angle of shear bands in geodynamic numerical models of brittle deformation. Tectonophysics 484, 36–47. doi:10.1016/j.tecto.2009.08.042 instead of Kaus (2009) as referred to in the text.

Boris Kaus, Sep.18 2014

Please also note the supplement to this comment:

http://www.solid-earth-discuss.net/6/C969/2014/sed-6-C969-2014-supplement.pdf

Interactive comment on Solid Earth Discuss., 6, 1949, 2014.