se-20200-79-RC2
Response to Reviewer Comments

Patrick Sanan, for all authors

We thank the reviewer, Mikito Furuichi, very much for detailed and thoughtful comments. We reproduce these comments here and provide our own comments and responses. Line numbers in our responses refer to the revised manuscript.

General comments:

This paper presents the benchmark experiment with the direct and iterative solvers for the Stokes flow and elastic problems targeted by the solid earth simulation. The authors especially focus on the ILDL factorization which is not yet commonly used in the numerical solid earth community. Their performance test showed the tradeoff relations among the robustness, time to solution, and memory cost. This paper is well organized and presented results may motivate the computational geoscientist to utilize ILDL in their own geodynamics and seismic applications. Thus, this paper essentially fits the scope of the method paper of Solid Earth (SE). On the other hand, there is some room for improvement in presentation and experimental design. The author claims that the robustness of ILDL is the advantage over the iterative ABF solver, but supporting experimental data is found only in the extreme case which solves 10 inclusions of 10^6 viscosity contrast within 32^3 elements simulation. In other cases, iterative ABF solver shows better results in time-to-solution, memory usage, parallel performance. On the other hand, ILDL shows practical advantages against direct solver in memory cost. Thus, in conclusion, the ILDL solver is found to be the potentially good alternative of direct solver rather than an iterative solver. So, the expected reader would be the user of direct solver. However, their performance analysis is presented mainly for ILDL vs iterative ABF solver rather than vs direct solver, especially in the parallel performance section. I encourage the author to continue this work, but the presentation should be improved and more detailed performance analysis should be addressed before I recommend this for publication in SE.

We argue that these “extreme” cases are in fact those of greatest interest to many practitioners, in particular in geodynamics. It’s become fairly common practice to abstract and characterize some of the difficulties of challenging heterogeneous coefficient structures with multiple-sinker problems, as they conveniently offer two “knobs” for contrast and geometrical complexity (number of sinkers). Each parameter tends to stress solvers in different ways.

We have attempted to stress in the paper (e.g. lines 257, 321, and 396) that in addition to the lower memory footprint, a perhaps even greater advantage of tools like ILDL preconditioning is their relative ease and robustness of use compared to an ABF solver.

The ABF solver presented here took years to tune, for experts. While they do indeed provide “optimal” solvers in many of the ways optimality is defined, they certainly have not shown themselves to be optimal in terms of solvers that most practitioners can understand and implement enough to incorporate. As stated above, the readers with the most to gain from ILDL preconditioning are those who rely on direct solvers, but who run into a memory limitation. Whereas before, the only good option was a risky exploration of solvers like ABF (or monolithic multigrid solvers), we aim to demonstrate the availability and performance of a far more incremental change to the solver stack.

It would have been tempting to not include the ABF solves at all - again, we agree that the most likely user of ILDL preconditioners would be those for whom this solver is practically out of reach, and we agree that the closer comparison is with direct solvers. However, we feel that it is important to clearly show what is attainable with a more invasive and complex implementation, particularly as we also believe strongly in the value of composable and hierarchical solver frameworks (such as the PETSc-based one used here) which offer a way forward to making such solvers easier to use, though there is clearly a long way to go.

The parallel section certainly leaves many things to future work. Our main objective, in analogy to existing work (and practice) which uses ILU subdomain solves within a block Jacobi or additive Schwarz preconditioner, is to simply demonstrate that a similar approach is available (with many of the same drawbacks) for indefinite problems, and the within a composable software environment, this is in many ways easier than one might expect to experiment with. Also see the response to point 17, below.

Detail comments:

1. In introduction: Several sentences sound your opinion rather than the objective view (e.g. “This is unfortunate” in line 102). Such phrases are not appropriate for the research paper.

We agree that we veered too far into this territory, and have thus edited the introduction to remove several opinion-based statements, and have made others more objective.

2. In introduction: Please more review the progress and difficulty in direct solvers, although the author mainly reviews the recent progress of iterative solver.

While there is certainly still research going on, the field of direct solvers could be considered to be much more mature, and is extensive. As such we’ve added a comprehensive review article, Davis et al. 2016, for the interested reader. We also note that the references specifically on incomplete LDL^T factorization can guide the interested reader to that subset of the direct solver literature which is most relevant (methods using complete LDL^T factorizations).

3. In line 45: The hieratical grid system with such as AMR [Rudi et.al. 2015] worked well as the solution of highly variable viscosity problem with controlling the coefficient.

We note that the referenced work uses *smoothed* viscous inclusions in their benchmarks, so while addressing the problem of large viscosity contrasts, does not involve actual discontinuities in the coefficient field, so that coefficient always varies sharply across a single finite element - this is currently mentioned in the paper on line 277.

Nevertheless, AMR-based methods show great promise. However, on a practical level, AMR methods are very challenging to implement, perhaps even more so than ABF methods. In contrast, we believe that the approaches highlighted in this work offer a tool which promises to be more useful to the large class of users who may want to move beyond using a direct solver, but only have the time or resources to experiment with something like ILDL preconditioners, which correspond to the ILU preconditioners which practitioners have been successfully using, albeit only for the positive-definite problems for which they are designed, for many years.

4. In line 95: Since expected readers of this journal are not specialists in linear algebra, a more comprehensive review is needed. For example, how much memory was saved against direct solver with increasing/decreasing the time-to-solution in the past successful application?

While this paper attempts to give some detailed information on the algorithms being used, we ultimately hope that the experiments (and provided source code) will be intelligible to readers without much linear algebra background, in the comparisons they offer to direct solvers, especially.

Some of the included references contain some information (Hagemann and Schenk 2006, for example), but a key part of the contribution of this paper is to address a general lack of this sort of information in the literature. Indeed, there aren't as many past successful applications as we believe there should be, and this paper is one step to try to expose these solvers. We perform a study of the relevant tradeoffs in the context of a set of physical applications, as opposed to in the context of a set of fixed-sized matrices representing a wide array of applications, which are more typical in the published work on these solvers.

5. In line 211: Delete the space after "("

This sentence has been reworded in response to comments from another reviewer.

6. In line 253: The spectral analysis for scaled pressure mass matrix can cite [1]

This citation has been added.

7. In Numerical experiment: I think that the experiment starts with x=0. This problem setting is suitable for steady-state solution. But in practice, we solve the timestepping/nonlinear problems. Thus, it is interesting if ILDL largely outperforms the direct solver from the second step. The solution of the previous step will be a good initial gauss for reducing the iteration of ABF and ILDL.

A simple motivation for using zero initial guesses is that this represents a uniformly-available and in some sense "worst-case" setup, which would be interpretable in the context of the most readers' problems.

In the case of a nonlinear solve, one is typically computing an update step (e.g. in Newton's method). There is no obviously generally useful initial guess for the linear solves; for example, the previous step would not be useful, as the solver has just updated the approximate solution with what it considers to be a good multiple of this search direction.

For nonlinear problems solved at each timestep, it is common practice to use a previous solution as an initial condition, but this is an orthogonal issue to the one of initializing solutions to the linear solves which may be performed within the nonlinear solver.

In the case of providing starting guesses for linear, time-dependent problems, an initial guess might provide faster convergence if one were relying on an absolute convergence tolerance. However, this approach is sensitive to scaling of the equations and as such the definition of an absolute tolerance is problem-dependent and must be carefully chosen, if used at all. The results in this paper are intended to be robust to system scaling (and as such we do not emphasize scalings or units - for more see the response to Reviewer 1, which is another motivation for not considering absolute convergence tolerances here.

8. In line 264: In practice, direct solver is mainly used in 2D problems. Also, in memory capacity, the difference in maximum element size in 3D (40ˆ3 for PARADISO < 48ˆ3 for ILDL) seems to be trivial but that in 2D (252ˆ2 < 332ˆ2) is significant in scientific application. Then, the experiment in 2D should worth considering in SE.

We very much agree that 2D problems with large memory footprints are relevant in practice. We performed these experiments (and note that they can also be performed using our included, open source code), but did not believe that they would be as interesting to the reader as the 3D problems which we've chosen to devote manuscript space to. This was not unexpected, as direct solvers exhibit better scaling for 2D problems than 3D ones, in particular with regards to memory footprint. This implies that far fewer readers would benefit from a deeper look at the 2D case, as there are fewer gains to be made over a direct solver. Those readers who have run into the limits of direct solver performance for 2D problems would likely be better served by investigating the more complex alternatives: ABF solvers, monolithic multigrid solvers, and emerging nonlinear solvers based on pseudotransient continuation.

Also see Figure 1 in our response to Reviewer 1, for a 2D experiment which we removed from an earlier draft.

9. In line 317: It is confusing that ABF does not fail to converge in Figure 1. Why not plot the case with contrast = 10ˆ6 with 8 inclusions?

This has been clarified in the text (line 325, and see also our response to a similar concern from Reviewer 1).

The plot resulting from changing the viscosity contrast 10^4 to 10^6 would look very similar.

10. In line 318: Do we really need to solve the problem with over 10 inclusions in 32ˆ3? The accuracy of such a setting seems to be a useless solution in physics. In addition, to check the robustness, SINKER box test of [May and Moresi, 2008] is better than this setting.

It is true that at these grid resolutions, the inclusions may well be under-resolved, and if one were interested in physically-relevant flow fields for these problems, finer grids should be used. We note that in applications, particularly those like geodynamics wherein coefficient structure emerges and changes with time, solvers must be able to handle under-resolved cases gracefully.

We use the multiple-sinker problem not because of its direct physical relevance, but because it presents a very useful abstraction of difficult coefficient structures that can appear in practice.

The advantages of this benchmark over something like SINKER are

- Distribution of interface alignment is uniformly distributed, so effects of grid alignment are less of a concern
- By changing the number of sinkers, the geometric complexity of the coefficient structure is quantifiable and adjustable.

On this second point, it can be observed that the spectrum of the operator, on which the convergence of the solver depends critically (See Elman, Silvester, Wathen 2005, Chapter 5) has a wider range of eigenvalues (worse conditioning) when the viscosity contrast is increased, and less tightly-cluster eigenvalues when the number of inclusions is increased. Since conditioning and spectral clustering are (for symmetric systems) key predictors of convergence for Krylov methods, this provides a very useful benchmark.

The SINKER benchmark might be an interesting complement, because of the complexity of flow induced by the corners of the rectangular inclusion.

11. In Figure 1: Sample glyphs are difficult to see.

These glyphs are only intended to give an impression of the flow field, and note that the images are high resolution and can be zoomed in on in a PDF file.

12. In Figure 1: What is the message from the peak memory foot point? Why memory size in Table 2 is not enough?

Table 1 only covers a single coefficient structure. We include the memory plot in Figure 1 in order to show, in a quick-to-see way, that the memory behavior is insensitive to the coefficient structure and to make the scaling behavior (the slope of the curves) apparent, hopefully giving the reader a clear idea of the memory footprint gains available over a range of problems.

13. In Table 2: For a fair performance comparison, it should be noted that the number of iterations independent from the DOF for ABF.

Table 2 is concerned with the effect of the drop tolerance on the ILDL-preconditioned solve, but Table 1 includes iterations counts for the ABF solver. This independence of iteration count on problem size for the ABF preconditioned solves isn't explicitly noted elsewhere, as the scalability of the full solves implies this.

14. In line 320: Since your ABF is based on Jacobi smoother and Arnoldi type Krylov method, more smoothing iteration or avoiding rounding error of GMRES are promising to gain the convergence even with 10ˆ6 problem. It is interesting to see the performance of ABF with increasing the number of inner smoothing iterations to converge 10ˆ6 problem (I argue that such simple tuning is out of the expertise.). Whether such robust ABF can solve the 10ˆ6 problem faster than the ILDL method or not, is the matter for ILDL to be the alternative of ABF.

It is indeed true that ABF solvers can be tuned to deal with large viscosity contrasts by increasing the amount of computational work done by the inner multigrid solver. For a given problem, which solver offers the fastest time-to-solution can indeed depend on the amount of effort spent in tuning the solver. The main point we wish to make, though, is that ILDL-preconditioned solves are less sensitive to changes in the coefficient structure than any given ABF solver. It is our anecdotal observation that the performance of the ILDL-preconditioned solves tends to at least degrade incrementally with viscosity contrast, whereas the ABF solve can completely stagnate at a certain contrast, requiring the sorts of tunings mentioned.

15. In line 297: Please write Eqs. (5), (11), and the norm should be a consistent form.

We have changed the notation in the mentioned equations and added a note on line 305.

16. In line 353: Additive Schwarz Method (ASM) should be noted.

Fixed.

17. In "Using ILDL within a parallel preconditioner": Since ILDL is worth investigating as an alternative of direct solver PRADISO rather than ABF solver, the performance on SMP system (openMP) is more interesting than distributed memory parallelization (MPI). Please reconsider the way of presentation. Since ABF is inherently suitable for the distributed memory parallelization, Table 3 did not show any advantage of ILDL.

We agree that this is a very interesting future avenue, and is an ongoing project in terms of research and software development. There are a few references at the end of the paper (line 410) on recent work, which we more properly introduce as mentioned in the next comment. Implementations of ILDL preconditioners which function in shared memory parallel environments (e.g. OpenMP) or fine-grained parallel environments (e.g. on GPUs) show obvious promise in terms of reducing the time-to-solution of the ILDL-preconditioned solves presented here, by parallelizing operations (though not notably affecting algorithmic properties like number of iterations). Most importantly, we note that future parallel implementations will not notably impact memory usage, which we have emphasized as the key limiting resource for practitioners relying on direct solvers.

18. In lines in 400-404: These lines seem to be a jump in the context. Please introduce them in more detail if you want to address them. By the way, "incomplete LDL" should be ILDL

We have tried to reword the concluding statements (line 407 and onwards) on extensions to the algorithms to make the presentation flow better.

ILDL is a synonym for "incomplete LDL^T", but we have made this change.

[1] P. P. Grinevich and M. A. Olshanskii, An iterative method for the Stokes-type problem with variable viscosity, SIAM Journal on Scientific Computing, 31 (2009), pp. 3959– 3978