se-2020-79
Author's Response

Patrick Sanan, for all authors

We again thank the reviewers and everyone involved in the adiminstration of SED for an illuminating and enjoyable review process.

Here, we include full responses to the two reviewers, and a version of our revised manuscript with all differences automatically highlighted.

The reviewer responses include line number references to all important changes, but note that these line numbers are with respect to the revised article PDF submitted, *not* the marked-up version at the end of this document (which has additional lines due to the diff notation).

se-20200-79-RC1
Response to Reviewer Comments

Patrick Sanan, for all authors

We deeply thank the reviewer, Marcin Dabrowski, for insightful and thorough comments. We reproduce these comments here and provide our own comments and responses inline. Line numbers in our responses refer to the revised manuscript.

> General comments
>
> The paper addresses problems relevant to the scope of SE and includes some interesting novel concepts regarding the numerical solution of 3D mechanical problems. The scientific methods and assumptions are sound and the presented conclusions are justified. The authors give credit to previous related work and they clearly delineated their contribution. The paper is well written and properly structured, the title is informative and clear, and the abstract provides a good summary of the work. Below I present my specific comments and technical corrections. I would encourage the author to include a more detailed presentation of the studied numerical setups within the main body of the manuscript, according to my detailed suggestions below.

> Specific comments
> For the Taylor-Hood element, the static elasticity in the mixed finite element formulation produces a symmetric indefinite system. It is maybe worth noting that for FEM discretization with piecewise discontinuous pressure field such as in the case of the Crouzeix-Raviart element family, the pressure mass matrix can be easily inverted on the element level and by performing block Guassian elimination a positive definite system can be obtained that allows for using the highly robust sparse Cholesky factorization.

In the discussion of the ineffectiveness of ILU preconditioning (line 289), we have added a citation to Dabrowski et al., 2008 to highlight this approach, which indeed needed to be mentioned.

> The author claimed that the previous incarnations of ILDL' were not necessarily robust(l.93-94). Could the authors just briefly mentioned the major improvements within the recent ILDL' implementation? What improvements exactly have made them robust in the recent years?

This mainly refers to ILDL preconditioning without the weighted-matching step, or just applying ILU or incomplete Cholesky factorizations directly. As in the Chow and Saad 1997 reference cited on line 284, ILU factorizations perform very unreliably. Hagemann and Schenk's 2006 paper presents some experiments which compare the effect of different pivoting strategies.

> What is exactly meant by "coefficient structure" (for example l. 115)? I guess that this is not just the sparsity pattern.

This is intended to refer to the functional form of the coefficients, that is how the material properties vary over the domain. Of interest in the context of solver robustness is whether these coefficients have large global variation, large local variations, and geometrically simple (roughly, "low frequency") or complex distribution of these variations.

We've further clarified the use of this term on line 113.

1x1 and 2x2 blocks are mentioned in the context of pivoting for both LDL' and ILDL'. I am actually wondering whether the natural blocking inherent to the problem due to its dimensionality is retained during this operation? It is stated that fill-reducing reordering is performed block-wise. Which blocks are exactly meant here? I would guess that the ones related to the problem dimensional (say 3x3 blocks in the case of 3D problems). How is it ensure that the blocking due to the symmetric maximum weighted matching preprocessing is retained during the subsequent fill-in reducing reordering? I would suggest that this issue could be clarified in the manuscript.

The natural blocking ($2+1$ or $3+1$) is not directly retained during the factorization. All blocking and ordering at the level of the preconditioner is done with 2×2 and 1×1 blocks.

This has the advantage of requiring less information from the user, which is extremely desirable in the context of providing widely-applicable and robust methods. However, methods which take the specific saddle point structure of the problem into account do exist; for instance, the paper from Wubs and Thies (in the paper's references) takes into account what they call $\mathcal{F}$-matrix structure.

We have added a footnote (line 233) to emphasize which blocking is being discussed.

So what is exactly used as the Schur complement preconditioner for large coefficient jumps? The author mention "a scaled pressure mass matrix" in this context. What (viscosity) scaling is exactly used? If there is not enough space for explaining it, maybe the authors could refer to some other work here.

This has been clarified in the text (line 345) to note the exact scaling, $-\left(\frac{1}{\mu}+\frac{1}{\lambda}\right)$. As suggested by Reviewer 2, we have added a reference (line 254) to Grinevich and Olshanskii's 2009 paper, which discusses this preconditioner. Also see the response below to the comments about the C term.

The authors claim that sparse direct solution methods for indefinite systems using LDL' are expected to be highly competitive for 2D cases (l. 263). Is there any recent study showing their real performance (not just the theoretical scaling) that could be referred here?

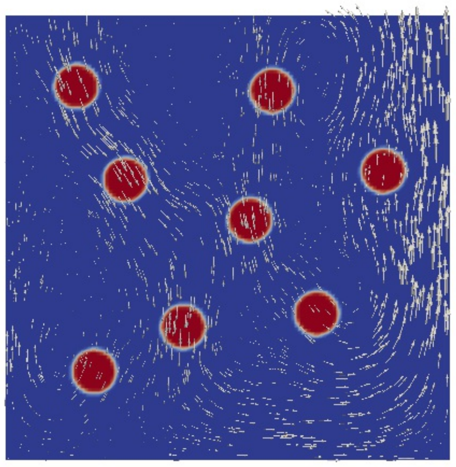
We include a 2D study, from an earlier draft of this paper, which hopefully gives a direct example of this, in Figure 1 of this response.

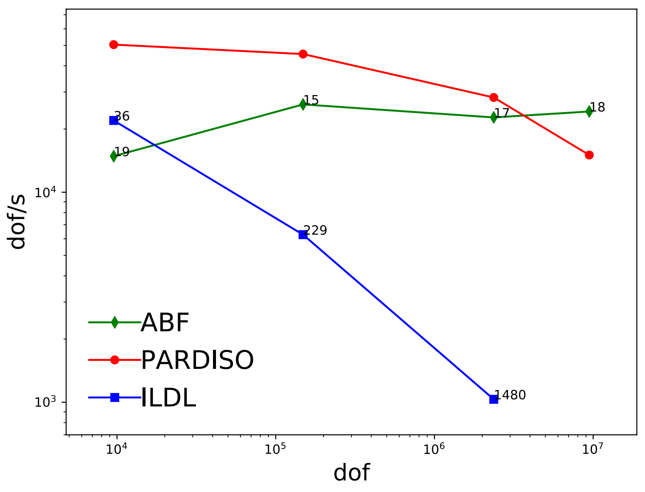
The authors make statements that variable coefficients on the level of individual elements (non-grid aligned coefficient jumps) are, loosely speaking, harder to solve. In what sense? Solution accuracy or solution time, or maybe both?

With sharply-varying fields, higher order convergence is harder to obtain. In practice, solution times are likely longer, both due to this lower convergence order requiring a finer mesh, or by the fact that multigrid methods appear to converge less quickly.

I would guess that referring to incomplete factorization preconditioners in l. 283, the authors specifically mean ILDL' rather than ILU or ICHOL, and that they perhaps make this statement in the context of geodynamics or, in general, geosciences.

Yes - this sentence was only meant to refer to incomplete factorization preconditioners for indefinite problems in geosciences (or even more generally in computational mechanics). We have updated it in the discussion around line 284.





	GMRES(60)/ILDL(1e-3)				PARDISO		FGMRES(30)/ABF			
Els.	Fill	Its.	Time [s]	Mem. [MB]	Time [s]	Mem. [MB]	Lvls.	Its.	Time [s]	Mem. [MB]
32^2	2.9	36	4.34E-01	27	1.89E-01	<1024	3	19	6.43E-01	<1024
128^2	4.4	229	2.37E+01	324	3.27E+00	404	5	15	5.68E+00	335.00
512^2	5.7	1480	2.29E+03	5879	8.37E+01	7059	7	17	1.04E+02	5292.00
1024^2					6.28E+02	31266	8	18	3.90E+02	20846.00

Figure 1: Experiments comparing solvers for a 2D stationary Stokes flow problem with dense ($1.1\times$), viscous ($\eta_1 = 10^4, \eta_0 = 1$) inclusions, discretized with $\mathbb{Q}_2 - \mathbb{Q}_1$ finite elements. Extremely short runs do not always report accurate memory usage on the test system.

The comparative study of Gould (2007) is mentioned in line 292 to justify the choice of PARDISO. I am wondering where there could be any more up-to-date performance studies for sparse direct solvers of symmetric indefinite systems.

Unfortunately, we couldn't find a good, more-recent study (though this would also be of great interest to us). We hope that the older study at least convinces the reader that we have chosen a highly-competitive direct solver, to provide a meaningful comparison with other options. We have slightly modified the statement in the paper (line 300), to imply that we choose it as a competitive solver in general, and that the cited paper offers some concrete, though not complete, evidence of this.

I totally agree with the authors that the choice of norm is important for matrices characterized by large condition numbers such as in the case of the studied Stokes problem with strongly variable viscosities. In this respect, the authors choose to use the true residual 2-norm rather than the norm induced by the preconditioning. It is perhaps outside the scope of this study, but, in my view, given that the authors have access to highly accurate solutions obtained using the direct solver approach, it would be quite interesting to check and compare the solution error between GMRES(60)/ILDL & FGMRES(30)/ABF, say in the energy norm.

Choosing a norm is usually a compromise, and here we chose one that we thought would be the most representative of something like "actual solver performance", in terms of minimizing a quantity (true residual norm) which most people would agree corresponds well to minimization of the quantity of actual interest. However, as noted in the paper (line 308), in practice a different (quasi-)norm would likely be used, either because of computational expedience or because it better represented the application's idea of an accurate solution. We do think it's probably outside the scope of the paper to present the results in additional norms, but agree that it would be interesting to interrogate and should be, if a practitioner relies on a specific norm.

As a related point here, we have devoted considerable effort to making the experiments here reproducible, with publicly-available (and usable under a BSD-2 license) source code, to help address the problem that users will often want or need to further interrogate the experiments presented. Here, additional norms could be examined by running the application code (with the help of our reproduction supplement), and using PETSc command line options to monitor a different norm or (if the desired norm is not supported), even modifying the C code. While obviously the investment of time and effort to run any code is non-trivial, as readers we very much appreciate the ability to examine source code to answer the common question of how a description in a paper ultimately translates to the implementation, and if interested in extending the method, to be able to directly compare to an existing implementation. As scientific software becomes more and more complex and relies on larger and larger software stacks, the notion that reproducibility (and re-implementability) is implied by a technically-complete description of algorithms becomes less and less valid.

Do the solve times reported in the tables for the ILDL' preconditioning include the time spent on computing the ILDL' preconditioner? Actually, it would interesting to see how this time compares to the time spent on iterations.

The solve times include both the setup and solve time. The motivation for this is as mentioned on on line 114 - for most of the applications we envision being relevant (3-dimensional nonlinear problems with large-enough memory footprints that direct solvers become problematic), the system is only solved once.

The setup time usually dominates the application time, very roughly requiring $50-90\%$ of the total solve time, for the experiments in this paper. If running the included code, one can observe this by using PETSc's logging feature (use `-log_view` as an option), and then can observe the amount of time spent in `PCSetUp`, where the factorization is computed.

We agree that this is under-reported in the paper (especially given the prevalence of reporting these times separately in computational science literature, whether or not is really relevant). Thus, we have added a note on line 297 and a new column of setup times in Table 1.

Regarding the numerical setup, I would claim that what really matters is the fraction of the inclusion. With increasing inclusion fraction, as in the case of the setup studied in Fig. 2, a natural transition towards porous media like systems occurs (technically speaking, I am wondering how well 100 inclusions can be resolved using a 32ˆ3 computational mesh). Such physical systems are characterized by strongly localized flows, which might be harder to solve compared to the suspension type of flow typically obtained for low concentration. It would be actually interesting to see how well the presented methods work when the gravity load is replaced by an ambient pressure gradient prescribed through boundary traction.

None of the experiments presented here includes a particularly high volume fraction of inclusions.

The intent of these benchmarks is not to stress the solvers by imposing a complex domain (as for instance would be the case for a high volume fraction of rigid inclusions, with only the interstitial space meshed). Rather, it is to explore the solver performance on a simple domain, but with coefficient structures which vary across non-grid-aligned discontinuities. The multiple sinker benchmark is convenient, but one could also have explored, say, a sinusoidal boundary between two materials, varying the coefficient jump across the boundary, and the frequency of the boundary.

Technical corrections

l. 11-14 This sentence seems a bit convoluted. I would actually guess that something might be missing here.

A typo (extra "to") has been fixed and this sentence (line 11) has been simplified and split into two.

l. 22-23...the coefficient structure is made increasingly challenging – I would suggest formulating it more precisely; What "complex topologies" have been addressed in this study?

We have changed this sentence to specifically mention the multiple-inclusion scenario (line 24).

l. 33 This is maybe not so critical, but compressible quasi-static linear elasticity is not exactly an example of a problem with a divergence free displacement field. In addition, it may indeed represent a saddle point problem, but in some numerical formulations it may be straightforwardly cast as a positive-definite problem.

This was indeed incorrectly expressed - to amend this, we have added a sentence to mention the interesting and computationally-relevant fact that systems requiring divergence-free flow/displacement fields can be seen as limiting cases of systems which penalize volume changes (line 30).

We have chosen to focus on incompressible or nearly-so examples in this work, as outside of this context, there is less motivation to introduce elasticity in mixed form (even at the continuous level).

l.71...the nonzero entries of the factors are restricted to those for Aˆk – This could be stated a bit more precisely.

Fixed to "A^{k+1}" and wording clarified (line 71).

l.72-73 I find the end part of this sentence unclear.

The has been reworded to be more concise (line 72), as the point of this passage is simply to point out that only a small number of parameters are required for various variants of ILU preconditioning.

l. 109 In contrast to the previous ILDL studies previous mentioned above...- please remove the second instance of “previous”

Fixed.

l. 134 One could consider using the transpose for one of the vectors in $n * \sigma * n$, etc in eq. 3 & 4.

We opt not to do this, to try to keep the notation uncluttered. We believe it is at least consistent notation, in the sense that $u \cdot v = u^T v$ for two vectors, $u \cdot A = A^T u$ for a vector and a tensor (thought of as a matrix) and similarly $T \cdot u = Tu$.

To make this section more clear, we have added a description of the boundary conditions as a partition of the boundary into free-slip and free surface (zero stress) regions (line 134).

l. 179 Is it really necessary to replace τ with dev(σ) in eq. 3? Given that the t and n vectors are perpendicular ($\langle t, n\rangle = 0$), $t' * \sigma * n = t' * (\tau - p * I) * n = t' * \tau * n - p * tn = t' * \tau * n$

This is true, and indeed this requires no special treatment in our code, so we have removed this statement.

l. 210 permutation (a map from rows to columns) – I would think that the permutation operates within the rows and within the columns, and not from rows to column.

This was confusingly written and we've removed the mention of the row-column map (line 209), as the interested reader is better served by reading about the details of the matching in the references. Briefly, the problem of permuting the matrix is re-cast as a matching problem: the matrix is interpreted as a weighted bipartite graph, where entries correspond to edges between rows and columns. The maximum weighted matching gives (for a nonsingular matrix) a subset of the edges such that each row and column is involved exactly once, which can thus be interpreted as a permutation. This subset is sought which maximizes an objective (the product of the entries).

l. 215 If one wishes to find a symmetric permutation, one can only change the order oft he diagonal entries. – If I am getting it right, a symmetric permutation preserves the symmetry of the matrix. I guess that with changing the order of the diagonal entries,the order of the entire rows and columns is also changes (not just the order of the diagonal entries). Anyway, could “non-symmetric” permutations be considered in the current context?

A symmetric permutation of a square matrix M is indeed of the form PMP^T, where P is a permutation matrix, and preserves the symmetry of the matrix while moving entire rows and columns.

We have not considered non-symmetric permutations in this context, but we do not categorically disregard them. The aim of the permutations is to provide a good pivoting strategy, and the current approach seems to scale close to optimally in practice while still retaining system symmetry (thus allowing one to store only L and D, as opposed to two triangular factors, and allowing the use of methods like QMR or MINRES which require symmetric systems). Thus, exploring non-symmetric permutations becomes less appealing.

However, the question of relaxing the symmetry requirements is a very interesting one! On the practical level, this would be highly desirable for applications, for instance finite difference schemes (including finite volume schemes on orthogonal grids) for the Stokes equations, which don't produce a symmetric system.

Interesting future work could address the usage algorithms which can extend the approaches presented here (for instance, exploring the use of the multi-level ILU as implemented in ILUPACK) to non-symmetric, indefinite systems which arise in computational geosciences.

l. 293-4 Through a custom interface we use PARDISO (Kuzmin et al., 2013) – This looks a bit repetitive with respect to the previous sentence.

Modified to be more concise (line 299).

l. 298 The choice or norm allows is...- Please fix.

Fixed.

Table 1 - I would suggest that the volume fraction of the inclusions could be given. The viscosity is shown without the unit, and this problem could be easily solved by showing the viscosity ratio. Is the relative density dimensionless? Is it defined as $(\rho_{\text{incl}} - \rho_{\text{host}})/\rho_{\text{host}}$? Is it actually relevant given that the model is linear? I would suspect that changing the relative density should only result in a rescaled velocity, and it should, hopefully, produce no appreciable changes to the course of numerical iterations. Is "fill" defined as the ratio between the non-zero entries in the ILDL' factor with respect to the non-zero entries of the original matrix (the triangular part of it, including the diagonal)? Is it necessary to use the scientific notation when time is reported? Maybe giving the total dof count could be useful.

We haven't focused on volume ratio (though several are computed in the remainder of this response), because these ratios are low and we don't believe that this is a factor stressing the solver. We choose the multiple-inclusion problem as a benchmark not because of its direct relevance in application (where higher volume fractions are a key consideration) but because of its usefulness as an abstraction of difficult coefficient structure. See also our response to Reviewer 2's general comments.

However, the presentation wasn't clear enough to make the volume fraction obvious to the reader. As such, we've added the inclusion radius to the captions of Figures 1,3, and 4. This information was also available at the very end of our reproduction supplement, where we have updated the command-line options to include the default (0.1) inclusion radius.

We have changed the caption of Table 1 to mention only the viscosity ratio, and added a column for total DOFs. The relative density is dimensionless, and is defined as $\rho_{\text{incl}}/\rho_{\text{host}}$. Indeed, as this is a linear problem, most simple scalings of any of the parameters have little effect on the solver performance, which motivates the fact that we do not focus heavily on units or scalings in this paper, but on variations in the material coefficients.

"Fill" is defined as the number of nonzeros in the L factor in the LDL^T factorization, relative to the number of nonzeros in the strictly upper-triangular part of the matrix being factored. We have added a note in the paper to make this concrete (line 201). Fill is not reported with scientific notation, though drop tolerance is (which we thought was more readable).

Figure 1 - I would suggest a more detailed description of the numerical setups, both in the caption and in the main body of the manuscript. What is the volume fraction of the inclusions? What is meant by (Vel. scaled 1/3x)? Isn't it that the scaling of the quiver lengths is in no obvious way absolute? I think that it would be useful to show gridlines in the plots. I guess that the dashed line in the Peak Memory Footprint shows the maximum available RAM during the numerical tests, but it would be useful to explain it in the caption. The curve styles are not well visible in the legend. It could also be explicitly explained in the caption that ABF(a), ABF(b),...refer to setups a, b, c...(at a first glance it may look as if it were some variants of the solvers).

The volume fractions of the inclusions can be computed from the inclusion count and radius (which are specified in Section 1.7 in the reproduction supplement).

- Single sinker of radius 0.25 $\implies$ volume fraction of 0.06

- 3 sinkers of radius 0.1 $\implies$ volume fraction of 0.01
- 8 sinkers of radius 0.1 $\implies$ volume fraction of 0.03

These are of course low volume fractions, in the context of problems concerned with interstitial flow (where indeed, other models than a pure Stokes model may be appropriate, e.g. including a Darcy-type term). Our use of the multiple-inclusion problem is motivated by its usefulness as an abstraction of coefficient structure which can affect solver performance, as further discussed in our response to the general comments from the second reviewer.

The captions about the velocity scaling are meant to signify that the first two plots have the same scaling, and the second two a different one, but as pointed out, the absolute scaling of these velocities is not very meaningful, so we have removed these notes to reduce clutter.

Grid lines have been added to all graphs in the paper. The flow plots already have grid lines, though one has to zoom in to see them.

Notes have added to the plots specifying that the dashed black lines are indeed the maximum RAM available.

All legend entries have been modified to hopefully make the line styles more clear. The caption of Figure 1 has been changed to refer directly to the coefficient structures (a)-(d).

> l. 317-8...the ABF solver fails to converge. – It is not clear to me where this can be seen in Fig. 1 (I can't really see any missing data for ABF)

This was intended to mean that when using even more inclusions than are presented in Figure 1 are added, the performance continues to degrade, and thus there are missing ABF entries in Figure 2 (right). This has been clarified (line 326).

> Figure 2 - What is the volume fraction of the inclusions as their number is increase? Given that the numerical resolution is kept constant (32ˆ3) I would guess that it is increased. In my opinion, this should be explicitly stated in the caption and also in the main body of the manuscript. In fig. 1 for 32ˆ3 the overall solver performance in terms of dof/s fell in to the range between 5*10ˆ3 and 10ˆ4, which is consistent with the time reported in table 1. However, in fig. 2, even in the previously studied case of the viscosity ratio of 10ˆ4, the performance is between 10ˆ-2 and 10ˆ-1. I would guess that this could be some technical mistake. In my opinion, it would be useful to show gridlines and maybe use a slightly large font for the legend entries.

The volume fraction indeed increases with the number of inclusions. The inclusion are of radius 0.05 (in the unit cube), and as such for the maximum of 140 inclusions, representing about 7% of the volume. We've added a note to this effect in the caption of Figure 2.

There was indeed an error in our plotting script (an error in the calculation for the total number of DOFs). We have fixed the error and made the y axes uniform between the two sub-plots in Figure 2.

The legends have been increased in size in Figure 2, and grid lines have been added to all graphs in the paper.

> l.324 "...varying to drop tolerance" – Please fix.

Fixed.

> l. 326 System scaling is mentioned in the footnote. Please explain what system scaling(physical, algebraic, ..) is exactly meant here.

This has been clarified (line 333) to refer to a (newly-numbered) equation in Section 3, describing the preprocessing performed before the drop tolerance is applied.

l.339.. and C is the term (depending on λ as in Eq.(9)). – I would guess that the outer brackets are not necessary here. Could the author hint what they actually use for the C term?

The parentheses were indeed a typo, and we agree that the presentation was unclear.

The matrix $-C$ arises from a term in the weak form like $-\int_\Omega q^T \frac{1}{\lambda} p dv$, hence is just another scaled pressure mass matrix, scaled with coefficient $-\frac{1}{\lambda}$, which is ultimately added to the mass matrix scaled with $-\frac{1}{\mu}$, that arises in the same way that the $-\frac{1}{\eta}$ term does for the Stokes problem (as an easy to compute yet spectrally equivalent approximation for the $-B^T K^{-1} B$ term in the Schur complement).

This passage has been reworded in the manuscript (line 343), and more explicit descriptions of the weightings for pressure mass matrices have been added elsewhere (e.g. line 253).

Readers who may be interested in reimplementing the method, or simply wanting to see the direct expression of the formulae in C code, can also examine the source code (at the time of this writing, see `femixedspace.c:2987` at `bitbucket.org/psanan/exsaddle`).

l. 340 Figure 4 shows a similar experiment using a scenario which is perhaps more typical in applications. – Please explain the boundary conditions used in this setup in the main body of the manuscript.

We have added this description and a short note in the initial description of the elasticity problem to highlight that in this case we use inhomogeneous boundary conditions; these are a simple modification of the free-slip conditions, adding terms to the righthand side to specify a given normal displacement as opposed to a zero normal displacement.

Figure 3 – Maybe the Lame parameters μ and λ could be scaled by $\rho * g * L$. A colorbar for the color-coded pressure and gridlines would be a nice addition to this figure.

We have not focused on scaling parameters, as global scalings of these do not affect linear solves, and only relative scalings affects the solvers considered here; indeed, it is one of the great advantages of the ILDL preconditioners, shared with the direct solvers, that they can largely automatically address scaling issues. In each case, the maximum pressure (red) is about 1, and the minimum is a small number (corresponding to a zero pressure at the top, free surface). We have added color bars for the pressure fields, and grid lines for the graphs. There are already grid lines (albeit faint) in the 3d plots.

Figure 4 – It is of small relevance to the studied topic, but the deformed wire mesh implies a substantial deformation that could hardly be accommodated elastically by any geomaterial. But maybe this could be treated as an exaggerated mesh deformation.The elastic moduli are given with no units.

This being a linear problem, it is indeed hopefully still a relevant (and easier to visualize) experiment, even when using an unrealistically-large deformation.

We have not emphasized the units or absolute values of the parameters, as while these are obviously of great importance in actual applications, the applicability and effectiveness of the solvers discussed in this paper are crucially dependent on relative coefficient variability, and essentially invariant to scalings.

Marcin Dabrowski

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 10ˆ6 viscosity contrast within 32ˆ3 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

Pragmatic Solvers for 3D Stokes and Elasticity Problems with Heterogeneous Coefficients: Evaluating Modern Incomplete LDL^T Preconditioners

Patrick Sanan[1], Dave A. May[2], Matthias Bollhöfer[3], and Olaf Schenk[4]

[1]Department of Earth Sciences, ETH Zurich, Sonneggstrasse 5, Zürich 8092, Switzerland
[2]Department of Earth Sciences, University of Oxford, South Parks Road, Oxford OX1 3AN, United Kingdom
[3]Institute for Computational Mathematics, TU Braunschweig, D-38106 Braunschweig, Germany
[4]Advanced Computing Laboratory, Università della Svizzera italiana, Via Buffi 6, Lugano 6900, Switzerland

Correspondence: Patrick Sanan (patrick.sanan@erdw.ethz.ch)

Abstract. The need to solve large saddle point systems within computational Earth sciences is ubiquitous. Physical processes giving rise to these systems include porous flow (the Darcy equations), poroelasticity, elastostatics, and highly viscous flows (the Stokes equations). The numerical solution of saddle point systems is non-trivial since the operators are indefinite.

Primary tools to solve such systems are direct solution methods (exact triangular factorization) or Approximate Block Factorization (ABF) preconditioners. While ABF solvers have emerged as the state-of-the-art scalable option, they are invasive solvers requiring splitting of pressure and velocity degrees of freedom, a multigrid hierarchy with tuned transfer operators and smoothers, machinery to construct complex Schur complement preconditioners, and the expertise to select appropriate parameters for a given coefficient regime – they are far from being "black box" solvers. Modern direct solvers, which robustly produce solutions to almost any system, do so at the cost of rapidly growing time and memory requirements for large problems, especially in 3D. Incomplete LDL^T (ILDL) factorizations, with symmetric maximum weighted matching preprocessing, used as preconditioners for Krylov (iterative) methods, have emerged as an efficient means to solve indefinite systems. These methods have been developed within the numerical linear algebra community but have yet to become widely used in ~~non-trivial~~ applications, despite their practical potential~~; they~~ . They can be used whenever a direct solver can, only requiring an assembled operator, yet can offer comparable or superior ~~to performance , with the added benefit of having~~ performance with a much lower memory footprint. In comparison to ABF solvers, they only require the specification of a drop tolerance and thus provide an easy-to-use addition to the solver toolkit for practitioners.

Here, we present solver experiments employing incomplete LDL^T factorization with symmetric maximum weighted matching preprocessing to precondition operators, and compare these to direct solvers and ABF-preconditioned iterative solves. To ensure the comparison study is meaningful for Earth scientists, we utilize matrices arising from two prototypical problems, namely Stokes flow and quasi-static (linear) elasticity, discretized using standard mixed finite element spaces. Our test suite targets problems with large coefficient discontinuities across non-grid-aligned interfaces, which represent a common, challenging-for-solvers, scenario in Earth science applications. Our results show: (i) as the coefficient structure is made increasingly challenging~~(high contrast , complex topology)~~, by introducing high contrast and complex topology with a multiple-inclusion

benchmark, the ABF solver can break down, becoming less efficient than the ILDL solver before breaking down entirely; (ii) ILDL is robust, with a time-to-solution that is largely independent of the coefficient topology and mildly dependent on the coefficient contrast; (iii) the time-to-solution obtained using ILDL is typically faster than that obtained from a direct solve, beyond 10^5 unknowns; (iv) ILDL always uses less memory than a direct solve.

1 Introduction

Saddle point systems frequently arise in the context of constrained minimization problems. Many physical processes relevant to the Earth sciences fall within such a minimization framework. Possibly the most widely used relates to the variational statement which seeks to constrain a vector field (e.g. displacement / velocity) to be divergence free. These statements can be viewed, both numerically and physically, as limiting cases of scenarios in which volume change is penalized. Such formulations naturally introduce a pressure-like (scalar) variable to constrain the displacement / flow (vector) field. Specific examples include: mixed Darcy problems involving the unknowns Darcy flux and saturation pressure (porous flow, groundwater flow, oil and gas reservoirs); poroelasticity (geomechanics, reservoirs engineering, bore hole stability); compressible / incompressible quasi-static linear elasticity (crustal deformation targeting on inter-seismic periods); and incompressible viscous flow (dynamics of the mantle, lithosphere, glaciers, ice sheets). Other relevant (but more generic) applications giving rise to discrete problems of saddle point type include PDE-constrained optimization, weak imposition of boundary conditions (e.g. contact, fault constitutive behaviour) and matching conditions between different model domains (e.g. Beavers-Joseph matching conditions between fluid-solid regions).

Solution techniques for saddle point systems have been extensively studied (Benzi et al., 2005; Benzi and Wathen, 2008; Loghin and Wathen, 2004). Nevertheless, ~~saddle point systems have a reputation for being~~ a saddle point system may be challenging to solve ~~– this is to a large part attributed to the fact that the~~ because the discrete problem (~~e.g.~~ i.e. the matrix), while often symmetric, is *indefinite*; this structure precludes the use of many standard approaches (like classical multigrid).

Saddle point problems too large to be practically solved via sparse direct solution techniques (e.g. LU or LDL^T factorization)~~are deemed to require~~ (Davis et al., 2016) can be solved with highly-specialized solvers ~~. Indeed, if algorithmically optimal methods are sought, this opinion is justified. As such, the~~ which exhibit better scaling in both time and memory required. The development of highly scalable, optimal preconditioners for the solution of large-scale variable viscosity Stokes systems arising in ice sheet modeling (Isaac et al., 2015) and geodynamics (May et al., 2014; Rudi et al., 2015) is mature.

However, the *practical* usage of saddle point solvers does not always favor ~~the above agenda~~these approaches. Maximum problem sizes of interest are typically fixed or modest (e.g. $< O(10^8)$ DOFs); algorithmic or parallel scalability may not be to be prized to the exclusion of all else; time-to-solution(s) may not dominate the time required to set up and tune (by hand) a specialized solver; or computational resources available may be modest (e.g. single compute node with 100 GB RAM and unlimited walltime, or few low memory compute nodes with ~ 400 cores with walltime restricted to < 24 hrs). Specialized, optimal solvers often lack robustness as problem parameters or problem types are varied, though progress is being made in this regard (e.g. Rudi et al., 2017).

Solvers can trade some algorithmic performance for robustness and/or ease of use. A striking example is the persistent use of incomplete LU (ILU) preconditioning (e.g. Wathen (2015) §6.1, Benzi (2002) §3). With the help of an easy-to-apply-and-store approximate inverse to the system matrix, obtained by discarding terms from a full factorization (as used in a direct solver), the solution may be iteratively updated until it approaches the solution. While ILU-preconditioned Krylov methods are neither algorithmically scalable nor completely robust, the method is ubiquitous for several reasons.

1. *Only an assembled operator is required.* Many efficient preconditioners require auxiliary information, typically concerning the physical domain of an underlying PDE. For example, multigrid solvers require a hierarchy of grids and transfer operators and FETI-DP (Farhat et al., 2001) methods require access to finite element subdomains. However, purely algebraic solvers, which only require an assembled operator (matrix), are always in demand, as they can be applied ~~much~~ more broadly and allow for greater ease of algorithmic experimentation.

2. *Reasonable performance is observed for a large class of relevant problems.* Some variants of ILU preconditioning reduce the condition number of a standard second-order elliptic PDE, discretized with finite differences or finite elements, from $O(h^2)$ to $O(h)$ (Benzi, 2002) As such, the preconditioner has been used in many applications, alone or as a subdomain preconditioner for a block-Jacobi preconditioner, for example in subsurface flow (Mills et al., 2007).

3. *The methods are tunable, with a small number of parameters.* ILU-preconditioned Krylov methods typically only expose a small number of parameters to the user, depending on the variant employed (Saad, 2003, Ch. 10). For instance, in ILU(k) methods, the nonzero entries of the factors are restricted to those of ~~A^k, one may drop any~~ A^{k+1}. One may also drop entries below a given threshold ~~,~~ or use more complicated approaches such as ILUT (Saad, 1994) ~~. Multi-level~~ or multi-level inverse-based dropping strategies (Bollhöfer and Saad, 2006)~~accept a drop tolerance for factorization as well as a potentially different drop tolerance~~.

   Denser factors typically produce a better approximate inverse, which typically makes for a better preconditioner. Thus, a simple trade-off exists between iteration counts and memory usage. This is in contrast to direct solvers, which do not offer the user any control over memory usage and which always (over-)solve to machine precision.

   Often, an additional choice is available of an ordering strategy such as approximate minimum degree ordering, nested dissection, reverse Cuthill-McKee (RCM), or others. Finally, a choice must be made of the Krylov method itself; this study uses right-preconditioned GMRES or FGMRES~~, but this is rarely a critical choice and many options are usually available, as Krylov methods are far simpler to implement than direct solvers or incomplete factorization preconditioners.~~
   .

4. *Tools are widely available in software.* Because of the conveniences described in points 1 and 3, ILU preconditioners have made their way into numerous software packages, including MATLAB (MATLAB, 2019), ILUPACK (Bollhöfer and Saad, 2006), UMFPACK (Davis, 2004), WSMP (Gupta et al., 1997), PETSc (Balay et al., 2019a, b), TRILINOS (Heroux et al., 2005), and many others, as they often provide a reasonable default preconditioner.

5. *The method is well-discussed in the practical literature.* Potential users are likely to have access to a performance study which includes the effect of ILU preconditioning, with documentation of the parameter choices employed. ~~A valid criticism of ILU preconditioning is that part of its popularity is simply due to its heritage; it was studied early and hence there is robust literature which directly demonstrates its use.~~

Incomplete LDL^T (ILDL) preconditioners arise as incomplete versions of direct solvers for indefinite systems, which use the factorization $A = LDL^T$, where L is lower-triangular and D is block diagonal. Here the term "incomplete" implies that factorization is only approximate. ILDL methods with symmetric maximum weighted-matching preprocessing (see Section 3) have emerged in the numerical linear algebra community as an analogous method to ILU methods, but for symmetric indefinite systems; they are more robust than ILU or previous incarnations of ILDL. Recently, comparison studies characterizing such approaches have appeared: Hagemann and Schenk (2006) provide a study of the effectiveness of ILDL preconditioning with respect to a matrix "zoo"; Greif et al. (2015) performed a similar study with their SYM-ILDL package; Multilevel ILDL preconditioners with symmetric weighted matching preprocessing have been shown to be effective when applied to the Helmholtz equation (Bollhöfer et al., 2009); Schenk et al. (2008) showed that a similar approach can also be used to effectively compute a few interior eigenvalues of a large indefinite matrix arising from the Anderson model of localization. Recent work on sparse inverse covariance matrix estimation highlights how ILDL preconditioners can be preferable to highly efficient direct solvers, due to their much lower memory footprints (Bollhöfer et al., 2019a).

Despite these studies, the applicability of ILDL preconditioners is less well-known outside the numerical linear algebra community~~. This is unfortunate, as~~, even though these modern methods bring incomplete factorization approaches for indefinite symmetric systems in line with other popular methods, in terms of robustness and ease of use. It should also be emphasized that points 1 and 3 hold for the ILDL method.

1.1 Motivations and Outline

This paper addresses points 2, 4, and the beginnings of 5 in the context of using ILDL preconditioners with symmetric maximum weighted matching ordering applied to saddle point systems arising from the spatial discretization of a class of PDEs relevant to the solid Earth.

In contrast to the previous ILDL studies ~~previous~~ mentioned above, which mostly focus on the robustness of preconditioners across a corpus of matrices representing individual instances of different applications, here we focus on a deeper examination of a specific class of PDEs commonly used within the Earth sciences. Specifically, we wish to examine the saddle point problems arising from stationary Stokes flow with highly heterogeneous viscosity structure, and systems arising from the static linear elasticity, also with large coefficient jumps. Particular attention is paid to physical problems with ~~challenging~~large, non-grid-aligned coefficient jumps, as these constitute ~~some of the more~~ challenging systems of interest within the Earth sciences. This focus allows new insight into the effect of varying problem size and parameters ~~(in particular, coefficient structure)~~ on the performance of solvers. In particular, the spatial distribution of material parameters, which we refer to as "coefficient structure", is highlighted.

We only examine the scenario in which a given linear system need only be solved for a single right hand side. This is typical when solving nonlinear systems of equations (often within time-stepping algorithms) when a Jacobian and residual are assembled and used to compute a step.

The saddle point operators arising from the Stokes and static elasticity (in mixed form) systems are presented in Section 2. Section 3 describes the incomplete ILDL factorization preconditioner with maximum symmetric weighted matching preprocessing. Section 4 describes an Approximate Block Factorization (ABF) preconditioner and a sparse direct solver, which serve as representatives of the two classes of alternative approaches in common use and which we will compare ILDL against. Section 5 presents experiments which characterize the performance of these ILDL-preconditioners, direct and the ABF-preconditioned solves applied to the Stokes and elasticity problems for a variety of synthetic model configurations involving multiple inclusions of differing material parameters (with respect to the surrounding medium).

2 Prototypical problems and saddle point systems

2.1 Stokes flow

Conservation of momentum and mass for an incompressible creeping fluid in a domain Ω with boundary $\partial\Omega$ are given by

$$\begin{aligned} -\nabla \cdot \tau + \nabla p &= \rho \hat{g}, \\ -\nabla \cdot u &= 0, \end{aligned} \tag{1}$$

where u and p are the velocity and pressure, respectively. The forcing term is associated with buoyancy variations; ρ is a spatially-varying density and $\hat{g}$ is the gravity vector. For the isotropic media we consider here, the deviatoric stress τ is related to the strain rate $\dot{\varepsilon}[u]$ via

$$\tau = 2\eta\dot{\varepsilon}[u], \quad \dot{\varepsilon}[u] = \tfrac{1}{2}\left(\nabla u + (\nabla u)^T\right), \tag{2}$$

where η is a spatially-varying effective shear viscosity. The system given by Eq. (1) is closed with appropriate boundary conditions specified on the normal and tangential components of the velocity and stress (σ). In this work, we consider the following boundary conditions, partitioning the boundary into free-slip and free surface (zero stress) regions:

$$u \cdot n = 0, \quad t \cdot \tau \cdot n = 0 \qquad x \in \Gamma_D, \tag{3}$$

$$n \cdot \sigma \cdot n = 0, \quad t \cdot \sigma \cdot n = 0 \qquad x \in \Gamma_F, \tag{4}$$

where $\sigma = \tau - pI$ is the total stress, n, t are the normal (outward pointing) and tangent vector to the boundary $\partial\Omega$, for which $\Gamma_D \cap \Gamma_F = \emptyset$ and $\Gamma_D \cup \Gamma_F = \partial\Omega$.

2.1.1 Discrete problem

We use inf-sup stable mixed finite elements (FE) to obtain discrete solutions of Eq. (1). A full description of the variational (weak) problem associated with incompressible Stokes flow can be found in Elman et al. (2005). The discrete Stokes problem

~~$\mathcal{A}$~~ $\mathcal{A}_{\text{Stokes}}$ is denoted by

$$\begin{bmatrix} K & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \quad \text{or } \cancel{\mathcal{A}}\mathcal{A}_{\text{Stokes}} v = \cancel{\mathcal{F}} b, \tag{5}$$

where ~~K~~ K is the discrete gradient of the deviatoric stress tensor and ~~B, B^T~~ B and B^T are the discrete gradient and divergence operators~~respectively. We note that the~~ , respectively. The FE discretization results in ~~K~~ K being symmetric positive definite~~,~~ ~~thus $\mathcal{A}$~~ ; thus, $\mathcal{A}_{\text{Stokes}}$ is a symmetric, indefinite operator.

2.2 Static linear elasticity

The conservation of momentum for an elastic solid in static equilibrium in a domain Ω with boundary $\partial\Omega$ is given by

$$-\nabla \cdot \hat{\sigma} = \rho \hat{g}, \tag{6}$$

where ρ and $\hat{g}$ were defined previously in Section 2.1, and $\hat{\sigma}$ is the total stress, which we assume to obey the linear, isotropic constitutive relation

$$\begin{aligned} \hat{\sigma} &= 2\mu\varepsilon[u] + \lambda \operatorname{Tr}(\varepsilon[u])\, I \\ &= 2\mu\varepsilon[u] + \lambda \nabla \cdot u\, I, \end{aligned} \tag{7}$$

where u is the displacement and the (linear) strain tensor $\varepsilon[u]$ is given by

$$\varepsilon[u] = \tfrac{1}{2}\left(\nabla u + (\nabla u)^T\right) \tag{8}$$

and $\operatorname{Tr}(\cdot)$ denotes the trace operator. The particular form of the constitutive relationship adopted is defined in terms of the two *Lamé* parameters (λ, μ). The first Lamé parameter, λ, characterizes compressibility; as $\lambda \to \infty$, the material becomes incompressible. The second Lamé parameter, μ, is equivalent to the shear modulus (often denoted G) and characterizes resistance to shearing.

Describing materials which are incompressible in some or all of Ω is problematic with the formulation given in Eqs. (6) and (7) since the last term in Eq. (7) behaves like $\infty \times 0$ in the incompressible limit. The latter scenario is relevant when considering plasticity models, or the presence of fluids (e.g. within a crack along the subduction interface). Even when large but finite values for λ (e.g. the equivalent Poisson ratio $\nu = \frac{\lambda}{2(\lambda+\mu)} > 0.49$) are used, "locking" may occur when using standard finite difference, finite volume, or finite element spatial discretizations (Brezzi and Fortin, 1991), rendering the displacement solutions meaningless. The issues in the incompressible limit can be resolved by grouping the problematic terms into a new auxiliary pressure variable, $p = -\lambda \nabla \cdot u$. Then, if we decompose the stress as $\hat{\sigma} = \tau - pI$, Eqs. (6) and (7) can be cast as the following mixed (u, p) problem (Brezzi and Fortin, 1991):

$$\begin{aligned} -\nabla \cdot \tau + \nabla p &= \rho(x)\hat{g}, \\ -\nabla \cdot u - \frac{1}{\lambda} p &= 0, \end{aligned} \tag{9}$$

with stress τ given by

$$\tau = 2\mu\varepsilon[u]. \tag{10}$$

One will observe that form is similar to the Stokes system described in Section 2.1; u now represents displacements, strain is considered instead of strain rate, and pressure and divergence of u are now related by the material parameter λ. This implies that the degree of coupling between u and p within the conservation of mass may be spatially variable as λ need not be constant throughout Ω. One should also note that τ in Eq. (10) is only deviatoric in regions where $\lambda \to \infty$, cf. Stokes where τ is strictly deviatoric everywhere in Ω since $\nabla \cdot v = 0$ is imposed throughout the domain. The boundary conditions given by Eqs. (3), (4) are valid for the mixed elasticity problem, with the velocity v interchanged for the displacement u, and σ interchanged with $\hat{\sigma}$. ~~Additionally, since the stress decomposition in the mixed elasticity formulation is not strictly deviatoric-volumetric (e.g. volumetric terms still live within $\hat{\sigma}$), we replace τ in~~ We also consider an inhomogeneous version of Eq. (3) ~~with dev($\hat{\sigma}$)~~to specify a normal displacement.

2.2.1 Discrete problem

The discrete mixed (u, p) static elasticity problem ~~$\mathcal{A}_\mathrm{L}$~~ $\mathcal{A}_\mathrm{Elasticity}$ is denoted by

$$\begin{bmatrix} K & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \quad \text{or } \mathcal{A}_\mathrm{L} \mathcal{A}_\mathrm{Elasticity} v = \mathcal{F} b, \tag{11}$$

where K is the discrete gradient of the deviatoric stress tensor~~and B, B^T~~ , B and B^T are the discrete gradient and divergence operators, and C is a ~~$1/\lambda$~~ $\frac{1}{\lambda}$-weighted discrete pressure mass matrix. The FE discretization results in ~~K~~ K and C being symmetric positive definite and ~~, thus $\mathcal{A}_\mathrm{L}$~~ thus $\mathcal{A}_\mathrm{Elasticity}$ is again a symmetric, indefinite operator. To ensure the discrete problem is stable when the continuum is incompressible (or near to this limit), be it locally or globally (as determined by the value of λ), as per the discrete Stokes system (Section 2.1.1), the FE basis functions used to discretize the displacement u and pressure p cannot be chosen arbitrarily. Rather, an inf-sup stable pair of FE basis functions must be used.

3 Incomplete LDL^T (ILDL) preconditioning, with symmetric maximum weighted matching ordering, for saddle point matrices

Linear systems involving indefinite symmetric matrices are, in general, more difficult to solve than their positive-definite counterparts. This is partially due to the lack of positive-definite (inner product) structure. Diagonal entries which are small, zero, and/or not of constant sign make pivoting and numerical solution more challenging. Roughly speaking, though, if one can cast them as block systems which are in some sense better behaved, grouping problematic diagonal entries with better-behaved ones, robustness can be regained.

ILDL preconditioners arose as incomplete versions of *direct* solvers for indefinite systems, which use factorization $A = LDL^T$, after permutation and scaling. Here, D is block diagonal with 1×1 and 2×2 blocks, and L is lower triangular (Duff et al., 1991). Weighted matchings were first observed to be effective, *static* approximations to pivoting order by Olschowka and Neumaier (Olschowka and Neumaier, 1996). Duff and Koster introduced fast algorithms (Duff and Koster, 1999) and with the addition of Bunch-Kaufman pivoting (Bunch and Kaufman, 1977), highly efficient sparse direct solvers, both in terms of solution time and memory footprint, where made available (Duff and Pralet, 2005; Schenk and Gärtner, 2006). These have become the standard for the direct solution of sparse indefinite systems (Li and Demmel, 2003; Schenk and Gärtner, 2006).

By limiting the number of non-zeros (the "fill") in L, one can obtain an approximate factorization to be used as a preconditioner for a Krylov method (Hagemann and Schenk, 2006). In this work, we consider "fill" to be the ratio of the number of nonzero entries in L, relative to in the strictly upper-triangular part of the matrix being factored. Wubs and Thies present results for the special case of Stokes $\mathcal{F}$-matrices, arising from a simple finite-difference scheme ~~(?)~~(Wubs and Thies, 2011). A closely-related approach which we do not investigate here is that of signed incomplete Cholesky factorization preconditioners (Scott and Tůma, 2014).

Permutation and scaling based on symmetric maximum weighted matching algorithms have shown to nearly or completely eliminate the need for pivoting in the factorization process, thus giving rise to very efficient methods (Duff and Pralet, 2005). ~~We thus describe these considerations in further detail, to give a flavor of the sophisticated methods which are now available.~~ Numerical stability of incomplete factorization can be enhanced by permuting large elements onto the diagonal of a matrix. One may pose this task as a *(perfect) maximum weighted matching* procedure, ~~finding a permutation (a map from rows to columns)~~ producing a matrix permutation which maximizes the product of the absolute values of the diagonal entries. This can be accomplished via the Kuhn-Munkres algorithm (Laird and Giles, 2002; Munkres, 1957) with a complexity of $O(N^{1+\alpha}\log N), \alpha < 1$ for sparse matrices arising from finite difference or finite-element discretizations (Gupta and Ying, 1999); in practice, however the complexity typically scales linearly with N (Schenk and Gärtner, 2006).

If one wishes to find a *symmetric* permutation, one can only change the order of the diagonal entries. Nonetheless, one can extract cycles from the maximum matching and apply these symmetrically to move large entries *close* to the diagonal, in particular close to small or zero diagonal entries. If these cycles are decomposed into 1×1 and 2×2 cycles, one can then define a blocking wherein diagonal entries may be small, leading to poor conditioning, but 2×2 diagonal blocks have large off-diagonal entries, making these blocks suitable pivots for a block elimination process (for more, and some useful diagrams, see (Bollhöfer et al., 2009, §2.2)).

This preprocessing is usually so effective as to not require any further pivoting (though additional Bunch-Kaufman pivoting is included in implementations, for maximum robustness) and in practice, the algorithm to solve the weighted-matching problem scales linearly in time, providing an extremely efficient method, far more attractive than methods without preprocessing steps.

Once this ordering preprocessing has been performed, a standard fill-reducing ordering may be performed on the full system. Thus, the complete factorization of a matrix A may be represented as

$$\Pi^T \hat{P}^T \hat{D} A \hat{D} \hat{P} \Pi = A', \quad A' = LDL^T + E,$$

$$\Pi^T \hat{P}^T \hat{D} A \hat{D} \hat{P} \Pi = A', \quad A' = LDL^T + E, \tag{12}$$

Where Π is a fill-reducing permutation, $\hat{P}$ and $\hat{D}$ are a permutation and scaling arising from the symmetric maximum weighted matching preprocessing, and E is an error introduced by the incomplete factorization process, which produces the incomplete factors L and D used in the preconditioner.

The ingredients in the preconditioner include the following components, each of which can be addressed separately in software.

- A reordering and scaling preprocessing step to reduce fill and the need for pivoting.
- An additional block-wise[1] fill-reducing reordering
- A factorization stage which computes and stores L and D, with respect to some drop tolerance, estimate of $||L||$, or specified fill pattern.
- A routine to quickly solve $LDL^T x = b$ by (block) forward- and back-substitution.

Despite the sophistication of the algorithms just discussed, practical usage of an ILDL preconditioner, given robust solver software, reduces to specification of only a few parameters; the user typically only needs to understand that there is a tradeoff between fill and the strength of the preconditioner, which they can control with some simple parameters (here, a drop tolerance).

4 Approximate block factorization (ABF) preconditioning

Approximate Block Factorization (ABF) solvers provide a powerful class of methods for the solution of saddle point systems. These solvers define preconditioners by exploiting a block LDU factorization of the saddle point matrix, with respect to the pressure and velocity blocks (Benzi et al., 2005, (5)). Approximately inverting the block-triangular or block-diagonal factors (often with available scalable solvers) provides a natural way to define approximate inverses, constructed from approximate solvers on a single field. For more, see details we refer to Elman et al. (2005).

We choose a particular ABF solver as a representative of this class. In particular, we consider an upper block-triangular preconditioner

$$\begin{bmatrix} K & B \\ & \hat{S} \end{bmatrix}, \tag{13}$$

where $\hat{S}$ is an approximation to the Schur complement S given by $S = -C - B^T K^{-1} B$ (and noting that $C = 0$ in the Stokes case). The approximate solver on the viscous block is a geometric multigrid method, with a direct solve via UMFPACK (Davis, 2004) on the coarse level. Smoothing is accomplished by 8 Chebyshev-Jacobi iterations (Hu et al., 2003), where GMRES is used to estimate the maximum eigenvalue λ_{max} of the preconditioned operator. The Chebyshev polynomial is tuned to the interval $[0.2\lambda_{\text{max}}, 1.1\lambda_{\text{max}}]$. The approximate Schur complement solver $\hat{S}$ is a single application of an ILU preconditioner formed from a ~~scaled~~ $\frac{1}{\lambda}$-weighted pressure mass matrix plus the $(2,2)$ block (which is zero in the case of Stokes); this

[1]meaning the 1×1 and 2×2 blocks as used in the previous step, not any other, physically-inspired blocking

is simple but is known to produce a spectrally-equivalent, hence scalable, preconditioner (Grinevich and Olshanskii, 2009). This preconditioner was chosen based on experience using these solvers for applications in geodynamics, where it has shown to be scalable and efficient. For problems with large non-grid-aligned coefficient jumps, more elaborate Schur complement preconditioners have also been developed in recent years (Elman, 1999; May and Moresi, 2008; Rudi et al., 2015, 2017). The ABF solver chosen here often shows superior performance for all but the smallest problem sizes, but relies on much more machinery set up in the application: the solver is aware of pressure and velocity blocks and a hierarchy of grids, transfer operators, and rediscretized operators. In addition, auxiliary operators must be defined to implement a Schur complement preconditioner, used here and in most competitive ABF solvers.

5 Numerical experiments

To provide a concrete and reproducible set of experiments, we use a $\mathbb{Q}_2 - \mathbb{Q}_1$ (Taylor-Hood) mixed finite element code[2], making use of the PETSc (Balay et al., 2019a, b) library. It solves the Stokes and elasticity systems in the unit square (2D) and cube (3D). We focus on 3D problems, as in the 2D case, sparse direct solution methods are expected to be highly competitive, with time to solution scaling as $O(N^{3/2})$ (as opposed to $O(N^2)$ for 3-dimensional problems) and expected fill scaling as $O(N \log N)$ (as opposed to $O(N^2)$ for 3-dimensional problems) (George, 1973). For Stokes flow tests, free slip boundary conditions are imposed everywhere ($u \cdot n = 0, t \cdot \tau \cdot n = 0$) except the top boundary of the domain, where a free surface is prescribed ($n \cdot \sigma \cdot n = t \cdot \tau \cdot n = 0$); this implies a non-singular system matrix (Elman et al., 2005, Ch.5, p. 215). The elasticity tests also include experiments with non-zero Dirichlet boundary conditions (specified displacements).

Experiments involving both Stokes and elasticity systems are defined by a "multiple inclusion" configuration. That is, the domain is partitioned into a set of N non-overlapping spheres each with radius R. By providing parameters to control the (non-grid-aligned) coefficients (viscosity / Lamé parameters and density) contrast between the N spherical inclusions and the surrounding medium, this model configuration provides a useful way to characterize two major factors which impact solver performance: coefficient jumps across arbitrary interfaces and the geometric complexity of these interfaces. Dealing with these factors is of primary important in designing solvers for realistic Earth science applications. This discontinuous coefficient field is projected onto the quadrature points used to evaluate the bilinear / linear forms required by the finite element method. This projection has the effect of making the coefficient field vary on the length scale of a single finite ~~elements~~element; loosely speaking, this makes the problem "harder" and less amenable to solution with higher-order methods as the mesh is refined, but this is nonetheless consistent with the way that such problems are often solved in practice (Gerya and Yuen, 2003; May et al., 2015).

We compare three solver configurations: GMRES preconditioned with ILDL (see Section 3); sparse direct; and FGMRES preconditioned with ABF (described in Section 4). We do not extensively compare to standard ILU preconditioning, or to ILDL preconditioning without symmetric maximum weighted matching preprocessing, as these preconditioners are very unreliable for indefinite problems (Chow and Saad, 1997). This characteristically poor performance has likely contributed to the fact that

[2]Source code publicly available; see the "Code availability" section at the end of this paper, and the supplement on reproducibility.

incomplete factorization preconditioners for indefinite mechanical problems have not been championed, before this work, as a viable ~~alternative for practical preconditioning~~practical approach, even though software tools have now developed to the point of making them robust options. We note that when using a mixed finite element space with a discontinuous pressure space, along with an appropriate penalty (or bulk viscosity) term C, one can in some cases transform and solve a symmetric positive definite problem, for which incomplete Cholesky preconditioning is effective and robust (Dabrowski et al., 2008).

All linear algebra is dispatched through the PETSc API. We use a wrapper[3] to provide an interface between PETSc and internal functions in ILUPACK[4] which perform symmetric maximum weighted matching permutation and scaling, prior to a factorization step using a block elimination process with a simple threshold-based dropping strategy; entries in the L factors less than a given value (after scaling and permutation) are dropped during the factorization process. Ordering with METIS (Karypis and Kumar, 1998) by nodes, with respect to the blocked system, proved robust and is used everywhere in this work. In these experiments, available multi-level ILDL options did not seem to offer enhanced performance. Note that, as mentioned in Section 1.1, we focus on applications where a given system is solved only once, and hence report solve times which include the setup (factorization) and Krylov (iterative) solve. The setup time is typically the majority of the solve time; Table 1 reports these times directly.

~~There are many packages for the sparse~~ For direct solution of ~~linear systems;~~ sparse linear systems, we choose PARDISO (Schenk and Gärtner, 2004; Schenk and Gärtner, 2006; Kuzmin et al., 2013) ~~based on~~ as a highly-competitive package, for instance as demonstrated in the comparative study of Gould et al. (2007) examining performance respect to total, serial (single-)solve time for symmetric indefinite systems with 10000 or more DOFs. Through a custom interface[5] ~~we use PARDISO (Kuzmin et al., 2013) to provide~~ this provides a direct solver for symmetric indefinite systems, using the same weighted-matching ordering used by ILUPACK and the ILDL preconditioners considered here.

All iterative solves use right-preconditioned GMRES or FGMRES and share a common convergence criterion: a reduction of 10^6 in the true residual 2-norm $\|b-Ax\|_2$, where A is A_{Stokes} or $A_{\text{Elasticity}}$. In practice, Krylov methods which take advantage of symmetric structure, e.g. MINRES or QMR, may be attractive. The choice ~~or norm allows~~ of norm is important because we consider ill-conditioned linear operators for which convergence in a preconditioned norm often fails to imply convergence in the true residual norm. In practice, different norms are usually used. These include preconditioned residual norms or a quasi-norm in the case of QMR (Freund and Nachtigal, 1991).

Most of the computations were performed on single compute node of the Euler II cluster at ETH Zurich. Each compute node is a dual-socket Intel Xeon E5-2680v3 nodes, each with 64 GB of memory. Numerical experiments used a single MPI-rank and a single OpenMP thread. Experiments as reported in Figure 4 were performed on the Leonhard cluster at ETH Zurich, using dual-socket Intel Xeon E5-2697v4 nodes, with 128 GB or more memory. Experiments as reported in Section 5.3 were performed on Piz Daint at the Swiss National Supercomputing Center, using 6 MPI ranks per dual-socket Intel Xeon E5-2695v4 compute node with 64 GB of memory.

[3]Source code publicly available; see the "Code availability" section at the end of this paper, and the supplement on reproducibility.
[4]Free academic licenses available; see the “Code availability" section at the end of this paper, and the supplement on reproducibility.
[5]Source code publicly available; see the "Code availability" section at the end of this paper, and the supplement on reproducibility.

		GMRES(60)/ILDL(1e-3)					PARDISO		FGMRES(30)/ABF			
Els.	DOFs	Fill	Its.	Tot. Time [s]	Setup Time [s]	Mem. [MB]	Time [s]	Mem. [MB]	Lvls.	Its.	Time [s]	Mem. [MB]
8^3	15,468	2.0	14	2.27E+00	1.96E+00	127	1.42E+00	163	2	22	2.46E+00	125
16^3	112,724	2.9	45	~~5.27~~6.18E+01	5.30E+01	851	3.78E+01	1743	3	24	2.71E+01	1104
24^3	368,572	3.7	112	4.77E+02	2.72E+02	3076	2.98E+02	7289	3	19	7.74E+01	2330
32^3	859,812	4.6	226	1.96E+03	1.07E+03	8100	1.55E+03	21856	4	17	2.04E+02	9587
40^3	1,663,244	4.5	420	6.41E+03	2.63E+03	15513	6.67E+03	52376	4	17	3.10E+02	10426
48^3	2,855,668	5.5	568	1.31E+04	9.61E+03	30126	-	-	4	16	6.33E+02	17842
56^3	4,513,884	-	-	-	-	-	-	-	4	15	7.01E+02	28219
64^3	6,714,692	-	-	-	-	-	-	-	5	19	6.28E+03	40519

Table 1. 3D stationary Stokes flow, with 3 denser (relative density 1.2) spherical inclusions of radius 0.1 viscosity ~~$\eta_1 = 10^4$~~ $\eta_1/\eta_0 = 10^4$ times higher than in a surrounding medium ~~of viscosity $\eta_0 = 1$~~in the unit cube. Iteration counts accompany data points in the graph. Missing data correspond to runs which failed due to insufficient available memory. See also Figure 1.

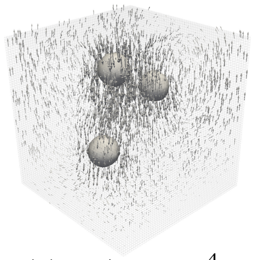
5.1 3D Stokes flow

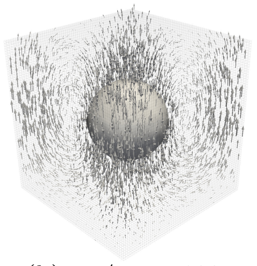
Examples of 3-dimensional Stokes flow in cubic domains, for problem sizes ranging from 8^3 to 64^3 $\mathbb{Q}_2 - \mathbb{Q}_1$ elements, are presented in Table 1 and Figure 1. Here, one can see comparable performance between the direct solve and the ILDL-preconditioned solves, across all the problems tested; however, the ILDL-preconditioned solve requires less memory and has the additional advantage of allowing for a loosening of the solve tolerance if desired. Due to its lower memory requirements, we were able to solve larger problems with the ILDL-preconditioned approach. The ABF solver typically provides the best time to solution, yet lacks robustness with respect to the problem parameters, in addition to relying on much more auxiliary information and many more parameters (in particular, an auxiliary operator for the Schur complement preconditioner, and a grid hierarchy and tuned parameters for the multigrid hierarchy).

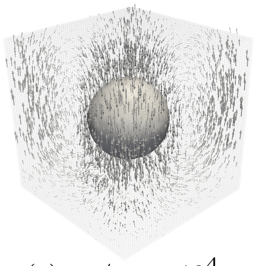
As shown in Figure 1, increasing the number of inclusions degrades the performance of the ABF solver. This trend continues, as additional inclusions are added, until eventually the ABF solver fails to converge~~. ~~; Figure 2 demonstrates ~~the effectof further increasing the number of inclusions~~this effect, with a viscosity contrast of $\eta_1/\eta_0 = 10^6$ (a typical cutoff value for even-higher contrasts arising in geodynamical modeling). Here we can directly observe a regime in which ILDL-preconditioned iterative solves not only provide a simpler alternative to ABF solves, but a more robust one which also outperforms a direct solve in terms of time-to-solution and memory footprint.

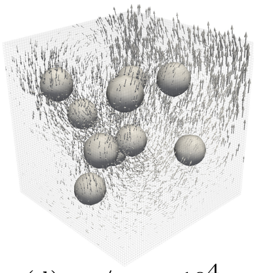
The iterative solver gives the user control over time-memory trade-offs by varying the drop tolerance. Table 2 shows the effect of varying ~~to ~~the drop tolerance with a 32^3 element experiment as pictured in Figure 1(a). Comparable times to solution are observed over a fairly broad range of drop tolerances, and experiments like this lead us to recommend default drop tolerances in the $10^{-3} - 10^{-4}$ range[6].

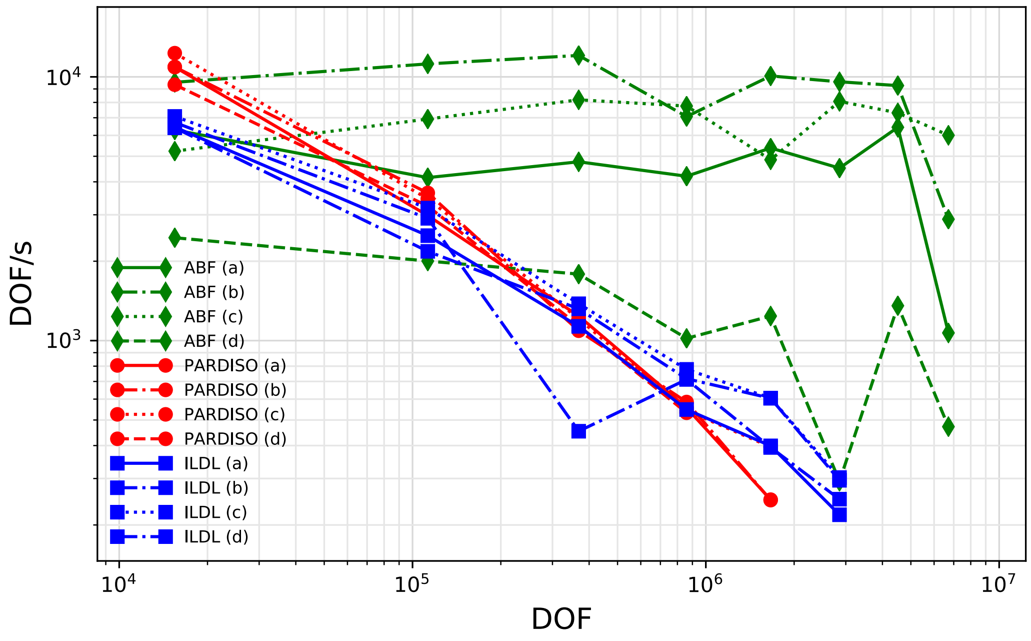
[6]~~Note that dropping ~~Dropping is performed on a ~~system which has already been ~~scaled and permuted system, ~~hence ~~as in Eq. (12). Hence, these values have meaning independent of the original scaling of the system.


(a) $\eta_1/\eta_0 = 10^4$.


(b) $\eta_1/\eta_0 = 100$.


(c) $\eta_1/\eta_0 = 10^4$.


(d) $\eta_1/\eta_0 = 10^4$.



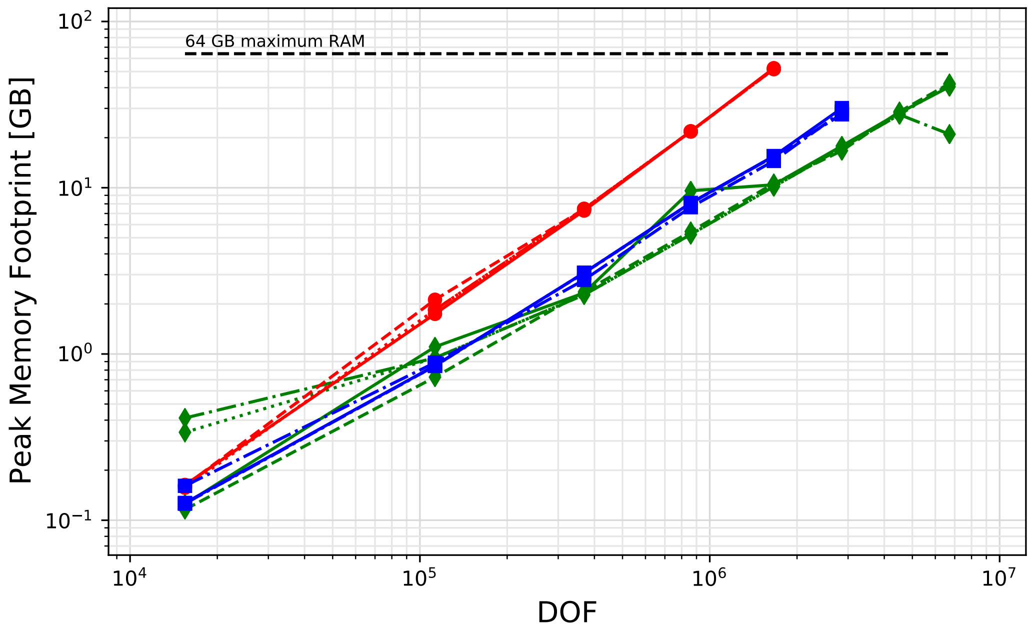


Figure 1. Additional experiments, corresponding to coefficient structures labelled (a)-(d), analogous to those in Table 1. Inclusion radii are 0.25 for single inclusions, and 0.1 otherwise, in the unit cube. Solver performance is assessed in terms of degrees of freedom solved to the prescribed tolerance (10^{-6} relative error in the true residual norm) over the solution time, and by peak memory footprint.

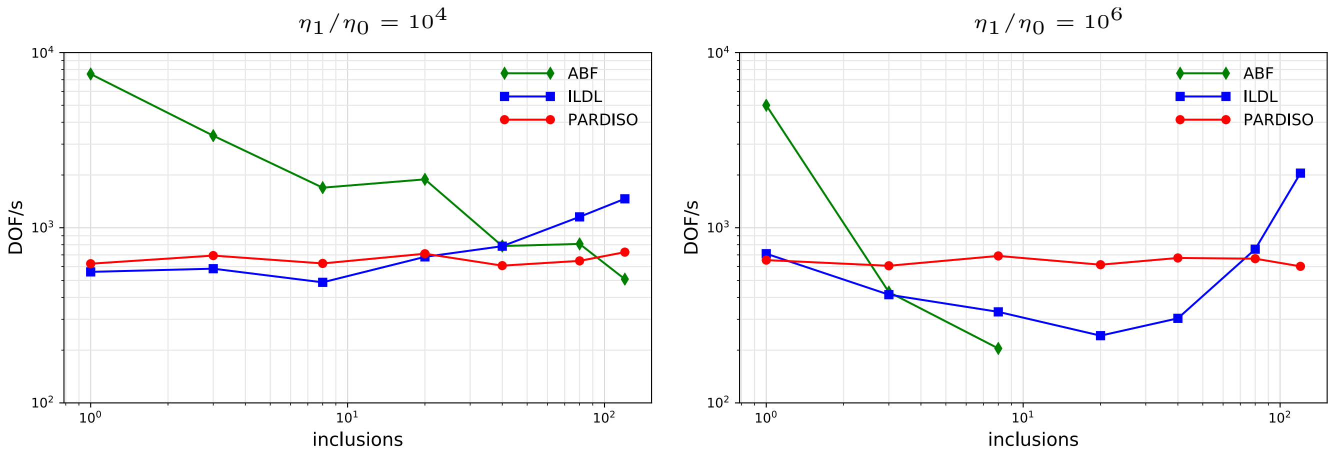


Figure 2. The effect of increasing the number of viscous inclusions on the effectiveness of solvers, for a 32^3 element simulation. Up to 140 non-overlapping inclusions of radius 0.05 are placed randomly in the unit cube. ILDL preconditioning becomes more competitive for more challenging structures, showing much greater robustness to viscosity structure than ABF solvers while maintaining a lower memory footprint than direct solution. Missing ABF data indicate that the solver failed to converge. Memory footprints are similar to those shown in Figure 1; the direct solve (red) requires approximately $4\times$ as much memory as the ILDL-preconditioned solve.

drop tol.	**fill**	**Solve Time [s]**	**Iterations**	**Max. Mem. [MB]**
$1 \cdot 10^{-5}$	17	1.0696e+04	10	21072
$5 \cdot 10^{-5}$	13	6.2200e+03	22	16582
$1 \cdot 10^{-4}$	11	4.0494e+03	35	14321
$5 \cdot 10^{-4}$	6.1	1.9075e+03	134	9666
$1 \cdot 10^{-3}$	4.6	1.6031e+03	226	8433
$5 \cdot 10^{-3}$	2.1	1.1421e+03	652	6585
$1 \cdot 10^{-2}$	1.4	1.1657e+03	1015	6479
$5 \cdot 10^{-2}$	0.39	-	(>10000)	-

Table 2. The effect of varying the drop tolerance parameter (see Section 3) for a 32^3 element system and right-preconditioned GMRES(60)/ILDL solver as in Table 1. Loosening the drop tolerance increases the iteration count and reduces the fill and hence memory footprint.

5.2 3D elasticity experiments

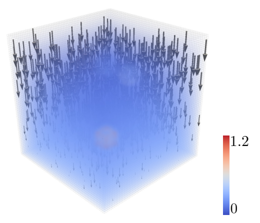
Figure 3 shows the results of similar experiments to those in Section 5.1 with spherical inclusions with different material properties, here varying λ inside the inclusions, creating areas of near-incompressibility. This corresponds to a very high Poisson ratio $\nu = \frac{\lambda}{2(\lambda+\mu)}$ close to $\frac{1}{2}$, the challenging case where standard (non-mixed) finite element methods tend to exhibit locking. The effect of varying this parameter is markedly different to that of varying η in the Stokes case (analogous to μ in the Lamé case); performance is not as dependent on this parameter for any of the solvers, and the attained ~~dof~~DOF/s for the ILDL solver degrades much more slowly. Again, we observe marked reduction in memory consumption for the ILDL solver, as compared to the direct solver, along with a notable performance gain from using ILDL preconditioning, relative to direct solution. ABF preconditioning is still advantageous, but may not be available in practice and may require an expert to set up. ~~We note that the~~ The ABF preconditioner is not identical to that used for the Stokes problem, with μ substituted for η~~; to achieve good performance, the Schur complement preconditioner is different, due to the nonzero (2,2) block in the elasticity system. That is, to~~. To obtain a spectrally equivalent preconditioner, we ~~needed to build a preconditioner~~ build $\hat{S}$ (Cf. equation 13) ~~as an approximate inverse of $M_\mu - C$, where M_μ is a finite element mass matrix weighted by μ and C is the term (depending on λ as~~ to include the $-C$ term in the Schur complement, where $-C$ arises from the finite element discretization of the $-\frac{1}{\lambda}p$ term in Eq. (9)~~).~~. Thus, $\hat{S}$ is a $-\left(\frac{1}{\mu}+\frac{1}{\lambda}\right)$-weighted pressure mass matrix.

The experiments in Figure 3 are chosen to emphasize the similarity of the elasticity and Stokes problems, yet elasticity problems are commonly prescribed with non-zero boundary displacements, amounting to non-zero Dirichlet boundary conditions. Figure 4 shows a similar experiment using a scenario which is perhaps more typical in applications. Inhomogeneous Dirichlet boundary conditions are used to specify non-zero normal displacements on two opposing boundaries; the top boundary is stress-free and the remaining boundaries are free-slip. The results parallel those in the previous set of experiments; the more-invasive ABF solver produces the fastest solution, but preconditioning with ILDL and symmetric weighted matching preprocessing gives a vastly superior option to a direct solve, while using no additional information beyond the specification of a single drop tolerance. Memory usage is very similar to the plots shown in Figure 3.

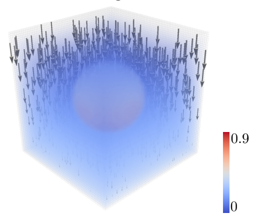
5.3 Using ILDL within a parallel preconditioner

An obvious limitation of the results presented thus far, and of the particular implementation of the ILDL decomposition that we have employed, is that they have focused on single-process (i.e. “sequential”) usage. However, most scientific computation is now performed with some degree of multi-process (or multi-thread) parallelism.

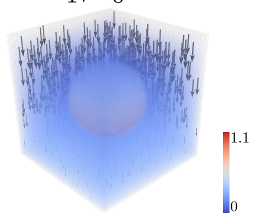
A well-known and often-used approach to extend a sequential preconditioner to a parallel preconditioner is to employ a domain decomposition method (Smith et al., 2004) wherein the computational domain is decomposed into possibly-overlapping patches where local preconditioners can be applied before the results are used to update the global solution. The simplest such preconditioner is the block Jacobi method, with non-overlapping subdomains, and a natural extension is the Additive Schwarz Method (ASM), wherein overlapping subdomains, here defined in terms of finite elements, are used to define subsolves.



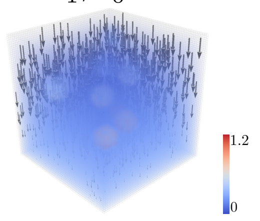
(a) 3 inclusions, $\lambda_1/\lambda_0 = 10^4$



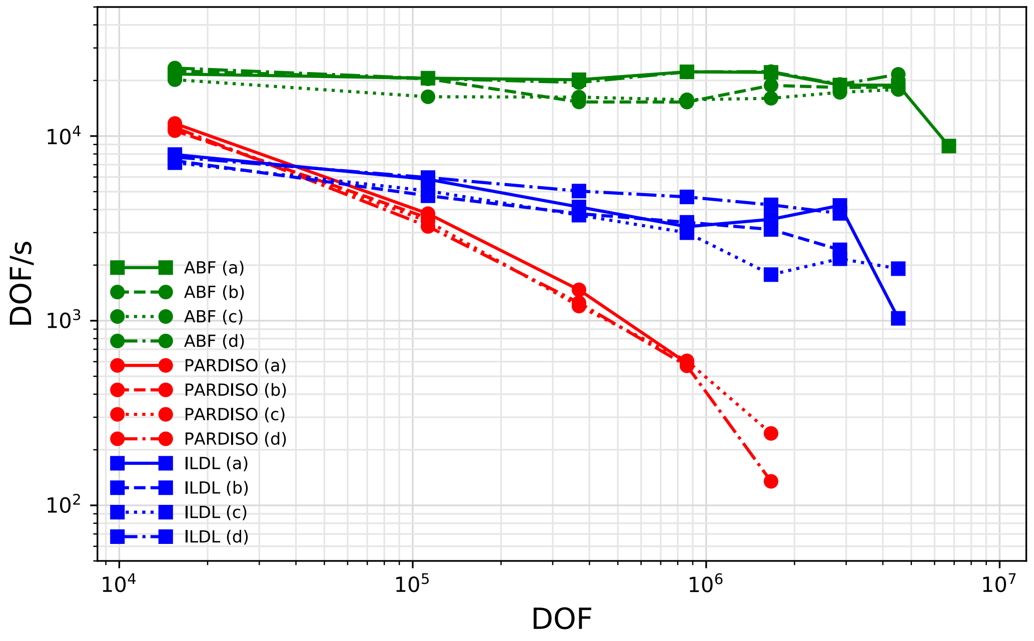
(b) 1 inclusion, $\lambda_1/\lambda_0 = 10^2$



(c) 1 inclusion, $\lambda_1/\lambda_0 = 10^4$



(d) 8 inclusions, $\lambda_1/\lambda_0 = 10^4$



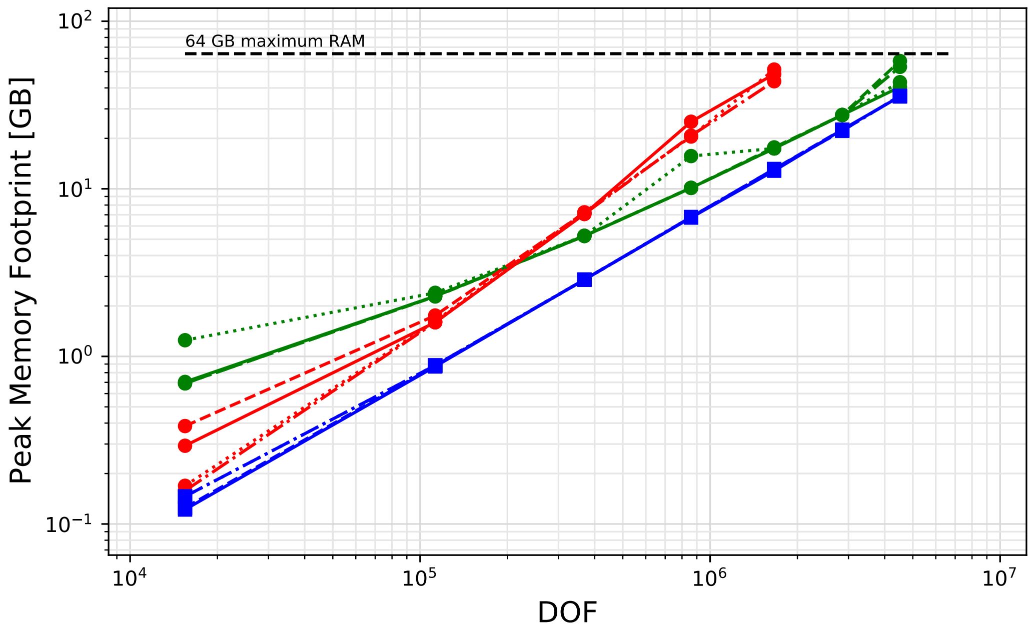
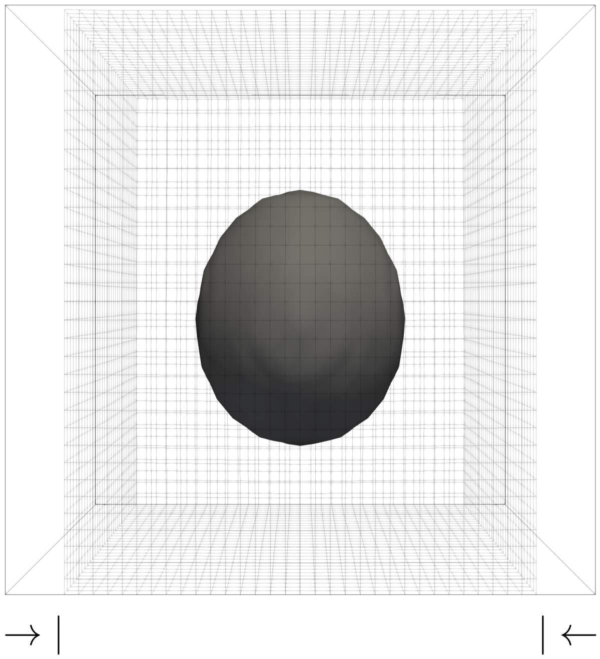


Figure 3. Experiments showing performance of ILDL preconditioning for the elasticity system, holding $\mu_0 = \mu_1 = 1, \lambda_0 = 1$ constant and varying λ_1 inside inclusions or radius 0.25 (single inclusions) or 0.1. Arrows show displacement, and the pressure field is plotted volumetrically.



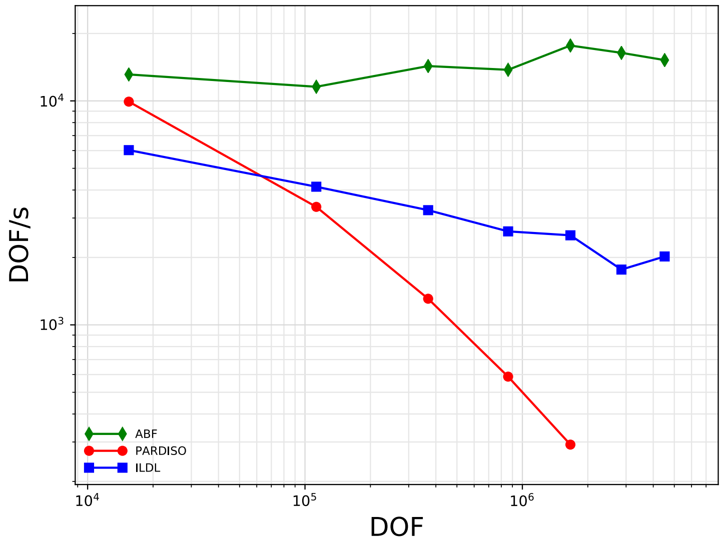


Figure 4. An experiment showing the performance of ILDL preconditioning for an elasticity problem with a heterogeneous medium in compression; the outer box shows the reference (undeformed) state and the wire mesh shows the deformed state. The surrounding medium has a Poisson ratio of $1/3$ (~~$\lambda = 2, \mu = 1$~~$\lambda = 2,\ \mu = 1$), and the originally-spherical (radius 0.25) inclusion is almost incompressible, with a Poisson ratio greater than 0.4999 (~~$\lambda = 10^4, \mu = 1$~~$\lambda = 10^4,\ \mu = 1$). Boundary conditions are free-slip everywhere except the top, which is stress free. The results show that ILDL preconditioning offers substantially better-scaling performance (with a lower memory footprint) than a direct solver, without the auxiliary information, implementation, and tuning required for the even-better-scaling ABF solver. These experiments were run on a slightly different cluster than the preceding ones (see Section 5), so solve times are not directly comparable.

ILDL preconditioning can be used to provide approximate inverses to local subproblems, much as ILU preconditioning is commonly used in the corresponding positive-definite case. We note that this block Jacobi/ILU preconditioning is practical useful for a wide range of problems despite, like ILU itself, it not being perfectly scalable or robust.

This is possible by leveraging the same software wrappers used in the sequential experiments in this paper, and indeed within the code used here, amounts simply to ~~specifiying~~ specifying a few command line options. As a proof of concept, table 3 compares iteration counts and time-to-solution solving a Stokes problem with a block Jacobi or ASM preconditioner, with various choice of sub-preconditioner. As in the rest of this work, comparison is made with a well-tuned ABF solver. [7] No attempt has been made to optimize the subdomain solvers here - the drop tolerance was simply adopted from the sequential case. For the isoviscous case shown, the block Jacobi method and ASM method are comparable, but when a viscosity jump is added, the ASM solver can still converge, albeit slowly, while the block Jacobi approach becomes much slower. These simple

[7] And note that the largest ABF solve here required the use of PCTELESCOPE (May et al., 2016) to define the coarsest grid on a subset of ranks, another wrinkle in the effective use of multigrid solvers.

			GMRES(60)/Block Jacobi/ILDL(1e-3)		**GMRES(60)/ASM/ILDL(1e-3)**		**FGMRES(30)/ABF**		
η_1/η_0	Els.	MPI Ranks	Its.	Time [s]	Its.	Time [s]	Lvls.	Its.	Time [s]
1	32^3	8	219	8.0564e+01	84	8.2991e+01	4	9	1.2685e+01
	48^3	27	693	1.7149e+02	586	2.4185e+02	4	9	1.5954e+01
	64^3	64	1310	2.6762e+02	1843	5.2592e+02	5	9	1.3151e+01
	96^3	216	4668	7.6356e+02	7903	1.6136e+03	5	9	1.6056e+01
	128^3	125			4613	5.9810e+03			
		216			6587	3.6521e+03			
		512	6126	1.1899e+03	15396	4.8914e+03	5	9	1.3296e+01
		729			19387	4.3029e+03			
10^2	32^3	8	709	1.6289e+02	155	9.3547e+01	4	12	2.1862e+01
	48^3	27	14892	2.5563e+03	832	2.8261e+02	4	12	2.6607e+01
	64^3	64	-	$> 1.4e+04$	17636	6.9489e+03	5	14	2.1098e+01
	96^3	216	-	$> 2.8e+04$	$> 10^5$	2.5548e+04	5	13	2.3065e+01
	128^3	512	-	$> 2.8e+04$	$> 10^5$	2.6805e+04	5	14	2.3120e+01

Table 3. Data for MPI parallel solves, using a block Jacobi and 1-element overlapping ASM preconditioners, each with ILDL subdomain preconditioners, compared with an ABF solver as used throughout this paper. These solves correspond to the single-sinker Stokes case as in Figure 1(a)-(b), but now with larger problem sizes made possible by the distributed memory-environment. These show the feasibility of using incomplete factorizations to create an simple-to-apply parallel preconditioner for symmetric saddle point systems, albeit one which shows the same non-optimal scaling and parameter sensitivity familiar from the use of subdomain ILU or ICC preconditioners in parallel for definite systems. The problem sizes and number of ranks are chosen to demonstrate weak scaling (constant problem size per rank), and a strong scaling test is shown for one problem. Note that these experiments were conducted on a different cluster than those in previous sections (see Section 5), so times to solution are not directly comparable.

experiments demonstrate that ILDL preconditioners can be used within parallel preconditioners to solve problems too large for a single computational node. As in the sequential case, one can sacrifice some performance, with respect to a complex solver relying on more machinery and great expertise in tuning (ABF) to be able to quickly use a simpler (from the user's perspective) and more widely applicable solver.

6 Conclusions

The efficient solution of symmetric indefinite linear systems is an important task in many physical ~~modelling~~ modeling applications in the Earth sciences and beyond, particularly solving PDE in mixed formulations. Approximate Block Factorization (ABF) preconditioned solvers (or other scalable options) including nested multilevel solves are well-known to be efficient for sufficiently-large problems, but require invasive code modifications and expertise in implementation and tuning, so might not be practical to implement or evaluate. Further, these solvers may not be robust to challenging coefficient structures. The ABF solver we used for comparison here, for example, performs extremely well for small numbers of viscous inclusions but fails to converge for larger numbers. Advances in algorithms and software for direct solution of symmetric indefinite systems have, in recent years, brought direct solution for these systems to a level of performance and robustness on par with their counterparts for definite systems. These advances carry over to incomplete factorization preconditioning, though this is much less well-known; this work presents much-needed results on the effectiveness of these solvers in challenging parameter regimes. The Krylov methods studied here, using ILDL preconditioning with a symmetric maximum weighted matching preprocessing step, require only a single drop tolerance parameter for the preconditioner, and can be useful across problem types, as seen here between Stokes and elasticity, and indeed even across "matrix market"-style corpora (Hagemann and Schenk, 2006; Greif et al., 2015). They also show useful robustness to viscosity structure, outperforming our representative ABF solver for larger numbers of viscous inclusions. Further, ILDL-preconditioned Krylov methods can be a preferable choice to direct solves: one trades some parameter selection and less-robust performance for a large reduction in memory footprint and often extra performance.

The results in this paper show that if one is employing a direct solver for a symmetric, indefinite problem, such as a Stokes or elasticity problem, an ILDL-preconditioned iterative solver is worth investigating. The preconditioner requires only an assembled operator and can be quickly used in situations that an ABF solver must be arduously selected, integrated, and tuned, and can offer greater robustness to coefficient structure. An ILDL-preconditioned iterative solver typically remains competitive or even superior to direct solution, in terms of ~~dof~~DOF/s computed for larger problems while using $3\times - 5\times$ less memory. This alternative can be investigated quickly as only an assembled operator and one or two parameters need be provided.

~~We~~ To conclude, we mention avenues for further development. Firstly, we have focused on single-level ILDL preconditioning, but ILUPACK includes a multilevel ILDL preconditioner (Bollhöfer and Saad, 2006; Schenk et al., 2008), available through the same wrappers, and PARDISO (Schenk and Gärtner, 2004; Kuzmin et al., 2013) also includes a multi-recursive iterative solver which uses multi-level ILDL preconditioning. These techniques are related to an algebraic multigrid (AMG) approach; further investigation is warranted of these and other highly-automated AMG approaches (Metsch, 2013) which may provide the scalability associated with multilevel methods while retaining the robustness and ease of use associated with factorization-based approaches. ~~We~~

Secondly, we have focused on sequential computation and application of ILDL factorizations, and shown how these may be extended to the parallel case by using simple domain-decomposition-based preconditioners.

However, recent work has shown the promise of computing and applying ~~incomplete LDL^T~~ ILDL factorizations directly in modern parallel environments, taking advantage of multiple threads, distributed memory subdomains (MPI ranks), and/or GPUs (Aliaga et al., 2014, 2016a, b, 2017; Bollhöfer et al., 2019b). As these developments make their way into software packages, these algorithms will become even more attractive for applications in the Earth sciences and beyond.

Code availability. Our solvers utilize functionality from ILUPACK [8], PARDISO [9], and PETSc [10]. PETSc is open-source under a BSD-2 license; ILUPACK and PARDISO are closed source and offer complimentary academic licenses.

[8]http://ilupack.tu-bs.de

[9]www.pardiso-project.org

[10]www.mcs.anl.gov/petsc

PETSc represents the highest level within our solver stack; all underlying solver implementations are utilized through PETSc function calls (e.g. `KSPSolve()`). To support this, we provide a public, open source wrapper around ILUPACK so that it can be used as a preconditioner (`PC`) implementation within PETSc [11]. Through a custom interface (again public and open source) we use PARDISO to provide a direct solver for symmetric indefinite systems, using the same weighted-matching ordering [12]. The code which performs the discretization of the Stokes and elasticity problems, configures the solvers and generates the post-processed flow/displacements fields is publicly available, open source[13]. Also see the supplement to this paper, which provides additional instructions and details to reproduce, extend, and apply the experiments and tools discussed above.

Author contributions. All authors contributed collaboratively to the development of the project. P. Sanan, D. May and O. Schenk contributed to the writing of the manuscript. P. Sanan and D. May conceived and designed the scope and experiments presented, including the demonstration code. M. Bollhöfer provided code and support to interface with ILUPACK.

Competing interests. We declare there are no competing or conflicts of interest.

Acknowledgements. P. Sanan acknowledges financial support from the Swiss University Conference and the Swiss Council of Federal Institutes of Technology through the Platform for Advanced Scientific Computing (PASC) program. D. May acknowledges financial support from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC Grant Agreement Number 279925 and the Alfred P. Sloan Foundation through the Deep Carbon Observatory (DCO) Modeling and Visualization Forum. P. Sanan acknowledges Radim Janalik who helped developed the PETSc interface to PARDISO.

[11] http://bitbucket.org/psanan/PCILUPACK
[12] https://bitbucket.org/psanan/petsc/branch/psanan/pardiso-3.12.4
[13] http://bitbucket.org/psanan/exsaddle

References

Aliaga, J. I., Bollhöfer, M., Dufrechou, E., Ezzatti, P., and Quintana-Ortí, E. S.: Leveraging Data-Parallelism in ILUPACK using Graphics Processors, in: 2014 IEEE 13th International Symposium on Parallel and Distributed Computing, pp. 119–126, https://doi.org/10.1109/ISPDC.2014.19, 2014.

Aliaga, J. I., Badia, R. M., Barreda, M., Bollhöfer, M., Dufrechou, E., Ezzatti, P., and Quintana-Ortí, E. S.: Exploiting Task and Data Parallelism in ILUPACK's Preconditioned CG Solver on NUMA Architectures and Many-Core Accelerators, Parallel Computing, 54, 97–107, 2016a.

Aliaga, J. I., Bollhöfer, M., Dufrechou, E., Ezzatti, P., and Quintana-Ortí, E. S.: A Data-Parallel ILUPACK for Sparse General and Symmetric Indefinite Linear Systems, in: European Conference on Parallel Processing, pp. 121–133, Springer, 2016b.

Aliaga, J. I., Barreda, M., Flegar, G., Bollhöfer, M., and Quintana-Ortí, E. S.: Communication in Task-Parallel ILU-Preconditioned CG Solvers using MPI+ OmpSs, Concurrency and Computation: Practice and Experience, 29, e4280, 2017.

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Karpeyev, D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H.: PETSc Web page, https://www.mcs.anl.gov/petsc, https://www.mcs.anl.gov/petsc, 2019a.

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Karpeyev, D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H.: PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 3.12, Argonne National Laboratory, https://www.mcs.anl.gov/petsc, 2019b.

Benzi, M.: Preconditioning Techniques for Large Linear Systems: A Survey, Journal of Computational Physics, 182, 418–477, https://doi.org/10.1006/jcph.2002.7176, 2002.

Benzi, M. and Wathen, A. J.: Some Preconditioning Techniques for Saddle Point Problems, in: Model Order Reduction: Theory, Research Aspects and Applications, edited by Schilders, W. H., van der Vorst, H. A., and Rommes, J., pp. 195–211, Springer Berlin Heidelberg, 2008.

Benzi, M., Golub, G. H., and Liesen, J.: Numerical Solution of Saddle Point Problems, Acta Numerica, 14, 1–137, https://doi.org/10.1017/S0962492904000212, 2005.

Bollhöfer, M. and Saad, Y.: Multilevel Preconditioners Constructed from Inverse-Based ILUs, SIAM Journal on Scientific Computing, 27, 1627–1650, 2006.

Bollhöfer, M., Grote, M., and Schenk, O.: Algebraic Multilevel Preconditioner for the Helmholtz Equation in Heterogeneous Media, SIAM J. Sci. Comput., 31, 2009.

Bollhöfer, M., Eftekhari, A., Scheidegger, S., and Schenk, O.: Large-Scale Sparse Inverse Covariance Matrix Estimation, SIAM Journal on Scientific Computing, 41, A380–A401, 2019a.

Bollhöfer, M., Schenk, O., and Verbosio, F.: High Performance Block Incomplete LU Factorization, 2019b.

Brezzi, F. and Fortin, M.: Mixed and Hybrid Finite Element Methods, Springer, New York, https://doi.org/10.1007/978-1-4612-3172-1, 1991.

Bunch, J. R. and Kaufman, L.: Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems, Mathematics of Computation, 31, 163–179, 1977.

Chow, E. and Saad, Y.: Experimental Study of ILU Preconditioners for Indefinite Matrices, Journal of Computational and Applied Mathematics, 86, 387–414, 1997.

Dabrowski, M., Krotkiewski, M., and Schmid, D. W.: MILAMIN: MATLAB-Based Finite Element Method Solver for Large Problems, Geochemistry, Geophysics, Geosystems, 9, 2008.

Davis, T. A.: Algorithm 832: UMFPACK V4. 3—An Unsymmetric-Pattern Multifrontal Method, ACM Transactions on Mathematical Software (TOMS), 30, 196–199, 2004.

Davis, T. A., Rajamanickam, S., and Sid-Lakhdar, W. M.: A Survey of Direct Methods for Sparse Linear Systems, Acta Numerica, 25, 383–566, https://doi.org/10.1017/S0962492916000076, 2016.

Duff, I., Gould, N., Reid, J., Scott, J., and Turner, K.: The Factorization of Sparse Symmetric Indefinite Matrices, JIMA, 11, 181–204, 1991.

Duff, I. S. and Koster, J.: The Design and Use of Algorithms for Permuting Large Entries to the Diagonal of Sparse Matrices, SIAM Journal on Matrix Analysis and Applications, 20, 889–901, https://doi.org/doi:10.1137/S0895479897317661, 1999.

Duff, I. S. and Pralet, S.: Strategies for Scaling and Pivoting for Sparse Symmetric Indefinite Problems, SIMAX, 27, 313–340, 2005.

Elman, H. C.: Preconditioning for the Steady-State Navier-Stokes Equations with Low Viscosity, SIAM J. Sci. Comput., 20, 1299–1316, https://doi.org/10.1137/S1064827596312547, 1999.

Elman, H. C., Silvester, D. J., and Wathen, A. J.: Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics, Numerical mathematics and scientific computation, Oxford University Press, Oxford, New York, 2005.

Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., and Rixen, D.: FETI-DP: A Dual–Primal Unified FETI Method-Part I: A Faster Alternative to the Two-Level FETI Method, International Journal for Numerical Methods in Engineering, 50, 1523–1544, 2001.

Freund, R. W. and Nachtigal, N. M.: QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems, Numerische Mathematik, 60, 315–339, 1991.

George, A.: Nested Dissection of a Regular Finite Element Mesh, SIAM J. Numer. Anal., 10, 1973.

Gerya, T. V. and Yuen, D. A.: Characteristics-Based Marker-in-Cell Method with Conservative Finite-Differences Schemes for Modeling Geological Flows with Strongly Variable Transport Properties, Physics of the Earth and Planetary Interiors, 140, 293–318, 2003.

Gould, N. I. M., Scott, J. A., and Hu, Y.: A Numerical Evaluation of Sparse Direct Solvers for the Solution of Large Sparse Symmetric Linear Systems of Equations, ACM Transactions on Mathematical Software, 33, 10–es, https://doi.org/10.1145/1236463.1236465, 2007.

Greif, C., He, S., and Liu, P.: SYM-ILDL: Incomplete LDL^T Factorization of Symmetric Indefinite and Skew-Symmetric Matrices, CoRR, abs/1505.07589, 2015.

Grinevich, P. P. and Olshanskii, M. A.: An Iterative Method for the Stokes-Type Problem with Variable Viscosity, SIAM Journal on Scientific Computing, 31, 3959–3978, 2009.

Gupta, A. and Ying, L.: On Algorithms for Finding Maximum Matchings in Bipartite Graphs (RC 21576), Tech. rep., IBM Research, 1999.

Gupta, A., Karypis, G., and Kumar, V.: Highly Scalable Parallel Algorithms for Sparse Matrix Factorization, IEEE Transactions on Parallel and Distributed systems, 8, 502–520, 1997.

Hagemann, M. and Schenk, O.: Weighted Matchings for Preconditioning Symmetric Indefinite Linear Systems, SIAM J. Sci. Comput., 28, 403–420, 2006.

Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., Salinger, A. G., Thornquist, H. K., Tuminaro, R. S., Willenbring, J. M., Williams, A., and Stanley, K. S.: An Overview of the Trilinos Project, ACM Trans. Math. Softw., 31, 397–423, https://doi.org/http://doi.acm.org/10.1145/1089014.1089021, 2005.

Hu, J., Tuminaro, R., Adams, M. F., and Brezina, M.: Parallel Multigrid Smoothing: Polynomial versus Gauss-Seidel, Journal of Computational Physics, 188, 593–610, 2003.

Isaac, T., Stadler, G., and Ghattas, O.: Solution on Nonlinear Stokes Equations Discretized by High-Order Finite Elements on Nonconforming and Anisotropic Meshes, with Application to Ice Sheet Dynamics, SIAM J. Sci. Comput., 37, 804–833, 2015.

Karypis, G. and Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs, SIAM Journal on Scientific Computing, 20, 359–392, 1998.

Kuzmin, A., Luisier, M., and Schenk, O.: Fast Methods for Computing Selected Elements of the Greens Function in Massively Parallel Nanoelectronic Device Simulations, in: Euro-Par 2013 Parallel Processing, edited by Wolf, F., Mohr, B., and Mey, D., vol. 8097 of *Lecture Notes in Computer Science*, pp. 533–544, Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-40047-6_54, 2013.

Laird, A. L. and Giles, M.: Preconditioned Iterative Solution of the 2D Helmholtz Equation, Tech. rep., University of Oxford, http://eprints.maths.ox.ac.uk/1216/, 2002.

Li, X. S. and Demmel, J. W.: SuperLU_DIST: A Scalable Distributed-Memory Sparse Direct Solver for Unsymmetric Linear Systems, ACM Transactions on Mathematical Software (TOMS), 29, 110–140, https://doi.org/10.1145/779359.779361, 2003.

Loghin, D. and Wathen, A. J.: Analysis of Preconditioners for Saddle-Point Problems, SIAM Journal on Scientific Computing, 25, 2029–2049, 2004.

MATLAB: 9.6 (R2019a), The MathWorks Inc., Natick, Massachusetts, 2019.

May, D. A. and Moresi, L.: Preconditioned Iterative Methods for Stokes Flow Problems Arising in Computational Geodynamics, Physics of the Earth and Planetary Interiors, 171, 33–47, https://doi.org/10.1016/j.pepi.2008.07.036, 2008.

May, D. A., Brown, J., and Le Pourheit, L.: pTatin3D : High-Performance Methods for Long-Term Lithospheric Dynamics, in: SC14, pp. 274–284, https://doi.org/10.1109/SC.2014.28, 2014.

May, D. A., Brown, J., and Le Pourhiet, L.: A Scalable, Matrix-Free Multigrid Preconditioner for Finite Element Discretizations of Heterogeneous Stokes Flow, Computer Methods in Applied Mechanics and Engineering, 290, 496–523, https://doi.org/10.1016/j.cma.2015.03.014, 2015.

May, D. A., Sanan, P., Rupp, K., Knepley, M. G., and Smith, B. F.: Extreme-Scale Multigrid Components Within PETSc, in: PASC '16: Proceedings of the Platform for Advanced Scientific Computing Conference, https://doi.org/10.1145/2929908.2929913, 2016.

Metsch, B.: Algebraic Multigrid (AMG) for Saddle Point Systems, Ph.D. thesis, Rheinischen Friedrich-Wilhelms-Universität Bonn, 2013.

Mills, R. T., Lu, C., Lichtner, P. C., and Hammond, G. E.: Simulating Subsurface Flow and Transport on Ultrascale Computers using PFLOTRAN, Journal of Physics: Conference Series, 78, 012 051, https://doi.org/10.1088/1742-6596/78/1/012051, 2007.

Munkres, J.: Algorithms for the Assignment and Transportation Problems, Journal of the Society for Industrial and Applied Mathematics, 5, 32–38, https://doi.org/10.1137/0105003, http://dx.doi.org/10.1137/0105003, 1957.

Olschowka, M. and Neumaier, A.: A New Pivoting Strategy for Gaussian Elimination, Linear Algebra and Its Applications, 240, 131–151, https://doi.org/10.1016/0024-3795(94)00192-8, 1996.

Rudi, J., Ghattas, O., Malossi, A. C. I., Isaac, T., Stadler, G., Gurnis, M., Staar, P. W. J., Ineichen, Y., Bekas, C., and Curioni, A.: An Extreme-Scale Implicit Solver for Complex PDEs, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15, pp. 1–12, https://doi.org/10.1145/2807591.2807675, 2015.

Rudi, J., Stadler, G., and Ghattas, O.: Weighted BFBT Preconditioner for Stokes Flow Problems with Highly Heterogeneous Viscosity, SIAM Journal on Scientific Computing, 39, S272–S297, 2017.

Saad, Y.: ILUT: A Dual Threshold Incomplete LU Factorization, Numerical Linear Algebra with Applications, 1, 387–402, https://doi.org/10.1002/nla.1680010405, 1994.

Saad, Y.: Iterative Methods for Sparse Linear Systems, SIAM, 2nd edn., https://doi.org/10.1016/S1570-579X(01)80025-2, 2003.

Schenk, O. and Gärtner, K.: Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO, Future Generation Computer Systems, 20, 475–487, 2004.

Schenk, O. and Gärtner, K.: On Fast Factorization Pivoting Methods for Sparse Symmetric Indefinite Systems, Electronic Transactions on Numerical Analysis, 23, 158–179, 2006.

Schenk, O., Bollhöfer, M., and Römer, R. A.: On Large-Scale Diagonalization Techniques for the Anderson Model of Localization, SIAM review, 50, 91–112, 2008.

Scott, J. and Tůma, M.: On Signed Incomplete Cholesky Factorization Preconditioners for Saddle-Point Systems, SIAM Journal on Scientific Computing, 36, A2984–A3010, https://doi.org/10.1137/140956671, 2014.

Smith, B., Bjorstad, P., and Gropp, W.: Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 2004.

Wathen, A. J.: Preconditioning, Acta Numerica, 24, 329–376, https://doi.org/10.1017/S0962492915000021, 2015.

Wubs, F. W. and Thies, J.: A Robust Two-Level Incomplete Factorization for (Navier-)Stokes Saddle Point Matrices, SIAM Journal on Matrix Analysis and Applications, 32, 1475–1499, https://doi.org/10.1137/100789439, 2011.