COMPARISON BETWEEN ALGEBRAIC AND MATRIX-FREE GEOMETRIC MULTIGRID FOR A STOKES PROBLEM ON ADAPTIVE MESHES WITH VARIABLE VISCOSITY

THOMAS C. CLEVENGER
tcleven@clemson.edu, Clemson University

TIMO HEISTER
heister@sci.utah.edu, University of Utah and Clemson University

Abstract. Problems arising in Earth’s mantle convection involve finding the solution to Stokes systems with large viscosity contrasts. These systems contain localized features which, even with adaptive mesh refinement, result in linear systems that can be on the order of 100+ million unknowns. One common approach for preconditioning to the velocity block of these systems is to apply an Algebraic Multigrid (AMG) v-cycle (as is done in the ASPECT software, for example), however, with AMG, robustness can be difficult with respect to problem size and number of parallel processes. Additionally, we see an increase in iteration counts with adaptive refinement when using AMG. In contrast, the Geometric Multigrid (GMG) method, by using information about the geometry of the problem, should offer a more robust option.

Here we present a matrix-free GMG v-cycle which works on adaptively refined, distributed meshes, and we will compare it against the current AMG preconditioner (Trilinos ML) used in the ASPECT[2] software. We will demonstrate the robustness of GMG with respect to problem size and show scaling up to 24576 cores and 2.2B unknowns. All computations are run using the open source, finite element library deal.II.[1]

1. Introduction

The major bottleneck of computations of processes in Earth’s mantle convection is the solution of Stokes systems for velocity and pressure over a large physical domain. These systems often have large variation in their coefficients, as well as highly localized features requiring adaptive mesh refinement to yield high enough resolution while still being computationally feasible to solve. Even still, there is a desire to solve on meshes containing 100M to over 1B unknowns, which the current state-of-the-art methods are not equipped to handle.

The current state-of-the-art solvers often rely on Algebraic multigrid methods when solving the elliptic problems that occur when preconditioning Stokes finite element discretizations (see, e.g., the works of Geenen et al.[9] and Kronbichler et al.[13]). While these methods can be very powerful for smaller problems, they tend to deteriorate with highly adaptive meshes and when distributing the problem over a large number of processors. Also, these methods require the storing of matrices which can itself be a major bottleneck in finite element computations. One way to attack these problems is by switching to a geometric multigrid (or

Key words and phrases. geometric multigrid, matrix-free finite elements, parallel computing, adaptive mesh refinement, Stokes equations.
These methods have the advantage that, by using geometric information about the problem, there should be less deterioration with highly adaptive meshes. They also allow for the possibility of matrix-free matrix-vector operations.

Here we will present a comparison of a Stokes application using the geometric multigrid method presented in the work of Clevenger et al.\cite{6} with the current algebraic-based method described in the work of Kronbichler et al.\cite{13} (based on Trilinos ML) and currently in use inside the mantle convection code ASPECT\cite{2}. We will demonstrate the advantages of the geometric method over the algebraic method, as well as show weak and strong scalability of the geometric algorithm on the sinker benchmark used in the work of May et al.\cite{14} and Rudi et al.\cite{16}.

1.1. Software. For all computations we will be using the open-source library deal.II\cite{1} which offers scalable parallel algorithms for finite element computations. The deal.II library uses functionality from other libraries such as Trilinos\cite{11} (for linear algebra, including Trilinos ML AMG preconditioner) and p4est\cite{5} (for mesh partitioning).

As mentioned above, we will be comparing our solver to the one used by ASPECT. ASPECT is an open-source library written on top of deal.II for the specific purpose of solving problems related to the earth’s mantle. The overall goal of this work is a full scale replacement of the current matrix-based Stokes solve in ASPECT with the matrix-free method presented here.

2. Stokes Equations

We will consider the Stokes equations in the form

\begin{align}
-\nabla \cdot (2\mu \varepsilon(u)) + \nabla p &= f \quad \text{on } \Omega \\
\nabla \cdot u &= 0 \quad \text{on } \Omega \\
u &= 0 \quad \text{on } \partial \Omega,
\end{align}

where \( u \) denotes the fluid’s velocity, \( p \) it’s pressure, and \( \mu(x) \) it’s viscosity. Here we use the strain-rate tensor \( \varepsilon(u) = \frac{1}{2} \left( \nabla u + (\nabla u)^T \right) \) as we will be considering non-constant viscosity. The right hand side \( f \) is a forcing term that, inside ASPECT, typically comes from temperature variation, but for our purposes here, we will only be considering Stokes with velocity and pressure variables and a manufactured right hand side. Finally we are considering incompressible flow with homogeneous Dirichlet boundary conditions, though in the future there should be functionality for the general expression \( \nabla \cdot u = g \), as well as for no normal flux boundary constraints \( \nabla u \cdot n = 0 \).

3. Discretization and Linear Solver

Following traditional finite element methods, and using the notation in the work of Kronbichler et al.,\cite{13} we seek coefficients \( u_j \) and \( p_j \) where, for finite element shape functions \( \varphi^u_j \) and \( \varphi^p_j \),

\begin{align}
\mathbf{u}_h &= \sum_{j=0}^{N_u} u_j \varphi^u_j \\
p_h &= \sum_{j=0}^{N_p} p_j \varphi^p_j
\end{align}

such that the weak formulation

\begin{align}
(\varepsilon(\varphi^u_i), 2\mu(x) \varepsilon(\varphi^u_j))_\Omega - (\nabla \cdot \varphi^u_i, p_h)_\Omega - (\varphi^p_i, \nabla \cdot \mathbf{u}_h)_\Omega &= (\varphi^u_i, f)_\Omega \\
(\varphi^p_i, \nabla \cdot \mathbf{u}_h)_\Omega &= 0
\end{align}

holds for each \( 0 \leq i \leq N_u \) and \( 0 \leq l \leq N_p \), where \( N_u \) and \( N_p \) are the total number of degrees of freedom for velocity and pressure respectively. We will choose the shape functions from the Taylor-Hood elements \([Q_{k+1}]^{\text{dim}} \times Q_k, k \geq 1\), which are known to be stable for the system considered here.\cite{8, 4}
Solving this system for coefficients \( U = \{ u_i \} \) and \( P = \{ p_j \} \) is then equivalent to solving the block linear system

\[
\begin{pmatrix}
A & B^T \\
B & 0
\end{pmatrix}
\begin{pmatrix}
U \\
P
\end{pmatrix}
= \begin{pmatrix}
F \\
0
\end{pmatrix}
\]

where

\[
A_{ij} = \int_\Omega \varepsilon(\varphi_i^u) : (2\mu \varepsilon(\varphi_j^u)) \quad B_{ij} = -\int_\Omega \varphi_i^p (\nabla \cdot \varphi_j^u) \quad F_j = \int_\Omega \varphi_j^w f.
\]

We will consider the following preconditioner

\[
P = \begin{pmatrix}
A & B^T \\
0 & -S
\end{pmatrix}
\]

where \( S = BA^{-1}B^T \) is the Schur complement.\[7\] When used as a right preconditioner for the system matrix in (4), we have the following preconditioned system

\[
AP^{-1} = \begin{pmatrix}
A & B^T \\
B & 0
\end{pmatrix}
\begin{pmatrix}
A & -A^{-1}B^TS^{-1} \\
0 & S^{-1}
\end{pmatrix}
= \begin{pmatrix}
I & 0 \\
BA^{-1} & I
\end{pmatrix}.
\]

The preconditioned system \( AP^{-1} \) has only 1 distinct eigenvalue \( \lambda = 1 \) and will converge in at most 2 iteration for an appropriate Krylov subspace method. This is a common block preconditioner used for Stokes solves, especially in the field of mantle convection (see, e.g., the work of May et al.,\[14\] the work of Kronbichler et al.,\[13\] and the work of Rudi et al.\[16\]) although it is not the only possible choice (cf. the work of Silvester\[17\]).

Computing exact representations for \( A^{-1} \) and \( S^{-1} \) is highly impractical, therefore we seek approximate \( \hat{A}^{-1} \) and \( \hat{S}^{-1} \) for the preconditioner \( P \).

### 3.1. Choosing \( \hat{A}^{-1} \).

Since \( A \) comes from a vector Laplacian equation, multigrid would appear to be a logical choice given that these methods are widely known to have convergence independent of mesh size \( h \) for elliptic boundary value problems.\[3, 19\]

Currently in ASPECT, \( \hat{A}^{-1} \) is approximated by 1 AMG V-cycle for each Krylov subspace iteration, however the AMG method is not based on the bilinear form \( a(\cdot, \cdot) \) in (3), but instead we consider the bilinear form

\[
\hat{a}(u, v) = \sum_{i=1}^d (2\mu \varepsilon([u]_{de_d}), \varepsilon([v]_{de_d})) = (2\mu \nabla u, \nabla v)
\]

which ignores the off diagonal blocks of \( A \). The reasons for using this partial coupling of velocity components are the following:

(i) \( \hat{a}(\cdot, \cdot) \) is spectrally equivalent to \( a(\cdot, \cdot) \) with constant viscosity (see (7) below),

(ii) AMG methods, which depend on the sparsity structure of the underlying matrix, tend to deteriorate when coupling vector components in higher order computations,\[9\] and

(iii) the resulting matrix \( \hat{A} \) will have far fewer entries (1/3 the entries in 3D, see, e.g., Figure 1) which results in smaller memory overhead as well as faster AMG v-cycles.

However, one can show that for \( \mu : \Omega \to \mathbb{R}, \mu > 0 \) and bounded,

\[
\frac{\mu_{\min}}{2\mu_{\max}} \leq \frac{2\mu \varepsilon(u), \varepsilon(v)}{(2\mu \nabla u, \nabla v)} \leq \frac{\mu_{\max}}{2\mu_{\min}}
\]

implying that if \( \mu_{\max}/\mu_{\min} \) is large, then \( (2\mu \nabla u, \nabla v) \) as an approximation for \( (2\mu \varepsilon(u), \varepsilon(v)) \) will deteriorate. We will define \( DR(\mu) := \mu_{\max}/\mu_{\min} \) as the dynamic ratio of viscosity, as this term will be useful later.
We will test this implementation of \( \tilde{A}^{-1} \) with a GMG method developed in the work of Clevenger et al.[6] We consider a matrix-free v-cycle based on a degree 4 Chebyshev smoother with one smoothing step per level. The level matrices of the GMG v-cycle are based on the fully coupled system \( A \) since, as we are computing each matrix entry on-the-fly at the quadrature level, using the strain rate tensor only consists of adding the off diagonal term in the correct place and dividing by 2; essentially, it is for free. For the computations below, a typical application of the Stokes matrix operator is roughly 10x faster matrix-free than matrix-based for a 3D, \( Q_2 \) application.

3.1.1. Viscosity Averaging. As discussed in the work of Heister et al.,[10] we will be using the harmonic averaging of viscosity over all quadrature points on a cell. Since in many applications the viscosity will not come from a functional representation, we must have a way to project this cell-wise viscosity on the active mesh to a viscosity on the meshes throughout the level hierarchy. We accomplish this by transferring these coefficients using the same grid restriction operator as discussed in the work of Clevenger et al.,[6] the work of Brenner and Scott,[4] and the work of Janssen and Kanschat,[12] based on a degree 0 DG element. Essentially this involves averaging the active cells viscosity, and then setting the the viscosity of a parent cell in the hierarchy to the arithmetic average of the viscosity of each of its children.

3.2. Choosing \( \tilde{S}^{-1} \). A common choice for approximating \( S = BA^{-1}B^T \) is a weighted pressure mass matrix \( M_p \), where \( M_p = (\mu^{-1} \varphi_i^p, \varphi_i^p) \). [17, 7, 13] The reasons for this is that \( S \) and \( M_p \) are spectrally equivalent for constant viscosity,[7] making \( M_p^{-1} \) a good approximation to \( S^{-1} \), while \( M_p^{-1} \) is far easier to compute. The application of \( \tilde{S}^{-1} \) then is a simple CG solve with an ILU preconditioner (for matrix-based AMG method) or a Chebyshev iteration (for matrix-free GMG method), converging in between 1-5 iterations, and, compared with \( \tilde{A}^{-1} \), is not computationally significant.[13] Since the application of \( P \) now requires a CG solve whose iteration count may change between applications, a flexible Krylov subspace method must be used for the outer iteration. We use the flexible variant of GMRES here.

It should be noted that this Schur complement approximation begins to break down for large DR(\( \mu \)) (see the work of Rudi et al.[16]) and a more sophisticated may be required in those cases.
4. Results

4.1. Benchmark Problem. We give results demonstrating the scalability of the GMG-based method developed here, as well as a comparison with the AMG-based method used in ASPECT. The test problem used is the “Sinker” benchmark described in the work of May et al. [14] and the work of Rudi et al. [16] It consists of solving the Stokes problem (1) over the unit cube domain, where there exists \( n \) randomly positioned “sinkers” of higher viscosity throughout the domain. By specifying \( \text{DR}(\mu) \), we define a smooth viscosity by

\[
\mu(x) = X(x)\mu_{\text{min}} + (1 - X(x))\mu_{\text{max}},
\]

Here \( \mu_{\text{min}} = \text{DR}(\mu)^{-1/2} \), \( \mu_{\text{max}} = \text{DR}(\mu)^{1/2} \), \( c_i \) are the center of each sinker, \( \delta = 200 \) controls the exponential decay of the viscosity, and \( \omega = 0.1 \) is the diameter of the sinkers. The right hand side is given by \( f(x) = (0, 0, \beta(X(x) - 1)) \) with \( \beta = 10 \) and we use homogeneous Dirichlet boundary conditions for the velocity. Physically, this represents gravity pulling down the high viscosity sinkers. Figure 2 gives a representation of both the velocity and the pressure solution of this benchmark.

The problem difficulty can be increased by increasing \( n \) or \( \text{DR}(\mu) \). Table 1 gives the iterations required to reduce the residual of the outer GMRES solve by 1e6 for different values of these parameters. We see that both AMG and GMG deteriorate as both \( n \) and \( \text{DR}(\mu) \) increase, with GMG being slightly more robust. This problem can likely be addressed using methods derived in the work of Rudi et al. [16], where it was shown that this deterioration is due to the approximation loss in the Schur complement solve, and a more sophisticated Schur complement approximation was proposed based on least squares communicators. For the remaining results, we will only consider \( n = 4 \) and \( \text{DR}(\mu) = 1e4 \), as this is within the range of problems where our Schur complement approach is sufficient.

All timings in this section were from computations run on Stampede2 at The University of Texas at Austin’s Texas Advanced Computing Center. We will be using the Intel Xeon Platinum 8160 (Skylake) nodes which have 48 cores and 192GB per node. They support AVX-512 instructions allowing for vectorization over 8 doubles. The deal.II version used is 9.1.0-pre, and we compile using gcc 7.1.0, intel-mpi 17.0.3. The p4est version is 2.0.0, the Trilinos version is 12.10.1 and the ASPECT version is 2.1.0-pre.

Each line in the timing plots connect the median time of 5 distinct runs, with all 5 runs shown as points. We will test strong scaling (problem size stays constant, number of cores increase) and weak scaling (problem size per core stays constant) for key parts of the computation. Tests run on adaptively refined meshes will include the model for partition imbalance described in the work of Clevenger et al. [6] This model should represent the ideal scaling we can expect to see for the GMG method given the imbalance of cells in the level hierarchy.

4.2. GMG scaling. Figure 3(a) gives the strong scaling for an application of the Stokes block preconditioner from a globally refined mesh with between 5-8 refinement levels, and Table 2 gives the corresponding GMRES iteration counts during the solve. The dashed scaling lines here are all computed based on the data point at 48 cores/6.7M DoFs, therefore the plot represents both strong and weak scaling. We see scaling to around 15-68K DoFs/core for the preconditioner and roughly constant iteration counts. Figure 3(b) gives the timings for the GMRES solve\(^1\), and unsurprisingly (since the iteration counts are almost constant) we see the same scaling. Figure 3(c) gives the speedup and efficiency for the GMRES solve.

\(^1\)Note that now the dashed line is no longer only based on the 48 cores/6.7M DoFs data point as we increase 1 iteration from the 6.7M DoF run to the 53M DoF run. These runs will have an extra application of the preconditioner which will result in slower runtimes.
4.3. AMG/GMG Comparison. For a comparison between the AMG and GMG preconditioners, we will consider an adaptively refined mesh, where for each refinement, the number of cells are roughly doubled. We start with a mesh of 4 global refinements and create each new mesh using a Kelly estimator to refine roughly 1/7 of the cells from the previous refinement cycle, doubling the number of cells in our mesh. Table 3 gives the runtimes for such a mesh with 5 levels of adaptive refinement on 48 cores (18.5M degrees of freedom). The “Setup” time includes the distribution of the degrees of freedom, setting up of any sparsity patterns necessary, as well as the setup of the data structures required for the matrix-free GMG transfer. Here, our GMG method requires roughly 2x the work for distributing the degrees of freedom (more objects related to DoFs, must distribute DoFs on level hierarchy), but does not need to build any sparsity patterns. This results in roughly equivalent setup times between AMG and GMG, with GMG being slightly faster. In theory, we should easily be able to lower the requirement of 2x the work in DoF distribution. The “Assemble” timing includes all matrix assembly (system matrix, preconditioner matrix, AMG setup) as well as assembling the right hand side of the linear system and vectors/tables related to the matrix-free operators. Here is where we see the largest advantage for the GMG method as it has no matrices to assemble, resulting in more than

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
DoFs\textbackslash{}Procs & 6.7M & 53M & 412M & 3.4B & 48 & 96 & 192 & 384 & 768 & 1536 & 3072 & 6144 & 12288 & 24576 \\
\hline
\hline
\end{tabular}
\end{table}
Figure 3. Strong and weak scaling for GMG, global refinement. Globally refined, 3D mesh with \([Q_2]^{\text{dim}} \times Q_1\) element.

|                  | AMG | GMG | factor |
|------------------|-----|-----|--------|
| Setup            | 12.6s | 10.3s | 1.2x   |
| Assemble         | 32.5s | 2.9s  | 11.2x  |
| Solve            | 38.6s | 14.8s | 2.6x   |
| Total            | 83.7s | 28.0s | 3.0x   |

Table 3. Timing comparison between AMG and GMG for an adaptively refined, 3D mesh, with 18.5M DoFs \((Q_2)^{\text{dim}} \times Q_1\) element) on one node (48 cores).

A 10x faster assembly. Combining setup and assembly with the linear solve, we have that the GMG method is around 3x faster for this problem.
Table 4. GMRES iterations required to reduce the residual by 1e6 on problems depicted in Figure 4(a).

| Procs | DoFs | AMG | GMG |
|-------|------|-----|-----|
| 48    | 18M  | 53  | 27  |
| 96    | 36M  | 56  | 27  |
| 192   | 72M  | 62  | 28  |
| 384   | 141M | 62  | 28  |
| 768   | 278M | 68  | 28  |
| 1536  | 551M | 75  | 28  |
| 3072  | 1.1B | 80  | 28  |
| 6144  | 2.2B | 83  | 28  |

However, for time dependent applications, many time steps will typically be solved without further refining the mesh, in which case, we no longer need to call the “Setup” functionality. Then the program time will be dominated by assembly and solve, in which case GMG will be about 4x faster here.

Expanding on this, we look at the weak scaling of each component up to 6,144 cores and mesh size of 2.2B degrees of freedom. Starting with the linear solve, as with the scaling plots for global refinement above, Figure 4(a) gives the timing for the preconditioner application, and Table 4 gives the number of GMRES iteration required in the solve. Here we see that, while the AMG preconditioner is cheaper to apply for all but the last data point, the iteration counts for GMG are much lower and stay constant while the AMG iteration counts increase by over 50%. Figure 4(b) shows the solve time and Figure 4(c) shows the speedup. The red dashed line in this plot represents the ideal weak scaling (black dashed line) multiplied by a factor representing the imbalance of the parallel mesh partition that occurs on computations using adaptive mesh refinement, explained in the work of Clevenger et al.[6] This factor represents the increase in runtime one can expect with the current mesh partition (as opposed to a fair distribution), and is both dependent on the mesh refinement scheme and the number of cores used. Here we see that, even with the imbalance of the partition, the scaling for GMG is more efficient than AMG, and there is near perfect efficiency when taking into account the imbalance of the mesh partition, which, according to the work of Clevenger et al.[6] should remain bounded.

The weak scaling of the setup is shown in Figure 5(a), and again we see that AMG and GMG are roughly equivalent, with GMG being slightly faster. The setup of the linear system should be optimized in the future since we are roughly on the same order of magnitude as the solve time, and efforts should be made to improve the scaling. The weak scaling of the assembly is shown in Figure 5(b). Unsurprisingly, the GMG assembly is much cheaper than AMG as there are no matrices to assemble.

Lastly, we look at a comparison for the memory consumption of each method (Table 5). These values represent an estimation of the memory consumptions (in MB) of the largest objects for each method on a globally refined mesh with 113K degrees of freedom on a single core. The number of vectors is a worst-case estimate based on the fact that we must store a few vectors for the linear system (system right-hand side, solution, etc.) as well as temporary vectors for the GMRES solve (typically around a restart length of 50). The number here is chosen to be 50 since the number of vectors is dominated by the GMRES restart length, and it should be noted that the number of iterations is typically higher for the AMG method and therefore, depending on the restart length, we often have to store fewer GMRES vectors for GMG. From the table we see that the AMG method requires roughly 4.3x more memory compared to GMG, and that the largest block of memory for GMG is taken up by vectors.
5. Conclusion

In this article we developed a Geometric Multigrid method for preconditioning of the velocity block in a Stokes solve with variable viscosity. The presented method was designed using a matrix-free framework and has capabilities to be run on adaptively refined meshes and in parallel. The parallel scalability of this method was shown for both strong and weak scaling, for both global and adaptive refinement, with up to 24,576 cores and up to 2.2B degrees of freedom. We performed a comprehensive comparison of the developed method with the AMG preconditioner currently used in the ASPECT code. The GMG preconditioner was shown to be both more robust (lower iteration counts and constant with mesh refinement) and having exhibited better weak scaling than the AMG method, resulting in roughly 3x faster computations than AMG at AMG’s most competitive point. The GMG method was also estimated to require around 3.6x less memory, with more optimizations expected in the future.
10 MATRIX-FREE GEOMETRIC MULTIGRID FOR A STOKES PROBLEM

Figure 5. Setup/Assembly of the linear system.

(a) Setup.

(b) Assembly.

| Memory (MB)   | AMG | GMG |
|---------------|-----|-----|
| Triangulation | 1.9 | 1.9 |
| DoFHandlers   | 2.8 | 5.7 |
| Constraints   | 1.0 | 2.7 |
| $A$           | 174.2 | -  |
| $B$ and $B^T$ | 31.4 | -  |
| $\hat{A}$     | 58.5 | -  |
| $\hat{S}$     | 1.4 | -  |
| Vectors(50)   | 85.0 | 85.0 |
| AMG matrices  | 59.8 | -  |
| Total         | 416.0 | 95.3 |

Table 5. Memory consumption required for major components of AMG and GMG for globally refined, 3D mesh, with 113K DoFs ($[Q_2]^\text{dim} \times Q_1$ element) on 1 cores.

Acknowledgments

The authors have been supported by NSF Award OAC-1835452 and by the Computational Infrastructure in Geodynamics initiative (CIG), through the NSF under Award EAR-0949446 and EAR-1550901 and The University of California – Davis.

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562,[18] Clemson University is acknowledged for generous allotment of compute time on Palmetto cluster. Lastly, we would like to thank Martin Kronbichler of the Technical University of Munich (TUM) for his extensive help in the understanding and implementation of matrix-free algorithms.
References

[1] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. The deal.II library, version 9.0. *Journal of Numerical Mathematics*, 2018, accepted.

[2] Wolfgang Bangerth, Juliane Dannberg, Rene Gassmoller, Timo Heister, et al. ASPECT v2.0.0 [software], May 2018.

[3] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *SIAM J. Sci. Comput.*, 20(5):967–975, 1983.

[4] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer, 2nd edition edition, 2002.

[5] C. Burstedde, L. C. Wilcox, and O. Ghattas. *p4est*: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM J. Sci. Comput.*, 33(3):1103–1133, 2011.

[6] Thomas C. Clevenger, Timo Heister, Guido Kanschat, and Martin Kronbichler. A flexible, parallel, adaptive geometric multigrid method for fem. submitted.

[7] Howard Elman, David J Silvester, and Andrew J Wathen. *Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics*. Oxford University Press, 2005.

[8] Alexandre Ern and Jean-Luc Guermond. *Theory and practice of finite elements*. Springer, 2010.

[9] T. Geenen, M. ur Rehman, S. P. MacLachlan, G. Segal, C. Vuik, A. P. van den Berg, and W. Spakman. Scalable robust solvers for unstructured le geodynamic modeling applications: Solving the stokes equation for models with large localized viscosity contrasts. *Geochemistry, Geophysics, Geosystems*, 10(9).

[10] Timo Heister, Juliane Dannberg, Rene Gassmller, and Wolfgang Bangerth. High accuracy mantle convection simulation through modern numerical methods II: realistic models and problems. *Geophysical Journal International*, 210(2):833–851, 05 2017.

[11] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.

[12] B. Janssen and G. Kanschat. Adaptive multilevel methods with local smoothing for $H^1$- and $H^{curl}$-conforming high order finite element methods. *SIAM J. Sci. Comput.*, 33(4):2095–2114, 2011.

[13] Martin Kronbichler, Timo Heister, and Wolfgang Bangerth. High accuracy mantle convection simulation through modern numerical methods. *Geophysical Journal International*, 191(1):12–29.

[14] D.A. May, J. Brown, and L. Le Pourhiet. A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 290:496 – 523, 2015.

[15] J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. J. Staar, Y. Ineichen, C. Bekas, A. Curioni, and O. Ghattas. An extreme-scale implicit solver for complex pdes: highly heterogeneous flow in earth’s mantle. In *SC ’15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, Nov 2015.

[16] Johann Rudi, Georg Stadler, and Omar Ghattas. Weighted bfbt preconditioner for stokes flow problems with highly heterogeneous viscosity. *SIAM Journal on Scientific Computing*, 39(5), 2017.

[17] David Silvester and Andrew Wathen. Fast iterative solution of stabilised stokes systems part ii: Using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367, 1994.

[18] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. Scott, and N. Wilkins-Diehr. Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(05):62–74, sep 2014.

[19] U. Trottenberg, C. W. Oosterlee, and Anton Schüller. *Multigrid*. Academic Press, 2001.