Robust Multigrid Solvers for Geodynamics

Patrick Sanan

patrick.sanan@[usi.ch,erdw.ethz.ch]

USI Lugano / ETH Zürich

With Dave A. May (ETHZ), Sascha M. Schnepp, Karl Rupp, Olaf Schenk (USI), Matthias Bollhöffer (TU Braunschweig), Paul J. Tackley (ETHZ)

September 14, 2016

German-Swiss Geodynamics Workshop

The GeoPC Project [1]

- Use and improve the latest algorithms and hardware to advance and accelerate **preconditioners for extreme scale Stokes flow with highly heterogenous viscosity structure**.
- **Attack the “full stack”**
  - Algorithmic innovation, scalable software, evaluation on leadership hardware, scientific application
  - Challenging, but exposes interesting synergies.

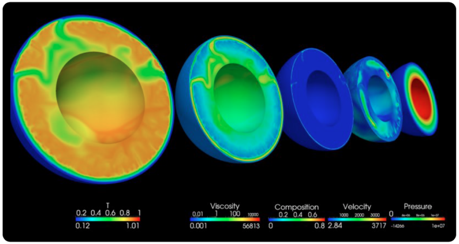


Image courtesy Paul J. Tackley (ETHZ)

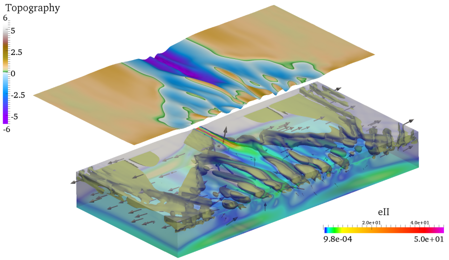


Image courtesy Laetitia Le Pourhiet (UPMC)

[1] http://www.pasc-ch.org/projects/projects/geopc/

Solver Robustness

- Abstractly, consider a family of problems $P(m, d) = 0$, to be solved.
- We have distinguished two inputs to P, $m$odel parameters and $d$ata.
- Solving the *forward problem* (our focus) means to determine d given m, and solving an *inverse problem* means the reverse.
- In m we include physical parameters as well as discretization parameters (problem size,etc.)
- Given a method to solve a problem, we can use various metrics (time-to-solution, etc.) to assess its *performance*
- **We are almost always interested in solving P for multiple values of m and d**

Solver Robustness

- Informally, *robustness* relates to the volume of a neighborhood in parameter-space (or data-space for the inverse problem) which surrounds a point with "good" performance with other points of "good" performance.
- That is, if the solver "works well," to what extent will it "work well" as we vary the parameters?
- The size of this window can vary greatly!
    - Sometimes one only cares that a solver is guaranteed to give a solution at all (uniform non-zero performance).
    - Sometimes problems are so large that any significant degradation in performance can leave problems intractable.

Why Would You Care?

- Especially in geodynamics, we can often never solve the "real" problem
  - Can only model a subset of processes
  - Can only efficiently solve a sub-region of the parameter space
- Thus, we always want to be able to push existing solvers hard!
- Considering our reliance on software, it is important that our tools be as flexible as possible - we can't write specialized code for everything
- Especially when practitioners often just want to "do science," robustness is fantastically valuable as it allows for more **encapsulation**.

The Fundamental Tradeoff

Structure $\Longleftrightarrow$ Efficient Algorithms

A Multigrid Smoother for Saddle-point Systems, Based on Local Incomplete Factorization

Motivation

- The stationary incompressible Stokes equations

$$-\nabla \cdot \boldsymbol{\tau} + \nabla p = \rho(\boldsymbol{x})\hat{\boldsymbol{g}}, \quad -\nabla \cdot \boldsymbol{u} = 0,$$

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

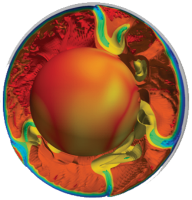
- Newton's Method and Picard iteration solve these with repeated solution of linear systems.
- Discretized with inf-sup stable mixed finite elements (here $\mathbb{Q}_2 - \mathbb{Q}_1$)

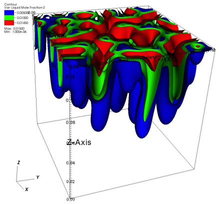
$$\begin{bmatrix} \mathbf{K} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}, \quad \text{or } \boldsymbol{\mathcal{A}}\boldsymbol{v} = \boldsymbol{\mathcal{F}},$$

- There is perpetual interest in being able to **robustly solve this system with respect to distributions of** η which exhibit large variation across arbitrary (non grid-aligned) interfaces.

Why Is This So Difficult?

- Assumptions of smoothness (or equivalently, rapidly decaying spectra) are built into the motivating assumptions of many numerical methods
  - Finite dimensional subspaces
  - Quadrature rules
  - Finite difference methods based on strong forms of PDE
- Traditional engineering methods for (Navier-)Stokes flow are often designed under the assumption of simpler viscosity structures.
- Nevertheless, many useful problems in PDE have this characteristic difficulty. Robustness to coefficient distribution is valued, even at the cost of deviation from optimal performance
  - Stokes Flow
  - Subsurface flow
  - Tomography


Crameri and Tackley 2013

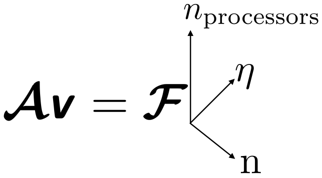


pflotran.org

Shepp-Logan phantom (Wikipedia)

Scalable Linear Solvers

- We focus on the solution of the linear system $\boldsymbol{\mathcal{A}} \boldsymbol{v} = \boldsymbol{\mathcal{F}}$, where most time is spent in many applications
- $\boldsymbol{\mathcal{A}}$ is not a single operator to be solved, but a 2-parameter family. We look for linear increase of DOF/s solved as the
  - problem size and degree of parallelism increase together (**weak scalability**)
  - degree of parallelism increases for a fixed problem size (**strong scalability**).
- In fact, one should consider an infinite dimensional space of relevant problem parameters (here, viscosity structures) and favor methods which behave uniformly well (here, this is what we mean by "robust").


$\boldsymbol{\mathcal{A}} \boldsymbol{v} = \boldsymbol{\mathcal{F}}$
$n_{\text{processors}}$
η
n

Existing Methods and Limitations

- Segregated Methods, Schur Complement Reduction (Can be less efficient than monolithic methods, oversolving)[Uzawa,...]

$$p = (\mathbf{B}^T\mathbf{K}^{-1}\mathbf{B})^{-1}(\mathbf{B}^T\mathbf{K}^{-1}\mathbf{F}), \quad u = \mathbf{F} - \mathbf{B}p$$

- Approximate Block Factorization, Fully Coupled approach (Can be finicky to implement and tune) [Sylvester, Wathen, Elman, Benzi, May, Moresi...]

$$\mathcal{A}^{-1} \approx \begin{bmatrix} \hat{\mathbf{K}} & \mathbf{B} \\ \mathbf{0} & \hat{\mathbf{S}} \end{bmatrix}^{-1},$$

- Full Saddle MG (Less theory, guarantees, tools) [Tackley, Gerya, Kaus...]
- Direct methods (Don't scale)
- Domain Decomposition (Difficult to implement properly, don't usually beat MG/ABF) [Klawonn, Pavarino, ...]
- Nestings, hybridizations, and nonlinear versions of all of the above (complexity, lack of guiding principles)

Smoothing and Preconditioning via Local Incomplete Factorization

- Incomplete factorizations like ILU can be **robust** as smoothers.
- These are not typically applied to indefinite operators, but with careful reordering and scaling[2], using ILUPACK [3], one can effectively use Incomplete LDL^T (ILDL) smoothers (and they also make excellent single-core preconditioners!)

$$\boldsymbol{\Pi}^T \hat{\mathbf{P}}^T \mathbf{D} \mathcal{A} \mathbf{D} \hat{\mathbf{P}} \boldsymbol{\Pi} = \mathcal{A}', \quad \mathcal{A}' = \mathbf{L} \mathbf{D} \mathbf{L}^T + \mathbf{E},$$

- Often a first-attempt parallel preconditioner or smoother is to additively combine local approximate inverses:

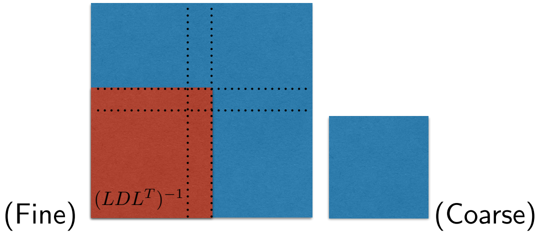
$$\mathcal{A}^{-1} \approx \sum_i \mathbf{R}_i^T \left(\mathbf{R}_i \mathcal{A} \mathbf{R}_i^T \right)^{-1} \mathbf{R}_i \approx \sum_i \mathbf{R}_i^T \mathbf{L}_i^{-T} \mathbf{D}_i^{-1} \mathbf{L}_i^{-1} \mathbf{R}_i$$

- Smoothing is by definition a local procedure so this is somewhat reasonable, but behavior at domain boundaries is problematic.

[2] Michael Hagemann and Olaf Schenk. “Weighted Matchings for Preconditioning Symmetrix Indefinite Linear Systems”. In: *SIAM J. Sci. Comput.* 28.2 (2006), pp. 403–420

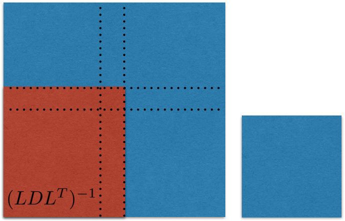
[3] `http://www.icm.tu-bs.de/ bolle/ilupack/`

A Hybrid Approach



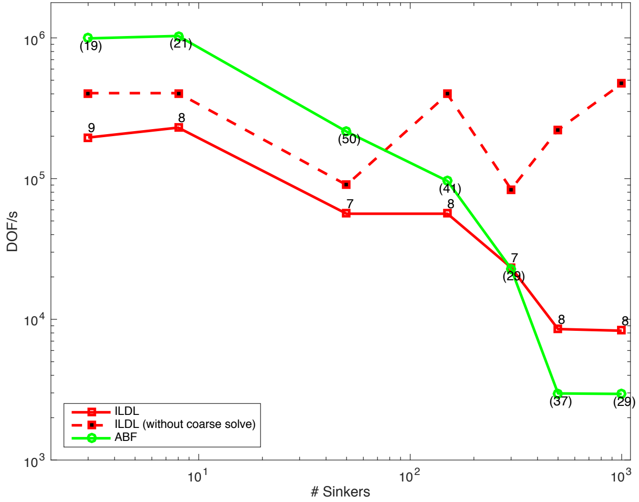
- As a first attempt at attempting to leverage the robustness of these factorizations, we construct a 2-level method which uses a scalable coarse grid solver.
- Fine level smoother: local ILDL smoothers, overlapped by 2 elements.
- Coarse grid solver: Elman-Slyvester-Wathen upper triangular preconditioner for FGMRES. Geometric multigrid with Galerkin coarsening on the viscous block. Approximates the inverse pressure Schur complement with block Jacobi/ILU(0) constructed from a pressure-weighted mass matrix.

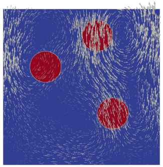
A Hybrid Approach, continued

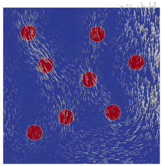
(Fine)  (Coarse)

- If iteration counts can be controlled, the resulting method will also be scalable.
- This is indeed what we see, if we use a tight tolerance on the coarse grid solve.
- The method is not uniformly faster (for all model configurations) than existing approaches, but does exhibit additional robustness to coefficient variation in some cases.

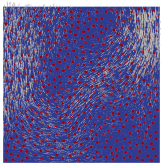
2D Robustness







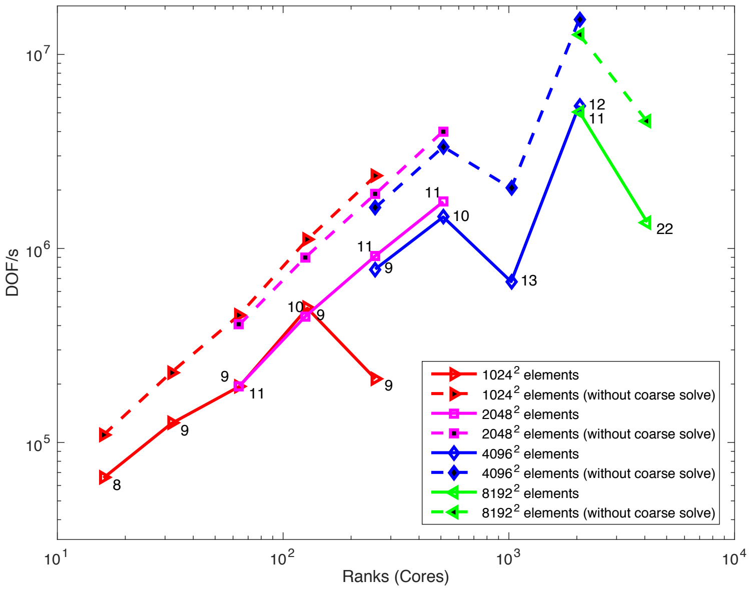
velocity scaled 10×



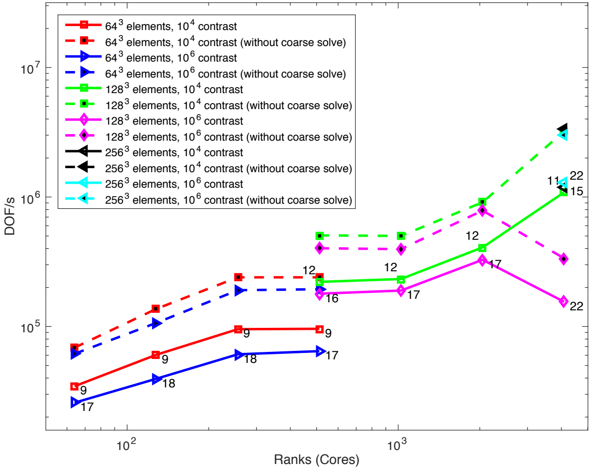
velocity scaled 100×

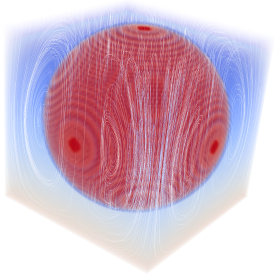
64 ranks (processors/cores), 2048^2 degrees elements (approx. 37 million DOF)

2D Scalability



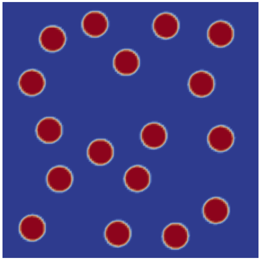
3D Scalability

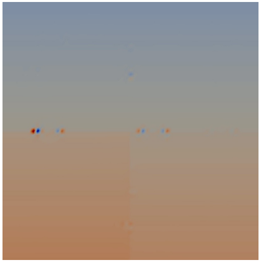




Assessment of The Hybrid Method

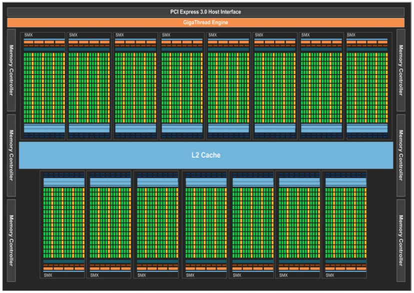
- The method we have shown is robust to some coefficient variation, but seems to suffer from problems common with overlapping domain decomposition.
- Convergence can be degraded when heterogeneities intersect domain boundaries.
- We focus on scalable methods, but one can use an ILDL decomposition as a preconditioner or smoother for a moderately-sized full saddle point problem, without much tuning.


Viscosity distribution (10^4 contrast)


p after one iteration

Opportunities for Acceleration

- Trend in High Performance Computing: adding coprocessors (GPUs/MICs) to compute nodes.



(From the NVIDIA Kepler GK110 white paper)

- Current coprocessors offer high performance ceilings but:
  - Require highly parallel algorithms
  - Use a PCI-express bus with bandwidth and latency restrictions, problematic for physics/algorithms involving global information transfer.
  - Have lower performance ceilings for memory bandwidth-bound operations (SpMV).
- However, coprocessors can apply **strong local solves**. This indicates potential to accelerate expensive local smoothers and solves and add robustness to existing methods.

Software Integration

- The space of solver parameters and available hardware is exploding exponentially. **Composable solver software** offers a way forward.
- The experiments above were carried out by registering a new preconditioner for use with PETSc[4], wrapping ILUPACK.
- With the newly-introduced `PCTELESCOPE` functionality[5], one can redestribute matrices to a smaller communicator with one GPU per rank, allowing for usage of GPU-enabled data types, provided by the PETSc interface to ViennaCL[6].

[4] `mcs.anl.gov/petsc`

[5] Dave A. May, Patrick Sanan, Karl Rupp, Matthew G. Knepley, and Barry F. Smith. "Extreme-Scale Multigrid Components Within PETSc". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '16. Lausanne, Switzerland: ACM, 2016, 5:1–5:12. DOI: `10.1145/2929908.2929913`

[6] viennacl.sourceforge.net

GPU-accelerated kernels

- Using a 3D $\mathbb{Q}_2$ viscous block operator $\mathbf{K}$, we can assess the benefit of moving (assembled) sparse matrix multiplication to the GPU using composable software tools:

	CPU (8 cpu cores)		GPU	
els.	Time (s)	GF/s	Time (s)	GF/s
4^3	6.3162e-03	15.5	3.6376e-02	1.76
8^3	6.4135e-02	9.50	9.4598e-02	5.16
16^3	5.2314e-01	8.18	2.8244e-01	13.5
24^3	1.7321e+00	7.99	9.7740e-01	13.1
28^3	2.7253e+00	7.96	1.5306e+00	13.3
32^3	4.0371e+00	7.93	2.3154e+00	13.0
36^3	5.7317e+00	7.89	3.2920e+00	13.0
40^3	7.7977e+00	7.90	4.6545e+00	12.6

- Modest performance improvement, limited by available memory bandwidth on the GPU, not peak floating point performance.

GPU-accelerated kernels, continued

- We see slightly better results for a similar experiment with a 3D linear elasticity operator, discretized with $\mathbb{Q}_2$ elements:

	CPU (8 cores)		GPU	
els.	Time (s)	GF/s	Time (s)	GF/s
4^3	8.89E−03	11.99	2.43E−02	4.40
8^3	1.27E−01	6.96	5.90E−02	14.99
12^3	4.15E−01	7.3	1.91E−01	15.91
24^3	3.15E+00	7.79	1.44E+00	17.09

- These translate to improved solve time for a complete multigrid solve using local Chebyshev smoothing:

els.	MG levels	T_{setup} (s)	Its.	T_{solve} (s)
CPU:				
8^3	2	1.12E−02	12	4.27E−02
12^3	3	4.41E−02	16	2.06E−01
24^3	3	1.88E−01	13	1.55E+00
48^3	4	1.29E+00	11	9.92E+00
GPU:				
8^3	2	5.49E−01	12	2.2813e-01
12^3	2	2.52E+00	16	2.3985e-01
24^3	3	4.94E+00	13	**1.28E+00**
48^3	4	3.58E+01	11	**6.66E+00**

Future Directions

- Develop matrix-free and otherwise optimized GPU operator kernels
- Offload the work of computing and applying ILDL factorizations to the GPU
  - In progress, headed for ViennaCL (ILU(0) variant already available in ViennaCL 1.7)
- Leverage modern domain decomposition methods to mitigate the effects of subdomain boundaries intersecting heterogeneities.
- Use communication-hiding techniques to allow other useful work to overlap the PCI-express bottleneck
  - Pipelined Flexible Krylov Methods [7]
  - Expanded Krylov bases ("Multi-Krylov methods") [8] [9], extended to asynchronous variants.

[7] P. Sanan, S.M. Schnepp, and D.A. May. "Pipelined, Flexible Krylov Subspace Methods". In: *SIAM Journal on Scientific Computing* 38.5 (2016), pp. C441–C470. DOI: 10.1137/15M1049130

[8] Tyrone Reis, Chen Greif, and Daniel Szyld. "GMRES with multiple preconditioners". URL: https://math.temple.edu/~szyld/reports/multipre_report.rev3.pdf

[9] Nicole Spillane. "An Adaptive Multi Preconditioned Conjugate Gradient Algorithm". URL: https://hal.archives-ouvertes.fr/hal-01170059/document

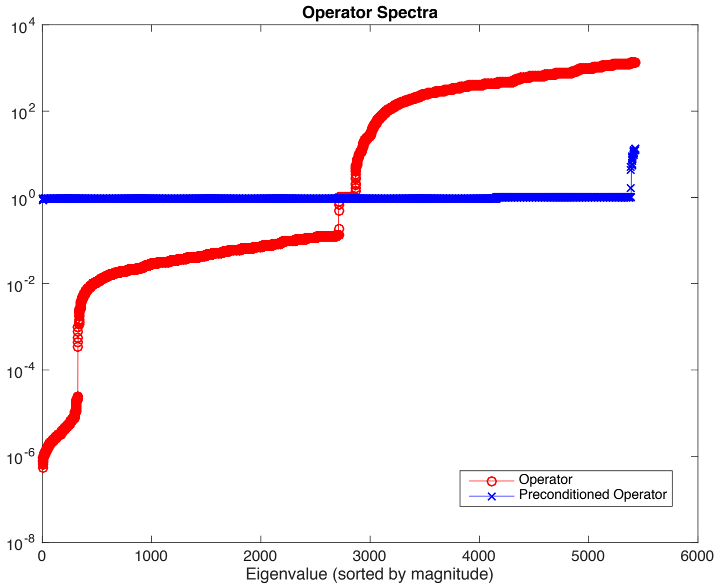
Adding Robustness to Existing Solvers

Adding Robustness to Existing Solvers

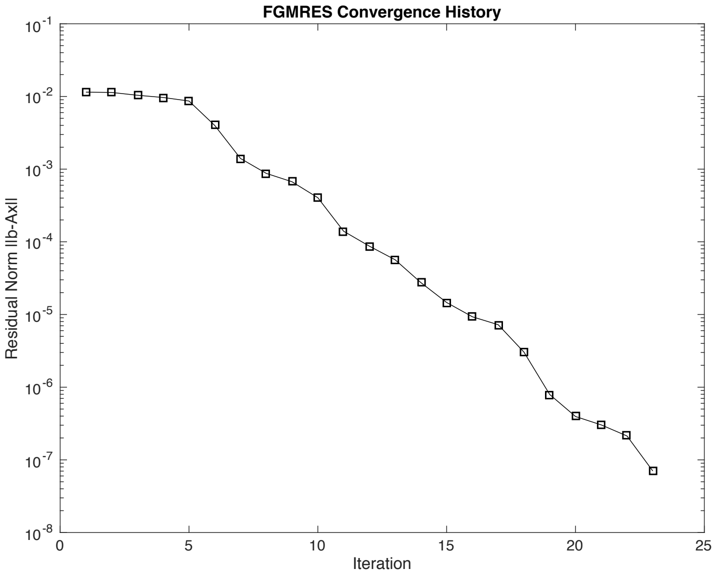
- Composition of solvers can be an effective way to “clean up” when solvers ”almost work”
- Krylov methods are often thought of as linear solvers, and preconditioners as an afterthought
- However, choosing and implementing a good preconditioner is usually the hardest part! (particularly if scalability is a concern)
- A better approach is to find a good solver, and use it as a preconditioner for a Krylov method
- Why do this? Because Krylov solvers work by finding minimal solutions (usually with respect to some norm) in low dimensional spaces. If your solver can reduce the error to a (usually) low-dimensional space, the Krylov method can do the rest for you

$$\min_{x_i - x_0 \in \kappa_i(A,b)} ||b - Ax||, \quad \kappa_k \doteq \{b, A, \ldots, A^{k-1}b\}$$

A small Stokes operator preconditioned with MG/ILDL



A Small Stokes operator preconditioned with MG/ILDL



Adding Robustness to Existing Solvers

- Why go to the trouble?
  - You have a good solver and want to make it more robust
  - You have a good solver, but you want to make it sloppier and cheaper (for example, your exact coarse grid solver is expensive)

Example : Wrapping the StagYY multigrid solver

- StagYY[10] includes a nicely-tuned Geometric multigrid solver
- Multigrid is famously "brittle", however. If any component fails to do its job, convergence stalls
- However, if the solver fails to reduce error in a low-dimensional subspace, a Krylov solver can clean up
- We do exactly this by leveraging PETSc

[10] Paul J. Tackley. "Modelling compressible mantle convection with large viscosity contrasts in a three-dimensional spherical shell using the yin-yang grid". In: *Physics of the Earth and Planetary Interiors* 171.14 (2008). Recent Advances in Computational Geodynamics: Theory, Numerics and Applications, pp. 7 –18. ISSN: 0031-9201. DOI: `http://dx.doi.org/10.1016/j.pepi.2008.08.005`.

```
./stagyy par_mgk_test_1 -mgk_ksp_monitor_true_residual -mgk_ksp_type fgmres -mgk_ksp_rtol 1e-13 -mgk_ksp_view
```

```
**********************Time step :     1************************
********dt = 1.580E-05 ; total t = 1.580E-05
   Timestep fraction, diff & adv =   1.00000  0.02098
      Courant number, diff & adv =   0.80000  0.01679
for RHS - Inter-cell visc jump in x,y,z:  1.00E+00  1.23E+00  1.55E+03
    Multigrid coarse levels and #cpus:
global:   1x  64y  64z   1b; per node:   1x  64y  64z   1b on   1 cpus:  1x  1y  1z  1b
global:   1x  32y  32z   1b; per node:   1x  32y  32z   1b on   1 cpus:  1x  1y  1z  1b
global:   1x  16y  16z   1b; per node:   1x  16y  16z   1b on   1 cpus:  1x  1y  1z  1b
global:   1x   8y   8z   1b; per node:   1x   8y   8z   1b on   1 cpus:  1x  1y  1z  1b
  Residual norms for mgk_ solve.
  0 KSP Residual norm 8.505577832968e+09
rms (rhs/eta) :  7.89E-07
  Initial rrms :   0   1.00000 0.0000E+00 0.0000E+00 6.1774E+03 0.0000E+00
  Cycle & rrms :   1   1.00000 0.0000E+00 9.6278E+00 1.1191E+02 5.0422E-01
  1 KSP Residual norm 8.505567599480e+09
rms (rhs/eta) :  1.73E-08
  Initial rrms :   0   1.00000 0.0000E+00 6.7876E-01 7.8428E+00 3.5548E-02
  Cycle & rrms :   1   1.00000 0.0000E+00 7.6865E-03 4.3090E-02 3.5523E-04
  2 KSP Residual norm 8.006586370457e+09
rms (rhs/eta) :  8.55E-08
  Initial rrms :   0   1.00000 0.0000E+00 7.6119E-01 8.3983E+00 4.0178E-02
  Cycle & rrms :   1   1.00000 0.0000E+00 6.8283E-03 5.0793E-02 3.9055E-04
  3 KSP Residual norm 1.151158780350e+08
rms (rhs/eta) :  3.40E-07
  Initial rrms :   0   1.00000 0.0000E+00 6.1194E-01 6.1778E+00 2.8329E-02
  Cycle & rrms :   1   1.00000 0.0000E+00 1.3975E-02 3.7920E-02 4.0218E-04
  4 KSP Residual norm 2.196740387858e+06
...
```

```
Linear mgk_ solve converged due to CONVERGED_RTOL iterations 7
KSP Object:(mgk_) 1 MPI processes
  type: fgmres
    GMRES: restart=30, using Classical (unmodified) Gram-Schmidt Orthogonalization with no it
    GMRES: happy breakdown tolerance 1e-30
  maximum iterations=10000, initial guess is zero
  tolerances:  relative=1e-08, absolute=1e-50, divergence=10000.
  right preconditioning
  using UNPRECONDITIONED norm type for convergence test
PC Object:(mgk_) 1 MPI processes
  type: shell
    Shell: StagYY Veecycles PC
  linear system matrix = precond matrix:
  Mat Object:  StagYY Stokes Operator   1 MPI processes
    type: shell
    rows=16384, cols=16384
      has attached null space
Top flux and Nu =    32.439   32.439 ; Bot flux and Nu =     1.017    1.017
Temp : min =      0.142, mean =      0.945, max =      0.998
vel : min =  1.027E-10,  rms =  6.119E+00, max =  1.832E+01
Visc : min =  1.060E+00, mean =  4.569E+08, max =  5.075E+10
...
```

Next: Try to solve some more challenging systems!

More on coarse-grid solvers

- For large problems, proper treatment of a coarse grid solve is key - communication will eventually dominate computation
- One common way to utilize the robustness of an outer Krylov method is to use an inexact coarse grid solve, with reduced communication requirements
  - StagYY can use an iterative solve on the coarse grid
  - Similarly, the coarse grid solver can be Krylov methods[11], which in these instances can be accelerated by *pipelining*[12,13,14]
- The methods from the papers below are available (from command-line options!) in PETSc 3.7.

[11] D.a. May, J. Brown, and L. Le Pourhiet. "A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow". In: *Computer Methods in Applied Mechanics and Engineering* 290 (2015), pp. 496–523. ISSN: 00457825. DOI: 10.1016/j.cma.2015.03.014.

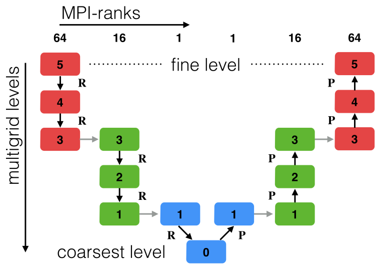
[12] P Ghysels, T J Ashby, K Meerbergen, and W Vanroose. "Hiding Global Communication Latency in the GMRES Algorithm on Massively Parallel Machines". In: *SIAM J. Sci. Comput.* 35.1 (2013), pp. 48–71.

[13] P. Ghysels and W. Vanroose. "Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm". In: *Parallel Computing* 40.7 (2014), pp. 224–238. ISSN: 01678191. DOI: 10.1016/j.parco.2013.06.001.

[14] P. Sanan, S.M. Schnepp, and D.A. May. "Pipelined, Flexible Krylov Subspace Methods". In: *SIAM Journal on Scientific Computing* 38.5 (2016), pp. C441–C470. DOI: 10.1137/15M1049130.

More on coarse-grid solvers

- The communication bottleneck can also be eased by *processor agglomeration*, an attempt to maintain a favorable balance of communication and computation by using only a subset of available distributed-memory nodes.
  - StagYY and many specialized solvers support this pattern
  - We recently introduced it as `PCTELESCOPE`, a reusable and composable component in PETSc[15]



[15] Dave A. May, Patrick Sanan, Karl Rupp, Matthew G. Knepley, and Barry F. Smith. "Extreme-Scale Multigrid Components Within PETSc". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '16. Lausanne, Switzerland: ACM, 2016, 5:1–5:12. DOI: 10.1145/2929908.2929913.

Software and Other Thoughts

Black Box → Toolbox

- One can relax the idea of robustness to include some human intervention
- If one has the software environment to do so, methods can be selected from a set of possibilities based on intuition, partial knowledge, and direct testing
- This allows one to potentially circumvent some of the fundamental tradeoff: each individual method may be more specific but less robust.
- Caveat: setting up the software environment this way can be time-consuming (though PETSc is very helpful for solvers)
- Caveat: some of the burden for doing the right thing has been pushed to the user. It is unfortunate to not always have a black box, but in some situations (optimal scalability) there simply is no black-box solver.

Conclusions

- We have shown some strategies that could be useful for making Stokes solvers more robust, particularly in the case of large coefficient variation
- There are inherent tradeoffs
  - (Automatic) robustness vs. tuning to a particular case
  - “Black Box” vs. “Toolbox”
- Reusable, reliable, composable, well-tested software is required

Thank you for your attention!

patrick.sanan@usi.ch
patrick.sanan@erdw.ethz.ch