Multilevel Stokes flow solvers

Adapting to heterogeneity and rheology

Jed Brown

Mathematics and Computer Science Division, Argonne National Laboratory

CIG Mantle/Lithosphere 2012-07-30

Intent of this talk

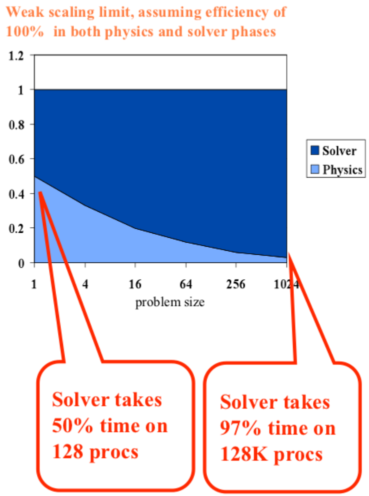
- observation: solver scalability is the bottleneck at scale
- “black box” solvers are not sustainable
  - optimal solvers must accurately handle all scales
  - optimality is crucial for large-scale problems
  - hardware puts up a spirited fight to abstraction
- introduce multilevel solver concepts
- outline ingredients that discretizations can provide to solvers
- discuss algorithmic trade-offs
- current state of solver software and what we are working on

Outline

Outline

It's *all* about algorithms (at the petascale)

- Given, for example:
  - a "physics" phase that scales as $O(N)$
  - a "solver" phase that scales as $O(N^{3/2})$
  - computation is almost all solver after several doublings
- Most applications groups have not yet "felt" this curve in their gut
  - as users actually get into queues with more than 4K processors, this will change

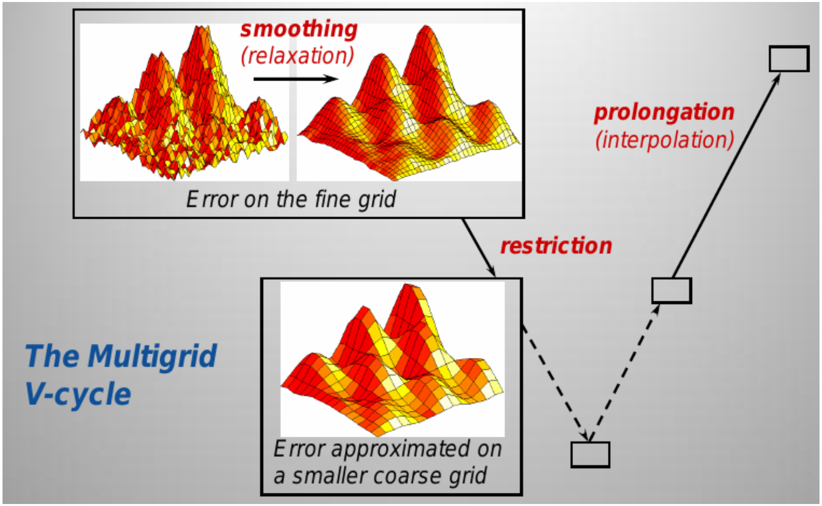


(c/o David Keyes)

Challenges for elliptic solvers

- multiscale material coefficients
  - long, thin high viscosity: transmit stresses long distances
  - “jelly sandwich”: release long-range stresses locally
- nonlinearity
  - plasticity: creates “jelly sandwich”
  - Newton linearization produces local anisotropy
  - heating: localization
  - coupling to other physical processes
- multilevel methods
  - need accurate coarse grids
  - need effective smoothers

Multigrid separates scales, feedback between scales



The Great Solver Schism: Monolithic or Split?

Monolithic

- Direct solvers
- Coupled Schwarz
- Coupled Neumann-Neumann (need unassembled matrices)
- Coupled multigrid

X Need to understand local spectral and compatibility properties of the coupled system

Split

- Physics-split Schwarz (based on relaxation)
- Physics-split Schur (based on factorization)
  - approximate commutators SIMPLE, PCD, LSC
  - segregated smoothers
  - Augmented Lagrangian
  - “parabolization” for stiff waves

X Need to understand global coupling strengths

- Preferred data structures depend on which method is used.
- Interplay with geometric multigrid.

Outline

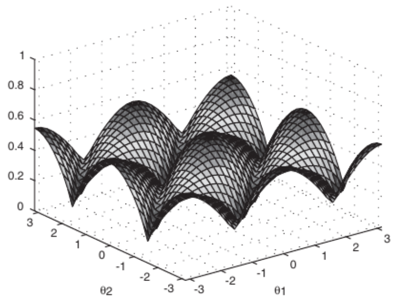
Three schools of thought

- Multigrid (Brandt, Hackbusch, . . .)
  - originally for resolved/asymptotic spatial discretizations
  - textbook: reach discretization error in one F-cycle
  - matrix-light/free, good for bandwidth
  - FAS well-developed for nonlinear problems
- Multilevel Domain Decomposition (Mandel, Dohrmann, Widlund)
  - leverage direct subdomain solvers, minimize communication
  - rapid coarsening $\kappa(P^{-1}A) \sim \left(1 + \log \frac{H}{h}\right)^{2(L-1)}$
  - often formulated only as two-level methods
  - typically with domain-conforming coefficients
  - lightly developed for nonlinear (e.g. ASPIN [Cai and Keyes])
- Multiscale Finite Elements (Babuska, Arbogast, . . .)
  - local preprocessing to construct coarse space
  - rarely/never revisit fine space
  - mostly restricted to linear problems

Computable Convergence Measures

- Prolongation $P : V_{\text{coarse}} \to V_{\text{fine}}$
- Restriction $R : V_{\text{fine}} \to V_{\text{coarse}}$
- $I - PR : V_{\text{fine}} \to V_{\text{fine}}$ removes part of vector visible in coarse space
- Error iteration matrix $I - M^{-1}A$, worst-case convergence factor is $\lambda_{\max}$
- "Interpolation must be able to approximate an eigenvector with error bound proportional to the size of the associated eigenvalue."
  - $\max_x \|x\|_{(I-PR)S(I-PR)} / \|x\|_A$
- What can we do before we have prolongation P?

Compatible Relaxation



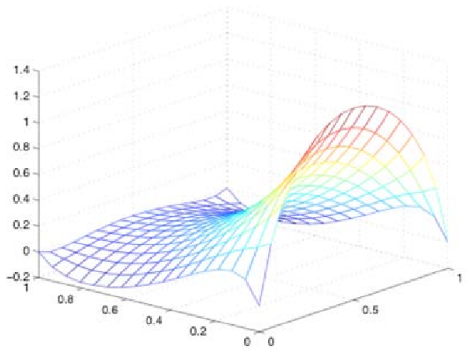
[Livne 2004]

- Apply smoother subject to constraint $\hat{R}x = 0$
  1. $\tilde{x}_n = x_{n-1} + S_A^{-1}\big(r(x_{n-1})\big)$
  2. $x_n = \tilde{x}_n + S_R^{-1}\big(\hat{R}\tilde{x}_n)\big)$
- Method to determine when coarse space is rich enough
- Slow to relax points/regions good candidates for coarse points/aggregates
- If subdomain solves used for smoothing, only interfaces are candidates

Coarse basis functions

- $\|PRx\|_A + \|(I-PR)x\|_A \le C\,\|x\|_A$
- "decompose any x into parts without increasing energy much"
- near-null spaces must be represented exactly (partition of unity)
- number of rows of R determined already, usually $P = R^T$
- energy minimization with specified support [Wan, Chan, Smith; Mandel, Brezina, Vanek]
- smoothed aggregation: $P_{\text{smooth}} = (I - \omega D^{-1}A)P_{\text{agg}}$
- classical AMG: each fine point processed independently
- domain decomposition/multiscale FEM: solve subdomain problems

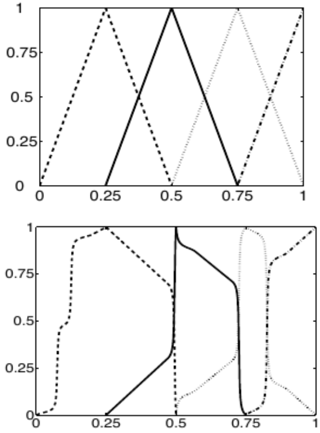
Example: BDDC/FETI-DP coarse basis function



[Mandel and Sousedik 2010]

- only low-order continuity between subdomains
- corrected by more technical subdomain smoother

Why I like subdomain problems



[Arbogast 2011]

- subassembly avoids explicit matrix triple product $A_{\text{coarse}} \leftarrow P^T A_{\text{fine}} P$
- can update the coarse operator locally (e.g. local nonlinearity)
- need not assemble entire fine grid operator
- can coarsen very rapidly (at least in smooth regions)
- lower communication setup phase

Complication for saddle point problems

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$$

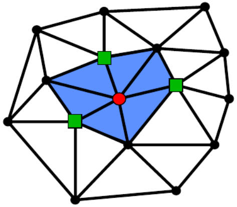
- want uniform stability for coarse problem
  - respect inf-sup condition, similar to fine grid
- want exact representation of volumetric mode
  - i.e. we can't cheat on conservation while upscaling
- to be rigorous, we need to evaluate face integrals
  - self-similar coarse discretizations are attractive
- heuristic algebraic coarsening also possible [Adams 2004]

Nonlinear problems

- matrix-based smoothers require global linearization
- nonlinearity often more efficiently resolved locally
- nonlinear additive or multiplicative Schwarz
- nonlinear/matrix-free is good if

$$C = \frac{(\text{cost to evaluate residual at one point}) \cdot N}{(\text{cost of global residual})} \sim 1$$

  - finite difference: $C < 2$
  - finite volume: $C \sim 2$, depends on reconstruction
  - finite element: $C \sim$ number of vertices per cell
- larger block smoothers help reduce C

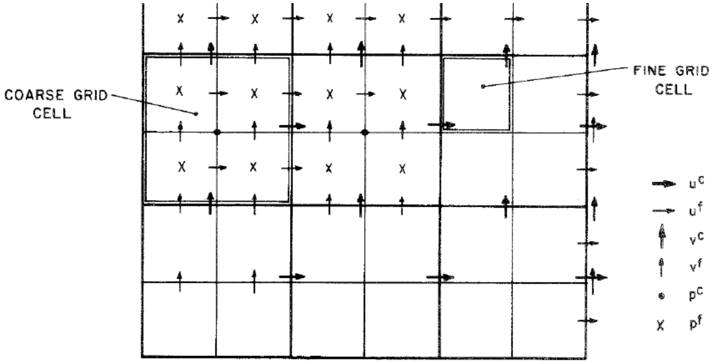


Smoothing for saddle point systems

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$$

- pressure has no self-coupling
- pressure error modes not spectrally separated
- approaches
  - block smoothers (Vanka)
  - splitting with approximate Schur complement
  - amplify fine-grid modes

Vanka block smoothers



- solve pressure-centered cell problems
  (better for discontinuous pressure)
- robust convergence factor ~ 0.3 *if* coarse grids are accurate
- 1D energy minimizing interpolants easy and effective
- can use assembled sparse matrices, but more efficient without

Changing Associativity: Distributive Smoothing

$$PAx = Pb \qquad\qquad APy = b, \quad x = Py$$

- Normal Preconditioning: make PA or AP well-conditioned
- Alternative: amplify high-frequency modes
  - Multigrid smoothers only need to relax high-frequency modes
  - Easier to do when spectrally separated: h-ellipticity
    - pointwise smoothers (Gauss-Seidel) and polynomial/multistage methods
  - Mechanics: form the product PA or AP and apply "normal" method
  - Example (Stokes)

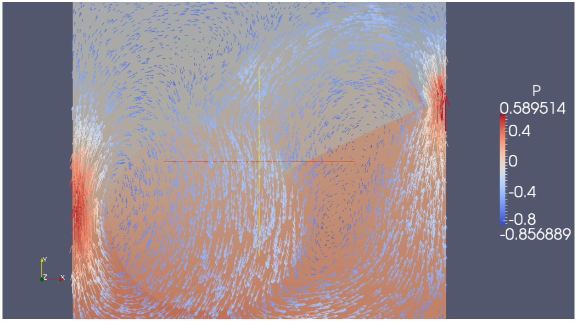
$$A \sim \begin{pmatrix} -\nabla^2 & \nabla \\ \nabla\cdot & 0 \end{pmatrix} \quad P \sim \begin{pmatrix} 1 & -\nabla \\ 0 & -\nabla^2 \end{pmatrix} \quad AP \sim \begin{pmatrix} -\nabla^2 & \text{“0”} \\ \nabla\cdot & -\nabla^2 \end{pmatrix}$$

- Convergence factor 0.32 (as good as Laplace) for smooth problems

Coupled MG for Stokes, split smoothers

$$J = \begin{pmatrix} A & B^T \\ B & C \end{pmatrix}$$

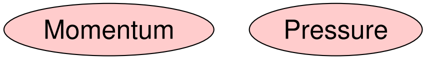
$$P_{\text{smooth}} = \begin{pmatrix} A_{\text{SOR}} & 0 \\ B & M \end{pmatrix}$$



```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_fieldsplit_block_size 3
-mg_levels_pc_fieldsplit_0_fields 0,1
-mg_levels_pc_fieldsplit_1_fields 2
-mg_levels_fieldsplit_0_pc_type sor
```

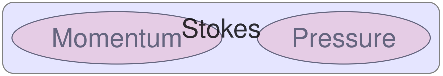
Outline

Multi-physics coupling in PETSc



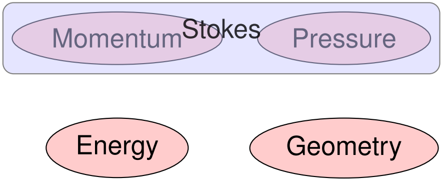
- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting

Multi-physics coupling in PETSc



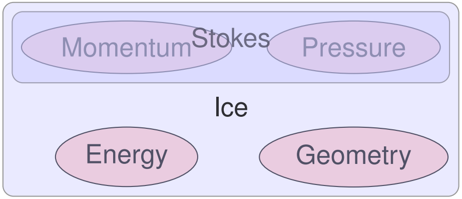
- package each "physics" independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting

Multi-physics coupling in PETSc



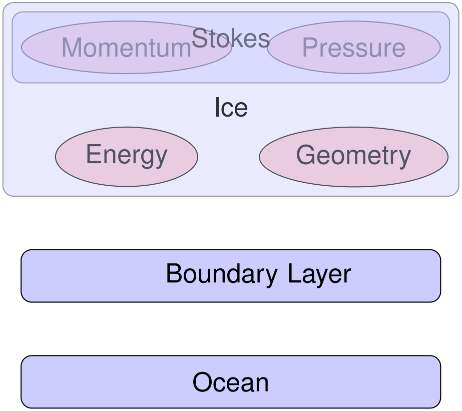
- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting

Multi-physics coupling in PETSc



- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting

Multi-physics coupling in PETSc



- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format for each physics (e.g. symmetric block size 3)
- matrix-free anywhere
- multiple levels of nesting

Splitting for Multiphysics

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

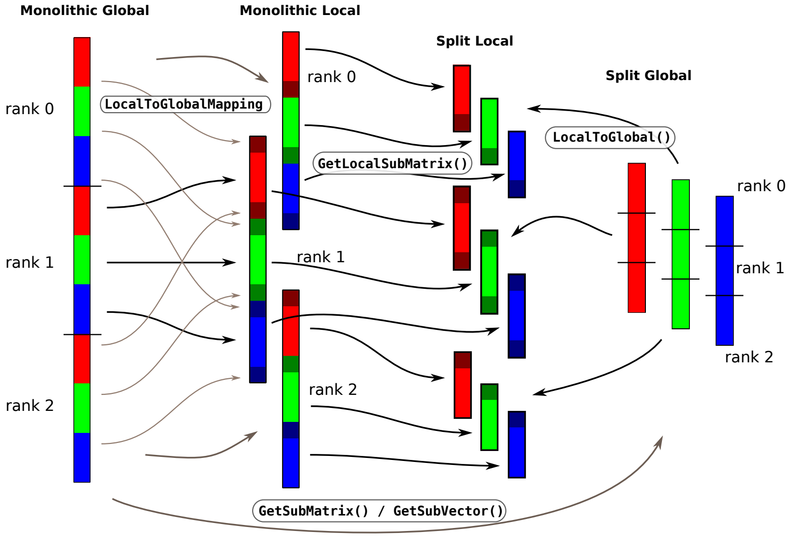
- Relaxation: `-pc_fieldsplit_type [additive,multiplicative,symmetric_multiplicative]`

$$\begin{bmatrix} A & \\ & D \end{bmatrix}^{-1} \qquad \begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \qquad \begin{bmatrix} A & \\ & 1 \end{bmatrix}^{-1} \left(1 - \begin{bmatrix} A & B \\ & 1 \end{bmatrix} \begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \right)$$

  - Gauss-Seidel inspired, works when fields are loosely coupled
- Factorization: `-pc_fieldsplit_type schur`

$$\begin{bmatrix} A & B \\ & S \end{bmatrix}^{-1} \begin{bmatrix} 1 & \\ CA^{-1} & 1 \end{bmatrix}^{-1}, \qquad S = D - CA^{-1}B$$

  - robust (exact factorization), can often drop lower block
  - how to precondition S which is usually dense?
    - interpret as differential operators, use approximate commutators



Work in Split Local space, matrix data structures reside in any space.

Multiphysics Assembly Code: Jacobians

```
FormJacobian_Coupled(SNES snes,Vec X,Mat J,Mat B,...) {
  // Access components as for residuals
  MatGetLocalSubMatrix(B,is[0],is[0],&Buu);
  MatGetLocalSubMatrix(B,is[0],is[1],&Buk);
  MatGetLocalSubMatrix(B,is[1],is[0],&Bku);
  MatGetLocalSubMatrix(B,is[1],is[1],&Bkk);
  FormJacobianLocal_U(user,&infou,u,k,Buu);         // single physics
  FormJacobianLocal_UK(user,&infou,&infok,u,k,Buk); // coupling
  FormJacobianLocal_KU(user,&infou,&infok,u,k,Bku); // coupling
  FormJacobianLocal_K(user,&infok,u,k,Bkk);         // single physics
  MatRestoreLocalSubMatrix(B,is[0],is[0],&Buu);
  // More restores
```

- Assembly code is independent of matrix format
- Single-physics code is used unmodified for coupled problem
- No-copy fieldsplit:
  `-pack_dm_mat_type nest -pc_type fieldsplit`
- Coupled direct solve:
  `-pack_dm_mat_type aij -pc_type lu -pc_factor_mat_solver_package mumps`

Quasi-Newton revisited: ameliorating setup costs

- Newton-Krylov with analytic Jacobian

Lag	FunctionEval	JacobianEval	PCSetUp	PCApply
1 bt	12	8	8	31
1 cp	31	6	6	24
2 bt		— diverged —		
2 cp	41	4	4	35
3 cp	50	4	4	44

pseudo-plastic rheology

`-snes_type qn`

`-snes_qn_scale_type jacobian`

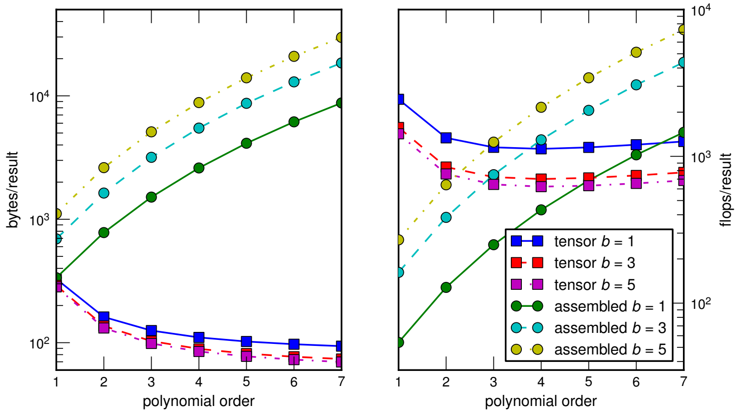
- Jacobian-free Newton-Krylov with lagged preconditioner

Lag	FunctionEval	JacobianEval	PCSetUp	PCApply
1 bt	23	11	11	31
2 bt	48	4	4	36
3 bt	64	3	3	52
4 bt	87	3	3	75

- Limited-memory Quasi-Newton/BFGS with lagged solve for H_0

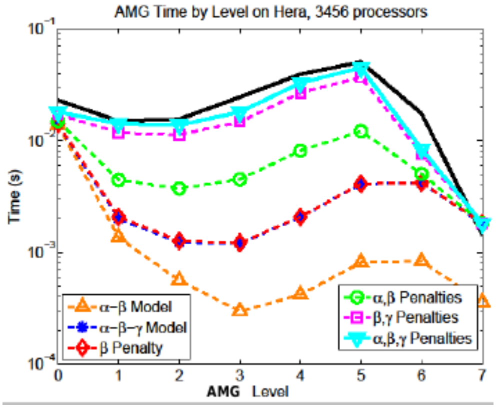
Restart	H_0	FunctionEval	JacobianEval	PCSetUp	PCApply
1 cp	10^{-5}	17	4	4	35
1 cp	preonly	21	5	5	10
3 cp	10^{-5}	21	3	3	43
3 cp	preonly	23	3	3	11
6 cp	10^{-5}	29	2	2	60
6 cp	preonly	29	2	2	14

Performance of assembled versus unassembled



- High order Jacobian stored unassembled using coefficients at quadrature points, can use local AD
- Choose approximation order at run-time, independent for each field
- Precondition high order using assembled lowest order method
- Implementation $> 70\%$ of FPU peak, SpMV bandwidth wall $< 4\%$

Coarse levels may not be cheaper than fine levels



[Gahvari, Schulz, Yang, Jordan, Gropp 2011]

- latency for longer-range communication outweighs smaller data
- very aggressive coarsening important to limit number of levels
- alternatives: additive multigrid, redundant coarse grids

Multilevel Solvers are a *Way of Life*

- ingredients that discretizations can provide
  - identify “fields”
  - topological coarsening, possibly for fields
  - near-null space information
  - “natural” subdomains
  - subdomain integration, face integration
  - element or subdomain assembly/matrix-free smoothing
- solver composition
  - most splitting methods accessible from command line
  - energy optimization for tentative coarse basis functions
  - algebraic form of distributive relaxation
  - generic assembly for large systems and components
  - working on flexibile “library-assisted” nonlinear multigrid
  - adding support for interactive eigenanalysis