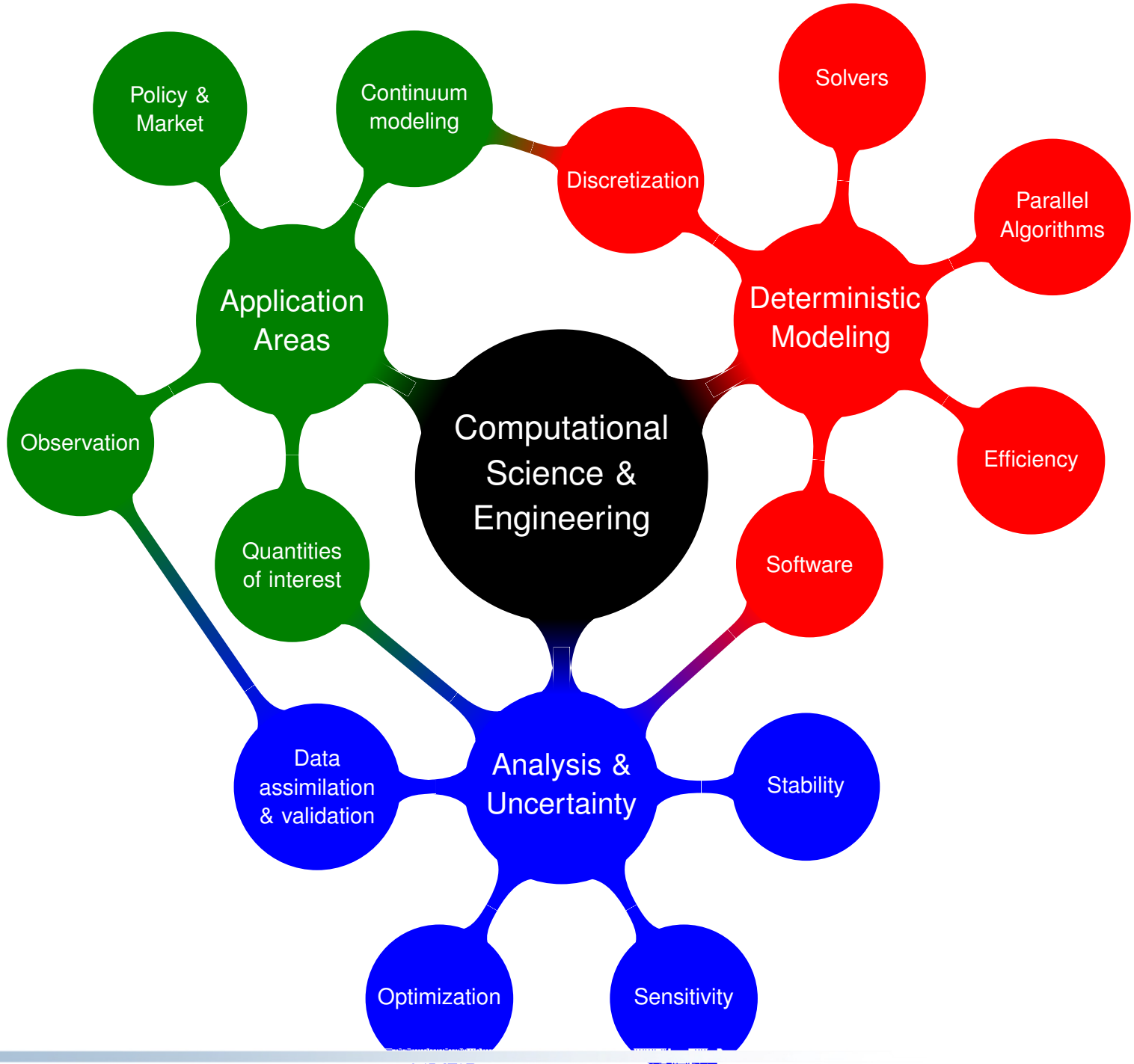
Exploiting structure in scientific computing

This talk:
http://59A2.org/files/20141211-Structure.pdf

Jed Brown jed@jedbrown.org (ANL and CU Boulder)

CU Boulder, 2014-12-11


Computational Science & Engineering
Application Areas
Policy & Market
Continuum modeling
Observation
Quantities of interest
Deterministic Modeling
Solvers
Discretization
Parallel Algorithms
Efficiency
Software
Analysis & Uncertainty
Data assimilation & validation
Stability
Optimization
Sensitivity

PETSc

- Portable Extensible Toolkit for Scientific computing
- Jed: User since 2004, developer since 2008
- Known for parallel (differential) algebraic solvers
- TAO: Toolbox for Advanced Optimization
- Used by many open source libraries and applications
- Deployment mechanism for algorithm development
- Industry: Abaqus, Boeing, Schlumberger, Shell, etc.
- Awards
  - 3 Gordon Bell Prizes have used PETSc (1999, 2003, 2004)
  - R&D100, E.O. Lawrence Award, SIAM/ACM CS&E Prize

This talk

- Implicitness: multiple time scales
- Coupling for multi-physics
- Multigrid solvers

- Parallelism and efficiency

State of implicitness

- Nature has many spatial and temporal scales
  - Porous media, structures, fluids, kinetics
  - Quasi-equilibrium processes
- Science/engineering problem statement does not weak scale
  - More time steps required at higher resolution
- Robust discretizations and implicit solvers are needed to cope
- Computer architecture is increasingly hierarchical
  - algorithms should conform to this structure
- Sparse matrices are comfortable, but outdated
  - Algebraic multigrid, factorization
  - Memory bandwidth-limited
- “black box” solvers are not sustainable: $\mathcal{O}(n^2)$ in 3D
  - optimal solvers must accurately handle all scales
  - optimality is crucial for large-scale problems
  - hardware puts up a spirited fight to abstraction

Model Coupling

$$f(u; v) = 0$$
$$g(v; u) = 0$$

- f and g developed independently

The Great Solver Schism: Monolithic or Split?

Monolithic

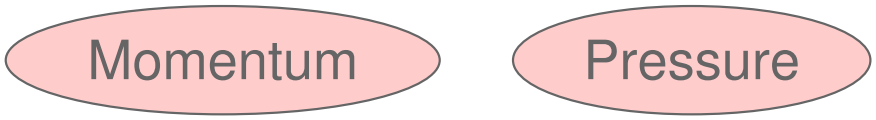
- Direct solvers
- Coupled Schwarz
- Coupled Neumann-Neumann (need unassembled matrices)
- Coupled multigrid
- X Need to understand local spectral and compatibility properties of the coupled system

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.

Multi-physics coupling in PETSc

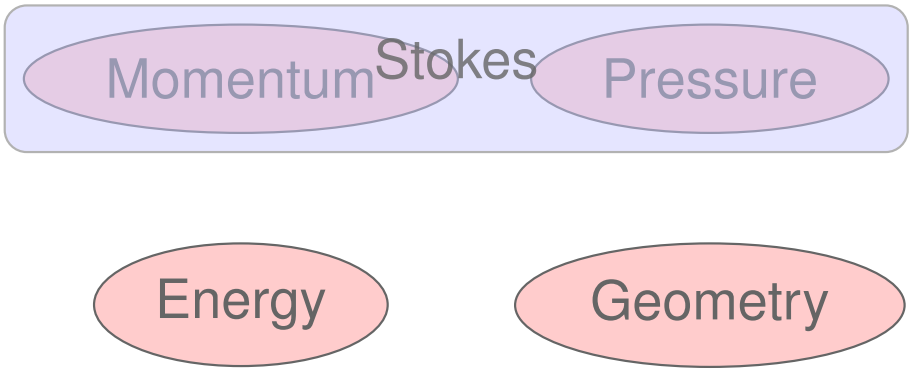


- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

Multi-physics coupling in PETSc

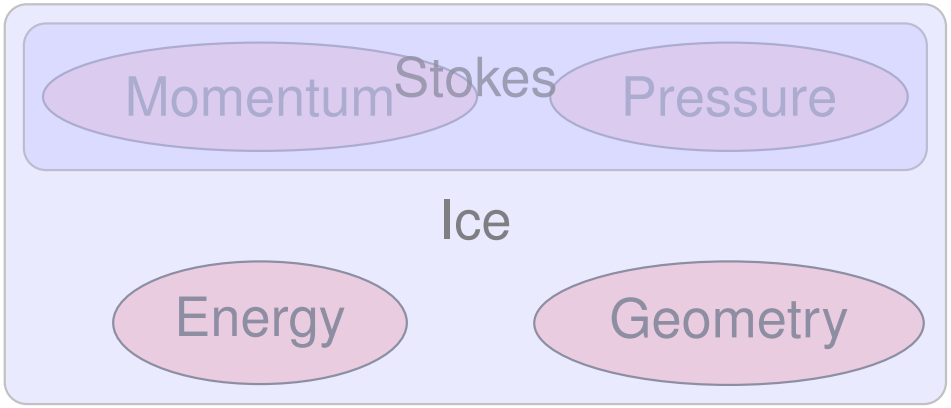
- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

Multi-physics coupling in PETSc



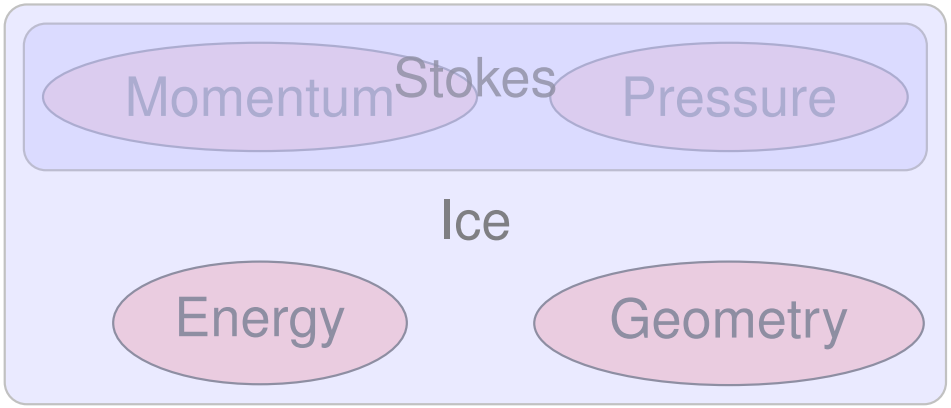
- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format 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

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

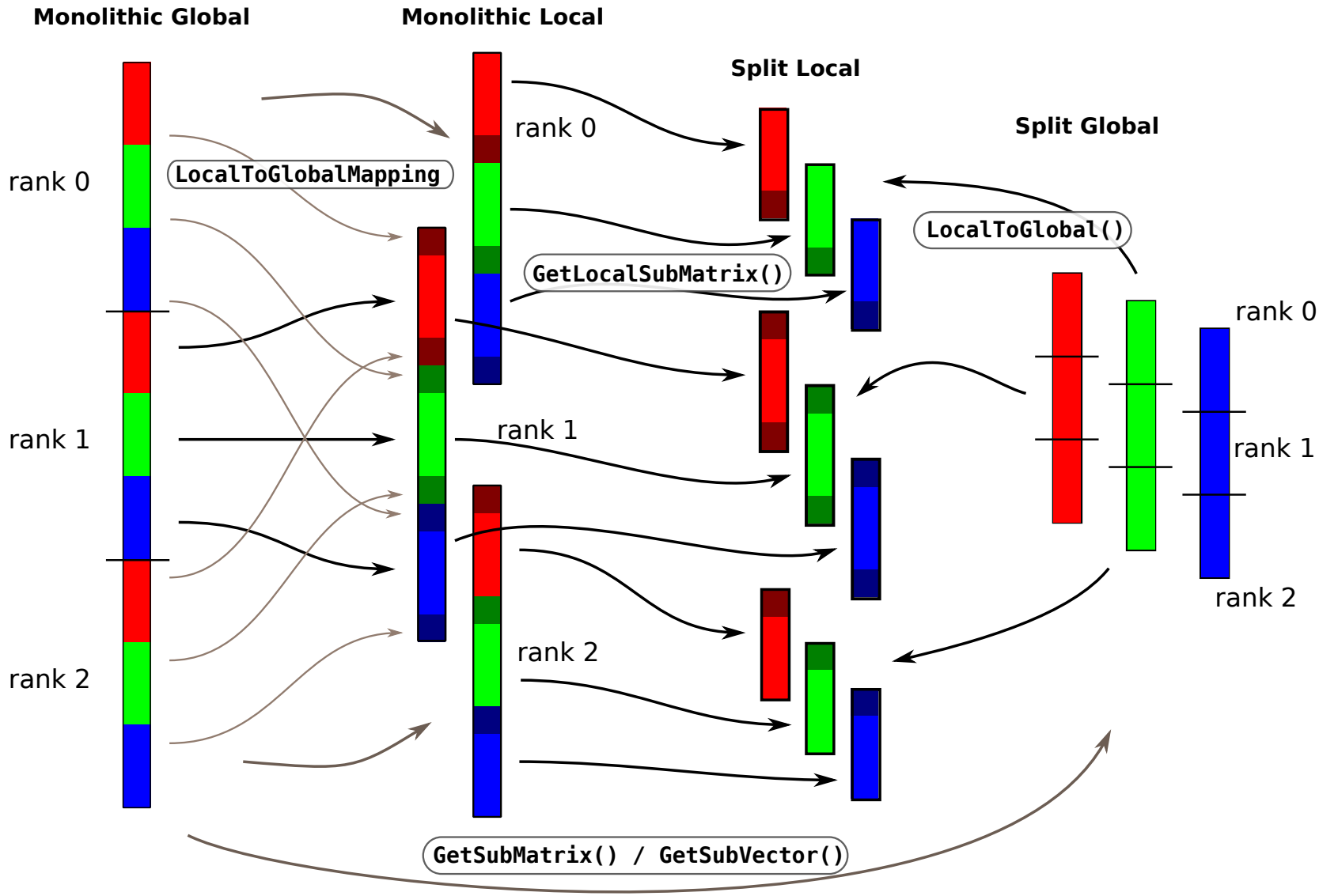
- Relaxation: `-pc_fieldsplit_type [additive,multiplicative,symmetric_multiplicative]`

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

  - 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
- “Composable Linear Solvers for Multiphysics” ISPDC 2012



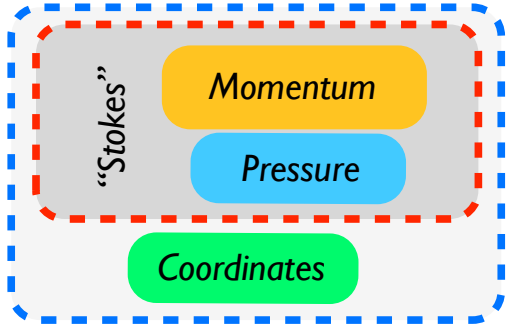
Work in Split Local space, matrix data structures reside in any space.

Stokes + Implicit Free Surface

$$\left[\eta D_{ij}(\boldsymbol{u})\right]_{,j} - p_{,i} = f_i$$

$$u_{k,k} = 0$$

$$\hat{x}_i = \hat{x}_i^{t-\Delta t} + \Delta t\, u_i(\hat{x}_i)$$



COORDINATE RESIDUALS

$$F_x := -u_i + \frac{\hat{x}_i}{\Delta t} - \frac{\hat{x}_i^{t-\Delta t}}{\Delta t}$$

[We use a full Lagrangian update of our mesh, with no remeshing]

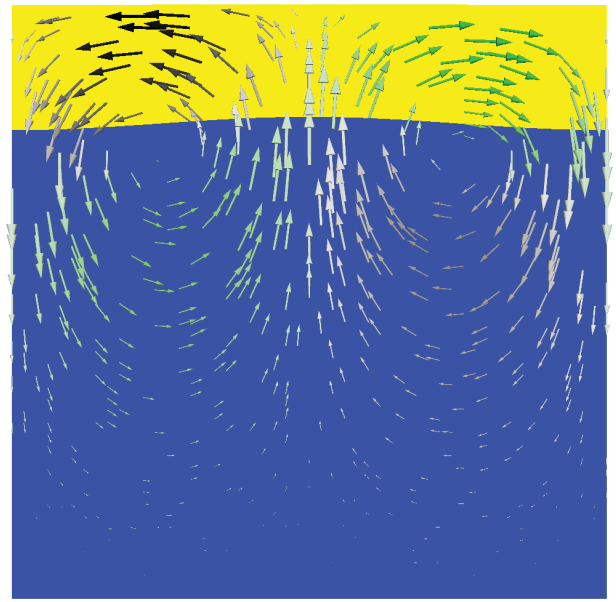
JACOBIAN

$$\mathcal{J}_{si} = \begin{bmatrix} A + \delta_{\hat{x}} A & B + \delta_{\hat{x}} B & J_{ac} \\ B^T + \delta_{\hat{x}} B^T & 0 & J_{bc} \\ -I & 0 & \frac{I}{\Delta t} \end{bmatrix}$$

Reuse stokes operators and saddle point preconditioners

NESTED PRECONDITIONER

$$\mathcal{P}_{si} = \begin{bmatrix} \left[\ \mathcal{P}_s^l\ \right] \\ \left[\ I\ \right]\left[\ -\frac{I}{\Delta t}\ \right] \end{bmatrix} \qquad \mathcal{P}_s^l = \begin{bmatrix} A & 0 \\ B^T & -S \end{bmatrix}$$



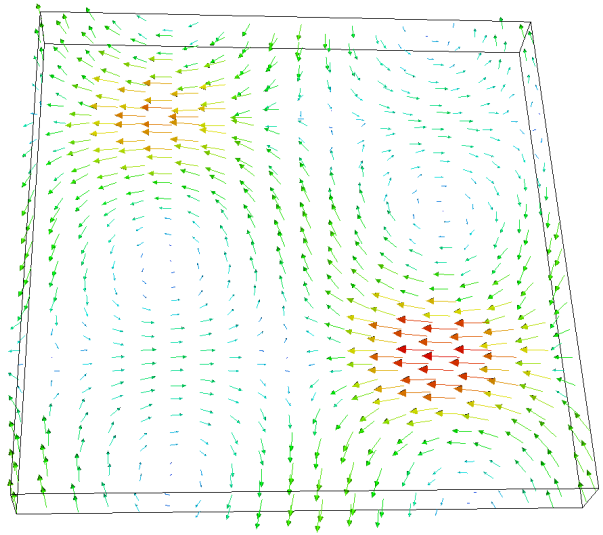
"Drunken seaman", Rayleigh Taylor instability test case from Kaus et al., 2010. Dense, viscous material (yellow) overlying less dense, less viscous material (blue).

Eigen-analysis plugin for solver design

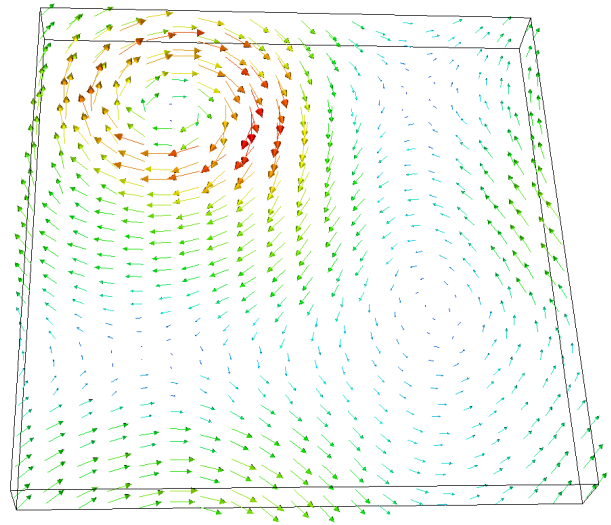
Hydrostatic ice flow (nonlinear rheology and slip conditions)

$$-\nabla\left[\eta\begin{pmatrix}4u_x+2v_y & u_y+v_x & u_z\\ u_y+v_x & 2u_x+4v_y & v_z\end{pmatrix}\right]+\rho g\nabla s=0, \qquad (1)$$

- Many solvers converge easily with no-slip/frozen bed, more difficult for slippery bed (ISMIP HOM test C)
- Geometric MG is good: $\lambda \in [0.805, 1]$ (SISC 2013)

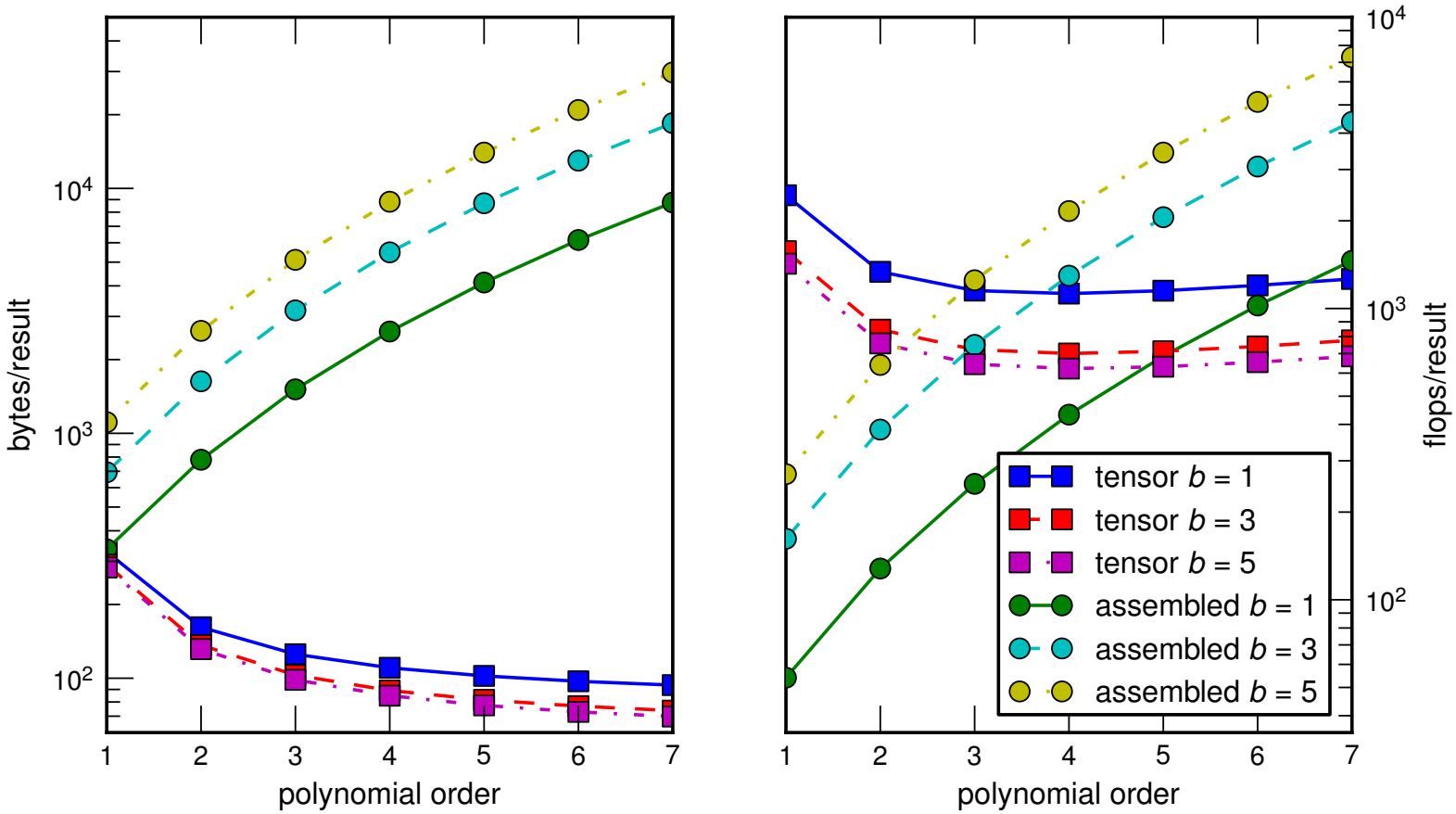


(a) $\lambda_0 = 0.0268$



(b) $\lambda_1 = 0.0409$

Performance of assembled versus unassembled



- Arithmetic intensity for Q_p elements
  - $< \frac{1}{4}$ (assembled), ≈ 10 (unassembled), $\gtrsim 5$ (hardware)
- store Jacobian information at Gauss quadrature points, can use AD

Hardware Arithmetic Intensity

Operation	Arithmetic Intensity (flops/B)
Sparse matrix-vector product	1/6
Dense matrix-vector product	1/4
Unassembled matrix-vector product, residual	$\gtrsim 8$

Processor	STREAM Triad (GB/s)	Peak (GF/s)	Balance (F/B)
E5-2680 8-core	38	173	4.5
E5-2695v2 12-core	45	230	5.2
E5-2699v3 18-core	60	660	11
Blue Gene/Q node	29.3	205	7
Kepler K20Xm	160	1310	8.2
Xeon Phi SE10P	161	1060	6.6
KNL (DRAM)	100	3000	30
KNL (MCDRAM)	500	3000	6

Q_2 tensor product optimization

- Reference gradient $\mathscr{D}_\xi = [\hat{D}\otimes\hat{B}\otimes\hat{B}, \hat{B}\otimes\hat{D}\otimes\hat{B}, \hat{B}\otimes\hat{B}\otimes\hat{D}]$
- $\nabla_\xi \mathbf{x} = (\mathscr{D}_\xi \otimes I_3)(\mathscr{E}_e \otimes I_3)\mathbf{x}$ (29%)
- Invert 3×3 at quad. points: $\nabla_{\mathbf{x}}\xi$ (7%)

$$A\mathbf{u} = \sum_{e\in N_{\text{el}}} \underbrace{\mathscr{E}_e^T}_{\text{scatter accum}} \overbrace{\mathscr{D}_\xi^T}^{\text{tensor 29\%}} \underbrace{\Lambda\Big((\nabla_{\mathbf{x}}\xi)^T(\omega\eta)(\nabla_{\mathbf{x}}\xi)\Big)}_{\text{independent at quadrature points 6\%}} \overbrace{\mathscr{D}_\xi}^{\text{tensor 29\%}} \underbrace{\mathscr{E}_e}_{\text{gather}} \mathbf{u}$$

- Pack 4 elements at a time in vector-friendly ordering
- Intrinsics, 30% of peak AVX (SNB) and FMA (Haswell)
- Similar structure in HPGMG-FE (`https://hpgmg.org`)

Operator	Flops	Pessimal Cache Bytes	Pessimal Cache F/B	Perfect Cache Bytes	Perfect Cache F/B	Time (ms)	GF/s
Assembled	9216	—	—	37248	0.247	42	113
Matrix-free	53622	2376	22.5	1008	53	22	651
Tensor	15228	2376	6.4	1008	15	**4.2**	1072
Tensor C	14214	5832	2.4	4920	2.9	—	—

Continental rifting

Rifting Video

[May, Brown, Le Pourhiet (SC14)]

Sparse linear algebra is dead (long live sparse . . .)

- Arithmetic intensity $< 1/4$
- Idea: multiple right hand sides

$$\frac{(2k \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)} + \texttt{sizeof(Int)}}, \quad k \ll \text{avg. nz/row}$$

- Problem: popular algorithms have nested data dependencies
  - Time step
    Nonlinear solve
    Krylov solve
    Preconditioner/sparse matrix
- Cannot parallelize/vectorize these nested loops
- Can we create new algorithms to reorder/fuse loops?
  - Reduce latency-sensitivity for communication
  - Reduce memory bandwidth (reuse matrix while in cache)

Sparse linear algebra is dead (long live sparse . . .)

- Arithmetic intensity $< 1/4$
- Idea: multiple right hand sides

$$\frac{(2k \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)} + \texttt{sizeof(Int)}}, \quad k \ll \text{avg. nz/row}$$

- Problem: popular algorithms have nested data dependencies
  - Time step
    Nonlinear solve
    Krylov solve
    Preconditioner/sparse matrix
- Cannot parallelize/vectorize these nested loops
- Can we create new algorithms to reorder/fuse loops?
  - Reduce latency-sensitivity for communication
  - Reduce memory bandwidth (reuse matrix while in cache)

Runge-Kutta methods

$$\dot{u} = F(u)$$

$$\underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix}}_{Y} = u^n + h \underbrace{\begin{bmatrix} a_{11} & \cdots & a_{1s} \\ \vdots & \ddots & \vdots \\ a_{s1} & \cdots & a_{ss} \end{bmatrix}}_{A} F \begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix}$$

$$u^{n+1} = u^n + b^T Y$$

- General framework for one-step methods
- Diagonally implicit: A lower triangular, stage order 1 (or 2 with explicit first stage)
- Singly diagonally implicit: all A_{ii} equal, reuse solver setup, stage order 1
- If A is a general full matrix, all stages are coupled, “implicit RK”

Method of Butcher (1976) and Bickart (1977)

- Newton linearize Runge-Kutta system at u^*

$$Y = u^n + hAF(Y) \qquad \left[I_s \otimes I_n + hA \otimes J(u^*)\right] \delta Y = RHS$$

- Solve linear system with tensor product operator

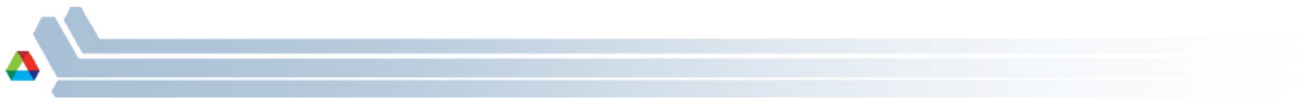
$$\hat{G} = S \otimes I_n + I_s \otimes J$$

  where $S = (hA)^{-1}$ is $s \times s$ dense, $J = -\partial F(u)/\partial u$ sparse
- SDC (2000) is Gauss-Seidel with low-order corrector
- Butcher/Bickart method: diagonalize $S = X \Lambda X^{-1}$
  - $\Lambda \otimes I_n + I_s \otimes J$
  - s decoupled solves
  - Complex eigenvalues (overhead for real problem)
- Problem: X is exponentially ill-conditioned wrt. s
- We avoid diagonalization
  - Permute $\hat{G}$ to reuse J: $G = I_n \otimes S + J \otimes I_s$
  - Stages coupled via register transpose at spatial-point granularity
  - Same convergence properties as Butcher/Bickart

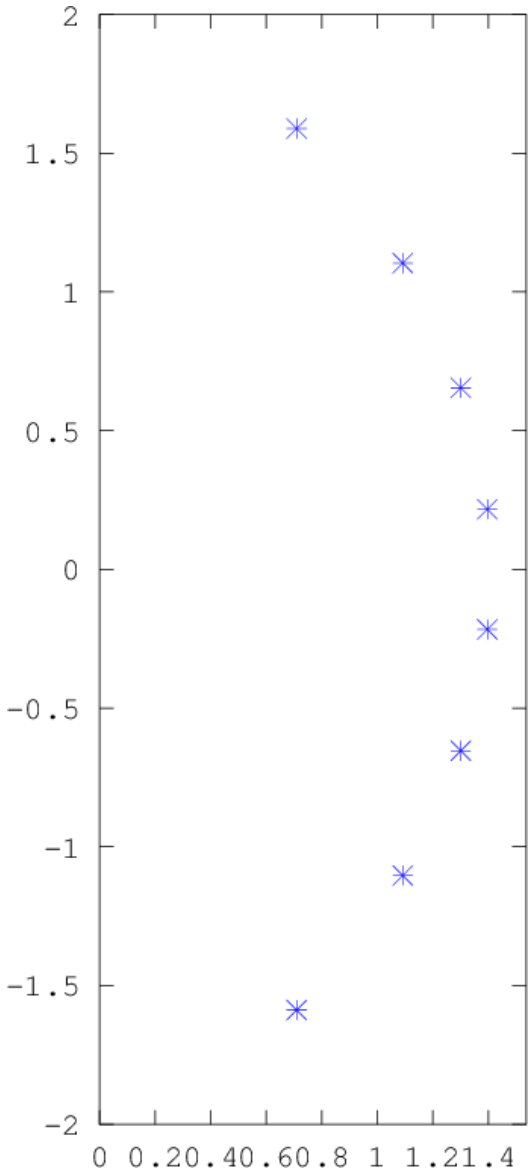
MatTAIJ: “sparse” tensor product matrices

$$G = I_n \otimes S + J \otimes T$$

- J is parallel and sparse, S and T are small and dense
- More general than multiple RHS (multivectors)
- Compare $J \otimes I_s$ to multiple right hand sides in row-major
- Runge-Kutta systems have $T = I_s$ (permuted from Butcher method)
- Stream J through cache once, same efficiency as multiple RHS
- Unintrusive compared to spatial-domain vectorization or s-step

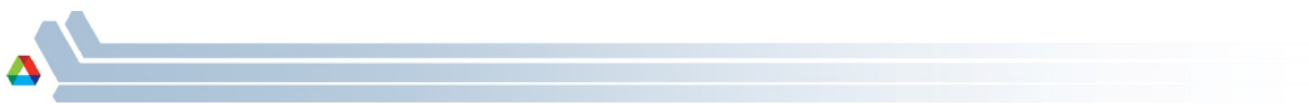


Multigrid for Implicit Runge-Kutta: Diffusion



- Prolongation: $P \otimes I_s$
- Coarse operator: $I_n \otimes S + (RJP) \otimes I_s$
- Larger time steps
- GMRES(2)/point-block Jacobi smoothing
- FGMRES outer

Method	order	nsteps	Krylov its.	(Average)
Gauss 1	2	16	82	(5.1)
Gauss 2	4	8	64	(8)
Gauss 4	8	4	44	(11)
Gauss 8	16	2	42	(21)



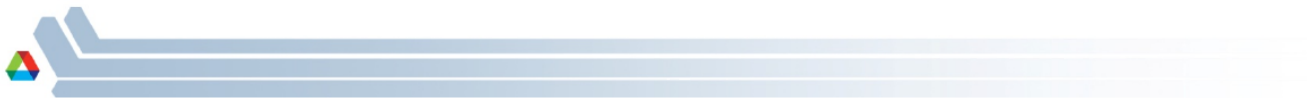
IMEX time integration in PETSc

- Additive Runge-Kutta IMEX methods

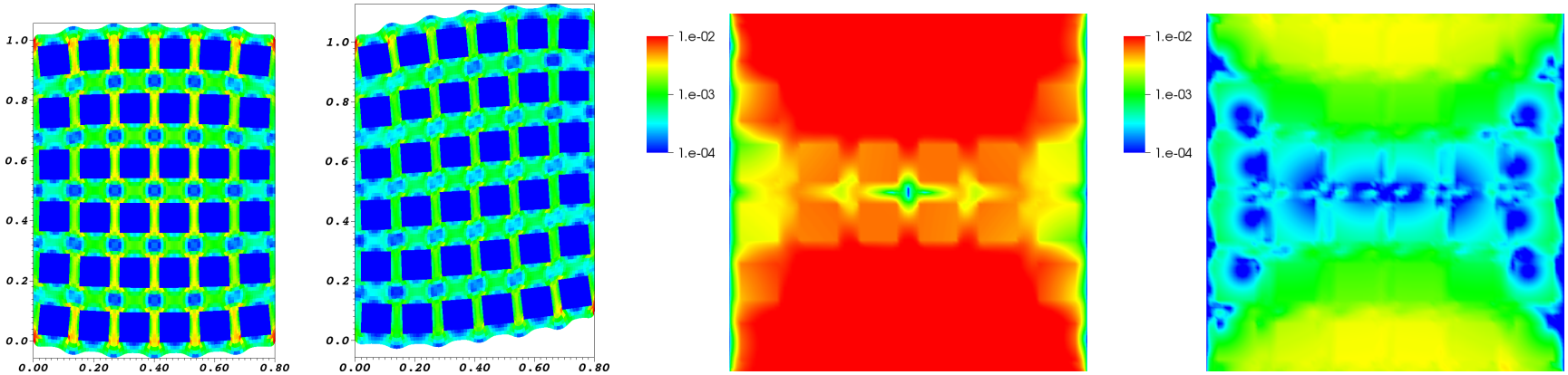
$$G(t, x, \dot{x}) = F(t, x)$$

$$J_\alpha = \alpha G_{\dot{x}} + G_x$$

  - User provides:
    - `FormRHSFunction(ts,`t`,`x`,`F`,void *ctx);`
    - `FormIFunction(ts,`t`,`x`,`$\dot{x}$`,`G`,void *ctx);`
    - `FormIJacobian(ts,`t`,`x`,`$\dot{x}$`,`α`,`J`,`J_p`,mstr,void *ctx);`
  - Can have L-stable DIRK for stiff part G, SSP explicit part, etc.
  - Orders 2 through 5, embedded error estimates
  - Dense output, hot starts for Newton
  - More accurate methods if G is linear, also Rosenbrock-W
  - Can use preconditioner from classical “semi-implicit” methods
  - FAS nonlinear solves supported
  - Extensible adaptive controllers, can change order within a family
  - Easy to register new methods: `TSARKIMEXRegister()`
- Single step interface so user can have own time loop
- Same interface for Extrapolation IMEX, LMS IMEX (in development)



τ corrections

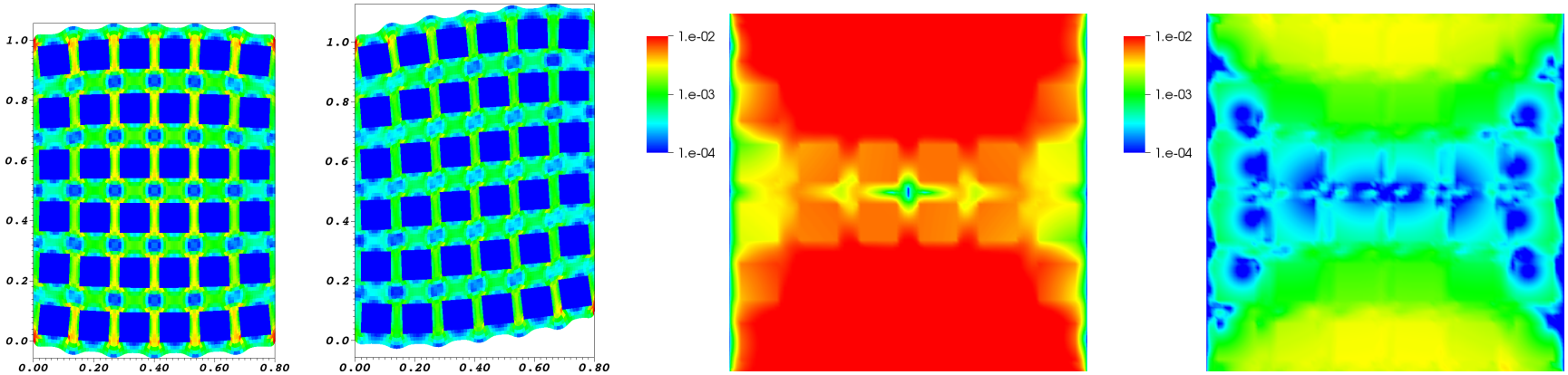


- Plane strain elasticity, $E = 1000, \nu = 0.4$ inclusions in $E = 1, \nu = 0.2$ material, coarsen by 3^2.
- Solve initial problem everywhere and compute $\tau_h^H = A^H \hat{I}_h^H u^h - I_h^H A^h u^h$
- Change boundary conditions and solve FAS coarse problem

$$N^H \acute{u}^H = \underbrace{I_h^H \acute{f}^h}_{\acute{f}^H} + \underbrace{N^H \hat{I}_h^H \tilde{u}^h - I_h^H N^h \tilde{u}^h}_{\tau_h^H}$$

- Prolong, post-smooth, compute error $e^h = \acute{u}^h - (N^h)^{-1} \acute{f}^h$
- Coarse grid *with* τ is nearly $10\times$ better accuracy

τ corrections



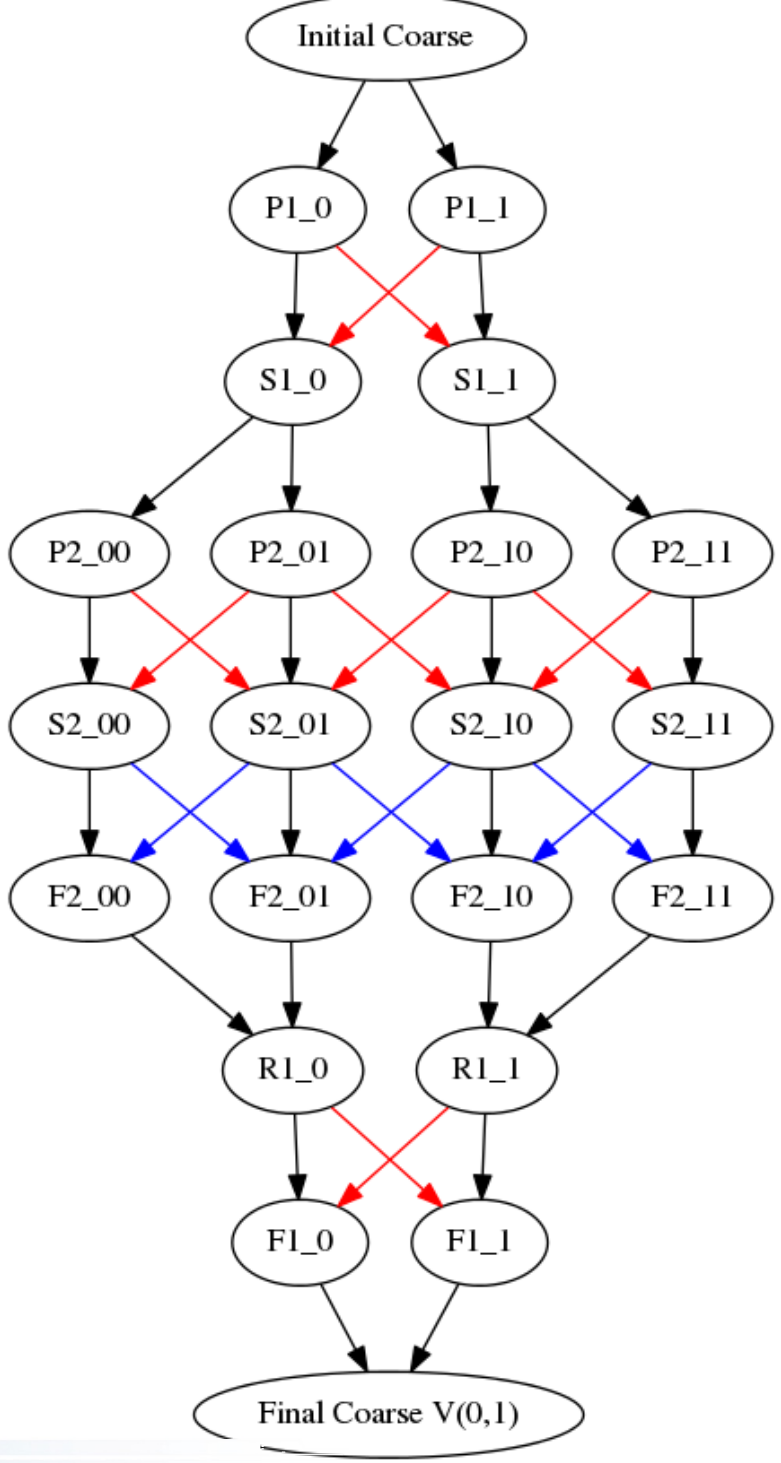
- Plane strain elasticity, $E = 1000, \nu = 0.4$ inclusions in $E = 1, \nu = 0.2$ material, coarsen by 3^2.
- Solve initial problem everywhere and compute $\tau_h^H = A^H \hat{I}_h^H u^h - I_h^H A^h u^h$
- Change boundary conditions and solve FAS coarse problem

$$N^H \acute{u}^H = \underbrace{I_h^H \acute{f}^h}_{\acute{f}^H} + \underbrace{N^H \hat{I}_h^H \tilde{u}^h - I_h^H N^h \tilde{u}^h}_{\tau_h^H}$$

- Prolong, post-smooth, compute error $e^h = \acute{u}^h - (N^h)^{-1} \acute{f}^h$
- Coarse grid *with* τ is nearly $10\times$ better accuracy

Low communication MG

- red arrows can be removed by τ-FAS with overlap
- blue arrows can also be removed, but then algebraic convergence stalls when discretization error is reached
- no simple way to check that discretization error is obtained
- if fine grid state is not stored, use compatible relaxation to complete prolongation *P*
- “Segmental refinement” by Achi Brandt (1977)
- 2-process case by Brandt and Diskin (1994)

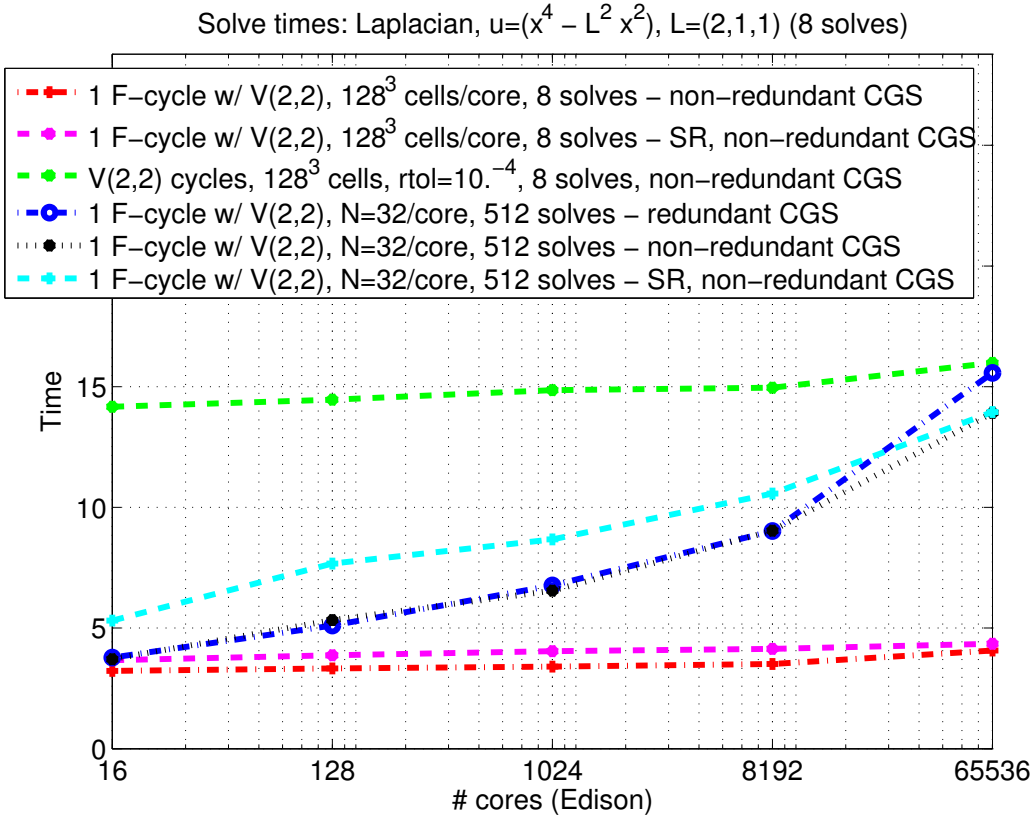


Segmental refinement: no horizontal communication

- 27-point second-order stencil, manufactured analytic solution
- 5 SR levels: 16^3 cells/process local coarse grid
- Overlap $=$ Base $+ (L - \ell)$Increment
  - Implementation requires even number of cells—round down.
- FMG with $V(2,2)$ cycles

Table: $\|e_{SR}\|_\infty / \|e_{FMG}\|_\infty$

Increment	Base 1	Base 2	Base 3
1	1.59	2.34	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00



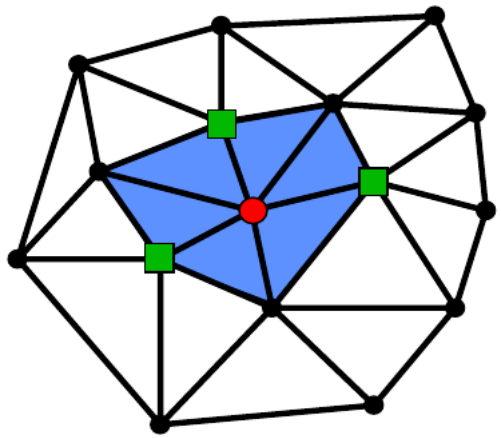
Adams, Brown, Knepley, Samtaney arXiv:1406.7808

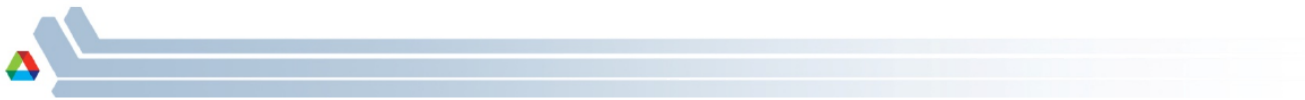
Nonlinear and matrix-free smoothing

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

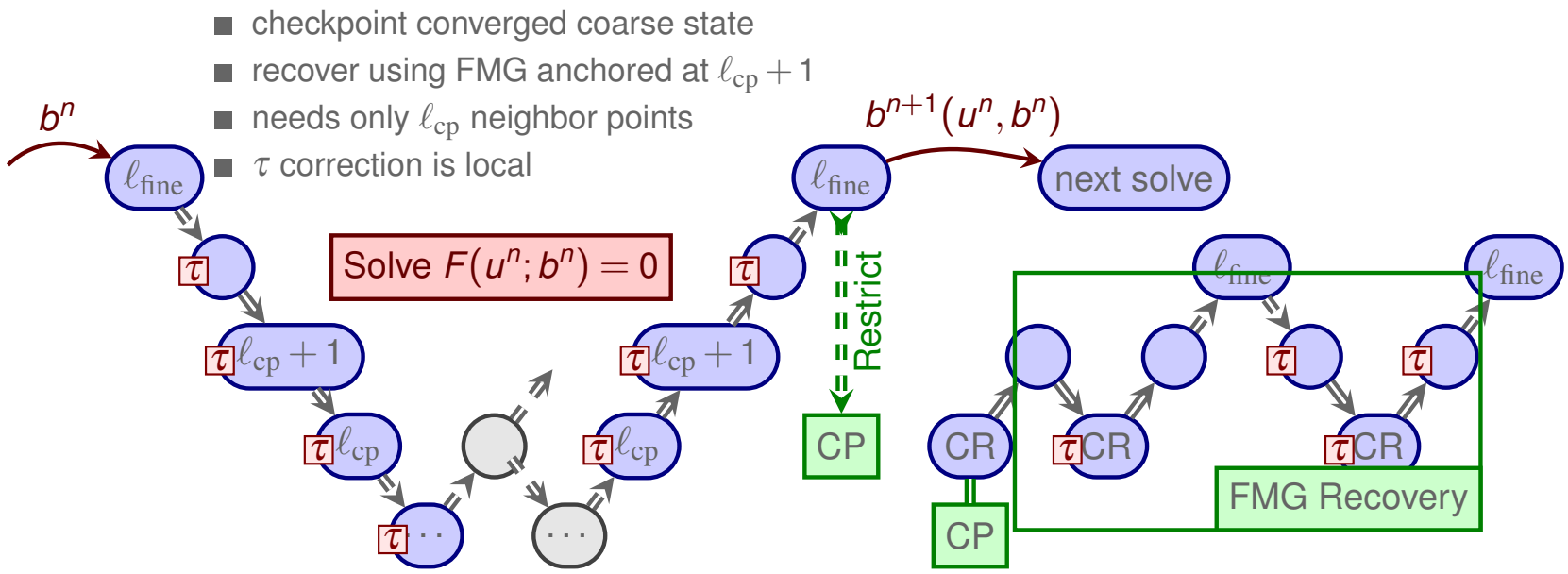
$$C = \frac{(\text{cost to evaluate residual at one “point”}) \cdot N}{(\text{cost of global residual})} \sim 1$$

  - 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
- additive correction (Jacobi/Chebyshev/multi-stage)
  - global evaluation, as good as $C = 1$
  - but, need to assemble corrector/scaling
  - need spectral estimates or wave speeds

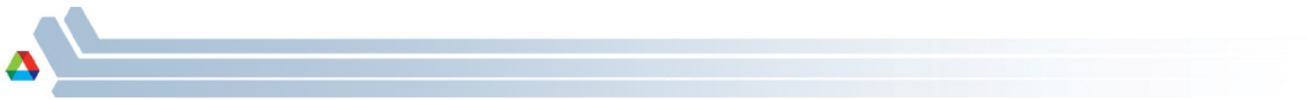




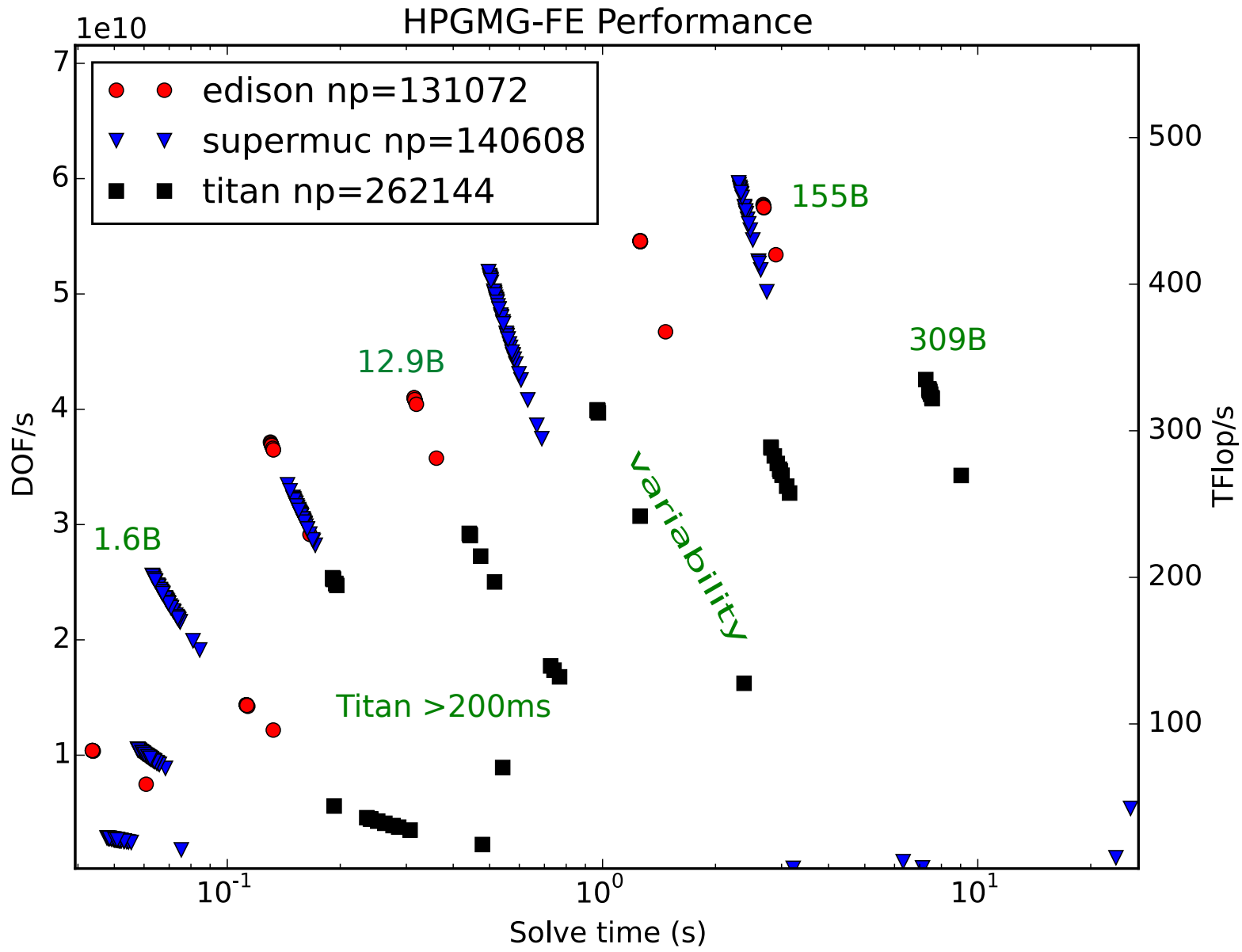
Multiscale compression and recovery using τ form



- Normal multigrid cycles visit all levels moving from $n \longrightarrow n+1$
- FMG recovery only accesses levels finer than ℓ_{CP}
- Only failed processes and neighbors participate in recovery
- Lightweight checkpointing for transient adjoint computation
- Postprocessing applications, e.g., in-situ visualization at high temporal resolution in part of the domain



HPGMG-FE https://hpgmg.org



Outlook

- Many more applications
  - subsurface, aerospace, engines, plasma, atmosphere/ocean, nuclear materials engineering, biological fluids and solvation
- Keep science objectives firmly in the driver's seat
- Do not compromise discretization quality
- Optimize convergence first, then implementation efficiency
- Library design: create maximally reusable components
- Users want performance versatility
  - fast solution on supercomputer
  - high-resolution on small workstation/cluster
- Robustness is always a challenge
- Huge scope remains at problem formulation
- Algorithmic optimality is crucial
- Real problems are messy
- Performance is always messy at scale
- Ideas are easy, implementation and practical issues are hard
- Maximize science per Watt

HPC Performance Modeling and Analysis

- Spring 2015, Special Topics
- Architectural roadmaps and modeling
- Performance variability
- Designing performance experiments
- Application case studies
- Target audience:
  - Advanced undergraduate and graduate students in Computer Science
  - Graduate students in simulation-based science or engineering
- `jed@jedbrown.org`