Exploits in Implicitness

hardware, problem structure, and library design

Jed Brown jedbrown@mcs.anl.gov (ANL and CU Boulder)
Collaborators in this work:
Mark Adams (LBL), Peter Brune (ANL), Emil Constantinescu (ANL),
Debojyoti Ghosh (ANL), Matt Knepley (UChicago),
Dave May (ETH Zürich, Lois Curfman McInnes (ANL),
Barry Smith (ANL))

SIAM PP, 2014-02-21

Why implicit?

- Nature has many spatial and temporal scales
  - Porous media, structures, fluids, kinetics
- Science/engineering problem statement does not weak scale
  - More time steps required at high 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- Sparse matrices are comfortable, but outdated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
  - optimal solvers must accurately handle all scales
  - optimality is crucial for large-scale problems
  - hardware puts up a spirited fight to abstraction

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

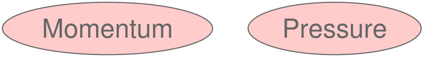
Split

- 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

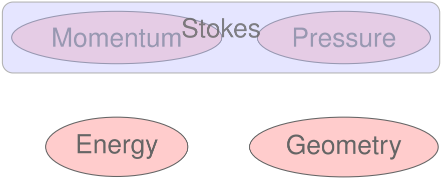


- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

Multi-physics coupling in PETSc

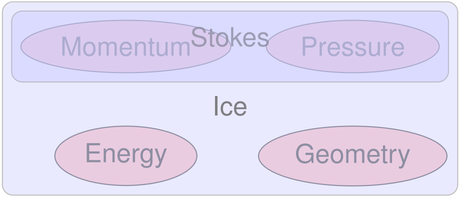
- package each "physics" independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

Multi-physics coupling in PETSc



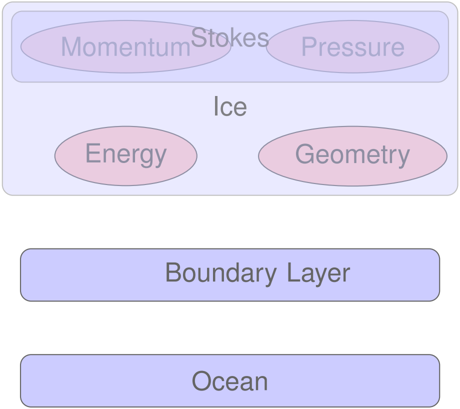
- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

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

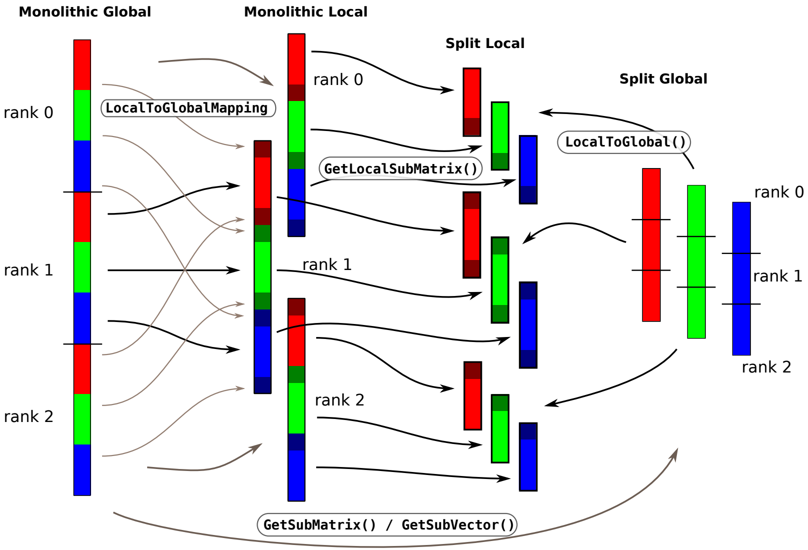
- Relaxation: `-pc_fieldsplit_type [additive,multiplicative,symmetric_multiplicative]`

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

  - 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



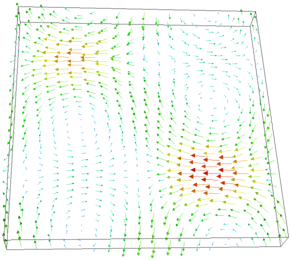
Work in Split Local space, matrix data structures reside in any space.

Eigen-analysis plugin for solver design

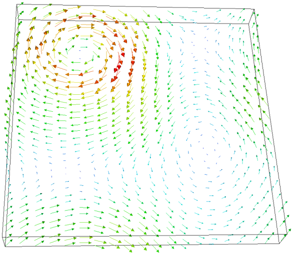
Hydrostatic ice flow (nonlinear rheology and slip conditions)

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

- 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) $\lambda_1 = 0.0409$

Plugins in PETSc

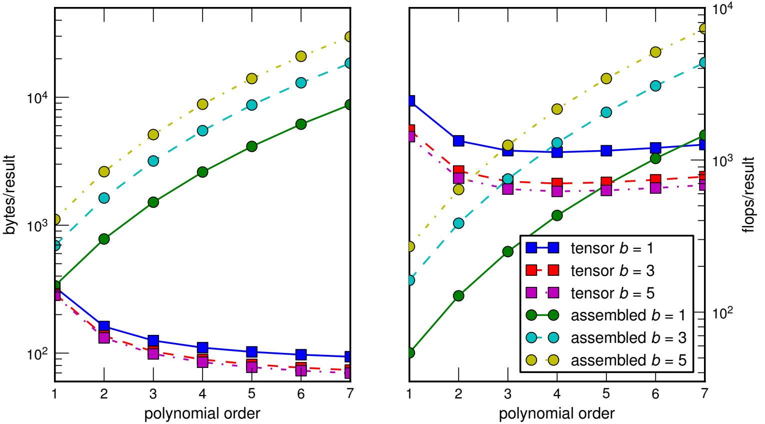
Philosophy: Everything has a plugin architecture

- Vectors, Matrices, Coloring/ordering/partitioning algorithms
- Preconditioners, Krylov accelerators
- Nonlinear solvers, Time integrators
- Spatial discretizations/topology*

Example

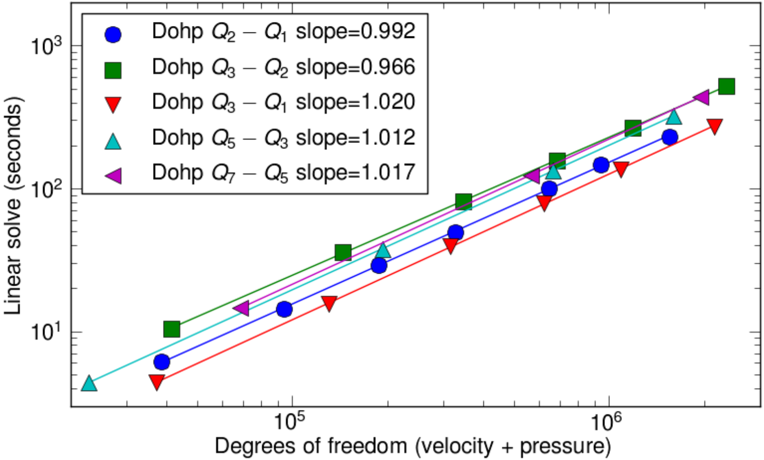
Third party supplies matrix format and associated preconditioner, distributes compiled shared library. Application user loads plugin at runtime, no source code in sight.

Performance of assembled versus unassembled



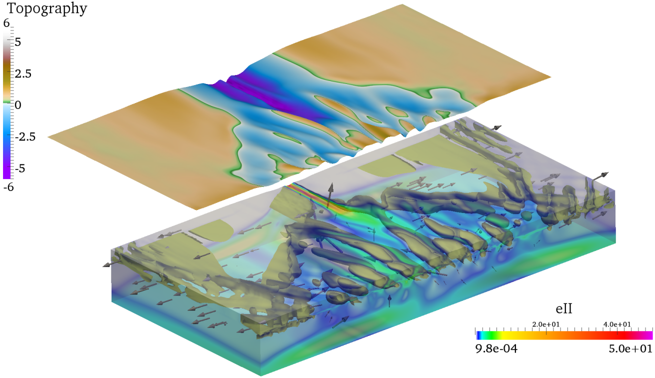
- Arithmetic intensity for Q_p elements
  - $< \frac{1}{4}$ (assembled), ≈ 10 (unassembled), ≈ 4 to 8 (hardware)
- store Jacobian information at Quass quadrature points, can use AD

Power-law Stokes Scaling



Only assemble Q_1 matrices, ML+PETSc smoothers for elliptic pieces
(fairly easy geometry and coefficients, Brown 2010 (J.Sci.Comput.))

pTatin3d: Long-term lithosphere dynamics



- Dave May (ETH Zürich), Laetitia Le Pourhiet (UPMC Paris)
- Visco-elasto-plastic rheology
- Material-point method for material composition, 10^{10} jumps
- Large deformation, post-failure analysis
- Free surface: $Q_2 - P_1^{\mathrm{disc}}$ (non-affine)

pTatin3d: Long-term lithosphere dynamics

- Assembled matrices: $9216F/38912B = 0.235F/B$
  - Problem size limited by memory
  - Mediocre performance, limited by memory bandwidth
  - Poor scalability within a node (memory contention)
  - Lots of experimentation with different algorithms
  - Multigrid: matrix-free on finest levels
- Matrix-free: $51435F/824B = 62.42F/B$
  - 81×27 element gradient matrix
  - Element setup computes physical gradient matrix
  - $1.5\times$ speedup when using all cores
- Tensor-product matrix-free: $16686F/824B = 20.25F/B$
  - Tensor contractions with 3×3 1D matrices
  - Tiny working set, vectorize over 4 elements within L1 cache
  - 30% of Haswell FMA peak, register load/store limited
  - $7\times$ speedup ($5\times$ speedup on Sandy Bridge AVX)

Hardware Arithmetic Intensity

Operation	Arithmetic Intensity (flops/B)
Sparse matrix-vector product	1/6
Dense matrix-vector product	1/4
Unassembled matrix-vector product	≈ 8
High-order residual evaluation	> 5

Processor	Bandwidth (GB/s)	Peak (GF/s)	Balance (F/B)
E5-2680 8-core	38	173	4.5
Magny Cours 16-core	49	281	5.7
Blue Gene/Q node	26	205	7.9
Tesla M2090	120	665	5.5
Kepler K20Xm	160	1310	8.2
Xeon Phi SE10P	161	1060	6.6

This is a dead end

- 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)
  - Implicit Runge-Kutta, creates tensor product structure
  - Full space/one-shot methods for PDE-constrained optimization

This is a dead end

- 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)
  - Implicit Runge-Kutta, creates tensor product structure
  - Full space/one-shot methods for PDE-constrained optimization

Beyond global linearization: FAS multigrid

p-Laplacian, $p = 4$, $c = 0.5$, MG-like preconditioners

Solv.	T	N. It	L. It	Func	Jac	PC	NPC
NK-MG	9.904	105	384	421	630	489	–
NK-MG-FAS	1.012	4	13	65	24	17	4
FASPIN	1.424	4	18	368	–	22	23
FAS	0.872	15	0	226	–	–	–
NCG-LFAS	1.376	8	0	400	–	–	25
NCG-RFAS	0.792	10	0	181	–	–	10
QN-LFAS	1.344	8	0	400	–	–	25
QN-RFAS	1.104	16	0	289	–	–	16
NGMRESL-FAS	0.684	7	0	128	–	–	8
NGMRESR-FAS	0.648	8	0	129	–	–	8

- Geometric coarse grids and rediscretization

Lagged quasi-Newton for nonlinear elasticity

Method	Lag	LS	Linear Solve	Its.	$F(u)$	Jacobian	P^{-1}
LBFGS	3	cp	`preonly`	18	37	5	18
LBFGS	3	cp	10^{-5}	21	43	6	173
LBFGS	6	cp	`preonly`	24	49	4	24
LBFGS	6	cp	10^{-5}	30	61	5	266
JFNK	0	cp	`preonly`	11	23	11	11
JFNK	0	cp	10^{-5}	8	69	8	60
JFNK	1	cp	`preonly`	15	31	8	15
JFNK	1	cp	10^{-5}	7	2835	4	2827
JFNK	3	cp	`preonly`	23	47	6	23
JFNK	3	cp	10^{-5}	7	3143	2	3135

- B and Brune, MC2013

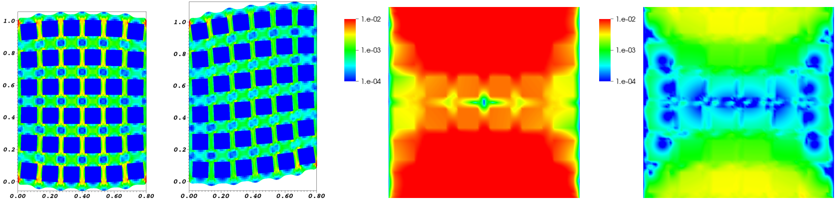
IMEX time integration in PETSc

- 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

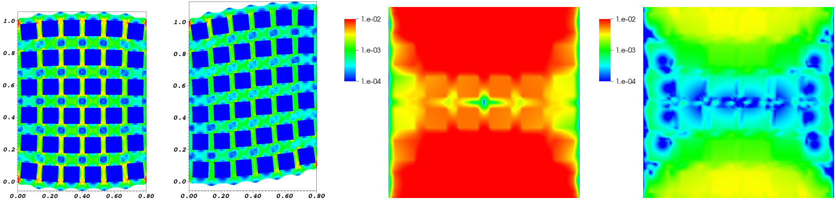


- Plane strain elasticity, $E = 1000, \nu = 0.4$ inclusions in $E = 1, \nu = 0.2$ material, coarsen by 3^2.
- Solve initial problem everywhere and compute $\tau_h^H = A^H \hat{I}_h^H u^h - I_h^H A^h u^h$
- Change boundary conditions and solve FAS coarse problem

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

- 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



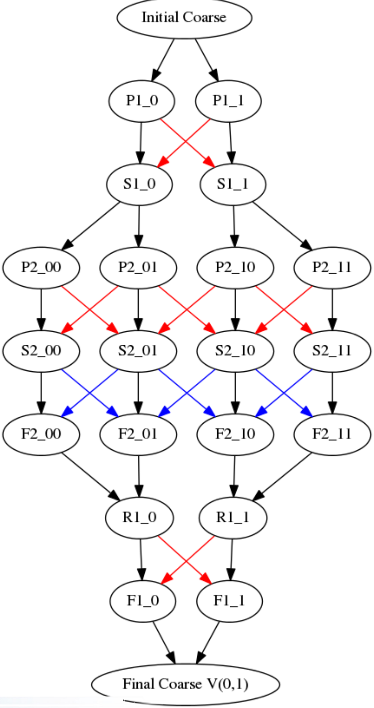
- Plane strain elasticity, $E = 1000, \nu = 0.4$ inclusions in $E = 1, \nu = 0.2$ material, coarsen by 3^2.
- Solve initial problem everywhere and compute $\tau_h^H = A^H \hat{I}_h^H u^h - I_h^H A^h u^h$
- Change boundary conditions and solve FAS coarse problem

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

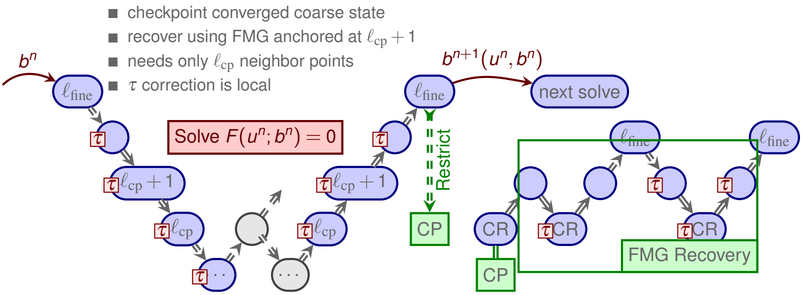
- Prolong, post-smooth, compute error $e^h = \acute{u}^h - (N^h)^{-1} \acute{f}^h$
- Coarse grid *with* τ is nearly $10\times$ better accuracy

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



Multiscale compression and recovery using τ form



- Normal multigrid cycles visit all levels moving from $n \to n+1$
- 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

- Maximize science per Watt
- Huge scope remains at problem formulation
- Raise level of abstraction at which a problem is formally specified
- Algorithmic optimality is crucial
- Improve matrix-free abstractions, robustness, diagnostics
- Better language/library support for aggregating