Tightly Coupled Solvers, Loosely Coupled Software

Multi-physics solvers and time integration in PETSc

Jed Brown, Emil Constantinescu, and Barry Smith

Argonne National Laboratory

2011-11-02

Multiphysics problems

Examples

- Saddle-point problems (e.g. incompressibility, contact)
- Stiff waves (e.g. low-Mach combustion)
- Mixed type (e.g. radiation hydrodynamics, ALE free-surface flows)
- Multi-domain problems (e.g. fluid-structure interaction)
- Full space PDE-constrained optimization

Software/algorithmic considerations

- Separate groups develop different "physics" components
- Do not know a priori which methods will have good algorithmic properties
- Achieving high throughput is more complicated
- Multiple time and/or spatial scales
  - Splitting methods are delicate, often not in asymptotic regime
  - Strongest nonlinearities usually non-stiff: prefer explicit for TVD limiters/shocks

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.

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

“Physics-based” preconditioners (semi-implicit method)

Shallow water with stiff gravity wave

h is hydrostatic pressure, u is velocity, $\sqrt{gh}$ is fast wave speed

$$h_t - (uh)_x = 0$$

$$(uh)_t + (u^2h + \frac{1}{2}gh^2)_x = 0$$

Semi-implicit method

Suppress spatial discretization, discretize in time, implicitly for the terms contributing to the gravity wave

$$\frac{h^{n+1} - h^n}{\Delta t} + (uh)_x^{n+1} = 0$$

$$\frac{(uh)^{n+1} - (uh)^n}{\Delta t} + (u^2h)_x^n + g(h^n h^{n+1})_x = 0$$

Rearrange, eliminating $(uh)^{n+1}$

$$\frac{h^{n+1} - h^n}{\Delta t} - \Delta t(gh^n h_x^{n+1})_x = -(\text{known})^n$$

Delta form

- Preconditioner should work like the Newton step: $-F(x) \mapsto \delta x$
- Recast semi-implicit method in delta form

$$\frac{\delta h}{\Delta t} + (\delta uh)_x = -F_0, \quad \frac{\delta uh}{\Delta t} + gh^n(\delta h)_x = -F_1, \quad \widehat{J} = \begin{pmatrix} \frac{1}{\Delta t} & \nabla\cdot \\ gh^n\nabla & \frac{1}{\Delta t} \end{pmatrix}$$

- Eliminate δuh

$$\frac{\delta h}{\Delta t} - \Delta t(gh^n(\delta h)_x)_x = -F_0 + (\Delta t F_1)_x, \quad S \sim \frac{1}{\Delta t} - g\Delta t \nabla\cdot h^n \nabla$$

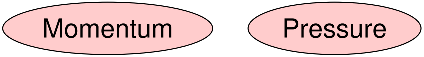
- Solve for δh, then evaluate

$$\delta uh = -\Delta t\left[gh^n(\delta h)_x - F_1\right]$$

- Fully implicit solver
  - Is nonlinearly consistent (no splitting error)
  - Implementation used same code as semi-implicit method
  - Efficient bifurcation analysis, steady-state analysis, data assimilation
- IMEX methods can also be high order, only need “stiff part” $\widehat{J}$

Outline

Multi-physics coupling in PETSc

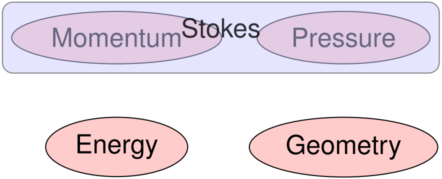


- package each "physics" independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

Multi-physics coupling in PETSc

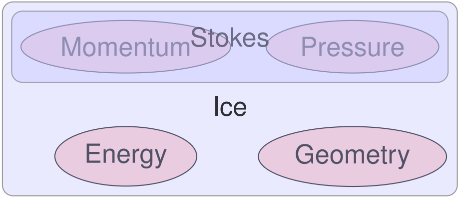
- package each "physics" independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

Multi-physics coupling in PETSc



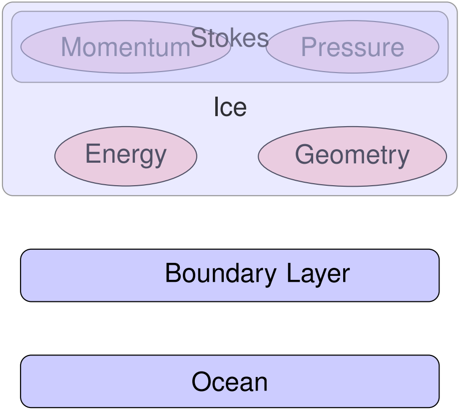
- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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

Multi-physics coupling in PETSc

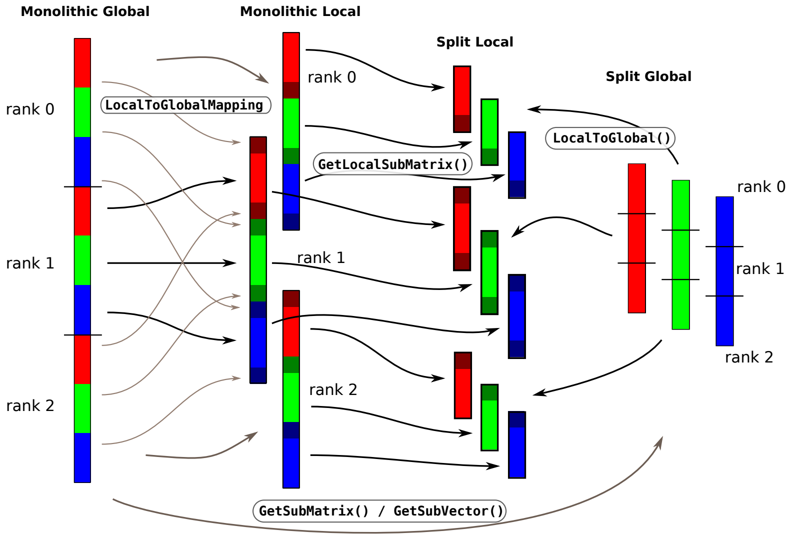


- package each “physics” independently
- solve single-physics and coupled problems
- semi-implicit and fully implicit
- reuse residual and Jacobian evaluation unmodified
- direct solvers, fieldsplit inside multigrid, multigrid inside fieldsplit without recompilation
- use the best possible matrix format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



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

`MatGetLocalSubMatrix(Mat A,IS rows,IS cols,Mat *B);`

- Primarily for assembly
  - `B` is not guaranteed to implement `MatMult`
  - The communicator for `B` is not specified, only safe to use non-collective ops (unless you check)
- `IS` represents an index set, includes a block size and communicator
- `MatSetValuesBlockedLocal()` is implemented
- MatNest returns nested submatrix, no-copy
- No-copy for Neumann-Neumann formats (unassembled across procs, e.g. BDDC, FETI-DP)
- Most other matrices return a lightweight proxy `Mat`
  - `COMM_SELF`
  - Values not copied, does not implement `MatMult`
  - Translates indices to the language of the parent matrix
  - Multiple levels of nesting are flattened

Outline

Conservative two-phase formulation

Find momentum density ρu, pressure p, and total energy density E:

$$(\rho u)_t + \nabla\cdot(\rho u \otimes u - \eta D u_i + p1) - \rho g = 0$$
$$\rho_t + \nabla\cdot\rho u = 0$$
$$E_t + \nabla\cdot\big((E+p)u - k_T \nabla T - k_\omega \nabla \omega\big) - \eta D u_i : D u_i - \rho u \cdot g = 0$$

- Solve for density ρ, ice velocity u_i, temperature T, and melt fraction ω using constitutive relations.
  - Simplified constitutive relations can be solved explicitly.
  - Temperature, moisture, and strain-rate dependent rheology η.
  - High order FEM, typically Q_3 momentum & energy, SUPG (yuck).
- DAEs solved implicitly after semidiscretizing in space.
- Preconditioning using nested fieldsplit

Relative effect of the blocks

$$J = \begin{pmatrix} J_{uu} & J_{up} & J_{uE} \\ J_{pu} & 0 & 0 \\ J_{Eu} & J_{Ep} & J_{EE} \end{pmatrix}.$$

- J_{uu} Viscous/momentum terms, nearly symmetric, variable coefficionts, anisotropy from Newton.
- J_{up} Weak pressure gradient, viscosity dependence on pressure (small), gravitational contribution (pressure-induced density variation). Large, nearly balanced by gravitational forcing.
- J_{uE} Viscous dependence on energy, very nonlinear, not very large.
- J_{pu} Divergence (mass conservation), nearly equal to J_{up}^T.
- J_{Eu} Sensitivity of energy on momentum, mostly advective transport. Large in boundary layers with large thermal/moisture gradients.
- J_{Ep} Thermal/moisture diffusion due to pressure-melting, $u \cdot \nabla$.
- J_{EE} Advection-diffusion for energy, very nonlinear at small regularization. Advection-dominated except in boundary layers and stagnant ice, often balanced in vertical.

How much nesting?

$$P_1 = \begin{pmatrix} J_{uu} & J_{up} & J_{uE} \\ 0 & B_{pp} & 0 \\ 0 & 0 & J_{EE} \end{pmatrix}$$

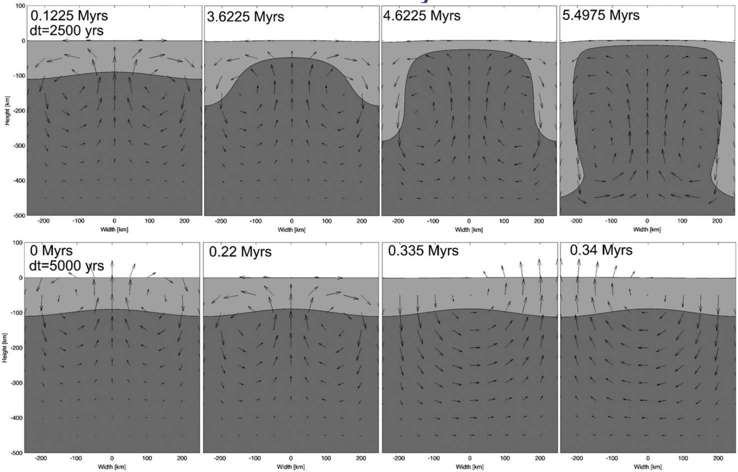
- B_{pp} is a mass matrix in the pressure space weighted by inverse of kinematic viscosity.
- Elman, Mihajlović, Wathen, JCP 2011 for non-dimensional isoviscous Boussinesq.
- Works well for non-dimensional problems on the cube, not for realistic parameters.

$$P = \begin{bmatrix} \begin{pmatrix} J_{uu} & J_{up} \\ J_{pu} & 0 \end{pmatrix} & \\ \begin{pmatrix} J_{Eu} & J_{Ep} \end{pmatrix} & J_{EE} \end{bmatrix}$$

- Inexact inner solve using upper-triangular with B_{pp} for Schur.
- Another level of nesting.
- GCR tolerant of inexact inner solves.
- Outer converges in 1 or 2 iterations.

- Low-order preconditioning full-accuracy unassembled high order operator.
- Build these on command line with PETSc `PCFieldSplit`.

The Drunken Seaman instability



- Subduction and mantle convection with a free surface.
- Free surface critical to long-term dynamics (e.g. mountain range formation)
- Advective 0.01 CFL for stability.
- Semi-implicit helps: Kaus, Mühlhaus, and May, 2010

Fully implicit free surface

- Velocity u, pressure p, Lagrangian/ALE coordinates x.

$$J = \begin{pmatrix} J_{uu} & J_{up} & J_{ux} \\ J_{pu} & 0 & J_{ux} \\ J_{xu} & 0 & J_{xx} \end{pmatrix}.$$

- Precondition with

$$P = \begin{bmatrix} \begin{pmatrix} J_{uu} & J_{up} \\ J_{pu} & 0 \end{pmatrix} & \\ \begin{pmatrix} J_{xu} & 0 \end{pmatrix} & J_{xx} \end{bmatrix} \qquad P_{\text{Stokes}} = \begin{pmatrix} J_{uu} & 0 \\ J_{pu} & B_{pp} \end{pmatrix}$$

- ```
  -dm_mat_type nest -pc_type fieldsplit
  -fieldsplit_stokes_pc_type fieldsplit
  -fieldsplit_stokes_pc_fieldsplit_type schur
  -fieldsplit_stokes_pc_fieldsplit_schur_factorization_type lower
  -fieldsplit_stokes_fieldsplit_u_pc_type mg
  ```
- ```
  -dm_mat_type aij -pc_type lu -pc_factor_mat_solver_package mumps
  ```

Outline

Coupling software

Applications

- Ice Flow
- Geodynamics

Implicit-Explicit time integration

Variational Inequalities

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);`
  - L-stable DIRK for stiff part G
  - Choice of explicit method, e.g. SSP
  - 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
  - Adaptive controllers, can change order within a family
  - Easy to register new methods
- Eliminate many interface quirks
- Single step interface so user can have own time loop

Outline

Variational Inequalities

- Supports inequality and box constraints on solution variables.
- Solution methods
  - Semismooth Newton
    - reformulate problem as a non-smooth system, Newton on subdifferential
    - Newton step solves diagonally perturbed systems
  - Active set
    - similar linear algebra to solving PDE
    - solve in reduced space by eliminating constrained variables
    - or enforce constraints by Lagrange multipliers
    - sometimes slower convergence or “bouncing”
- composes with multigrid and field-split
- demonstrated optimality for phase-field problems with millions of degrees of freedom

Outlook

- Unified algebraic interface for monolithic and nested formats
- Improves software modularity, but still manages stiff coupling
- Block and symmetric formats
- Multigrid inside or outside field splits
- Can use IMEX methods $g(t,x,\dot{x}) = f(t,x)$
- Variational inequalities
- Still to do:
  - Better preallocation for off-diagonal blocks
  - Nonlinear solvers for IMEX systems with structure
  - General/nonsymmetric pivoting in fieldsplit
  - Change of variables for fieldsplit
    (e.g. low-Mach Euler in conservative variables)