Numerical Libraries and Frameworks (PETSc)

Jed Brown jedbrown@mcs.anl.gov
Argonne National Lab and CU Boulder

ENES Workshop on Exascale Technologies, 2014-03-18
What can libraries offer?

- Code reuse
  - Porting/optimization to new architectures
  - ... but only the part of the problem solved by the library

- Easy experimentation with different methods
  - via run-time options (PETSc)
  - “black box” solvers are not sustainable
  - preconditioners, linear and nonlinear accelerators, time integrators

- Diagnostic and debugging support
  - Convergence monitors, error estimators, adaptive controllers
  - Compatibility checks
  - Eigen-analysis

- Communication with algorithm developers
  - Precise language to describe methods
  - Performance diagnostics

- Flexible coupling algorithms: beyond “first-order” splitting
Library or Framework?

<table>
<thead>
<tr>
<th>Library</th>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libraries provide a toolbox</td>
<td>Rapid development within a problem class</td>
</tr>
<tr>
<td>No assumptions about usage</td>
<td>End-to-end solution provides guidance and auxiliary tools</td>
</tr>
<tr>
<td>Any selection of libraries should be usable in combination</td>
<td>Opinionated</td>
</tr>
<tr>
<td><em>Extensible</em> libraries enable user to implement/extend</td>
<td>Hard to use in combination with other Frameworks</td>
</tr>
</tbody>
</table>

PETSc is a Library
<table>
<thead>
<tr>
<th>Library</th>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libraries provide a toolbox</td>
<td>Rapid development within a problem class</td>
</tr>
<tr>
<td>No assumptions about usage</td>
<td>End-to-end solution provides guidance and auxiliary tools</td>
</tr>
<tr>
<td>Any selection of libraries</td>
<td>Opinionated</td>
</tr>
<tr>
<td>should be usable in combination</td>
<td>Hard to use in combination with other Frameworks</td>
</tr>
<tr>
<td><em>Extensible</em> libraries enable</td>
<td></td>
</tr>
<tr>
<td>user to implement/extend</td>
<td></td>
</tr>
</tbody>
</table>

**PETSc is a Library**
Portable Extensible Toolkit for Scientific computing

Portable

- Runs *performantly* from laptop and iPhone to BG/Q and Titan
- Any compiler, any OS
- C, C++, Fortran 77 & 90+, Python, MATLAB
- Free to everyone: BSD-style license, open development

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.
Portable Extensible Toolkit for **Scientific computing**

- Computational Scientists and Engineers
  - Structural mechanics, CFD, Geodynamics, Subsurface flow, Reactor engineering, Fusion
  - Research (many countries, many agencies) and industry (oil and gas, aerospace, ABAQUS)
- Algorithm Developers (iterative methods and preconditioning)
  - Example: Ghysels’ pipelined Krylov methods
- Package Developers
  - SLEPc, TAO, Libmesh, MOOSE, FEniCS, Deal.II, etc
- Funding
  - Department of Energy (SciDAC, ASCR, collaborations)
  - National Science Foundation (CIG and others)
- Active development team with long-term commitment
- Hundreds of tutorial-style examples
- Hyperlinked manual, examples, and manual pages for all routines
- Lists: petsc-users@mcs.anl.gov, petsc-dev@mcs.anl.gov
- Support from petsc-maint@mcs.anl.gov
Solvers in climate

- “Pressure” solves for semi-implicit methods
  - Depends on separation between fastest wave and dynamics
- Time integration for atmospheric column physics
  - Currently swamped with splitting error
  - Stiff, positivity constraints, non-smoothness (freezing)
- Sea ice
  - Fast elastic wave speed ($v_p \approx 3 \text{ km s}^{-1}$)
  - Damped EVP model not converged at 120 subcycles, nor at 1200 (Lemieux et al. 2012)
- Land ice (Stokes and hydrostatic models with slippery bed)
  - PETSc: PISM (UAF, PIK), BISICLES (LBL, Chombo), ISSM (NASA)
- Improved stability for symplectic integration
- Accelerated spin-up (e.g., deep ocean)
  - Need to model unresolved-in-time processes
Impact of time step on autoconversion vs accretion partitioning (from Hui)

c/o Peter Caldwell (LLNL)

- Models calibrated for "efficient" time step
- No longer solving the PDEs we write down
- Expensive to recalibrate when discretization changes
- Calibration eats up a big chunk of the IPCC policy timeline
Sea Ice

\[(\rho hu)_t + \rho hfk \times u - \tau + \rho gh \nabla H_d - \nabla \cdot (\rho hu \otimes u - \sigma) = 0\]

\[\sigma = 2\eta \dot{e} + [(\zeta - \eta) \text{tr} \dot{e} - P/2]\]

- mildly nonsymmetric due to Coriolis (quasi-diagonal) and convection (small compared to viscous stresses)

- Nonlinear multigrid is less synchronous

<table>
<thead>
<tr>
<th>Method</th>
<th>Nonlinear its/stage</th>
<th>Linear its/stage</th>
<th>V-cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton-Krylov MG</td>
<td>6</td>
<td>30.44</td>
<td>30.44</td>
</tr>
<tr>
<td>FAS Newton/BJacobi/SOR</td>
<td>18.33</td>
<td>—</td>
<td>18.33</td>
</tr>
</tbody>
</table>

- Additive Runge-Kutta IMEX, error-based adaptivity, solver rtol $10^{-8}$

- Preliminary tests to 4096 cores of BG/Q and 64 fine-grid elements/process, less than 0.1 seconds/time step.
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}, \alpha, 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)
The Great Solver Schism: Monolithic or Split?

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

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

- Gauss-Seidel inspired, works when fields are loosely coupled

- **Factorization:** `-pc_fieldsplit_type schur`

\[
\begin{bmatrix}
A & B \\
S
\end{bmatrix}^{-1}
\begin{bmatrix}
1 & 1
\end{bmatrix}^{-1},
\quad S = D - CA^{-1}B
\]

- 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
Eigen-analysis plugin for solver design
Hydrostatic ice flow (nonlinear rheology and slip conditions)

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

- 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)

\((a) \; \lambda_0 = 0.0268 \quad \quad (b) \; \lambda_1 = 0.0409\)
Implicit Runge-Kutta for advection

Table: Total number of iterations (communications or accesses of $J$) to solve linear advection to $t = 1$ on a 1024-point grid using point-block Jacobi preconditioning of implicit Runge-Kutta matrix. The relative algebraic solver tolerance is $10^{-8}$.

<table>
<thead>
<tr>
<th>Family</th>
<th>Stages</th>
<th>Order</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crank-Nicolson/Gauss</td>
<td>1</td>
<td>2</td>
<td>3627</td>
</tr>
<tr>
<td>Gauss</td>
<td>2</td>
<td>4</td>
<td>2560</td>
</tr>
<tr>
<td>Gauss</td>
<td>4</td>
<td>8</td>
<td>1735</td>
</tr>
<tr>
<td>Gauss</td>
<td>8</td>
<td>16</td>
<td>1442</td>
</tr>
</tbody>
</table>

- Naive centered-difference discretization
A case for run-time configuration

- Simple build process
- Complete test suite without recompilation
- Cleaner provenance
  - Only need run-time configuration
  - No recompiles, only one binary to keep track of
  - Consistency checks in one place
- Simplified analysis/uncertainty quantification
  - More algorithms accessible
- More automated calibration
- Interface granularity is key to performance
Outlook

- **PETSc**: flexible, extensible, unintrusive
  - [http://mcs.anl.gov/petsc](http://mcs.anl.gov/petsc)
- Verification (converging the equations) encourages mathematicians
- Climate model components *should* become more library-like
  - Remove assumptions about environment
  - Improved modularity
  - Interfaces for configuration/calibration
  - Remove global variables (Fortran module variables)
- Tools need to make hard problems possible
  - Already many tools to make easy problems elegant
  - Ease of extending (versus DSLs/compilers)
- Strong-scaling necessity: ruthlessly shorten critical path
  - $2 \times$ increase in resolution requires at least $2 \times$ more steps
  - At fixed turn-around time, need twice as many steps/second
  - Algorithmic optimality is crucial