

PyLith

Brad Aagaard, Charles Williams, Matthew Knepley,
Sue Kientz and Leif Strand



May 12, 2009

PyLith

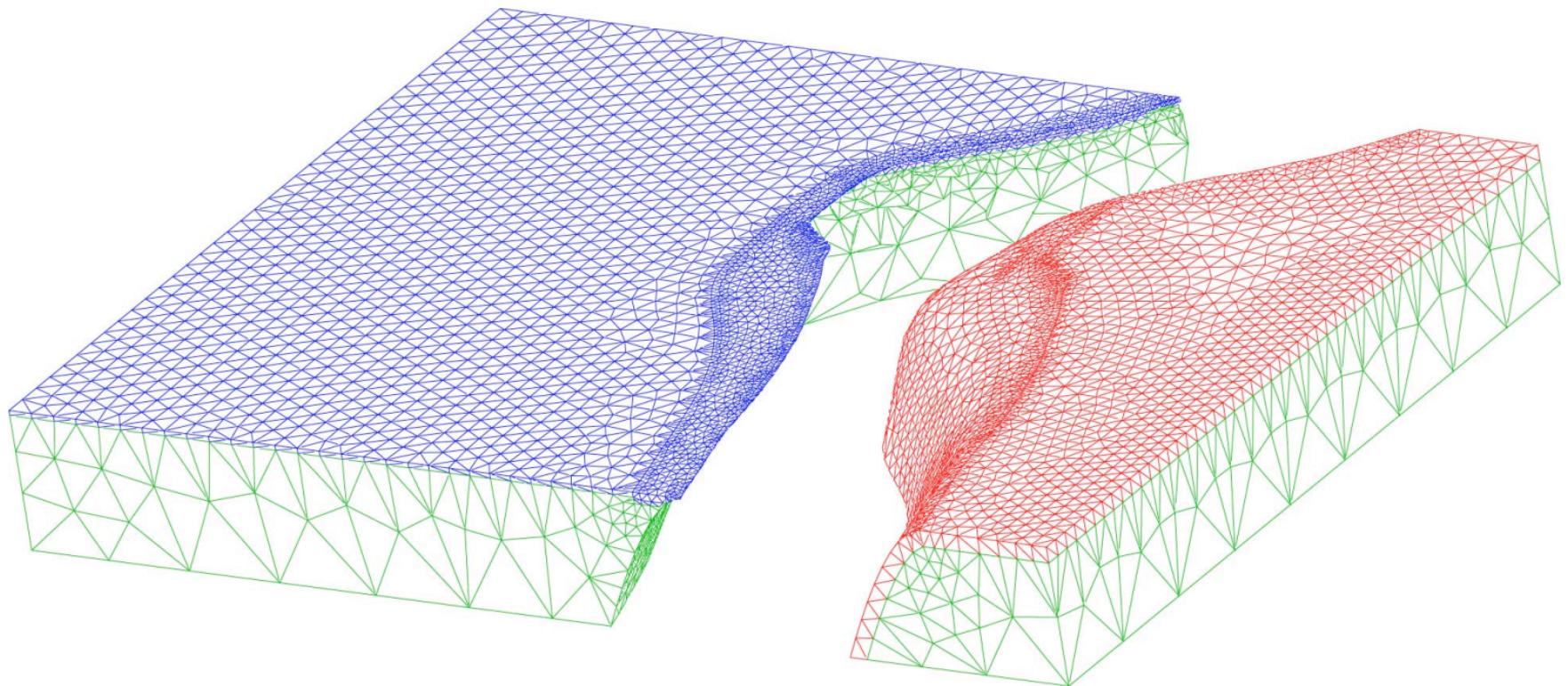
What is it good for?

- Elasticity problems where geometry does not change significantly
- Quasi-static crustal deformation
 - Strain accumulation associated with interseismic deformation
 - Post-seismic relaxation of the crust
 - Volcanic deformation associated with magma chambers and/or dikes
- Dynamic rupture and wave propagation
 - Kinematic (prescribed) earthquake ruptures
 - Local/regional ground-motion modeling

Example Application 1

Freed (Purdue), Ali (Purdue), and Freymueller (U. of AK, Fairbanks)

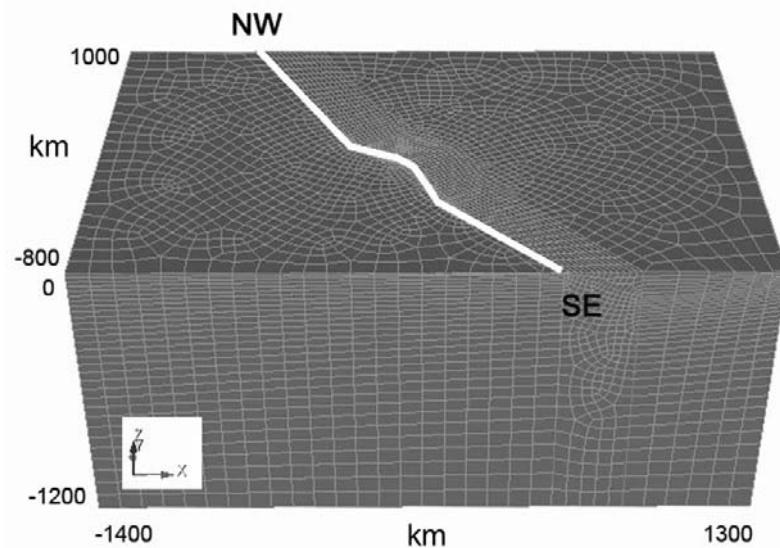
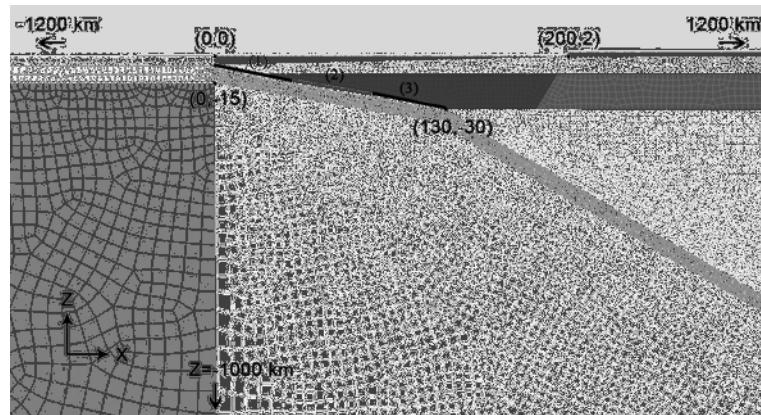
Contemporary strain and stressing rates across Alaska from
interseismic and postseismic processes



Example Application 2

Hsu (IESAS), Simons (Caltech), Williams (GNS Science), Casarotti (INGV)

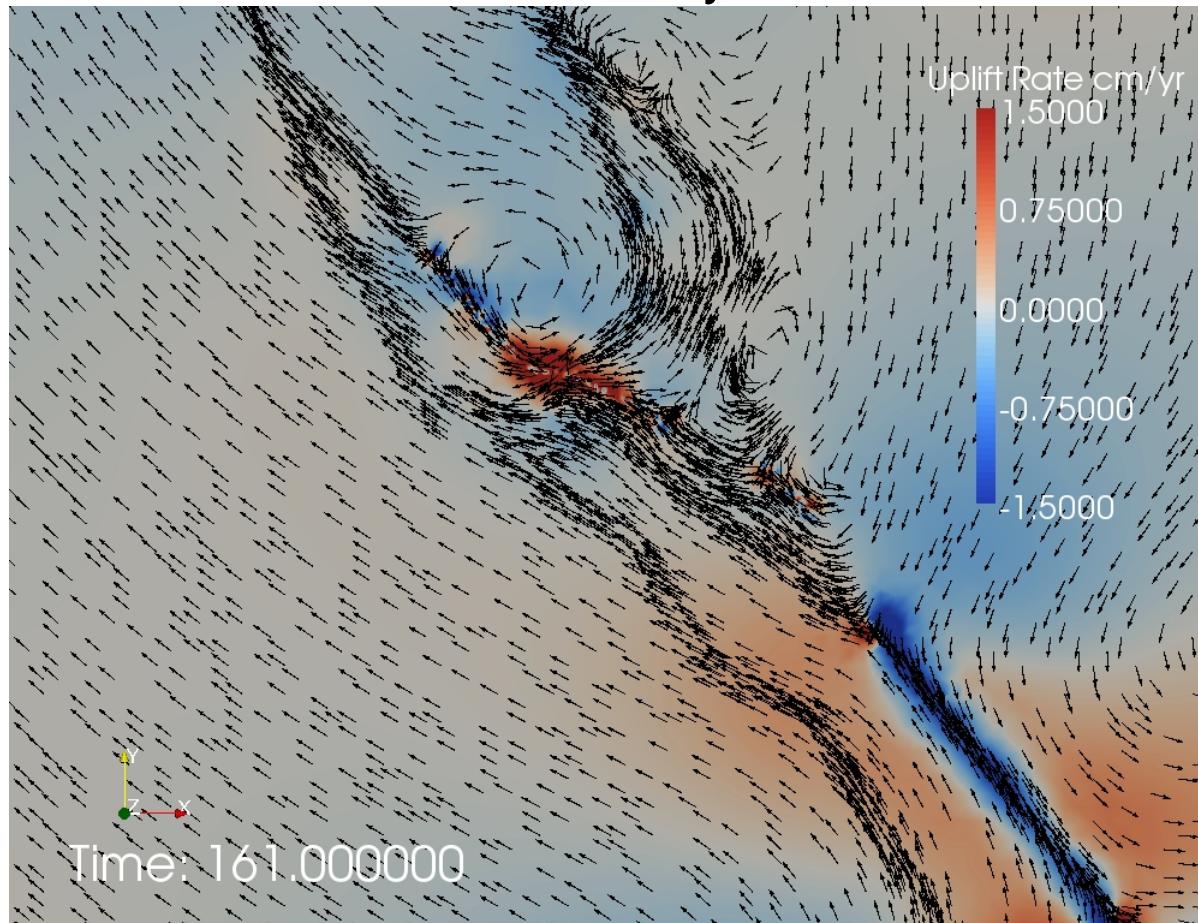
3-D FEM derived elastic Green's functions for the coseismic deformation of the 2005 Mw 8.7 Nias-Simeulue, Sumatra earthquake



Example Application 3

Williams (GNS Science), Gable (LANL), Hager (MIT), Meade (Harvard)
Aagaard (USGS), Knepley (ANL)

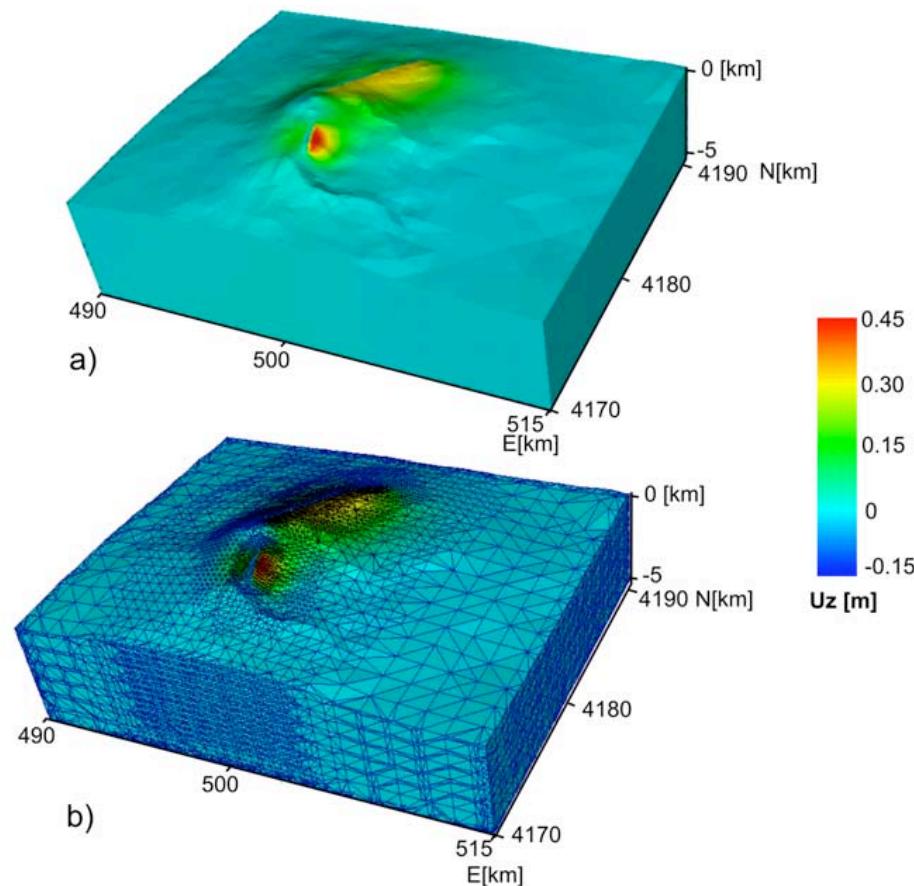
Modeling of Multiple Earthquake Cycles in Southern California Using
the SCEC Community Fault Model



Example Application 4

Currenti (INGV), Del Negro (INGV), Ganci (INGV), Williams (GNS Science)

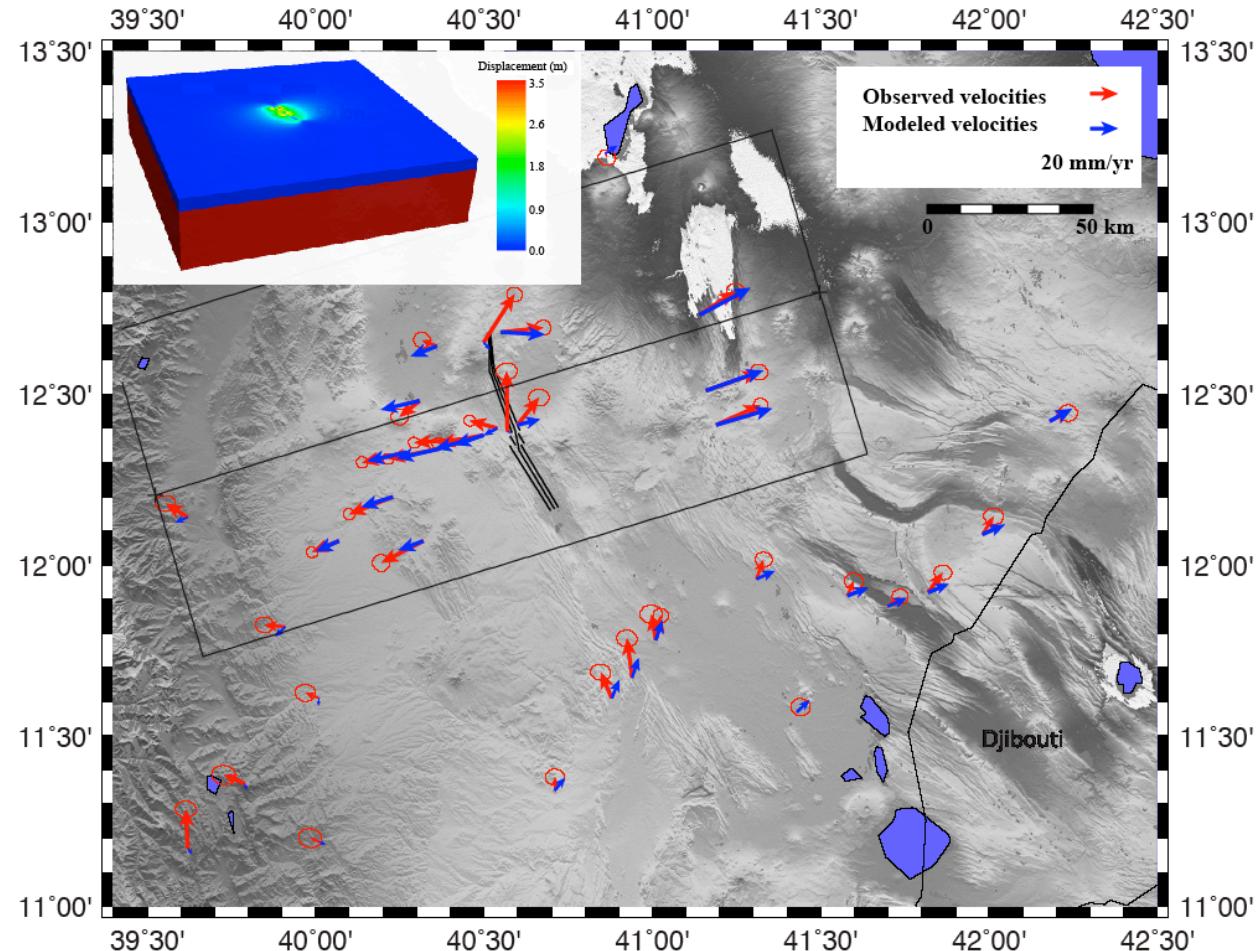
Modeling of Coulomb stress changes during the 2002-2003 Etna eruption



Example Application 5

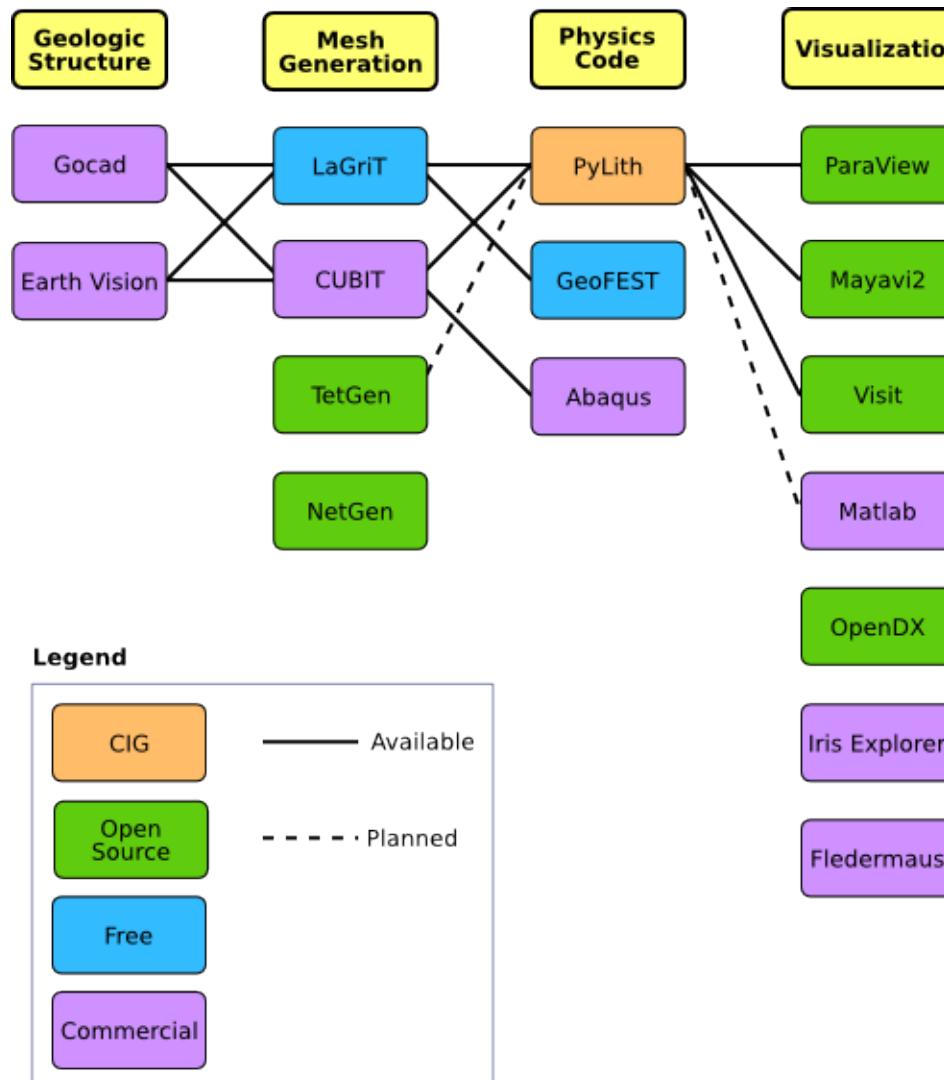
Buck (Columbia) and Nooner (Columbia)

Post-rifting relaxation at a subaerial spreading center



Crustal Deformation Modeling

Overview of workflow for typical research problem



Features in PyLith 1.3

- Spatial dimensions: 1-D, 2-D, or 3-D
- Time integration schemes
 - Implicit time stepping for quasi-static problems
 - Explicit time stepping for dynamic problems
- Bulk constitutive models
 - Elastic model (1-D, 2-D, and 3-D)
 - Linear and Generalized Maxwell viscoelastic models (3-D)
- Boundary and interface conditions
 - Dirichlet (prescribed displacement and velocity) boundary conditions
 - Neumann (traction) boundary conditions
 - Absorbing boundary conditions
 - Kinematic (prescribed slip) fault interfaces w/multiple ruptures
 - Gravitational body forces

Features in PyLith 1.3 (cont.)

- Automatic and user-controlled time stepping
- Ability to specify initial stress state
- Importing meshes
 - LaGriT: GMV/Pset
 - CUBIT: Exodus II
 - ASCII: PyLith mesh ASCII format (intended for toy problems only)
- Output: VTK files
 - Solution over volume
 - Solution over surface boundary
 - State variables (e.g., stress and strain) for each material
 - Fault information (e.g., slip and tractions)

PyLith 1.x: Planned Releases

Current productivity is about 2 feature releases per year

- PyLith 1.4: June 2009
 - Power-law viscoelastic rheology and PETSc nonlinear solvers
 - Ability to specify initial stress, strain, and state variables
 - Automatic, transparent nondimensionalization
 - Use SWIG for Python/C++ interface
- PyLith 1.5: anticipate release in Fall 2009
 - Fault constitutive behavior with several widely used friction models
- PyLith 1.6: anticipate release in Spring/Summer 2010
 - Time dependent boundary conditions
 - Large deformations and finite strain
- PyLith 1.7: Automation of 4-D Green's functions
- PyLith 1.8: Coupling of quasi-static and dynamic simulations

Motivation for Developing PyLith

- Available modeling codes
 - rarely solve the problem **you** want to solve
 - are often poorly documented
 - may not work correctly
- Current research demands larger, more complex simulations
- Want to avoid multiple, incompatible versions of the same code

PyLith Design Objective

Want a code developed for and by the community

- Modular
 - Users can swap modules to run the problem of interest
- Scalable
 - Code runs on one to a thousand processors efficiently
- Extensible
 - Expert users can add functionality to solve their problem without polluting main code

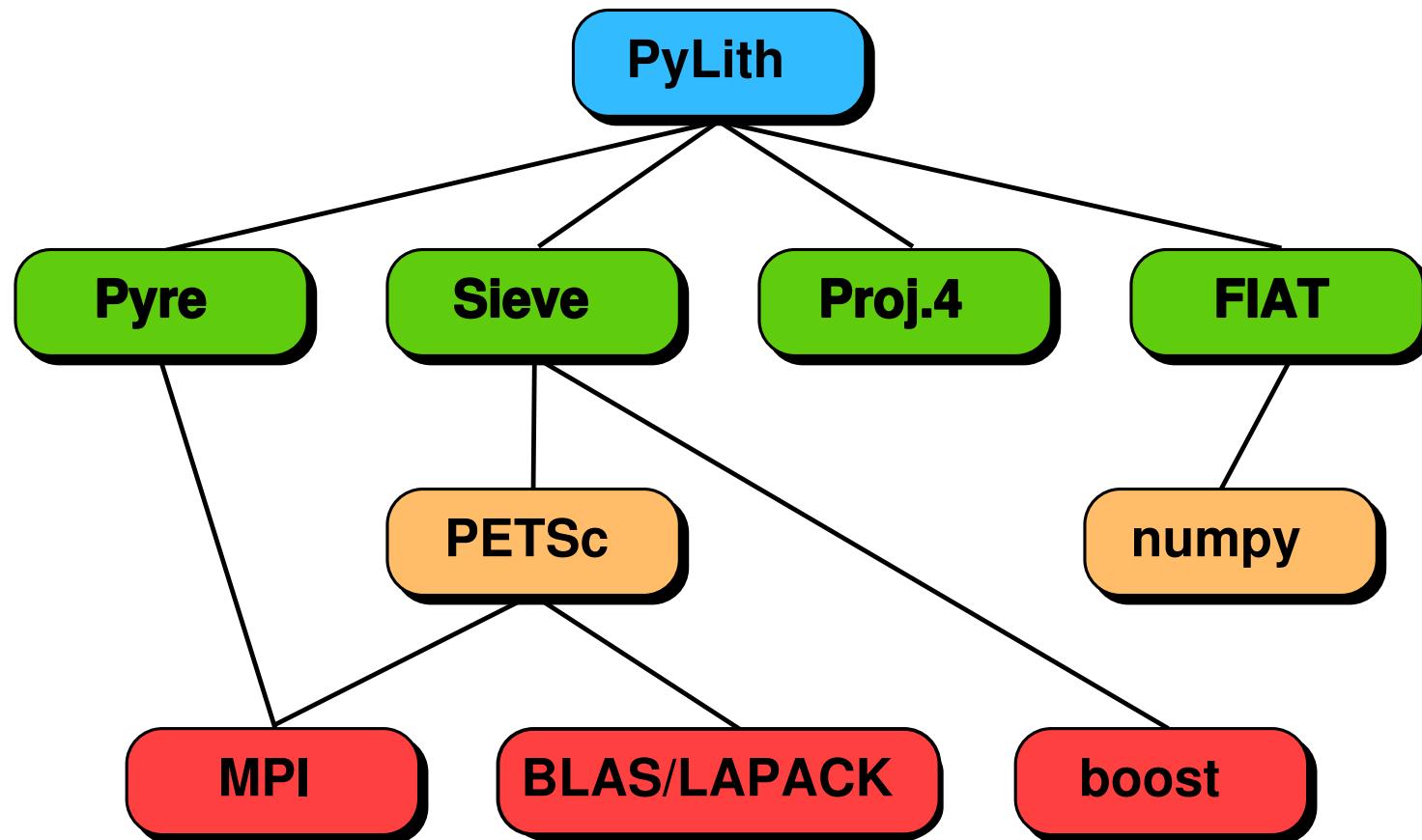
PyLith is a Community Code

Success of code depends on community participation

- End-users (anyone who uses the code)
 - Help define and prioritize features that should be added
 - Report bugs/problems and suggest improvements
- Expert users
 - Help test alpha versions of releases
 - Run benchmarks and report results
 - Contribute meshing and visualization examples to documentation
 - Add features following template (e.g., constitutive models)
- Developer
 - Define development strategy
 - Implement new features and tests
 - Write documentation

PyLith Design: Focus on Geodynamics

Leverage packages developed by computational scientists



PyLith Design: Code Architecture

Flexible and modular with good performance

- Top-level code written in Python
 - Expressive, high-level, object-oriented language
 - Dynamic typing allows adding additional modules at runtime
 - Convenient scripting
- Low-level code written in C++
 - Compiled (fast execution), object oriented language
- Bindings to glue Python & C++ together
 - Pyrex/pyrexembed generate C code for calling C++ from Python

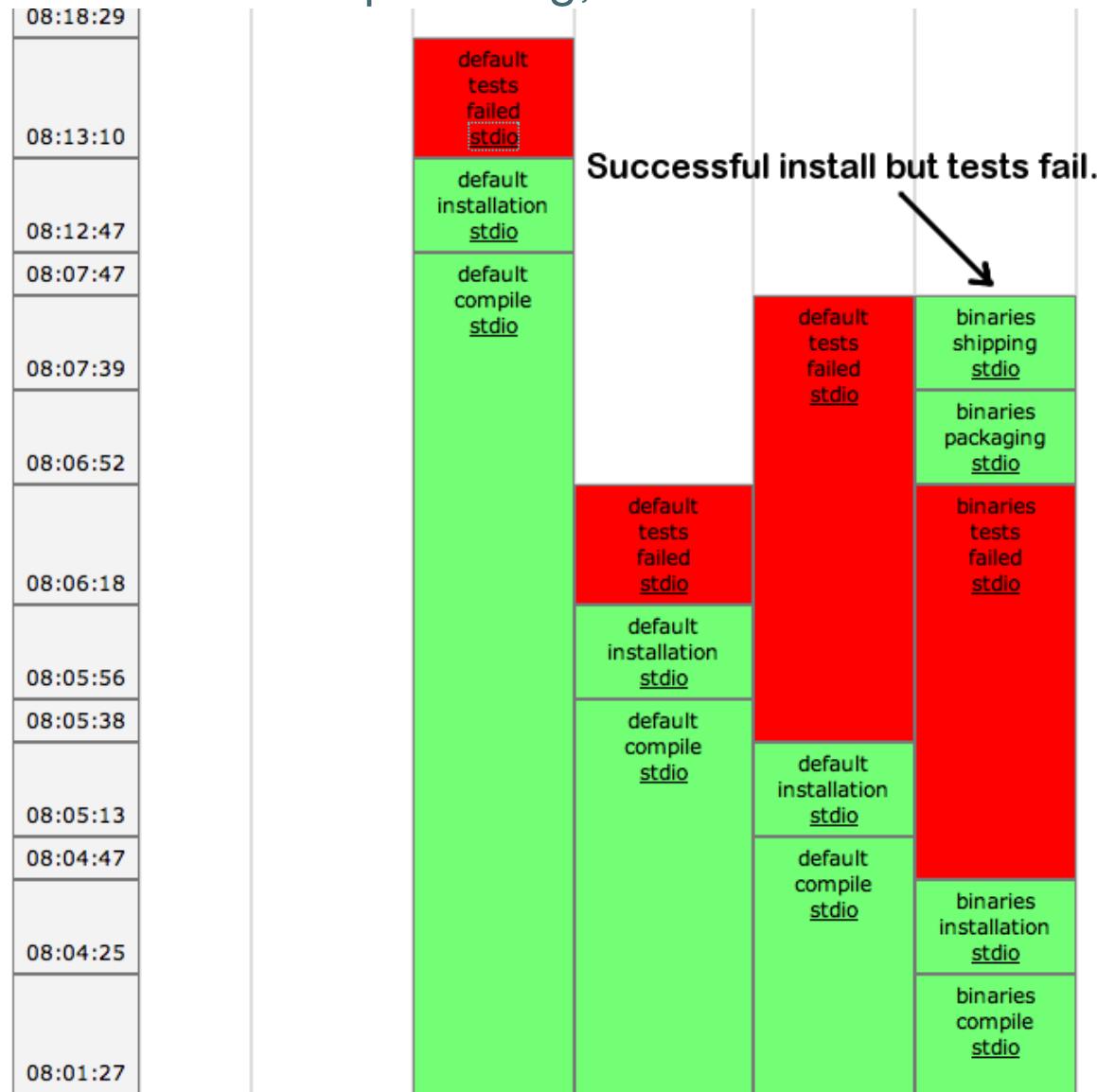
PyLith Design

Tests, tests, and more tests (>1100 in all)

- Create tests for nearly every function during development
 - Remove most bugs during initial implementation
 - Isolate and expose bugs at origin
- Create new tests to expose bugs reported
 - Prevent bugs from reoccurring
- Rerun tests whenever code is changed
 - Allows optimization of performance with quality control
 - Code continually improves

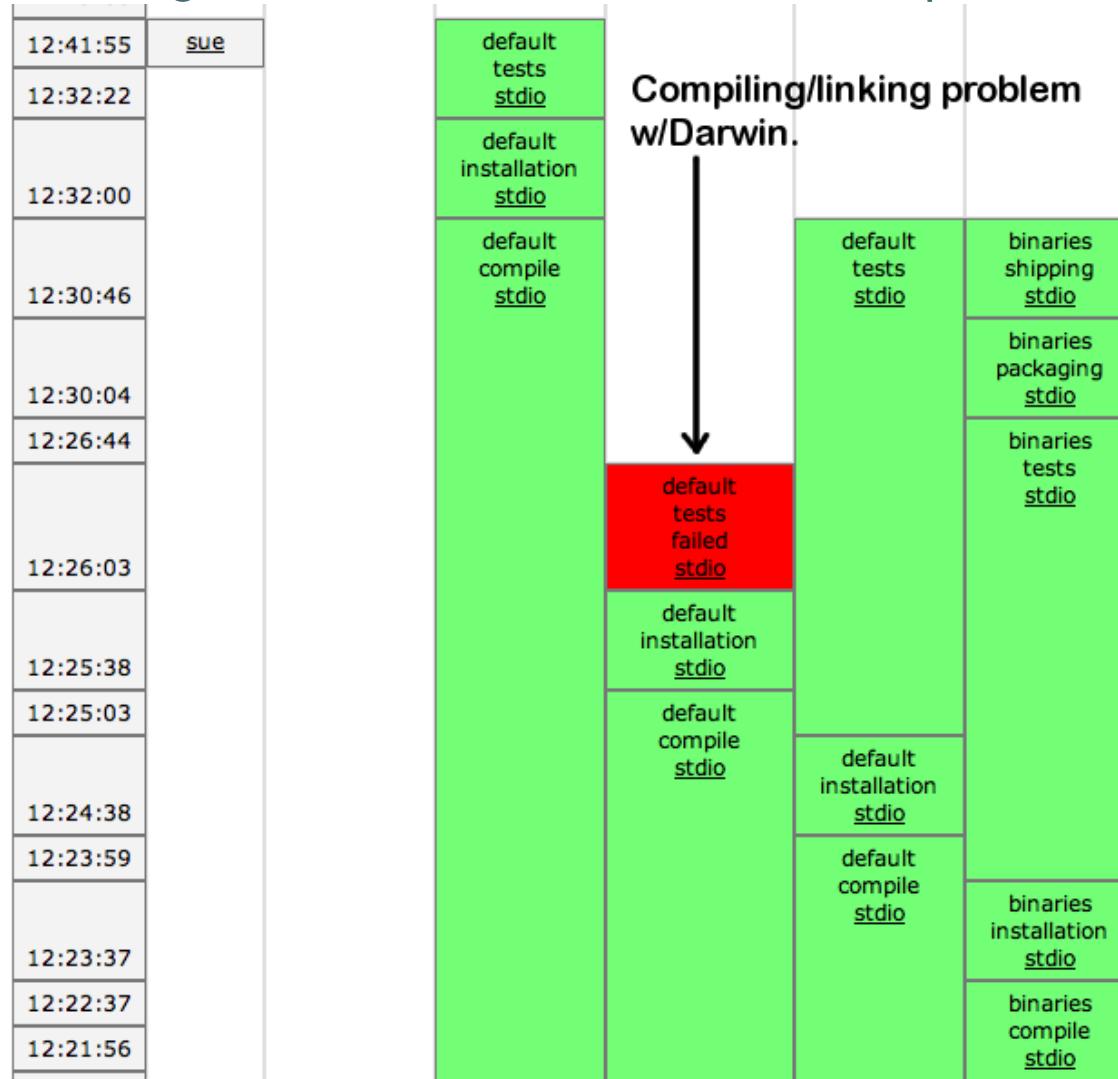
Example of Automated Building and Testing

Test written to expose bug, buildbot shows tests fail



Automated Building and Testing

Bug is fixed, buildbot shows tests pass



Implementation: Finite-Element Data Structures

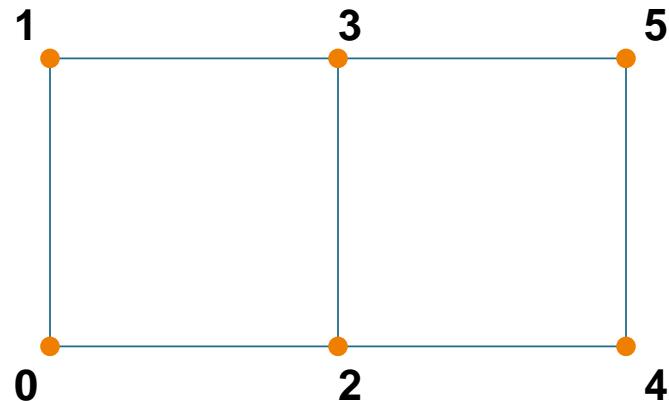
Use Sieve for storage and manipulating mesh information

- PyLith makes only a few MPI calls
- Data structures are independent of basis functions and reference cells
 - Same code for many cell shapes and types
 - Physics implementation limits code, not data structures
- Sieve routines force adhering to finite-element formulation
 - Do not have access to underlying storage
 - Manipulations must be done using Sieve interface
 - Only valid finite-element manipulation is allowed

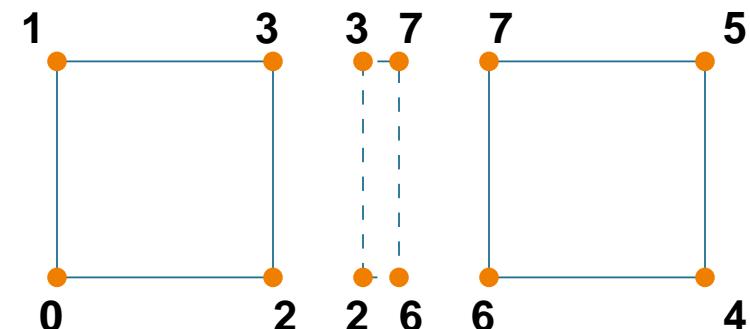
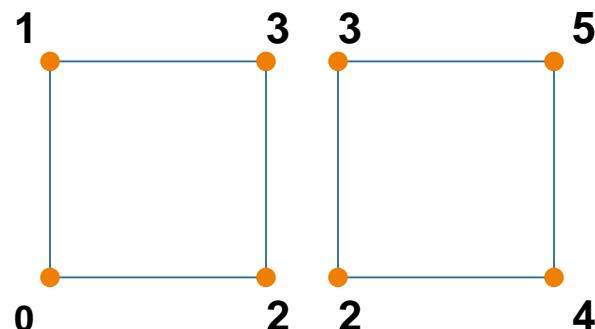
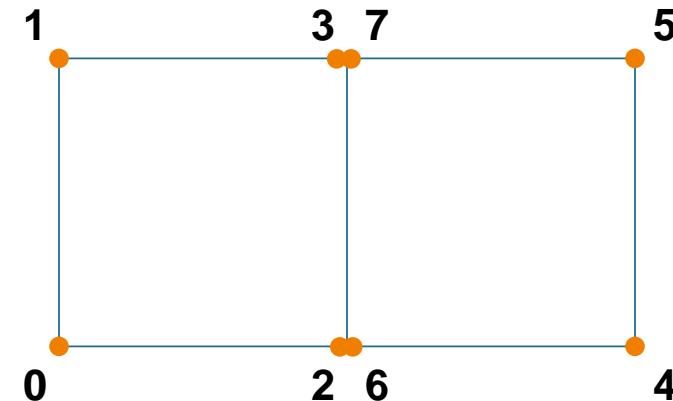
Implementation: Fault Interfaces

Use cohesive cells to control fault behavior

Original Mesh



Mesh with Cohesive Cell



Exploded view of meshes



Kinematic (prescribed) slip earthquake ruptures

Use Lagrange multipliers to specify slip

- System without cohesive cells

$$\underline{\mathbf{A}}\vec{u} = \vec{b}$$

- System with cohesive cells

$$\begin{pmatrix} \underline{\mathbf{A}} & \underline{\mathbf{C}}^T \\ \underline{\mathbf{C}} & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ \vec{L} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{D} \end{pmatrix}$$

- System with cohesive cells & conditioning

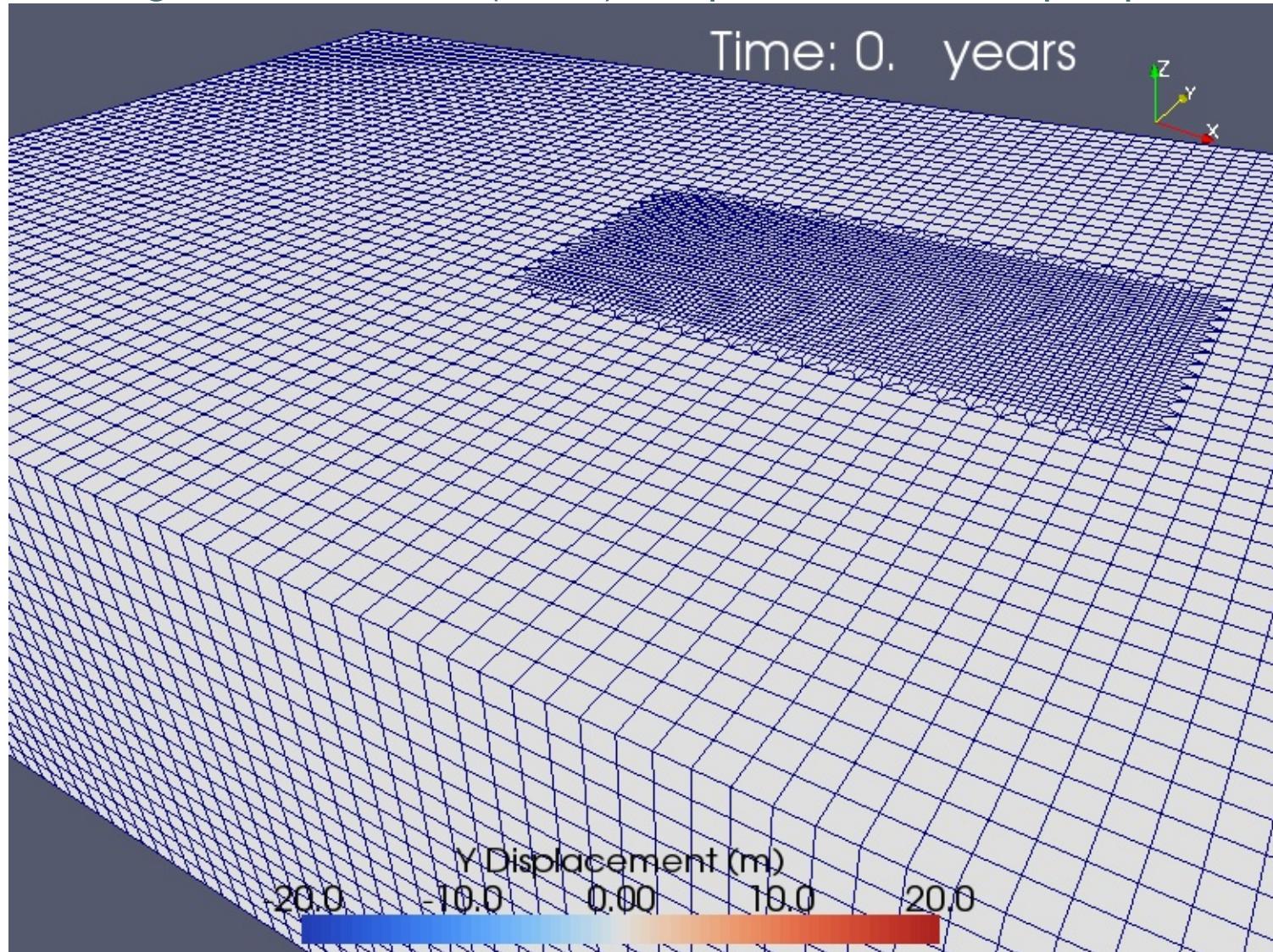
$$\begin{pmatrix} \underline{\mathbf{A}} & a\underline{\mathbf{C}}^T \\ \underline{\mathbf{C}} & 0 \end{pmatrix} \begin{pmatrix} \vec{u} \\ \frac{1}{a}\vec{L} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{D} \end{pmatrix}$$

Implementing Fault Slip with Lagrange multipliers

- Advantages
 - Fault implementation is local to cohesive cell
 - Solution includes forces generating slip (Lagrange multipliers)
 - Retains block structure of matrix (same number of DOF per vertex)
 - Offsets in mesh mimic slip on natural faults
- Disadvantages
 - Conditioned matrix is non-symmetric
 - Mixes displacements and forces in solution

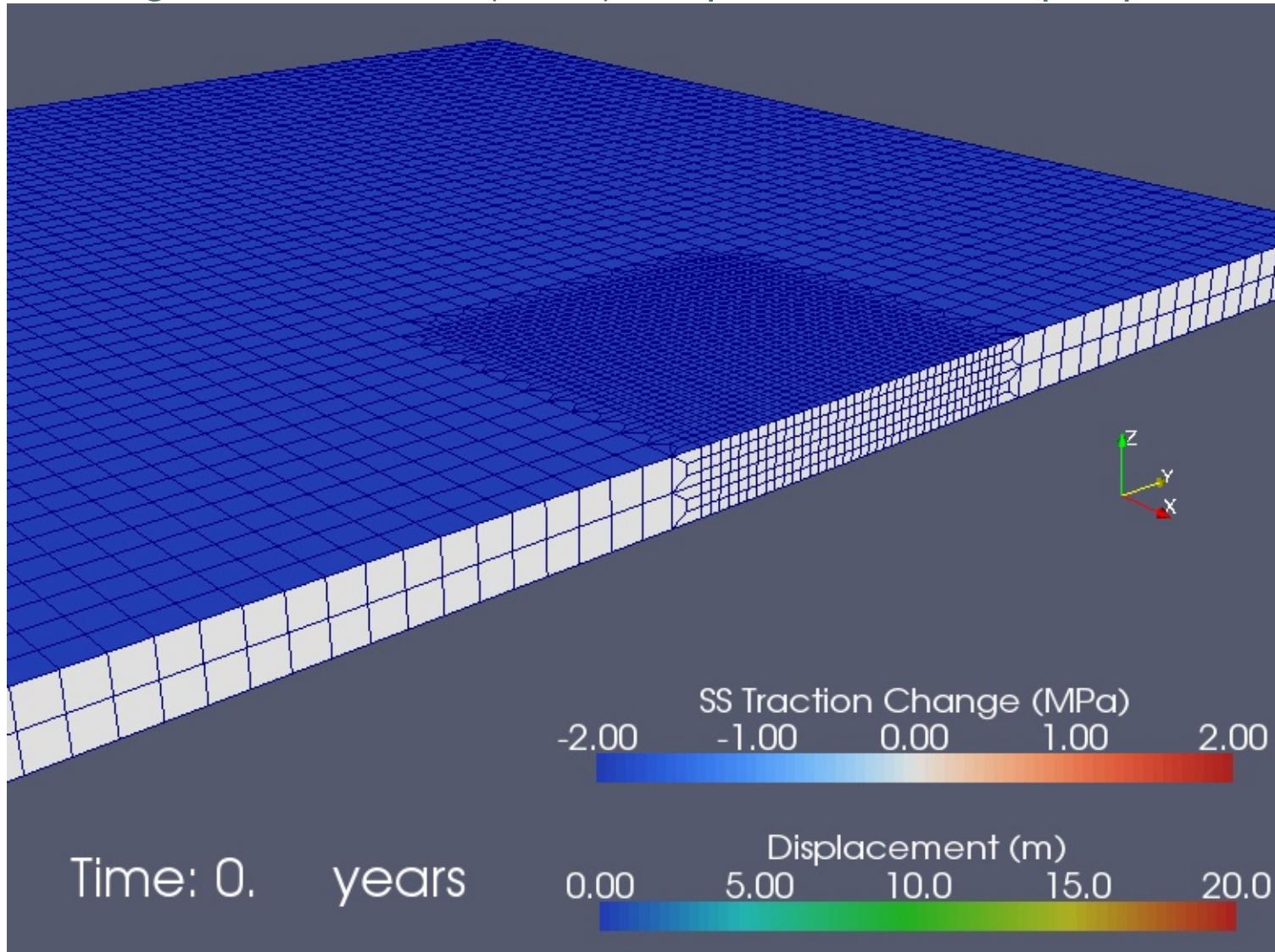
Benchmarking PyLith

Savage and Prescott (1978): Repeated Strike-slip ruptures



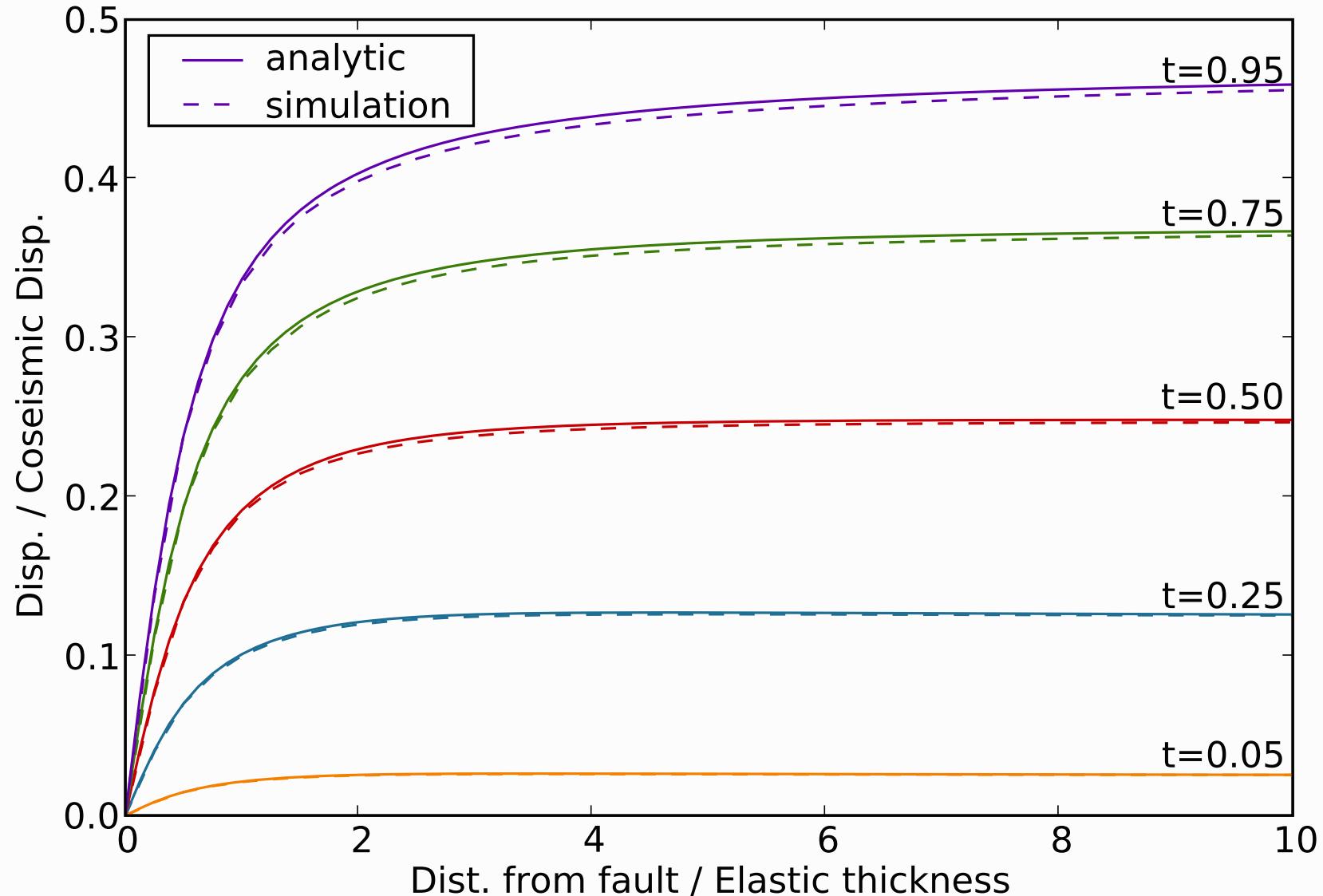
Benchmarking PyLith

Savage and Prescott (1978): Repeated Strike-slip ruptures



Benchmarking PyLith

Simulation closely matches analytical solution during 10th eq cycle



Running PyLith

Ingredients

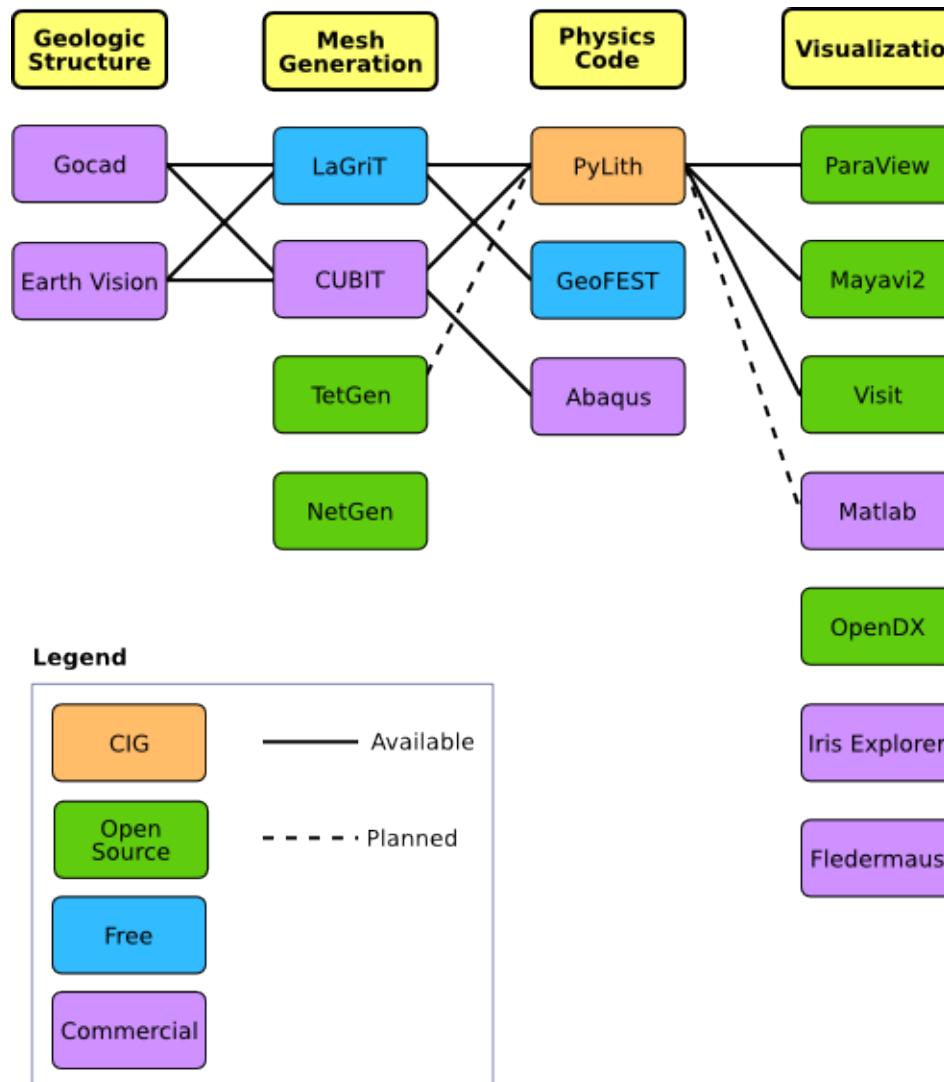
- Simulation parameters
- Finite-element mesh
 - Mesh exported from LaGriT
 - Mesh exported from CUBIT
 - Mesh constructed by hand (PyLith mesh ASCII format)
- Spatial databases for physical properties, boundary conditions, and rupture parameters
 - SCEC CVM-H, USGS Bay Area Velocity model, or simple ASCII files
 - Independent of discretization scheme and size

Useful Tips/Tricks

- Command line arguments
 - --help
 - --help-components
 - --help-properties
 - --petsc.start_in_debugger (run in xterm)
 - --nodes=N (to run on N processors on local machine)
- PyLith User Manual
- CIG Short-Term Tectonics mailing list
 - cig-short@geodynamics.org
- CIG bug tracking system
 - <http://www.geodynamics.org/roundup>

Crustal Deformation Modeling

Overview of workflow for typical research problem



Installing PyLith

Download from geodynamics.org or copy from CDROM

Recommend copying PyLith and ParaView from CDROM with INSTALL files.

Examples are in src/pylith/examples.