PyLith 1.0

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

Computational Infrastructure for Geodynamics

June 25, 2007

Outline

- Introduction to PyLith
  - Motivation & development objective
  - What does PyLith do?
- PyLith Design
  - Architecture and programming languages
  - Development strategy
- Features
  - Current release
  - Planned releases
- Example
- Feedback

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 1.0 Design Objective

Want to 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

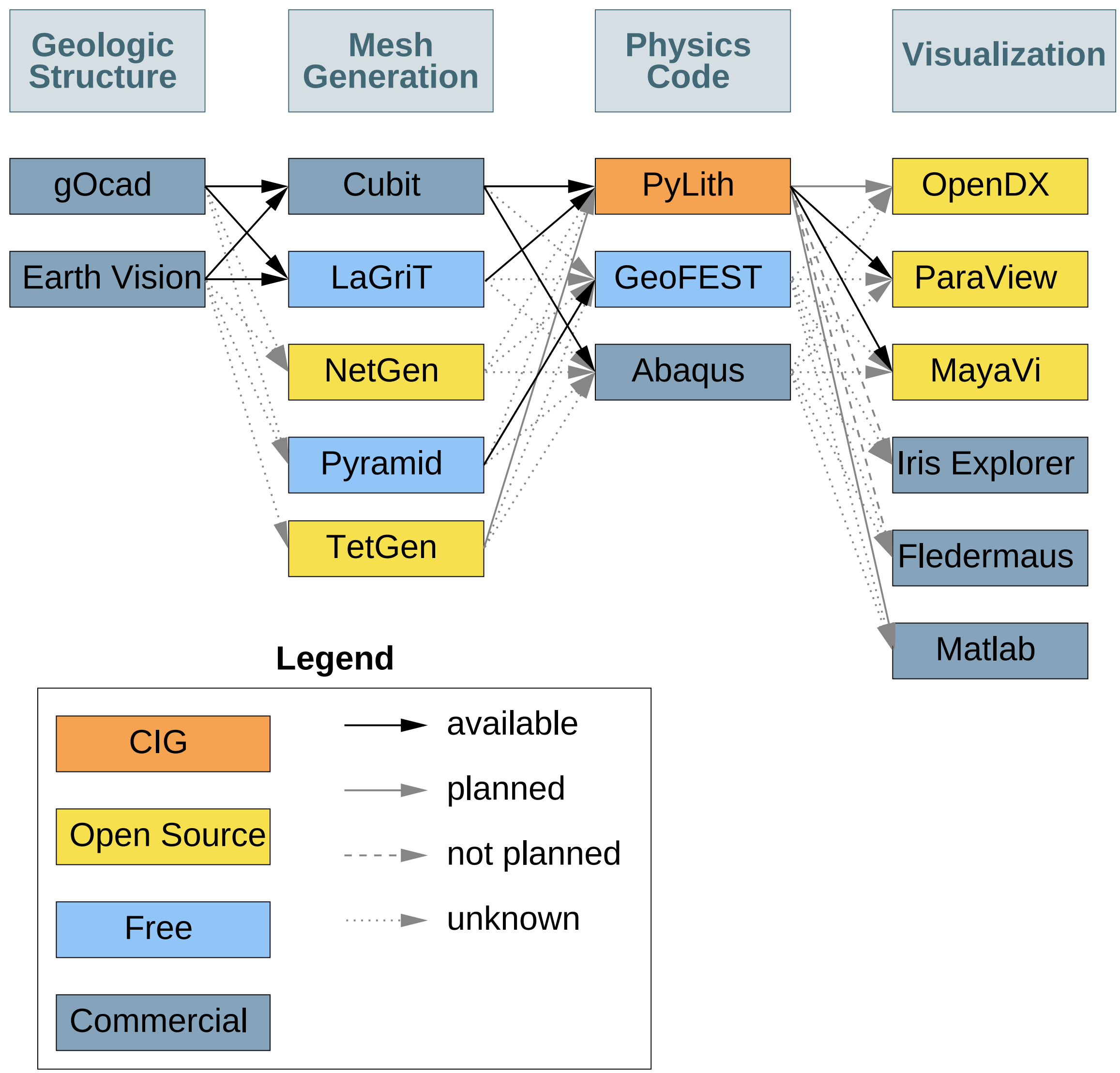
PyLith 1.0

What is it good for?

- Quasi-static crustal deformation
  - Interseismic deformation
  - Post-seismic deformation
- Dynamic rupture and wave propagation
  - Kinematic (prescribed) earthquake ruptures
  - Strong ground motion modeling

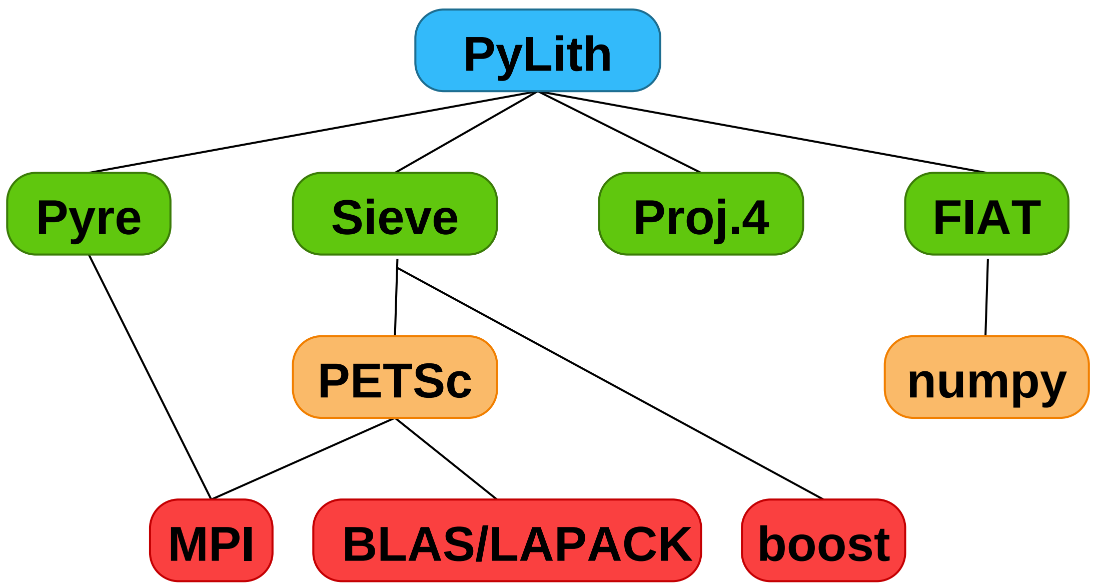
PyLith 1.0

Overview of workflow for typical research problem



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 (>500 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
  - Fast isolation of origin of bugs
  - Prevent bugs from reoccurring
- Rerun tests whenever code is changed
  - Allows optimization of performance with quality control
  - Code continually improves

PyLith 1.0: Features

- User-interface
  - Import meshes directly from LaGriT and CUBIT
  - Easy specification of parameters for boundary condition and fault conditions
- Applications
  - Quasi-static solution of interseismic crustal deformation
  - Dynamic solution of wave propagation for propagating ruptures
- Under the hood
  - Sieve - parallel data structures for storing and manipulating finite-element meshes
  - PETSc - large library of solvers and preconditioners
  - Pyre - science neutral simulation framework for easy access to user data and configuration

PyLith 1.x: Planned Releases

Quickly add important features back in

- PyLith 1.1: anticipate release in early Fall 2007
  - General
    - Expand output options and include state variables
    - Improve runtime and reduce memory usage
  - Dynamic problems
    - Add absorbing boundaries
    - Complete testing
  - Quasi-static problems
    - Add traction (Neumann) boundary conditions
    - Add viscoelastic models implemented in PyLith 0.8
- PyLith 1.2: anticipate release in late 2007 or early 2008
  - Implement frictional interfaces for faults
  - Add fault constitutive models
  - More under-the-hood improvements

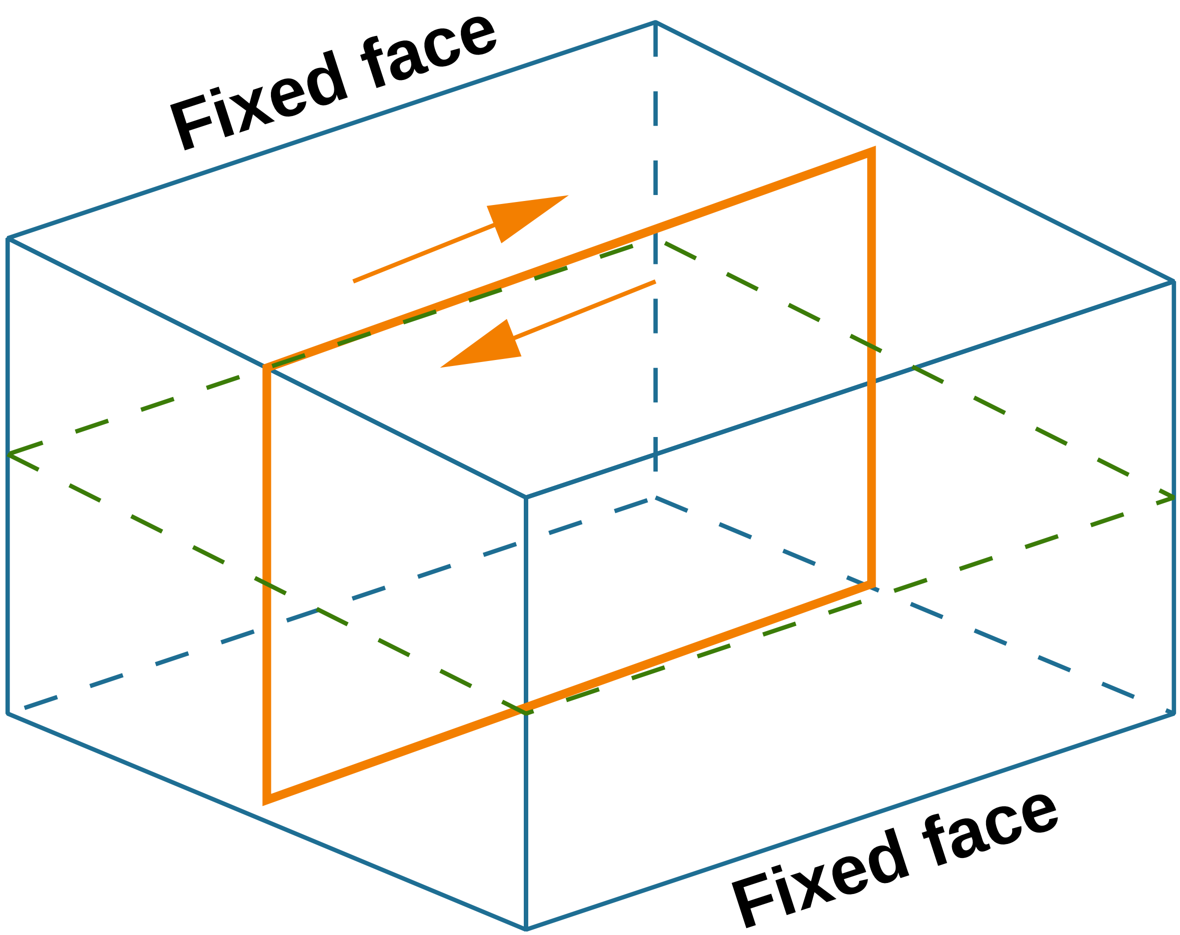
Running PyLith

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 boundary and fault conditions
  - Simple ASCII files specify spatial variation of parameters
  - Independent of discretization scheme and size

Example: Slip on a Vertical Strike-Slip Fault

examples/3d/hex8



Workflow for Example

1. Generate finite-element mesh using CUBIT (hex8 cells)

2. Create `.cfg` file with simulation parameters

3. Create containers for materials, boundary conditions, or faults (if necessary)

4. Create spatial database files with parameters for boundary conditions and faults

5. Run pylith

6. Visualize results with ParaView

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; not fully tested)
- PyLith 1.0 User Manual
- CIG Short-Term Tectonics mailing list
  - `cig-short@geodynamics.org`
- CIG bug tracking system
  - `http://www.geodynamics.org/roundup`

Feedback

We want your comments!

- PyLith 1.0 versus PyLith 0.8
  - Help prioritize adding features present in PyLith 0.8

- PyLith 1.0 versus other codes
  - You would like to be using PyLith, but . . .

- PyLith is designed to be a community code
  - Contribute bulk constitutive models
  - Contribute mesh importers
  - Contribute visualization exporters