PyLith Modeling Tutorial

Static Green's Functions

Charles Williams
Brad Aagaard
Matthew Knepley

COMPUTATIONAL INFRASTRUCTURE for GEODYNAMICS

June 27, 2017

Concepts Covered in this Session

- Simulation of a slow slip event (SSE) in Cascadia
- Usage of SimpleGridDB to specify fault slip
- Usage of a temporal database to specify variation of slip amplitude with time
- Solution output at a specified set of points (OutputSolnPoints)
- Postprocessing of HDF5 output using h5py
- Generation of synthetic data with user-specified noise
- Generation of Green's functions in 3D
- Simple linear inversion using numpy
- Plotting of inversion results using matplotlib and h5py

Green's Functions

- Compute deformation due to unit (i.e., 1 m) slip at fault vertices for use in an inversion for fault slip
  - Slip decreases **linearly** to 0 at surrounding vertices
  - Similar but not equivalent to uniform slip over a patch (Okada dislocation)
  - PyLith interpolates the responses to user-specified points using OutputSolnPoints output manager
- Provides ability to compute Green's functions with arbitrarily complex elastic structure and/or topography

Other Green's Functions Examples

- 2-D examples: `examples/2d/greensfns`
  - Example components
    1. Compute synthetic (fake) observations for an earthquake
    2. Compute displacements at sites for Green's functions
    3. Invert for fault slip
  - See Section 7.15 of the PyLith User Manual
- 3-D example: `examples/3d/hex8/step21`
  - Limited to computing displacements at sites for Green's functions
  - No inversion

Cascadia Green's Functions Example

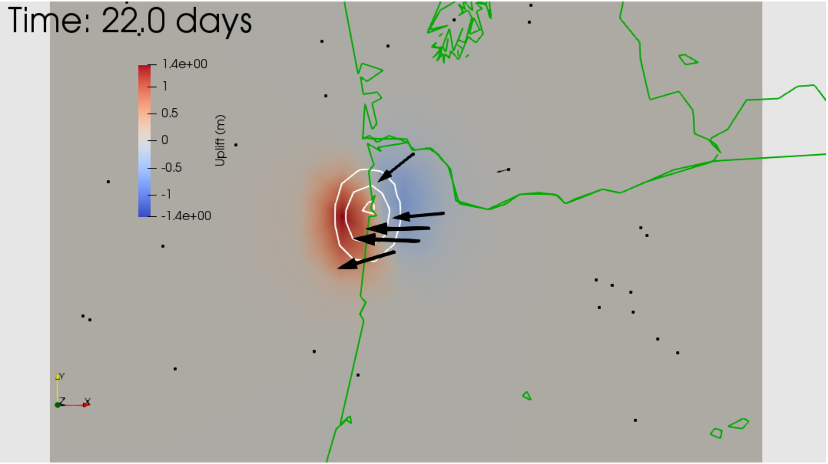
Simulated slow slip event plus inversion

Files are in `examples/3d/subduction`

1. Create a slip distribution that has a Gaussian shape spatially with a temporal variation, usting the Python script subduction/spatialdb/generate_slowslip.py
2. Run example `step06.cfg` to generate a synthetic slow slip event
3. Create synthetic observations with noise by running the Python script subduction/make_synthetic_gpsdisp.py
4. Compute displacements at sites for Green's functions by running `step07a.cfg` and `step07b.cfg`
5. Invert for fault slip using Python script subduction/slip_invert.py
6. Visualize inversion results using matplotlib Python package subduction/viz/plot_inversion_misfit.py and ParaView

Simulated Cascadia SSE

Time-varying slip on subduction interface



Simple Linear Inversion

Parameters

- G Green's function matrix
- d Unknown fault slip
- $d_{apriori}$ A priori estimate of fault slip
- u_{obs} Observed displacement
- D Penalty matrix
- θ Penalty parameter

The matrix G_{ij} gives displacement component i due to a unit of slip from component j.

Simple Linear Inversion

Equations

- Original system of equations:

$$Gd = u_{obs} \tag{1}$$

- Augmented system of equations:

$$G_a d = u_a, \text{ where } G_a = \begin{bmatrix} G \\ \theta D \end{bmatrix} \text{ and } u_a = \begin{bmatrix} u_{obs} \\ d_{apriori} \end{bmatrix} \tag{2}$$

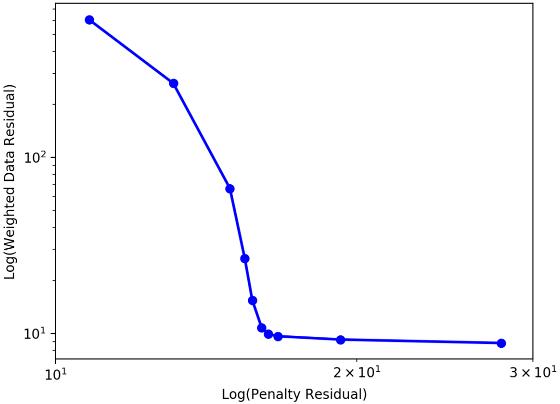
- Generalized inverse:

$$G^{-g} = \left(G_a^T G_a\right)^{-1} G_a^T \tag{3}$$

$$d_{est} = G^{-g} u_a \tag{4}$$

Inversion results

Plot of weighted data misfit vs. penalty misfit



Inversion results

Predicted slip distribution