Crustal Deformation Modeling Tutorial

PyLith Version 3

Brad Aagaard
Matthew Knepley
Charles Williams

June 27, 2017

PyLith v3.0

- Multiphysics through pointwise integration kernels
- Higher order spatial and temporal discretizations
- Adaptive time stepping via PETSc TS
- Improved fault formulation for spontaneous rupture (v3.1)

Aside: Finite-Element Method

Strong form to weak form

Solve governing equation in integrated sense:

$$\int_{\Omega} \psi_{trial} \cdot PDE \, d\Omega = 0, \tag{1}$$

by minimizing the error with respect to the unknown coefficients.

This leads to equations of the form:

$$\int_{\Omega} \psi_{trial} \cdot f_0(x,t) + \nabla \psi_{trial} \cdot f_1(x,t) \, d\Omega = 0. \tag{2}$$

Governing Equations

We want to solve equations in which the weak form can be expressed as

$$F(t, s, \dot{s}) = G(t, s) \tag{3}$$

$$s(t_0) = s_0 \tag{4}$$

where F and G are vector functions, t is time, and s is the solution vector.

Using the finite-element method and divergence theorem, we cast the weak form into

$$\int_\Omega \vec{\psi}_{trial} \cdot \vec{f}_0(t, s, \dot{s}) + \nabla\vec{\psi}_{trial} : \boldsymbol{f}_1(t, s, \dot{s})\, d\Omega = \int_\Omega \vec{\psi}_{trial} \cdot \vec{g}_0(t, s) + \nabla\vec{\psi}_{trial} : \boldsymbol{g}_1(t, s)\, d\Omega, \tag{5}$$

where $\vec{f}_0$ and $\vec{g}_0$ are vectors, and $\boldsymbol{f}_1$ and $\boldsymbol{g}_1$ are tensors.

Explicit Time Stepping

Explicit time stepping with the PETSc TS requires $F(t, s, \dot{s}) = \dot{s}$.

Normally $F(t, s, \dot{s})$ contains the inertial term ($\rho\ddot{u}$).

Therefore, when using explicit time stepping we transform our equation into the form:

$$F^*(t, s, \dot{s}) = \dot{s} = G^*(t, s) \tag{6}$$

$$\dot{s} = M^{-1}G(t, s). \tag{7}$$

Solving the Equations

Explicit time stepping requires a subset of the terms used in implicit time stepping.

- PETSc TS object provides time-stepping and solver implementations
  - Application code provides functions for computing RHS and LHS residuals and Jacobians
- Explicit time stepping
  - Compute RHS residual, $G(t, s)$
  - Compute lumped inverse of LHS, M^{-1}
  - No need to compute LHS residual, because $F(t, s, \dot{s}) = \dot{s}$
- Implicit time stepping (Krylov solvers)
  - Compute RHS residual, $G(t, s)$
  - Compute LHS residual, $F(t, s, \dot{s})$
  - Compute RHS Jacobian, $J_G = \frac{\partial G}{\partial s}$
  - Compute LHS Jacobian, $J_F = \frac{\partial F}{\partial s} + t_{shift} \frac{\partial F}{\partial \dot{s}}$

Example: Elasticity

$$\rho \frac{\partial^2 \vec{u}}{\partial t^2} = \vec{f}(\vec{x}, t) + \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(\vec{u}) \text{ in } \Omega, \tag{8}$$

$$\boldsymbol{\sigma} \cdot \vec{n} = \vec{\tau}(\vec{x}, t) \text{ on } \Omega_\tau, \tag{9}$$

$$\vec{u} = \vec{u}_0(\vec{x}, t) \text{ on } \Omega_u, \tag{10}$$

Implicit Time Stepping without Inertia

Displacement $\vec{u}$ is the unknown, $\vec{s} = \vec{u}$.

$$\underbrace{0}_{\vec{f}_0^u} = \int_\Omega \vec{\psi}^u_{trial} \cdot \underbrace{\vec{f}(\vec{x}, t)}_{\vec{g}_0^u} + \nabla \vec{\psi}^u_{trial} : \underbrace{-\boldsymbol{\sigma}(\vec{u})}_{\boldsymbol{g}_1^u} d\Omega + \int_{\Omega_\tau} \vec{\psi}^u_{trial} \cdot \underbrace{\vec{\tau}(\vec{x}, t)}_{\vec{g}_0^u} d\Omega_\tau \tag{11}$$

Different constitutive models are encapsulated in alternative kernels for $\boldsymbol{\sigma}(\vec{u})$.

Example: Elasticity (continued)

Explicit Time Stepping with Inertia

Form a first order equation using displacement $\vec{u}$ and velocity $\vec{v}$ as unknowns,

$$\vec{s}^T = \begin{pmatrix} \vec{u} & \vec{v} \end{pmatrix}^T$$

$$\int_\Omega \vec{\psi}^u_{trial} \cdot \underbrace{\frac{\partial \vec{u}}{\partial t}}_{\dot{u}} \, d\Omega = \int_\Omega \vec{\psi}^u_{trial} \cdot \underbrace{\vec{v}}_{\vec{g}^u_0} \, d\Omega, \tag{12}$$

$$\int_\Omega \vec{\psi}^v_{trial} \cdot \underbrace{\frac{\partial \vec{v}}{\partial t}}_{\dot{v}} \, d\Omega = \frac{1}{\vec{m}} \left(\int_\Omega \vec{\psi}^v_{trial} \cdot \underbrace{\vec{f}(\vec{x}, t)}_{\vec{g}^v_0} + \nabla \vec{\psi}^v_{trial} : \underbrace{-\boldsymbol{\sigma}(\vec{u})}_{\boldsymbol{g}^v_1} \, d\Omega + \int_{\Omega_\tau} \vec{\psi}^v_{trial} \cdot \underbrace{\vec{\tau}(\vec{x}, t)}_{\vec{g}^v_0} \, d\Omega_\tau \right), \tag{13}$$

$$\vec{m} = \int_\Omega \vec{\psi}^v_{trial} \cdot \underbrace{\rho}_{J^{vv}_{f0}} \vec{\psi}^v_{basis} \, d\Omega \tag{14}$$

Example: Elasticity (continued)

Implementing the governing equations involves a small set of simple kernels.

	Implicit	Explicit
$\vec{v}$	–	$\vec{g}_0^v$
$\vec{f}(\vec{x},t)$	$\vec{g}_0^u$	$\vec{g}_0^v$
$-\boldsymbol{\sigma}(\vec{u})$	$\boldsymbol{g}_1^u$	$\boldsymbol{g}_1^v$
$\vec{\tau}(\vec{x},t)$	$\vec{g}_0^u$	$\vec{g}_0^v$
$\vec{0}$	$\vec{f}_0^u$	–
ρ	–	$J_{f_0}^{uv}$

We also have simple kernels for the Jacobians needed in implicit time stepping.

Example: Elasticity Stress Kernels for Residual

- Volumetric Stress

```
for (i=0; i < _dim; ++i) {
   trace += disp_x[i] - initialstrain[i*_dim+i];
   meanistress += initialstress[i*_dim+i];
}
meanistress /= (PylithReal) _dim;
for (i = 0; i < _dim; ++i) {
   stress[i*_dim+i] += lambda * trace + meanistress;
}
```

- Deviatoric Stress

```
for (i=0; i < _dim; ++i) {
    meanistress += initialstress[i*_dim+i];
}
meanistress /= (PylithReal) _dim;
    for (i=0; i < _dim; ++i) {
      for (j=0; j < _dim; ++j) {
          stress[i*_dim+j] += mu * (disp_x[i*_dim+j] + disp_x[j*_dim+i]
              - initialstrain[i*_dim+j]) + initialstress[i*_dim+j];
    }
    stress[i*_dim+i] -= meanistress;
}
```

Example: Poroelasticity Neglecting Inertia

We assume a compressible fluid completely saturates a porous solid undergoing infinitesimal strain.

Elasticity equilibrium equation neglecting inertia:

$$0 = \vec{f}(\vec{x},t) + \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}(\vec{u},p_f) \text{ in } \Omega, \quad \boldsymbol{\sigma}\cdot\vec{n} = \vec{\tau}(\vec{x},t) \text{ on } \Omega_\tau, \vec{u} = \vec{u}_0(\vec{x},t) \text{ on } \Omega_u, \tag{15}$$

Mass balance of the fluid:

$$\frac{\partial\zeta(\vec{u},p_f)}{\partial t} = \gamma(\vec{x},t) - \nabla\cdot\vec{q}(p_f) \text{ in } \Omega, \quad \vec{q}\cdot\vec{n} = q_0(\vec{x},t) \text{ on } \Omega_q, p_f = p_0(\vec{x},t) \text{ on } \Omega_p, \tag{16}$$

Darcy's law:

$$\vec{q}(p_f) = -\kappa(\nabla p_f - \vec{f}_f), \quad \kappa = \frac{k}{\eta_f} \tag{17}$$

Constitutive behavior of the fluid:

$$\zeta(\vec{u},p_f) = \alpha(\nabla\cdot\vec{u}) + \frac{p_f}{M}, \quad \frac{1}{M} = \frac{\alpha-\phi}{K_s} + \frac{\phi}{K_f}, \tag{18}$$

Constitutive behavior of the solid (linear elasticity):

$$\boldsymbol{\sigma}(\vec{u},p_f) = \boldsymbol{C} : \boldsymbol{\epsilon} - \alpha p_f \boldsymbol{I} \tag{19}$$

Example: Poroelasticity Neglecting Inertia

Consider displacement $\vec{u}$ and fluid pressure p_f as unknowns, $\vec{s}^T = \begin{pmatrix} \vec{u} & p_f \end{pmatrix}^T$

$$\underbrace{0}_{\vec{f}_0^u} = \int_\Omega \vec{\psi}^u_{trial} \cdot \underbrace{\vec{f}(\vec{x},t)}_{\vec{g}_0^u} + \nabla\vec{\psi}^u_{trial} : \underbrace{-\boldsymbol{\sigma}(\vec{u},p_f)}_{\boldsymbol{g}_1^u} d\Omega + \int_{\Omega_\tau} \vec{\psi}^u_{trial} \cdot \underbrace{\vec{\tau}(\vec{x},t)}_{\vec{g}_0^u} d\Omega_\tau, \tag{20}$$

$$\int_\Omega \psi^p_{trial} \underbrace{\frac{\partial\zeta(\vec{u},p_f)}{\partial t}}_{f_0^p} d\Omega = \int_\Omega \psi^p_{trial} \underbrace{\gamma(\vec{x},t)}_{g_0^p} + \nabla\psi^p_{trial} \cdot \underbrace{\vec{q}(p_f)}_{\vec{f}_1^p} d\Omega + \int_{\Omega_q} \psi^p_{trial} \underbrace{(-q_0(\vec{x},t))}_{g_0^p} d\Omega_q. \tag{21}$$

Poroelasticity involves many of the same kernels as elasticity plus a few additional ones.

Finite-Element Discretization

Specify discretizations for solution fields and auxiliary fields

- Solution Fields
  Specify basis functions and quadrature for each field in solution.
- Auxiliary Fields
  - Fields associated with parameters and state variables for constitutive models & boundary conditions.
  - Populated from spatial databases.
  - Specify basis functions for each subfield in the auxiliary fields.
- PETSc DMPlex infrastructure unpacks/packs information to/from solution and auxiliary fields and calling finite-element kernels.

Summary of Multiphysics Implementation

We decouple the element definition from the fully-coupled equation, using pointwise kernels that look like the PDE.

Flexibility The cell traversal, handled by the library, accommodates arbitrary cell shapes. The problem can be posed in any spatial dimension with an arbitrary number of physical fields.

Extensibility The library developer needs to maintain only a single method, easing language transitions (CUDA, OpenCL). A new discretization scheme could be enabled in a single place in the code.

Efficiency Only a single routine needs to be optimized. The application scientist is no longer responsible for proper vectorization, tiling, and other traversal optimization.