Pointwise kernels for flexible implementation of crustal deformation physics in PyLith

Brad Aagaard (USGS), baagaard@usgs.gov
Matthew Knepley (SUNY, Buffalo)
Charles Williams (GNS Science)

PyLith Crustal Deformation Modeling Software

Target applications

Solves 2D and 3D problems associated with earthquake faulting and quasistatic and dynamic viscoelastic deformation.

- Short-term tectonics where geometry does not change significantly
- Spatial scales from 1 m to 10^3 km and temporal scales from 0.01 s to 10^5 years
- **Examples**
  - → Pre- and post-seismic deformation with viscoelastoplastic rheologies
  - → Ground-motion simulations with prescribed or spontaneous ruptures
  - → Calculation of 3D and 4D Green's functions
  - → Simulation of multiple earthquake cycles

Design Objectives

Accessible to the beginning user while empowering the advanced user.

Development meets all CIG "standard" best practices and most "target" best practices.

- **Modular**: Users can select modules to run the problem of interest
- **Scalable**: Code runs efficiently on one to thousands of cores
- **Extensible**: Advanced users can change physics to solve their problem

PyLith Releases

PyLith v2 versus v3

PyLith v3 is the result of a multi-year effort to add significant new capabilities for specifying physics and spatial and temporal discretizations.

Feature	Version 2.x (currently available)	Version 3.x (coming Fall 2020)
Governing equations	Hardwired (elasticity)	Flexible (elasticity, incompressible elasticity, poroelasticity)
Temporal discretization	Backward Euler, Newmark (central difference)	PETSc TS (primarily Runge Kutta)
Spatial discretization	Hardwired (1st order)	Flexible (tested up to 4th order)
Finite-element definition	PyLith	PETSc<

Availability

PyLith is open-source and distributed by CIG with 28 releases since 2007.

- Source code + installer utility to build PyLith and its dependencies
  - → Git (development version)
  - → Tarball (releases)
- Binary packages
  - → Linux (32-bit and 64-bit)
  - → OS X (Intel 10.6+)
- User manual with tutorials
- PyLith v2.2.1 downloads
  Binary: 1500+; User Manual: 2200+

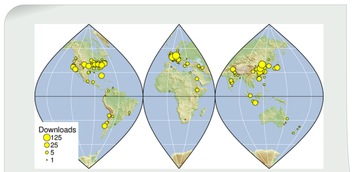

Fig. 2: Map of PyLith downloads in the past 12 months

Multiphysics Formulation Using Pointwise Kernels

Residual Formulation

To use PETSc time stepping algorithms, we put the governing equations in the form:

$$\vec{F}(t,s,\dot{s}) = \vec{G}(t,s), \quad \vec{s}(t_0) = \vec{s}_0 \tag{1}$$

where $\vec{F}$ and $\vec{G}$ are vector functions, t is time, and $\vec{s}$ is the solution vector. **Explicit time stepping with the PETSc TS requires that $\vec{F}(t,s,\dot{s}) = \dot{s}$.**

Using the finite-element method, we manipulate the weak form into integrals over the domain Ω into the form:

$$\int_\Omega \vec{\phi}\cdot\vec{f}_0(t,s,\dot{s}) + \nabla\vec{\phi} : \boldsymbol{f}_1(t,s,\dot{s})\, d\Omega = \int_\Omega \vec{\phi}\cdot\vec{g}_0(t,s) + \nabla\vec{\phi} : \boldsymbol{g}_1(t,s)\, d\Omega, \tag{2}$$

where $\vec{\phi}$ is the trial function, $\vec{f}_0$ and $\vec{g}_0$ are vectors, and $\boldsymbol{f}_1$ and $\boldsymbol{g}_1$ are tensors.

Multiple governing equations give rise to multiple equations of this form. The solution vector $\vec{s}$ may be comprised of several fields, such as displacement $\vec{u}$, velocity $\vec{v}$, pressure p, and temperature T.

PETSc DMPlex implements the computation of equation (2); we supply the physics via the pointwise kernels $\vec{f}_0(t,s,\dot{s})$, $\boldsymbol{f}_1(t,s,\dot{s})$, $\vec{g}_0(t,s)$, and $\boldsymbol{g}_1(t,s)$.

Jacobian Formulation

For implicit time stepping, we need the Jacobian (for example, the stiffness matrix) in addition to the residual. The Jacobian of $F(t,s,\dot{s})$ is $J_F = \frac{\partial F}{\partial s} + t_{shift}\frac{\partial F}{\partial \dot{s}}$ and the Jacobian of $G(t,s)$ is $J_G = \frac{\partial G}{\partial s}$. We put the Jacobians in the form:

$$J_F = \int_\Omega \vec{\phi}\cdot\boldsymbol{J}_{f0}(t,s,\dot{s})\cdot\vec{\psi} + \vec{\phi}\cdot\boldsymbol{J}_{f1}(t,s,\dot{s}) : \nabla\vec{\psi} + \nabla\vec{\phi}\cdot\boldsymbol{J}_{f2}(t,s,\dot{s})\cdot\vec{\psi} + \nabla\vec{\phi} : \boldsymbol{J}_{f3}(t,s,\dot{s}) : \nabla\vec{\psi}\, d\Omega \tag{3}$$

$$J_G = \int_\Omega \vec{\phi}\cdot\boldsymbol{J}_{g0}(t,s)\cdot\vec{\psi} + \vec{\phi}\cdot\boldsymbol{J}_{g1}(t,s) : \nabla\vec{\psi} + \nabla\vec{\phi}\cdot\boldsymbol{J}_{g2}(t,s)\cdot\vec{\psi} + \nabla\vec{\phi} : \boldsymbol{J}_{g3}(t,s) : \nabla\vec{\psi}\, d\Omega, \tag{4}$$

where $\vec{\psi}$ is a basis function.

Expressed in discrete form, the Jacobian for the coupling between solution fields s_i and s_j is

$$J^{s_i s_j} = J_0^{s_i s_j} + J_1^{s_i s_j} B + B^T J_2^{s_i s_j} + B^T J_3^{s_i s_j} B, \tag{5}$$

where B is a matrix of the derivatives of the basis functions, B^T is a matrix of the derivatives of the trial functions.

PETSc DMPlex implements the computation of equations (3) and (4); we supply the pointwise kernels $\boldsymbol{J}_{f0}(t,s,\dot{s})$, $\boldsymbol{J}_{f1}(t,s,\dot{s})$, $\boldsymbol{J}_{f2}(t,s,\dot{s})$, $\boldsymbol{J}_{f3}(t,s,\dot{s})$, $\boldsymbol{J}_{g0}(t,s)$, $\boldsymbol{J}_{g1}(t,s)$, $\boldsymbol{J}_{g2}(t,s)$, and $\boldsymbol{J}_{g3}(t,s)$.

Multiphysics Implementation Using Pointwise Kernels

Design

We decouple the finite-element definition from the weak form equation, using pointwise kernels that look like the PDE.

Each material and boundary condition contribute pointwise kernels.

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

Extensibility PETSc 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 few PETSc routines need to be optimized. The application scientist is no longer responsible for proper vectorization, tiling, and other traversal optimization.

Extensibility

Changing the physics involves:

1. Casting the governing equations in the form of equation (2)
2. Extending the "library" of residual and Jacobian kenels with additional kernels as necessary
3. Adding high-level Python/C++ code to define the governing equations
   - Define parameters.
   - Set kernels corresponding to governing equations.
4. Setting parameter values at runtime
   - Physical fields for solution
   - Constitutive parameters, initial conditions, and boundary condition values

Examples

Elasticity

Strong Form

$$\rho\frac{\partial^2\vec{u}}{\partial t^2} = \vec{f}(\vec{x},t) + \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}(\vec{u}) \text{ in } \Omega, \tag{6}$$
$$\boldsymbol{\sigma}\cdot\vec{n} = \vec{\tau}(\vec{x},t) \text{ on } \Gamma_\tau, \tag{7}$$
$$\vec{u} = \vec{u}_0(\vec{x},t) \text{ on } \Gamma_u, \tag{8}$$
$$\vec{u}^+ - \vec{u}^- = \vec{d} \text{ on } \Gamma_f, \tag{9}$$
$$\boldsymbol{\sigma}\cdot\vec{n} = -\vec{\lambda}(\vec{x},t) \text{ on } \Gamma_{f^+}, \tag{10}$$
$$\boldsymbol{\sigma}\cdot\vec{n} = +\vec{\lambda}(\vec{x},t) \text{ on } \Gamma_{f^-}. \tag{11}$$

where $\vec{u}$ is the displacement vector, ρ is the mass density, $\vec{f}$ is the body force vector, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, and t is time. We specify tractions $\vec{\tau}$ on surface Γ_τ, displacements $\vec{u}_0$ on surface Γ_u, and prescribe fault slip $\vec{d}$ on internal interface Γ_f.

Weak Form: Quasistatic (no inertia)

We use an implicit time integration formulation $F(t,s,\dot{s}) = 0$ with displacement $\vec{u}$ and Lagrange multiplier $\vec{\lambda}$ as the unknowns:

$$\int_\Omega \vec{\psi}^v_{trial}\cdot\underbrace{\vec{f}(\vec{x},t)}_{\vec{f}^v_0} + \nabla\vec{\psi}^v_{trial} : \underbrace{-\boldsymbol{\sigma}(\vec{u})}_{\boldsymbol{f}^v_1} d\Omega + \int_{\Gamma_\tau}\vec{\psi}^v_{trial}\cdot\underbrace{\vec{\tau}(\vec{x},t)}_{\vec{f}^v_0} d\Gamma + \int_{\Gamma_f}\vec{\psi}^{u^+}_{trial}\cdot\underbrace{\left(-\vec{\lambda}(\vec{x},t)\right)}_{\vec{f}^{u^+}_0} + \vec{\psi}^{u^-}_{trial}\cdot\underbrace{\left(+\vec{\lambda}(\vec{x},t)\right)}_{\vec{f}^{u^-}_0} d\Gamma = 0 \tag{12}$$

$$\int_{\Gamma_f}\vec{\psi}^\lambda_{trial}\cdot\underbrace{\left(-\vec{u}^+(\vec{x},t) + \vec{u}^-(\vec{x},t) + \vec{d}(\vec{x},t)\right)}_{\vec{f}^\lambda_0} d\Gamma = 0. \tag{13}$$

Weak Form: Dynamic

The dynamic elasticity and prescribed fault slip equations form a differential-algebraic set of equations (DAEs) with index 2. We use the PETSc implicit-explicit (IMEX) time integration formulation with displacement $\vec{u}$, velocity $\vec{v}$, and Lagrange multiplier $\vec{\lambda}$ as the unknowns. We solve the displacement-velocity and elasticity equations using explicit time integration. We introduce the Lagrange multiplier into the prescribed slip equation and solve it using implicit time integration.

$$\frac{\partial\vec{u}}{\partial t} = M_u^{-1}\int_\Omega\vec{\psi}^u_{trial}\cdot\underbrace{\vec{v}}_{\vec{g}^u_0} d\Omega, \tag{14}$$

$$\frac{\partial\vec{v}}{\partial t} = M_v^{-1}\int_\Omega\vec{\psi}^v_{trial}\cdot\underbrace{\vec{f}(\vec{x},t)}_{\vec{g}^v_0} + \nabla\vec{\psi}^v_{trial} : \underbrace{-\boldsymbol{\sigma}(\vec{u})}_{\boldsymbol{g}^v_1} d\Omega + M_v^{-1}\int_{\Gamma_\tau}\vec{\psi}^v_{trial}\cdot\underbrace{\vec{\tau}(\vec{x},t)}_{\vec{g}^v_0} d\Gamma + M_{v^+}^{-1}\int_{\Gamma_f}\vec{\psi}^{v^+}_{trial}\cdot\underbrace{\left(-\vec{\lambda}(\vec{x},t)\right)}_{\vec{g}^{v^+}_0} d\Gamma + M_{v^-}^{-1}\int_{\Gamma_f}\vec{\psi}^{v^-}_{trial}\cdot\underbrace{\left(+\vec{\lambda}(\vec{x},t)\right)}_{\vec{g}^{v^-}_0} d\Gamma, \tag{15}$$

$$\int_{\Gamma_f^-}\vec{\psi}^\lambda_{trial}\cdot\underbrace{\frac{1}{\rho(\vec{x})}\left(\vec{\lambda} - \vec{f}(\vec{x},t) - \frac{\nabla\rho(\vec{x})}{\rho(\vec{x})}\cdot\boldsymbol{\sigma}(\vec{u})\right)}_{\vec{f}^\lambda_0} + \nabla\vec{\psi}^\lambda_{trial} : \underbrace{\left(-\frac{1}{\rho(\vec{x})}\boldsymbol{\sigma}(\vec{u})\right)}_{\boldsymbol{f}^\lambda_1} d\Gamma + \int_{\Gamma_f^+}\vec{\psi}^\lambda_{trial}\cdot\underbrace{\frac{1}{\rho(\vec{x})}\left(\vec{\lambda} + \vec{f}(\vec{x},t) + \frac{\nabla\rho(\vec{x})}{\rho(\vec{x})}\cdot\boldsymbol{\sigma}(\vec{u})\right)}_{\vec{f}^\lambda_0} + \nabla\vec{\psi}^\lambda_{trial} : \underbrace{\left(+\frac{1}{\rho(\vec{x})}\boldsymbol{\sigma}(\vec{u})\right)}_{\boldsymbol{f}^\lambda_1} d\Gamma + \int_{\Gamma_f}\vec{\psi}^\lambda_{trial}\cdot\underbrace{\frac{\partial^2\vec{d}(\vec{x},t)}{\partial t^2}}_{\vec{f}^\lambda_0} d\Gamma = 0, \tag{16}$$

$$M_u = Lump\left(\int_\Omega\psi^u_{trial i}\underbrace{\delta_{ij}}_{J^v_0}\psi^u_{basis j}\, d\Omega\right), \tag{17}$$

$$M_v = Lump\left(\int_\Omega\psi^v_{trial i}\underbrace{\rho(\vec{x})\delta_{ij}}_{J^v_0}\psi^v_{basis j}\, d\Omega\right). \tag{18}$$

Quasistatic Incompressible Elasticity

Strong Form

$$0 = \vec{f}(\vec{x},t) - \vec{\nabla}p + \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}^{dev}(\vec{u}) \text{ in } \Omega, \tag{19}$$
$$0 = \vec{\nabla}\cdot\vec{u} + \frac{p}{\kappa(\vec{x})} \text{ in } V, \tag{20}$$
$$\boldsymbol{\sigma}\cdot\vec{n} = \vec{\tau}(\vec{x},t) \text{ on } \Gamma_\tau, \tag{21}$$
$$\vec{u} = \vec{u}_0(\vec{x},t) \text{ on } \Gamma_u, \tag{22}$$
$$\vec{u}^+ - \vec{u}^- = \vec{d} \text{ on } \Gamma_f, \tag{23}$$

where $\vec{u}$ is the displacement vector, p is the pressure (negative of mean stress), $\vec{f}$ is the body force vector, κ is the bulk modulus. $\boldsymbol{\sigma}$ is the Cauchy stress tensor, and t is time. We specify tractions $\vec{\tau}$ on surface Γ_τ, displacements $\vec{u}_0$ on surface Γ_u, pressure $\vec{p}_0$ on surface Γ_p, and prescribe fault slip $\vec{d}$ on internal interface Γ_f. For total incompressibility Poisson's ratio is 0.5, the bulk modulus is infinite, the volumetric strain is zero, and the pressure is finite.

Weak Form

We use an implicit formulation $F(t,s,\dot{s}) = 0$ with displacement $\vec{u}$, pressure p, and Lagrange multiplier $\vec{\lambda}$ as the unknowns:

$$\int_\Omega\vec{\psi}^u_{trial}\cdot\underbrace{\vec{f}(\vec{x},t)}_{\vec{f}^u_0} + \nabla\vec{\psi}^u_{trial} : \underbrace{\left(-\boldsymbol{\sigma}^{dev}(\vec{u}) + p\boldsymbol{I}\right)}_{\boldsymbol{f}^u_0} d\Omega + \int_{\Gamma_\tau}\vec{\psi}^u_{trial}\cdot\underbrace{\vec{\tau}(\vec{x},t)}_{\vec{f}^u_0} d\Gamma, = 0 \tag{24}$$

$$\int_\Omega\psi^p_{trial}\underbrace{\left(\vec{\nabla}\cdot\vec{u} + \frac{p}{\kappa(\vec{x})}\right)}_{f^p_0} d\Omega = 0. \tag{25}$$

$$\int_{\Gamma_f}\vec{\psi}^\lambda_{trial}\cdot\underbrace{\left(-\vec{u}^+(\vec{x},t) + \vec{u}^-(\vec{x},t) + \vec{d}(\vec{x},t)\right)}_{\vec{f}^\lambda_0} d\Gamma = 0. \tag{26}$$

Comments

- The pointwise kernels ($\vec{f}_0$, $\boldsymbol{f}_1$, and g_0) look like terms in the PDE.
- We can implement a variety of physics (quasistatic and dynamic elasticity and quasistatic incompressibility) using a relative small library of pointwise kernels.
- The implicit part of the implicit-explicit time integration for dynamic simulations only involves degrees of freedom associated with faults, whereas the explicit part involves degrees of freedom associated with the entire domain.