# 2021 PyLith Hackathon

June 7-10 and 14-16
Online

## Table of Contents



## Summary


The 2nd PyLith Hackathon held online June 7-10 and 14-16 involved 15 participants working on 5 different projects. The projects included earthquake cycle modeling, poroelasticity, static Green's functions, small strain and gravitational body forces, and integrating PETSc mesh importers and libCEED into PyLith, All projects made significant progress in implementing new features or transferring features from the previous PyLith release to the new multiphysics setup. Participants learned how to navigate the PyLith code base, how various features are implemented, how to make changes to the code, and implement tests. The core PyLith development team benefitted from discussions with the other participants about the technical aspects of the various projects as well as general discussions about PyLith design and implementing other new features. The online format was necessary due to the COVID-19 pandemic and certainly impacted productivity but provided greater flexibility in personal schedules and eliminated travel. A hybrid (online and in-person) format in which the in-person portion is preceded and followed by shorter remote portions may provide a good balance.



Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

## Participants

Brad Aagaard (USGS, PyLith developer)
Kali Allison (UC Davis)
Alexander Berne (Caltech)
Grant Block (University of New Mexico)
Jed Brown (University of Colorado Boulder)
Jiaqi Fang (Caltech)
Michael Gurnis (Caltech)
Taeho Kim (Caltech)
Rey Koki (University of Colorado Boulder)
Matthew Knepley (University at Buffalo, PyLith developer)
Shengduo Liu (Caltech)
Théa Ragon (Caltech)
Mousumi Roy (University of New Mexico)
Robert Walker (University at Buffalo, PyLith developer)
Charles Williams (GNS Science, PyLith developer)

## Format

Due to the COVID-19 pandemic we held the hackathon entirely online. We met for 7 days spread over two weeks. We held “core hours” each day from 1:00pm - 2:30pm PDT during which we met as one group to discuss progress and obstacles. We also used core hours for discussion of general topics, such as the code layout and design, Git workflow, and PyLith development plan.

The participants were organized into 5 groups around a project of mutual interest. Each group worked independently outside of core hours, meeting with developers at least once a day. Some groups worked closely together, meeting multiple times a day; others worked asynchronously usually due to disparate time zones.

### Tools

Wonder.me was used for the online meetings. The “room” contained a box for each group to meet independently as well as a box for a common meeting area. This made it easy for participants to interact within and across projects and for developers to meet with any project on short notice.

We used a shared Google Doc for meeting notes, including sharing links, creating TODO lists.

We used a Slack workspace with a general channel and separate channels for each for group. Slack was used to organize project meetings, report obstacles, and short communication outside core hours. Support for code blocks made it the preferred medium for sharing error messages, etc.

### Developer environment

Most participants used Visual Studio Code attached to a Docker container for an integrated development environment. External dependencies were provided by the Docker container; participants built CIG-related code (pythia, spatialdata, PETSc, and PyLith) in a persistent Docker storage volume. This made it easy for participants to update both PETSc and PyLith as necessary. The ability to simultaneously

view and edit files remotely using Visual Studio Code Live Share greatly facilitated implementing and debugging changes to the code.

# Projects

## Earthquake Cycle

*Kali Allison and Taeho Kim*

This project focused on adding the ability to solve multiple, coupled problems in a single PyLith simulation. The primary application will be coupling interseismic deformation solved via a quasistatic problem with coseismic deformation solved via a dynamic problem to resolve the earthquake cycle, including viscoelasticity, fault creep, propagating earthquake rupture, and radiated seismic waves. The group changed the top-level code to allow multiple, independent problems and created a full-scale test to verify the implementation. Testing multiple problems exposed some limitations in the full-scale test apparatus and led Brad Aagaard to implement a simpler, more flexible apparatus immediately following the Hackathon. Brad Aagaard worked with the group to design coupling between problems; the implementation will be part of ongoing development.

## Poroelasticity

*Grant Block, Shengduo Liu, Mousumi Roy, and Robert Walker*

This project focused initially on expanding the use of the new poroelastic formulation available in PyLith v3 (based on the new multiphysics support), and providing a hands-on user tutorial of the PyLith poroelastic capability. The group also produced a new point source representation for fluid injection, resulting in a new source class definition, and an analytical test case that will lead to a full-scale test.

Additionally, the group began working on a fully poroelastic fault rheology. This new model is flux conservative across the fault, with a fault pressure variable added to the existing Lagrange multiplier for fault mechanics.

## Static Green's Functions

*Alexander Berne and Théa Ragon*

The goal of this project was to transfer the implementation of static Green's functions in PyLith v2.2.2 to the new multiphysics setup in PyLith v3. The two main tasks included new implementations of the Green's function problem and fault slip impulses. The group completed initial drafts of these changes with construction of a full-scale test part of ongoing development.

## Small Strain and Gravitational Body Forces

*Jiaqi Fang and Michael Gurnis*

This project focused on verifying the small strain formulation used in PyLith v2.2.2 and transferring it to the new multiphysics setup in PyLith v3. The group confirmed that the general setup of small strain using Green-Lagrange strain and Piola Kirchoff stress formulation in v2.2.2 was correct. The group

implemented isotropic linear elasticity with small strain and constructed a test using the Method of Manufactured Solutions. Extending the formulation to accommodate all bulk rheologies is ongoing.

## Integrating libCEED and PETSc Mesh Importers into PyLith

*Jed Brown and Rey Koki*

This project aimed to compute the integrals for the finite element residual in PyLith using the libCEED libraries. libCEED provides a flexible, low-level interface for cellwise integration, and provides portability across architectures including GPUs. libCEED then leverages the PETSc DMPlex parallelism already in PyLith. Rey Koki added an interface for creating a PyLith mesh using the built-in DMPlex creation routines, and the group started developing a bulk elasticity residual implementation using the libCEED representation. The remaining work is making the geometric factors from PETSc available to libCEED for computing the residual. This plumbing will be provided in part by improvements to DMPlex integration with libCEED.