# Comparing poroelastostatics and poroelastodynamics: Numerics, solvers and algorithms

**Saumik Dana**
University of Southern California
Los Angeles, CA 90007
sdana@usc.edu

**ABSTRACT**

Understanding the causality between the events leading upto and post fault slip and the earthquake recording is important for seismic design and monitoring of underground structures, bridges and reinforced concrete buildings as well as climate mitigation projects like carbon sequestration and energy technologies like enhanced geothermal systems or oilfield wastewater disposal. While the events leading upto fault slip are typically governed by poroelastostatics, the events post fault slip can easily transition into poroelastodynamics territory due to runaway fault slip velocities. There are marked differences in the numerics of poroelastostatics and poroelastodynamics, and a simple switch from one algorithm to another based on fault slip velocities is not trivial. In fact, an understanding of expected fault slip velocities is critical apriori, as an algorithm which can seamlessly transition from time marching in poroelastostatics realm to poroelastodynamics realm and vice-versa is extremely difficult to achieve. We present the numerics of both physics and point out the differences between the two in this work.

## 1 Introduction

The earthquake cycle, from slow deformation associated with interseismic behavior to rapid deformation associated with earthquake rupture, spans spatial scales ranging from fractions of a meter associated with the size of contact asperities on faults and individual grains to hundreds of kilometers associated with plate boundaries [1]. Similarly, temporal scales range from fractions of a second associated with slip at a point during earthquake rupture to hundreds of years of strain accumulation between earthquakes. In many cases, earthquakes are triggered after pore pressure perturbations activate critically stressed seismogenic faults [1], not just due to natural causes like earth tides [2], rainfall [3], snowfall [4], typhoons [5], but also due to human activity [6]. As faults slip, the induced stress field spawns seismic waves, which travel through and around the earth and can be recorded with seismometers. There are several different kinds of seismic waves, and they all move in different ways. The two main types of waves are body waves (P- and S-) and surface waves. Body waves can travel through the Earth's inner layers, but surface waves can only move along the surface of the planet like ripples on water. Understanding the causality between the events leading upto and post fault slip and the earthquake recording is important for seismic design and monitoring of underground structures [7], bridges [8] and reinforced concrete buildings [9] as well as climate mitigation projects like carbon sequestration [10] and energy technologies like enhanced geothermal systems [11] or oilfield wastewater disposal [12].

The complexity of the many physical processes operating over this vast range of scales leads most researchers to focus on a narrow space-time window to isolate just one or a few processes; the limited spatial and temporal coverage of observations also often justifies this narrow focus. For quasi-static simulations [13–24], the inertial term is ignored and time dependence only enters through the constitutive models and the loading conditions. As a result, a quasi-static simulation is the solution to a series of static problems with potentially time-varying physical properties and boundary conditions. In dynamic simulations, we shall include the

inertial term to resolve the propagation of seismic waves, with an intended focus on applications for earthquake physics and ground-motion simulations. The only place where things can accelerate fast is on the fault and the only time it happens is when fault slips seismically ($> 1$ cm/sec). If the fault slip is aseismic or creeping ($< 0.1$ mm/sec), then accelerations are negligible and quasi-static and quasi-dynamic [25] approximations are valid. In our models, we know the fault is the location where things will happen, but we do not know a priori is whether it will be seismic, aseismic or in the transition region. While back-of-the-envelope estimations can be done from the strain energy accumulated in the rock surrounding the faults, the fault's friction properties, and the rock's elastic moduli, the more optimal and accurate option is to use the simulator to do this estimation by equipping it with the physics of elastodynamics.

### 1.1 Code development

The simulator is a mixed programming language hybrid MPI and OpenMP framework in which the open source geodynamics code [26, 27] written in Python and C++ is MPI-based and the in-house multiphase/multicomponent flow code written in C++ is OpenMP-based [13, 14]. There are three layers in which the code framework operates out of separate directories

- ✓ The high-level Python code in the geodynamics module makes use of Pyre [28] to link the sub-modules together at runtime
- ✓ The low-level C++ code performs operations on matrices and vectors in parallel using the Standard Template Library [29]
- ✓ The SWIG [30] code with all the .i files, and SWIG-generated python code and MODULE-NAME_wrap.cxx which wraps the Python code around the C++ code to create python modules.

The numpy library [31] allows direct manipulation in Python of multi-dimensional arrays, using memory allocated in C++. In general, Python's ability to easily expose a library API and the use of numpy arrays for data interchange, make it an excellent tool for combining libraries at a higher level to attack more complex multi-model, multi-physics, multi-scale problems [32]. Some of the code snippets are provided in A. Changes made to the C++ code entail a makefile related compilation of the code base. If there are changes to the C++ header files that need to be exposed to the input file, then those changes have to be copied to the corresponding module's .i file. This means SWIG has to be run again on that module to update MODULENAME_wrap.cxx and Python files of the module before building the executable using its Makefile. The central disadvantage for such multi-language code frameworks comes in debugging. There are certainly hurdles introduced into the build system, since different compilation and links steps are needed and many more tests are needed to verify interoperability.

Currently, the code base is hybrid MPI and OpenMP, in which the geodynamics code is MPI-parallel and the flow code is OpenMP-parallel. This hybrid framework, although accurate, brings in efficiency issues for large size simulations. While the geodynamics elements are distributed among MPI processes with each process running the same tasks on a different sub-grids of the main grid, the flow elements are all stacked together in the root process, and the tasks are distributed among threads in the root process. When the data structures port information from the geodynamics code to the flow code, they are assembled into the root process using the MPI_Gather command, and then transfered to the flow simulator. This porting can happen multiple times in each time step depending on the convergence requirements of the staggered solution algorithm, and this adds inefficiency to the overall computational run-time. The first option to increase efficiency is to port the OpenMP parallel functionality to MPI parallel on the flow side, and the second option is to offload the OpenMP parallel computations onto a GPGPU by incorporating CUDA and OpenCL libraries [33].

The second option is definitely more engaging as it leverages the massive computational power of the GPGPU. That being said, the advent of new, massively parallel architectures [34] has greatly increased the penalty for suboptimal non-vectorized code while providing much more energy efficient performance. They also increase the imbalance between flop rate and memory bandwidth or latency, so that thousands of flops can be needed to cover outstanding memory references. GPUs in particular have very high memory latency coupled with a wide bus, making the memory access pattern critical for good performance. In addition, the size of fast cache memory per core has shrunk dramatically, so that it cannot easily be used to hide irregular memory access.

## 2 Model equations and weak form

The governing partial differential equation for displacement $\mathbf{u} : \Omega \times [0, T] \rightarrow \mathbb{R}^3$ is the linear momentum balance *in the presence of inertia* given by

$$\nabla \cdot \boldsymbol{\sigma} + \rho_b \mathbf{g} = \rho_b \ddot{\mathbf{u}} \tag{1}$$

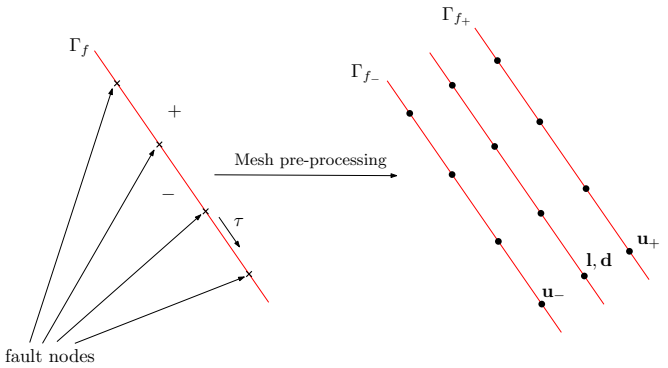


Figure 1: A fault surface $\Gamma_f$ (shown as a line in a 2D domain) is processed during mesh processing to create three surfaces: the positive side surface $\Gamma_{f+}$ containing $\boldsymbol{u}_+$, the negative side surface $\Gamma_{f-}$ containing $\boldsymbol{u}_-$, and the middle slip surface containing the fault effective traction vector $\boldsymbol{l}$ and the slip vector $\boldsymbol{d}$. The fault surface need not be planar i.e. it can be curved with the normal vector $\boldsymbol{n}$ varying in space on the fault surface.

with the constitutive laws relating poroelastic stress tensor $\boldsymbol{\sigma}$, effective stress tensor $\boldsymbol{\sigma}'$, strain tensor $\boldsymbol{\epsilon}$, volumetric strain $\epsilon = tr(\boldsymbol{\epsilon})$ and pore pressure $p$ given by

$$\begin{aligned} \boldsymbol{\sigma} &= \boldsymbol{\sigma}' - bp\mathbf{I} \\ \boldsymbol{\sigma}' &= \lambda\epsilon\mathbf{I} + 2G\boldsymbol{\epsilon} = \boldsymbol{D}\boldsymbol{\epsilon} \end{aligned} \tag{2}$$

with small strain kinematics dictating the relationship between strains and displacements as follows

$$\boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla^T\mathbf{u}) \tag{3}$$

with boundary and initial conditions given by

$$\begin{aligned} \mathbf{u} = \overline{\mathbf{u}} \text{ on } \Gamma_D,\ \dot{\mathbf{u}} = \dot{\overline{\mathbf{u}}} \text{ on } \Gamma_D,\ \boldsymbol{\sigma}^T\mathbf{n} = \overline{\mathbf{t}} \text{ on } \Gamma_N \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}),\ \dot{\mathbf{u}}(\mathbf{x}, 0) = \dot{\mathbf{u}}_0(\mathbf{x}) \end{aligned}$$

where $\Gamma_D$ and $\Gamma_N$ are Dirichlet and Neumann boundaries respectively, $\rho_b$ is the bulk density, $\lambda$ is a Lame constant, $G$ is the shear modulus, $b = 1 - \frac{K_b}{K_s}$ is the Biot parameter, $K_b \equiv \lambda + \frac{1}{3}2G$ is the drained bulk modulus, $K_s$ is the bulk modulus of the solid grains, $\mathbf{t}$ is the traction boundary condition, and $\mathbf{I}$ is the second order identity tensor. Following a conventional finite-element formulation, we construct the weak form by taking the dot product of the governing equation with a weighting function $\boldsymbol{\eta}$ and setting the integral over the domain equal to zero,

$$\int_\Omega \nabla\boldsymbol{\eta} : (\boldsymbol{\sigma}' - bp\mathbf{I})\ d\Omega - \int_\Omega \boldsymbol{\eta}\cdot\rho_b\boldsymbol{g}\ d\Omega - \int_{\Gamma_N} \boldsymbol{\eta}\cdot\overline{\mathbf{t}}\ d\Gamma - \int_\Omega \boldsymbol{\eta}\cdot\rho_b\ddot{\mathbf{u}}\ d\Omega = 0 \tag{4}$$

Using a domain decomposition approach, we consider the fault surface as an interior boundary between two domains. As shown in Fig. 1, we assign a fault normal direction to this interior boundary and "positive" and "negative" labels to the two sides of the fault, such that the fault normal is the vector from the negative side of the fault to the positive side of the fault. Slip on the fault is the displacement of the positive side relative to the negative side:

$$(\boldsymbol{u}_+ - \boldsymbol{u}_-) - \boldsymbol{d} = \mathbf{0} \text{ on } \Gamma_f, \tag{5}$$

Recognizing that the tractions on the fault surface are analogous to the boundary tractions, we add in the contributions from integrating the Lagrange multipliers $\boldsymbol{l} \equiv \boldsymbol{\sigma}'\boldsymbol{n}$ over the fault surface in Eq. (4) to get

$$\begin{aligned} &\int_\Omega \nabla\boldsymbol{\eta} : (\boldsymbol{\sigma}' - bp\mathbf{I})\ d\Omega - \int_\Omega \boldsymbol{\eta}\cdot\rho_b\boldsymbol{g}\ d\Omega - \int_{\Gamma_N} \boldsymbol{\eta}\cdot\overline{\mathbf{t}}\ d\Gamma - \int_\Omega \boldsymbol{\eta}\cdot\rho_b\ddot{\mathbf{u}}\ d\Omega \\ &+ \int_{\Gamma_{f_+}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_+\boldsymbol{n})\ d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_-\boldsymbol{n})\ d\Gamma = 0 \end{aligned} \tag{6}$$

where $p_+$ and $p_-$ are the pore pressures on the 'positive' and the 'negative' side of the fault. In the *absence of inertia* for the poroelastostatics case, the weak form (Eq. (6)) is reduced to

$$\int_\Omega \nabla\boldsymbol{\eta} : (\boldsymbol{\sigma}' - bp\mathbf{I})\, d\Omega - \int_\Omega \boldsymbol{\eta}\cdot\rho_b\boldsymbol{g}\, d\Omega - \int_{\Gamma_N} \boldsymbol{\eta}\cdot\bar{\boldsymbol{t}}\, d\Gamma + \int_{\Gamma_{f_+}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_+\boldsymbol{n})\, d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_-\boldsymbol{n})\, d\Gamma = 0 \tag{7}$$

### 2.1 Fault friction

The magnitude of the effective normal traction on the fault is

$$\sigma_n' = \boldsymbol{l}\cdot\boldsymbol{n} \tag{8}$$

A positive value of $\sigma_n'$ indicates that a tensile effective stress is transmitted across the fault surface. The Kuhn-Tucker conditions of contact mechanics are obeyed such that no penetration occurs and the effective normal traction stays compressive at the contact surface. The shear traction vector is, by definition, tangent to the fault surface and its magnitude is

$$\tau = |\boldsymbol{l} - \sigma_n'\boldsymbol{n}| \equiv |\boldsymbol{l} - (\boldsymbol{l}\cdot\boldsymbol{n})\boldsymbol{n}| \tag{9}$$

Shear tractions on the fault are limited by fault friction, or fault strength. A fault constitutive model is used to compute the frictional stress $\tau_f$ on the fault as

$$\tau_f = \begin{cases} \tau_c - \mu_f\boldsymbol{l}\cdot\mathbf{n}, & \boldsymbol{l}\cdot\mathbf{n} < 0, \\ \tau_c, & \boldsymbol{l}\cdot\mathbf{n} \geq 0 \end{cases} \tag{10}$$

where $\tau_c$ is the cohesive strength of the fault, and $\mu_f$ is the coefficient of friction, which is modeled as a function of displacement evolution on the fault. The most popular among the fault friction evolution models is the rate and state model [35, 36].

### 2.2 Fault pressure

We use the Mohr-Coulomb theory [37] to define the stability criterion for the fault. When the shear traction on the fault is below the friction stress, $\tau \leq \tau_f$, the fault does not slip. When the shear traction is larger than the friction stress, $\tau > \tau_f$, the contact problem is solved to determine the Lagrange multipliers and slip on the fault, such that the Lagrange multipliers are compatible with the frictional stress. Instead of assessing fault stability on both sides of the fault separately, we define a fault pressure $p_f$ that is a function of the pressures on the two sides as

$$p_f = \frac{p_- + p_+}{2} \tag{11}$$

By univocally defining the pressure at the fault, we also univocally define the effective traction at the fault (the Lagrange multiplier $\boldsymbol{l}$).

### 2.3 Normalization

We expect the displacements to be generally on the order of $mm$ to $m$, whereas the fault tractions will be on the order of $MPa$. Thus, if we use dimensioned quantities in SI units, then we would expect the solution to include terms that differ by up to 9 orders of magnitude. This results in a rather ill-conditioned system. This ill-conditioning is avoided by nondimensionalizing all of the quantities involved in the problem based upon user-specified scales [26, 27], facilitating the formation of well-conditioned systems of equations for problems across a wide range of spatial and temporal scales.

## 3 Computational framework for poroelastodynamics

### 3.1 Fully discrete system of equations for poroelastodynamics

In lieu of Eqs. (6), (11) and (5), the problem statement is to find $(\boldsymbol{u}, \boldsymbol{l})$ belonging to appropriate functional spaces satisfying the essential boundary conditions ($\boldsymbol{u} = \overline{\boldsymbol{u}}$ on $\Gamma_D$) such that

$$\int_\Omega \nabla\boldsymbol{\eta} : (\boldsymbol{\sigma}' - bp\mathbf{I})\, d\Omega - \int_\Omega \boldsymbol{\eta}\cdot\rho_b\boldsymbol{g}\, d\Omega - \int_{\Gamma_N} \boldsymbol{\eta}\cdot\bar{\boldsymbol{t}}\, d\Gamma + \int_\Omega \boldsymbol{\eta}\cdot\rho_b\ddot{\mathbf{u}}\, d\Omega$$
$$+ \int_{\Gamma_{f_+}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_f\boldsymbol{n})\, d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_f\boldsymbol{n})\, d\Gamma = 0, \tag{12}$$

$$\int_{\Gamma_{f_+}} \boldsymbol{\eta}\cdot\boldsymbol{u}_+\, d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}\cdot\boldsymbol{u}_-\, d\Gamma - \int_{\Gamma_f} \boldsymbol{\eta}\cdot\boldsymbol{d}\, d\Gamma = 0 \tag{13}$$

for all test functions $\boldsymbol{\eta}$ ($n_{\text{dim}} \times 1$ vector) belonging to the appropriate functional space satisfying $\boldsymbol{\eta} = \mathbf{0}$ on $\Gamma_u$. The displacement, the Lagrange multiplier, and the slip fields are approximated as follows:

$$\boldsymbol{u} \approx \boldsymbol{u}_h = \sum_{b=1}^{n_{\text{node}}} \eta_b \boldsymbol{U}_b, \quad \boldsymbol{l} \approx \boldsymbol{l}_h = \sum_{b=1}^{n_{f,\text{node}}} \eta_b \boldsymbol{L}_b, \quad \boldsymbol{d} \approx \boldsymbol{d}_h = \sum_{b=1}^{n_{f,\text{node}}} \eta_b \boldsymbol{D}_b,$$

where $n_{\text{node}}$ is the total number of nodes and $n_{f,\text{node}}$ is the number of Lagrange nodes. After substitution of the finite element approximations into the weak form of the problem (Eqs. (12)–(13)), we obtain the semi-discrete equations in residual form for all nodes $a$ and lagrange nodes $\bar{a}$:

$$\int_\Omega \boldsymbol{B}_a^T(\boldsymbol{\sigma}_h'^n - bp_h^n\mathbf{1})d\Omega - \int_\Omega \boldsymbol{\eta}_a^T\rho_{b,h}^n\boldsymbol{g}d\Omega - \int_{\Gamma_N}\boldsymbol{\eta}_a^T\bar{\boldsymbol{t}}d\Gamma + \int_\Omega \boldsymbol{\eta}_a^T\rho_{b,h}^n\ddot{\mathbf{u}}_h^n d\Omega$$
$$+ \int_{\Gamma_{f_+}} \boldsymbol{\eta}_a^T(\boldsymbol{l}_h^n - bp_{f,h}^n\boldsymbol{n})d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}_a^T(\boldsymbol{l}_h^n - bp_{f,h}^n\boldsymbol{n})d\Gamma = \mathbf{0}, \tag{14}$$

$$\int_{\Gamma_{f_+}} \boldsymbol{\eta}_{\bar{a}}^T\boldsymbol{u}_{h_+}^{n+1}d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}_{\bar{a}}^T\boldsymbol{u}_{h_-}^{n+1}\, d\Gamma - \int_{\Gamma_f}\boldsymbol{\eta}_{\bar{a}}^T\boldsymbol{d}_h^{n+1}d\Gamma = \mathbf{0} \tag{15}$$

where $\boldsymbol{\eta}$ and $\boldsymbol{B}$ are nodal matrices for the shape function and its gradient respectively. We employ explicit time stepping via Newmark's method [38] with a central difference scheme where the acceleration is given by

$$\ddot{\mathbf{u}}_h^n = \frac{\mathbf{u}_h^{n+1} - 2\mathbf{u}_h^n + \mathbf{u}_h^{n-1}}{(\Delta t)^2} \equiv \frac{d\mathbf{u}_h^n - \mathbf{u}_h^n + \mathbf{u}_h^{n-1}}{(\Delta t)^2} \tag{16}$$

the velocity is given by

$$\dot{\mathbf{u}}_h^n = \frac{\mathbf{u}_h^{n+1} - \mathbf{u}_h^{n-1}}{2\Delta t} \equiv \frac{d\mathbf{u}_h^n + \mathbf{u}_h^n - \mathbf{u}_h^{n-1}}{2\Delta t} \tag{17}$$

where $\Delta t$ is the time step and $d\mathbf{u}_h^n = \mathbf{u}_h^{n+1} - \mathbf{u}_h^n$. The fully discrete system of equations is

$$\boldsymbol{R}_{u,a}^{Dyn} = \int_\Omega \boldsymbol{B}_a^T(\boldsymbol{\sigma}_h'^n - bp_h^n\mathbf{1})d\Omega - \int_\Omega \boldsymbol{\eta}_a^T\rho_{b,h}^n\boldsymbol{g}d\Omega - \int_{\Gamma_N}\boldsymbol{\eta}_a^T\bar{\boldsymbol{t}}d\Gamma + \int_\Omega \boldsymbol{\eta}_a^T\rho_{b,h}^n\left(\frac{d\mathbf{u}_h^n - \mathbf{u}_h^n + \mathbf{u}_h^{n-1}}{(\Delta t)^2}\right)d\Omega$$
$$+ \int_{\Gamma_{f_+}} \boldsymbol{\eta}_a^T(\boldsymbol{l}_h^n - bp_{f,h}^n\boldsymbol{n})d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}_a^T(\boldsymbol{l}_h^n - bp_{f,h}^n\boldsymbol{n})d\Gamma = \mathbf{0} \tag{18}$$

$$\boldsymbol{R}_{l,\bar{a}}^{Dyn} = \int_{\Gamma_{f_+}} \boldsymbol{\eta}_{\bar{a}}^T\boldsymbol{u}_{h_+}^{n+1}d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}_{\bar{a}}^T\boldsymbol{u}_{h_-}^{n+1}\, d\Gamma - \int_{\Gamma_f}\boldsymbol{\eta}_{\bar{a}}^T\boldsymbol{d}_h^{n+1}d\Gamma = \mathbf{0} \tag{19}$$

We find the system Jacobian matrix by isolating the term for the temporal increment in displacement field at time step $n$ and Lagrange multipliers at time step $n+1$. The system of linear equations is:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{C}^T \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} d\boldsymbol{U} \\ \boldsymbol{L} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{R}_{u,a}^{Dyn} \\ \boldsymbol{R}_{l,\bar{a}}^{Dyn} \end{bmatrix} \tag{20}$$

where the block matrices are obtained via element-by-element assembly of the individual nodal contributions to the Jacobian:

$$\boldsymbol{K} = \int_{\Omega} \frac{1}{(\Delta t)^2} \boldsymbol{\eta}_a^T \rho_b \boldsymbol{\eta}_b \, d\Omega, \qquad \boldsymbol{C} = \int_{\Gamma_f} \boldsymbol{\eta}_a^T \boldsymbol{\eta}_{\bar{b}} \, d\Gamma \tag{21}$$

The matrix $\boldsymbol{C}$ is spectrally equivalent to the identity, because it involves integration of products of the basis functions. This makes the traditional LBB stability criterion trivial to satisfy by choosing the space of Lagrange multipliers to be exactly the space of displacements, restricted to the fault [27]. This means we simply need to know the distance between any pair of vertices spanning the fault, which can be expressed as a relative displacement, i.e., fault slip. To visualize the fault contribution to the linear system, we can write Eq. (20) as

$$\left[\begin{array}{ccc:c} \boldsymbol{K}_{rr} & \boldsymbol{K}_{r+} & \boldsymbol{K}_{r-} & \boldsymbol{0} \\ \boldsymbol{K}_{+r} & \boldsymbol{K}_{++} & \boldsymbol{0} & \boldsymbol{C}_+^T \\ \boldsymbol{K}_{-r} & \boldsymbol{0} & \boldsymbol{K}_{--} & -\boldsymbol{C}_-^T \\ \hdashline \boldsymbol{0} & \boldsymbol{C}_+ & -\boldsymbol{C}_- & \boldsymbol{0} \end{array}\right] \left[\begin{array}{c} d\boldsymbol{U}_r \\ d\boldsymbol{U}_+ \\ d\boldsymbol{U}_- \\ \hline \boldsymbol{L} \end{array}\right] = - \left[\begin{array}{c} \boldsymbol{R}_{u,r} \\ \boldsymbol{R}_{u,+} \\ \boldsymbol{R}_{u,-} \\ \hline \boldsymbol{R}_l \end{array}\right] \tag{22}$$

where the top row corresponds to displacement nodes excluding the fault positive and negative side nodes.

### 3.2 Numerical damping

In explicit time-stepping formulations for elasticity, boundary conditions and fault slip can excite short waveform elastic waves that are not accurately resolved by the discretization. To reduce these high frequency oscillations, numerical damping is introduced [26, 27], in which the kinematics connecting strains and displacements (Eq. (3)) is re-formulated in terms of an adjusted displacement using an artificial viscosity term $\eta$ as follows

$$\boldsymbol{u}^{adj} = \boldsymbol{u} + \eta \Delta t \, \boldsymbol{u}$$

### 3.3 Absorbing boundary conditions

We employ absorbing boundary conditions [26, 27] to prevent seismic waves reflecting off of a boundary by placing simple dashpots on the boundary. Normally incident dilatational and shear waves are perfectly absorbed. Waves incident at other angles are only partially absorbed. This boundary condition is simpler than a perfectly matched layer (PML) [39] boundary condition but does not perform quite as well, especially for surface waves. If the waves arriving at the absorbing boundary are relatively small in amplitude compared to the amplitudes of primary interest, this boundary condition gives reasonable results.

### 3.4 Time step

The displacement update in Eq. (18) is evidently explicit in nature, and the CFL condition [40] controls the maximum allowable time step. This condition states that the critical time step is the time it takes for the P wave to travel across the shortest dimension of a cell. In most cases this is the shortest edge length. However, distorted cells which have relatively small areas in 2-D or relatively small volumes in 3-D for the given edge lengths also require small time steps due to the artificially high stiffness associated with the distorted shape. As a result, we set the stable time step to be the smaller of the shortest edge length and a scaling factor times the radius of an inscribed circle in 2-D, and sphere in 3-D.

### 3.5 Time marching algorithm

As shown in Algorithm 1, the time marching requires inversion of both the mass matrix and damping matrix. Given that the maximum allowable time step is limited by the CFL condition, the marching would require substantially more time steps compared to the scenario without any time step size constraint. The

**Algorithm 1:** Time-marching in poroelastodynamics

$n \leftarrow 0; t \leftarrow 0;$
$\boldsymbol{d}_h^0 \leftarrow \boldsymbol{0};$ // Initialize fault slip at virgin state
$\boldsymbol{u}_h^0 \leftarrow \boldsymbol{u}_{0_h}; \boldsymbol{u}_h^{-1} \leftarrow \boldsymbol{u}_h^0 - \dot{\boldsymbol{u}}_{0_h} 2\Delta t;$ // Initial condition
$\boldsymbol{K} \leftarrow \boldsymbol{K}_d; \boldsymbol{C} \leftarrow \boldsymbol{C}_d;$ // Diagonalize mass and damping matrix
**while** $t < T$ **do**
  /* time marching till final time */
  **while** *Not converged* **do**
    /* Staggered solution algorithm loop */
    Solve flow problem for pressures;
    $\boldsymbol{d}_h^{n+1} \leftarrow \boldsymbol{d}_h^n;$ // Initialize fault slip for next time step
    $\boldsymbol{R}'_{u,a} \leftarrow \boldsymbol{R}_{u,a}|_{d\boldsymbol{u}_h^n=\boldsymbol{0}}; \boldsymbol{R}'_{l,\bar{a}} \leftarrow \boldsymbol{R}_{l,\bar{a}}|_{d\boldsymbol{u}_h^n=\boldsymbol{0}};$ // Get residuals corresponding to zero increment in displacement
    $d\boldsymbol{U}^* \leftarrow \boldsymbol{K_d}^{-1} \boldsymbol{R}'_{u,a};$ // Initial guess
    $d\boldsymbol{L} \leftarrow (\boldsymbol{C}_d(\boldsymbol{K}_{++}^{-1} + \boldsymbol{K}_{--}^{-1})\boldsymbol{C}_d^T)^{-1}(-\boldsymbol{R}'_{l,\bar{a}} + \boldsymbol{C}_d(d\boldsymbol{U}_+^* + d\boldsymbol{U}_-^*));$
    $\boldsymbol{L} \leftarrow \boldsymbol{L} + d\boldsymbol{L};$ // Update lagrange multipliers
    $d\boldsymbol{U} \leftarrow d\boldsymbol{U}^* - \boldsymbol{K}_d^{-1}\boldsymbol{C}_d^T d\boldsymbol{L};$ // Corrected increment
    $\boldsymbol{U} \leftarrow \boldsymbol{U} + d\boldsymbol{U};$ // Update displacements
    **for** *Loop over lagrange nodes* **do**
      $\tau \leftarrow |\boldsymbol{l}_h^{n+1} - (\boldsymbol{l}_h^{n+1} \cdot \boldsymbol{n})\boldsymbol{n}|;$ // Obtain shear stress on fault
      $\tau_f \leftarrow \tau_f|_{\boldsymbol{d}_h^n};$ // Compute fault friction based on the slip value at previous time step
      **while** $\tau > \tau_f$ **do**
        /* Satisfy fault constitutive law. If the Lagrange multipliers do not exceed the fault tractions allowed by the fault constitutive model, the increment in fault slip remains zero, and no adjustments to the solution are necessary */
        $\boldsymbol{U}_+ \leftarrow \boldsymbol{U}_+ - \boldsymbol{K}_{++}^{-1} \frac{\tau - \tau_f}{\tau} \boldsymbol{L};$
        $\boldsymbol{U}_- \leftarrow \boldsymbol{U}_- + \boldsymbol{K}_{--}^{-1} \frac{\tau - \tau_f}{\tau} \boldsymbol{L};$
        $\boldsymbol{d}_h^{n+1} \leftarrow \boldsymbol{u}_{h_+}^{n+1} - \boldsymbol{u}_{h_-}^{n+1};$ // Update fault slip
        $\tau_f \leftarrow \tau_f|_{\boldsymbol{d}_h^{n+1}};$ // Update fault friction
      $\boldsymbol{d}_h^{n+1} \leftarrow \boldsymbol{u}_{h_+}^{n+1} - \boldsymbol{u}_{h_-}^{n+1};$ // Update fault slip
  $n \leftarrow n + 1;$
  $t \leftarrow t + \Delta t;$

requirement to reduce the floating point operations towards inversion of matrices leads us to employ the lumping strategy [26, 27] to diagonalize the matrices. The lumping is performed on each submatrix of the matrix given in Eq. (22). At the virgin state, the displacement fields are initialized based on the initial conditions for the problem. At each time step, the residuals (Eqs. (18) and (19)) are evaluated by assuming the temporal increment in displacement is zero. These residuals are then used to compute increments in displacement and lagrange multipliers. At each time step, we first assume that the increment in the fault slip is zero, so that the Lagrange multipliers correspond to the fault tractions required to lock the fault. If the Lagrange multipliers exceed the fault tractions allowed by the fault constitutive model, we iterate to find the increment in slip that yields Lagrange multipliers that satisfy the fault constitutive model. On the other hand, if the Lagrange multipliers do not exceed the fault tractions allowed by the fault constitutive model, the increment in fault slip remains zero, and no adjustments to the solution are necessary. The process is iterative as the updated fault slip causes a change in the frictional stress, and the process is repeated with the new value of frictional stress.

## 4 Computational framework for poroelastostatics

### 4.1 Fully discrete system of equations for poroelastostatics

In lieu of Eqs. (7), (11) and (5), the problem statement is to find $(\boldsymbol{u}, \boldsymbol{l})$ belonging to appropriate functional spaces satisfying the essential boundary conditions ($\boldsymbol{u} = \overline{\boldsymbol{u}}$ on $\Gamma_D$) such that

$$\int_\Omega \nabla\boldsymbol{\eta} : (\boldsymbol{\sigma}' - bp\mathbf{I})\, d\Omega - \int_\Omega \boldsymbol{\eta}\cdot\rho_b\boldsymbol{g}\, d\Omega - \int_{\Gamma_N} \boldsymbol{\eta}\cdot\bar{\boldsymbol{t}}\, d\Gamma$$
$$+ \int_{\Gamma_{f_+}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_f\boldsymbol{n})\, d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}\cdot(\boldsymbol{l} - bp_f\boldsymbol{n})\, d\Gamma = 0, \tag{23}$$

$$\int_{\Gamma_{f_+}} \boldsymbol{\eta}\cdot\boldsymbol{u}_+\, d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}\cdot\boldsymbol{u}_-\, d\Gamma - \int_{\Gamma_f} \boldsymbol{\eta}\cdot\boldsymbol{d}\, d\Gamma = 0 \tag{24}$$

for all test functions $\boldsymbol{\eta}$ ($n_{\text{dim}} \times 1$ vector) belonging to the appropriate functional space satisfying $\boldsymbol{\eta} = \mathbf{0}$ on $\Gamma_u$. The displacement, the Lagrange multiplier, and the slip fields are approximated as follows:

$$\boldsymbol{u} \approx \boldsymbol{u}_h = \sum_{b=1}^{n_{\text{node}}} \eta_b \boldsymbol{U}_b, \quad \boldsymbol{l} \approx \boldsymbol{l}_h = \sum_{b=1}^{n_{f,\text{node}}} \eta_b \boldsymbol{L}_b, \quad \boldsymbol{d} \approx \boldsymbol{d}_h = \sum_{b=1}^{n_{f,\text{node}}} \eta_b \boldsymbol{D}_b,$$

where $n_{\text{node}}$ is the total number of nodes and $n_{f,\text{node}}$ is the number of Lagrange nodes. After substitution of the finite element approximations into the weak form of the problem (Eqs. (23)–(24)), we obtain the fully discrete equations in residual form for all nodes $a$ and lagrange nodes $\bar{a}$:

$$\boldsymbol{R}^{Stat}_{u,a} = \int_\Omega \boldsymbol{B}_a^T(\boldsymbol{\sigma}'^{n+1}_h - bp^{n+1}_h\mathbf{1})d\Omega - \int_\Omega \boldsymbol{\eta}_a^T\rho^{n+1}_{b,h}\boldsymbol{g}d\Omega - \int_{\Gamma_N}\boldsymbol{\eta}_a^T\bar{\boldsymbol{t}}d\Gamma$$
$$+ \int_{\Gamma_{f_+}} \boldsymbol{\eta}_a^T(\boldsymbol{l}^{n+1}_h - bp^{n+1}_{f,h}\boldsymbol{n})d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}_a^T(\boldsymbol{l}^{n+1}_h - bp^{n+1}_{f,h}\boldsymbol{n})d\Gamma = \mathbf{0} \tag{25}$$

$$\boldsymbol{R}^{Stat}_{l,\bar{a}} = \int_{\Gamma_{f_+}} \boldsymbol{\eta}^T_{\bar{a}}\boldsymbol{u}^{n+1}_{h_+}d\Gamma - \int_{\Gamma_{f_-}} \boldsymbol{\eta}^T_{\bar{a}}\boldsymbol{u}^{n+1}_{h_-}\, d\Gamma - \int_{\Gamma_f}\boldsymbol{\eta}^T_{\bar{a}}\boldsymbol{d}^{n+1}_h d\Gamma = \mathbf{0} \tag{26}$$

We find the system Jacobian matrix by isolating the term for the increments in displacements and Lagrange multipliers at time step $n+1$. The system of linear equations is:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{C}^T \\ \boldsymbol{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} d\boldsymbol{U} \\ d\boldsymbol{L} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{R}^{Stat}_{u,a} \\ \boldsymbol{R}^{Stat}_{l,\bar{a}} \end{bmatrix} \tag{27}$$

where the block matrices are obtained via element-by-element assembly of the individual nodal contributions to the Jacobian:

$$\boldsymbol{K} = \int_\Omega \boldsymbol{B}_a^T\boldsymbol{D}\boldsymbol{B}_b\, d\Omega, \qquad \boldsymbol{C} = \int_{\Gamma_f}\boldsymbol{\eta}_a^T\boldsymbol{\eta}_{\bar{b}}\, d\Gamma \tag{28}$$

To visualize the fault contribution to the linear system, we can write Eq. (27) as

$$\left[\begin{array}{ccc:c} \boldsymbol{K}_{rr} & \boldsymbol{K}_{r+} & \boldsymbol{K}_{r-} & \mathbf{0} \\ \boldsymbol{K}_{+r} & \boldsymbol{K}_{++} & \mathbf{0} & \boldsymbol{C}_+^T \\ \boldsymbol{K}_{-r} & \mathbf{0} & \boldsymbol{K}_{--} & -\boldsymbol{C}_-^T \\ \hdashline \mathbf{0} & \boldsymbol{C}_+ & -\boldsymbol{C}_- & \mathbf{0} \end{array}\right] \left[\begin{array}{c} d\boldsymbol{U}_r \\ d\boldsymbol{U}_+ \\ d\boldsymbol{U}_- \\ \hline d\boldsymbol{L} \end{array}\right] = - \left[\begin{array}{c} \boldsymbol{R}^{Stat}_{u,r} \\ \boldsymbol{R}^{Stat}_{u,+} \\ \boldsymbol{R}^{Stat}_{u,-} \\ \hline \boldsymbol{R}^{Stat}_{l} \end{array}\right] \tag{29}$$

where the top row corresponds to displacement nodes excluding the fault positive and negative side nodes.

**Algorithm 2:** Time-marching in poroelastostatics

$n \leftarrow 0$; $t \leftarrow 0$;
$\boldsymbol{d}_h^0 \leftarrow \boldsymbol{0}$; // Initialize fault slip at virgin state
$\boldsymbol{u}_h^0 \leftarrow \boldsymbol{u}_h^{Prestep}$; // Initial condition based on a elastic prestep solve
**while** $t < T$ **do**
  /* time marching till final time */
  **while** *Not converged* **do**
    /* Staggered solution algorithm loop */
    Solve flow problem for pressures;
    $\boldsymbol{d}_h^{n+1} \leftarrow \boldsymbol{d}_h^n$; // Initialize fault slip for next time step
    Solve system of equations using GMRES; // Krylov subspace solver
    $\boldsymbol{L} \leftarrow \boldsymbol{L} + d\boldsymbol{L}$; // Update lagrange multipliers
    $\boldsymbol{U} \leftarrow \boldsymbol{U} + d\boldsymbol{U}$; // Update displacements
    **for** *Loop over lagrange nodes* **do**
      $\tau \leftarrow |\boldsymbol{l}_h^{n+1} - (\boldsymbol{l}_h^{n+1} \cdot \boldsymbol{n})\boldsymbol{n}|$; // Obtain shear stress on fault
      $\tau_f \leftarrow \tau_f|_{\boldsymbol{d}_h^n}$; // Compute fault friction based on the slip value at previous time step
      **while** $\tau > \tau_f$ **do**
        /* Satisfy fault constitutive law. If the Lagrange multipliers do not exceed the fault tractions allowed by the fault constitutive model, the increment in fault slip remains zero, and no adjustments to the solution are necessary */
        $\boldsymbol{U}_+ \leftarrow \boldsymbol{U}_+ - \boldsymbol{K}_{++}^{-1} \frac{\tau - \tau_f}{\tau} \boldsymbol{L}$;
        $\boldsymbol{U}_- \leftarrow \boldsymbol{U}_- + \boldsymbol{K}_{--}^{-1} \frac{\tau - \tau_f}{\tau} \boldsymbol{L}$;
        $\boldsymbol{d}_h^{n+1} \leftarrow \boldsymbol{u}_{h_+}^{n+1} - \boldsymbol{u}_{h_-}^{n+1}$; // Update fault slip
        $\tau_f \leftarrow \tau_f|_{\boldsymbol{d}_h^{n+1}}$; // Update fault friction
      $\boldsymbol{d}_h^{n+1} \leftarrow \boldsymbol{u}_{h_+}^{n+1} - \boldsymbol{u}_{h_-}^{n+1}$; // Update fault slip
  $n \leftarrow n + 1$;
  $t \leftarrow t + \Delta t$;

### 4.2 Time marching algorithm

In many quasi-static simulations it is convenient to compute a static problem with elastic deformation prior to computing a transient response. Unlike the poroelastodynamics case, the heavy lifting for the time marching in poroelastostatics is done at the Krylov solver [41] stage to solve the system of Eqs. (27) as shown in Algorithm 2. From an algorithmic standpoint, it is more straightforward as PETSc [42] has a host of options in terms of solvers and preconditioners based on the specifics of the problem at hand, although generalized minimal residual method (GMRES) [43] is the most popular solver for the saddle point system of equations arising out of Eq. (27). Since the heavy lifting is done by the solver, a deeper understanding of Krylov subspace methods along with preconditioners is critical to optimize compute performance.

## 5 Conclusions and Outlook

We summarize the differences in the numerics of poroelastodynamics and poroelastostatics as follows

- ✓ Time marching in poroelastostatics is 'implicit' as the solution at a time step is dependent on evolution of the state variables at the same time step, as can be clearly seen in Eq. (25). On the other hand, time marching in poroelastodynamics is 'explicit' as the solution at the next time step is a function of evolution of state variables at the current time step, as is evident in Eq. (18)

- ✓ Explicit time stepping in poroelastodynamics case brings in time step size constraints in accordance with the CFL criterion, thereby restricting the ability to march forward quickly. On the other hand, the implicit time stepping in poroelastostatics brings in no such time step restriction, and we can gallop in time subject to the requirements of solution accuracy only
- ✓ In addition to time step restrictions, the wave nature of the poroelastodynamics physics brings in issues associated with short range high frequency oscillations polluting the solution and waves bouncing off the domain boundaries. The first of these issues is treated by adding numerical damping to the solution and the second issue is dealt with by imposing either absorbing boundary layers or perfectly matched boundary layers on the domain boundaries
- ✓ The heavy lifting in the poroelastostatics case is done by the iterative Krylov subspace solver, and careful attention needs to be given to the choice of solver and preconditioner. On the other hand, a direct solver suffices for poroelastodynamics.
- ✓ Time step restrictions in poroelastodynamics entails a brute force diagonalization of the mass and damping matrices unlike in the poroelastostatics case, where the diagonalization is done internally as a part of the Krylov subspace solution technique

## A Code snippets

```
# TimeDependent class
class TimeDependent(Problem):
  ...
  def run(self, app): #Solve time dependent problem
    from pylith.mpi.Communicator import mpi_comm_world
    comm = mpi_comm_world()
    ...
    stepCount = 0
    ...
    t = self.formulation.getStartTime()
    timeScale = self.normalizer.timeScale()
    while t < self.formulation.getTotalTime():    # Normal time loop
      self._eventLogger.stagePush("Prestep")
      tsec = self.normalizer.dimensionalize(t, timeScale)
      if 0 == comm.rank:
        self._info.log("Main time loop, current time is t=%s" % tsec)
      self.checkpointTimer.update(t)# Checkpoint if necessary
      dt = self.formulation.getTimeStep()# Get time step for advancing in time
      # adjust dt
      if t+dt>self.formulation.getTotalTime():
        dt = self.formulation.getTotalTime()-t
      dtsec = self.normalizer.dimensionalize(dt, timeScale)
      if 0 == comm.rank:
        self._info.log("Preparing to advance solution from time t=%s to t=%s with
    timescale=%s." % (tsec, tsec+dtsec, timeScale))
      if stepCount == 0:  #prestep only for initialization stepInitDisp, step calls
    prestep from inside
        self.formulation.prestep(t, dt)
      self._eventLogger.stagePop()
      if 0 == comm.rank:
        self._info.log("Advancing solution from t=%s to t=%s." % (tsec, tsec+dtsec))
      self._eventLogger.stagePush("Step")
      if stepCount == 0: # initialize disp field with IC BC for first call, then
    switch to coupled simulation
        self.formulation.stepInitDisp(t, dt, self.normalizer)
      elif stepCount > 0:
        self.formulation.step(t, dt, self.normalizer, stepCount)
      self._eventLogger.stagePop()
      if 0 == comm.rank:
        self._info.log("Finishing advancing solution from t=%s to t=%s." % (tsec, tsec
    +dtsec))
```

```
      self._eventLogger.stagePush("Poststep")
      self.formulation.poststep(t, dt)
      self._eventLogger.stagePop()
      # Update time
      t += dt
      stepCount += 1
    return
```

Listing 1: Python driver code for time marching

```
// Loop over geodynamics cells
for (SieveMesh::label_sequence::iterator c_iter=cellsBegin;c_iter!= cellsEnd;++
  c_iter) {
    ...
    const int pylithIndex = std::distance(cellsBegin, c_iter);//element #
    const scalar_array& BiotCoeff = _material->calcBiotCoeff();//biot coefficient
    double pressure = _material->getCellPressure(t, pylithIndex);
    pressure = pressure / (_normalizer->pressureScale()); //Pa to dimensionless
    //loop over quadrature points
    for (int iQuad = 0; iQuad < numQuadPts; ++iQuad) {
    //stressTot is total stress, stressEff is effective stress
        stressTot[iQuad * tensorSize + 0] = stressEff[iQuad * tensorSize + 0] -
  BiotCoeff[iQuad]*pressure;
        stressTot[iQuad * tensorSize + 1] = stressEff[iQuad * tensorSize + 1] -
  BiotCoeff[iQuad]*pressure;
        stressTot[iQuad * tensorSize + 2] = stressEff[iQuad * tensorSize + 2] -
  BiotCoeff[iQuad]*pressure;
        stressTot[iQuad * tensorSize + 3] = stressEff[iQuad * tensorSize + 3];
        stressTot[iQuad * tensorSize + 4] = stressEff[iQuad * tensorSize + 4];
        stressTot[iQuad * tensorSize + 5] = stressEff[iQuad * tensorSize + 5];
      ...
    }
    ...
}
```

Listing 2: Lower level C++ code snippet for computation of stress

## References


[1] Hiroo Kanamori and Emily E Brodsky. The physics of earthquakes. *Reports on Progress in Physics*, 67(8):1429, 2004.

[2] Christopher H Scholz, Yen Joe Tan, and Fabien Albino. The mechanism of tidal triggering of earthquakes at mid-ocean ridges. *Nature communications*, 10(1):1–7, 2019.

[3] Sebastian Hainzl, Toni Kraft, Joachim Wassermann, Heiner Igel, and E Schmedes. Evidence for rainfall-triggered earthquake activity. *Geophysical Research Letters*, 33(19), 2006.

[4] EK Montgomery-Brown, David R Shelly, and Paul A Hsieh. Snowmelt-triggered earthquake swarms at the margin of long valley caldera, california. *Geophysical Research Letters*, 46(7):3698–3705, 2019.

[5] ChiChing Liu, Alan T Linde, and I Selwyn Sacks. Slow earthquakes triggered by typhoons. *Nature*, 459(7248):833–836, 2009.

[6] Gillian R Foulger, Miles P Wilson, Jon G Gluyas, Bruce R Julian, and Richard J Davies. Global review of human-induced earthquakes. *Earth-Science Reviews*, 178:438–514, 2018.

[7] Youssef MA Hashash, Jeffrey J Hook, Birger Schmidt, I John, and Chiang Yao. Seismic design and analysis of underground structures. *Tunnelling and underground space technology*, 16(4):247–293, 2001.

[8] MJ Nigel Priestley, Frieder Seible, and Gian Michele Calvi. *Seismic design and retrofit of bridges*. John Wiley & Sons, 1996.

[9] Jack Moehle. *Seismic design of reinforced concrete buildings*. McGraw-Hill Education, 2015.

[10] Mark D Zoback and Steven M Gorelick. Earthquake triggering and large-scale geologic storage of carbon dioxide. *Proceedings of the National Academy of Sciences*, 109(26):10164–10168, 2012.

[11] William L Ellsworth, Domenico Giardini, John Townend, Shemin Ge, and Toshihiko Shimamoto. Triggering of the pohang, korea, earthquake (m w 5.5) by enhanced geothermal system stimulation. *Seismological Research Letters*, 90(5):1844–1858, 2019.

[12] Katie M Keranen and Matthew Weingarten. Induced seismicity. *Annual Review of Earth and Planetary Sciences*, 46:149–174, 2018.

[13] Birendra Jha and Ruben Juanes. Coupled multiphase flow and poromechanics: A computational model of pore pressure effects on fault slip and earthquake triggering. *Water Resources Research*, 50(5):3776–3808, 2014.

[14] Saumik Dana, Xiaoxi Zhao, and Birendra Jha. A two-grid computational framework for fast monitoring of fault stability and ground deformation in multiphase geomechanics. *Under review at Journal of Computational Physics*, 2021.

[15] Saumik Dana, Benjamin Ganis, and Mary F. Wheeler. A multiscale fixed stress split iterative scheme for coupled flow and poromechanics in deep subsurface reservoirs. *Journal of Computational Physics*, 352:1–22, 2018.

[16] Saumik Dana and Mary F Wheeler. Design of convergence criterion for fixed stress split iterative scheme for small strain anisotropic poroelastoplasticity coupled with single phase flow. *arXiv preprint arXiv:1912.06476*, 2019.

[17] Saumik Dana. System of equations and staggered solution algorithm for immiscible two-phase flow coupled with linear poromechanics. *arXiv preprint arXiv:1912.04703*, 2019.

[18] Saumik Dana, Joel Ita, and Mary F Wheeler. The correspondence between voigt and reuss bounds and the decoupling constraint in a two-grid staggered algorithm for consolidation in heterogeneous porous media. *Multiscale Modeling & Simulation*, 18(1):221–239, 2020.

[19] Saumik Dana, Xiaoxi Zhao, and Birendra Jha. Two-grid method on unstructured tetrahedra: Applying computational geometry to staggered solution of coupled flow and mechanics problems. *arXiv preprint arXiv:2102.04455*, 2021.

[20] Saumik Dana and Birendra Jha. A fault slip model to study earthquakes due to pore pressure perturbations. *arXiv preprint arXiv:2104.06257*, 2021.

[21] S. Dana and M. F. Wheeler. Convergence analysis of fixed stress split iterative scheme for anisotropic poroelasticity with tensor biot parameter. *Computational Geosciences*, 22(5):1219–1230, 2018.

[22] S. Dana and M. F. Wheeler. Convergence analysis of two-grid fixed stress split iterative scheme for coupled flow and deformation in heterogeneous poroelastic media. *Computer Methods in Applied Mechanics and Engineering*, 341:788–806, 2018.

[23] S. Dana. *Addressing challenges in modeling of coupled flow and poromechanics in deep subsurface reservoirs*. PhD thesis, The University of Texas at Austin, 2018.

[24] Lei Jin and Mark D Zoback. Fully dynamic spontaneous rupture due to quasi-static pore pressure and poroelastic effects: An implicit nonlinear computational model of fluid-induced seismic events. *Journal of Geophysical Research: Solid Earth*, 123(11):9430–9468, 2018.

[25] J. R. Rice. Spatio-temporal complexity of slip on a fault. *J. Geophys. Res.*, 98:9885–9907, 1993.

[26] B Aagaard, S Kientz, M Knepley, L Strand, and C Williams. Pylith user manual: version 2.1. 0. *Davis, CA: Computational Infrastructure of Geodynamics*, 2013.

[27] Brad T Aagaard, Matthew G Knepley, and Charles A Williams. A domain decomposition approach to implementing fault slip in finite-element models of quasi-static and dynamic crustal deformation. *Journal of Geophysical Research: Solid Earth*, 118(6):3059–3079, 2013.

[28] K Jarrod Millman and Michael Aivazis. Python for scientists and engineers. *Computing in Science & Engineering*, 13(2):9–12, 2011.

[29] Nicolai M Josuttis. *The C++ standard library: a tutorial and reference*. Addison-Wesley, 2012.

[30] Teresa L Cottom. Using swig to bind c++ to python. *Computing in Science & Engineering*, 5(2):88–97, 2003.

[31] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[32] David E Keyes, Lois C McInnes, Carol Woodward, William Gropp, Eric Myra, Michael Pernice, John Bell, Jed Brown, Alain Clo, Jeffrey Connors, et al. Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications*, 27(1):4–83, 2013.

[33] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, and Ahmed Fasih. Pycuda and pyopencl: A scripting-based approach to gpu run-time code generation. *Parallel Computing*, 38(3):157–174, 2012.

[34] Craig M Wittenbrink, Emmett Kilgariff, and Arjun Prabhu. Fermi gf100 gpu architecture. *IEEE Micro*, 31(2):50–59, 2011.

[35] J. R. Rice, N. Lapusta, and K. Ranjith. Rate and state dependent friction and the stability of sliding between elastically deformable solids. *J. Mech. Phys. Solids*, 49:1865–1898, 2001.

[36] Marion Y. Thomas, Nadia Lapusta, Hiroyuki Noda, and Jean-Philippe Avouac. Quasi-dynamic versus fully dynamic simulations of earthquakes and aseismic slip with and without enhanced coseismic weakening. *Journal of Geophysical Research: Solid Earth*, 119:1986–2004, 2014.

[37] J. C. Jaeger and N. G. W. Cook. *Fundamentals of Rock Mechanics*. Chapman and Hall, London, 1979.

[38] Nathan M Newmark. A method of computation for structural dynamics. *Journal of the engineering mechanics division*, 85(3):67–94, 1959.

[39] Ushnish Basu and Anil K. Chopra. Perfectly matched layers for time-harmonic elastodynamics of unbounded domains: theory and finite-element implementation. *Computer Methods in Applied Mechanics and Engineering*, 192(11-12):1337–1375, 2003.

[40] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[41] Henk A Van Der Vorst. Krylov subspace iteration. *Computing in science & engineering*, 2(01):32–37, 2000.

[42] Satish Balay, Shrirang Abhyankar, Mark Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, W Gropp, et al. Petsc users manual. 2019.

[43] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.