Abstraction in PyLith

Matthew Knepley[1], Brad Aagaard[2],
Charles Williams[3], Nicolas Barral[4]

[1]University at Buffalo
[2]United States Geological Survey, Menlo Park
[3]GNS Science, Wellington
[4]Imperial College, London

AGU Fall Meeting: Advances in Computational Geosciences
Washington D.C. December 11, 2018

PyLith. . .

PyLith. . .

simulates crustal deformation
focused on earthquake faulting,
both quasi-static and dynamic.

PyLith. . .

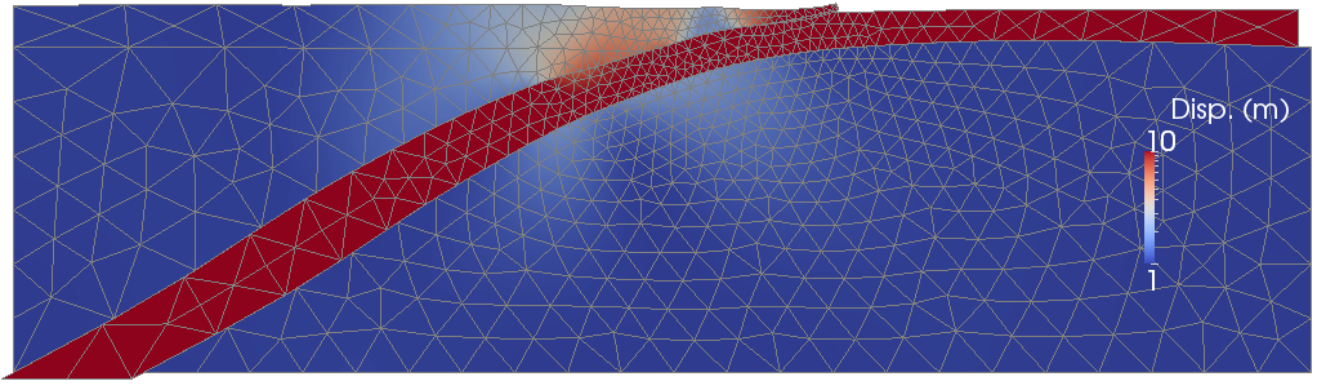
is actively developed since 2004,

with >1000 downloads per release.

PyLith. . .

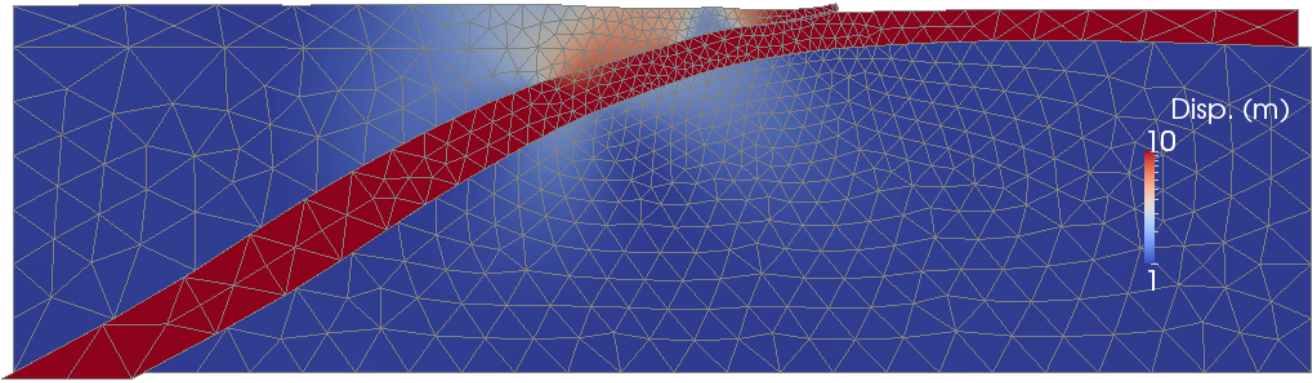
has over 180 citations (Scholar),

and 60 papers based on it (CIG).



PyLith supports. . .

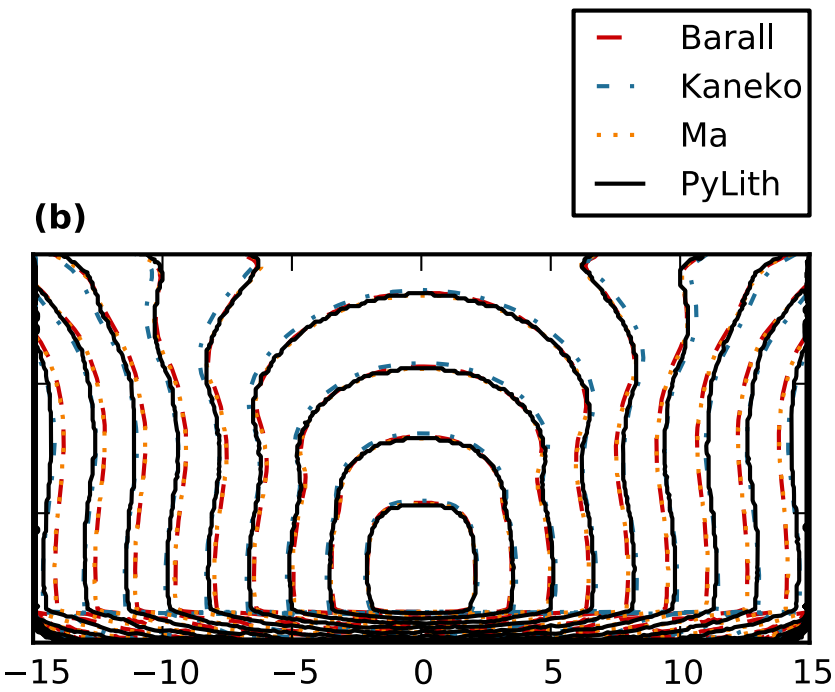
nonlinear fault rheologies,

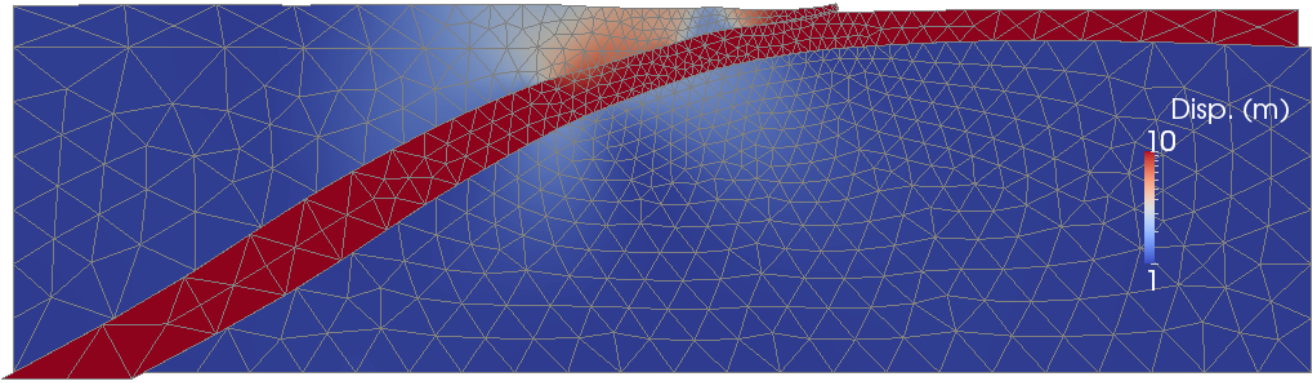


PyLith supports. . .

nonlinear fault rheologies,

flexible rupture definitions,



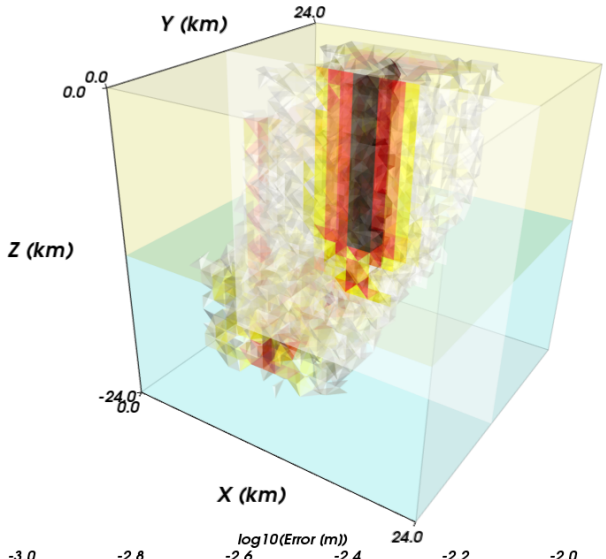


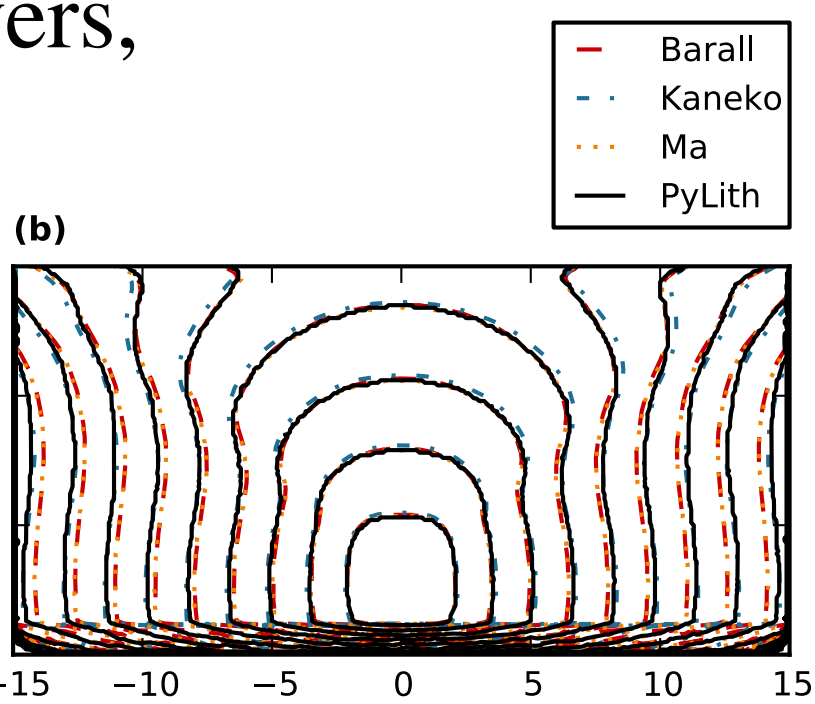
PyLith supports. . .

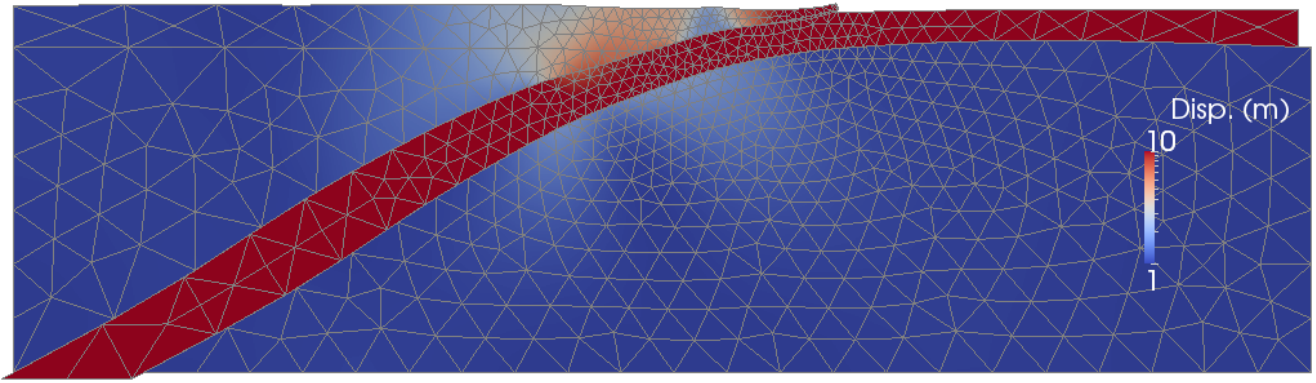
nonlinear fault rheologies,

flexible rupture definitions,

scalable FEM solvers,







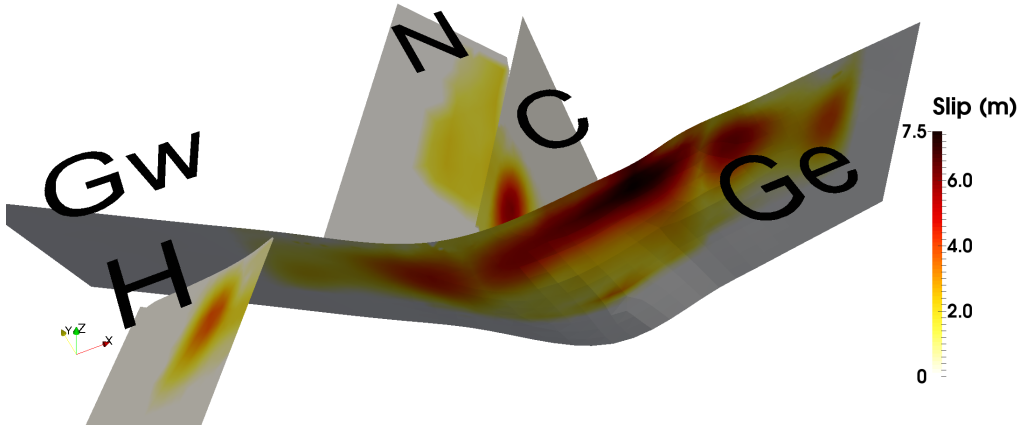
PyLith supports. . .

nonlinear fault rheologies,

flexible rupture definitions,

scalable FEM solvers,

and output over subsurfaces.



PyLith 3 will support. . .

PyLith 3 will support. . .

- higher order elements,

PyLith 3 will support. . .

higher order elements,

multiphysics problems,

PyLith 3 will support. . .

higher order elements,

multiphysics problems,

geometric MG on the full problem,

PyLith 3 will support. . .

higher order elements,

multiphysics problems,

geometric MG on the full problem,

and anisotropic AMR.

We want to enhance functionality,
while limiting complexity.

We want to enhance functionality,
while limiting complexity.

We achieve this goal through flexible,
generalizable interfaces.

Abstractions

- Equations
- Functions
- Adaptation

In Pylith 3, you specify equations pointwise.

$$
\begin{aligned}
F(u) = & \int_\Omega \psi \cdot f_0(u, \nabla u, x) + \\
& \int_\Omega \nabla \psi : \vec{f}_1(u, \nabla u, x) + \\
& \sum_\Gamma \int_\Gamma \psi^{bd} \cdot f_0^{bd}(u, \nabla u, x) + \\
& \sum_\Gamma \int_\Gamma \nabla \psi^{bd} : \vec{f}_1^{bd}(u, \nabla u, x)
\end{aligned}
$$

In Pylith 3, you specify equations pointwise.

Linear Isotropic Elasticity (u; λ, μ)

```
f1_u(...) {
  for (c = 0; c < Nc; ++c) {
    for (d = 0; d < dim; ++d) {
      f1[c*dim+d] += mu*(u_x[c*dim+d] + u_x[d*dim+c]);
      f1[c*dim+c] += lambda*u_x[d*dim+d];
    }
  }
}
```

In Pylith 3, you specify equations pointwise.

Large Deformation Elasticity (u, p; κ, μ)

```
f1_u(...) {
  Cof3D(cofu_x, u_x); Det3D(&detu_x, u_x);
  p = u[uOff[1]] + kappa * (detu_x - 1.0);
  for (c = 0; c < Nc; ++c) {
    for (d = 0; d < dim; ++d) {
      f1[c*dim+d] = mu * u_x[c*dim+d] + p * cofu_x[c*dim+d];
    }
  }}

f0_p(...) {
  Det3D(&detu_x, u_x);
  f0[0] = detu_x - 1.0;
}

f0_bd_u(..., n) {
  const PetscScalar wall_p = a[aOff[1]];
  Cof3D(cofu_x, u_x);
  for (c = 0; c < Nc; ++c) {
    for (d = 0; d < dim; ++d) f0[c] += cofu_x[c*dim+d] * n[d];
    f0[c] *= wall_p;
  }}
```

In Pylith 3, you specify equations pointwise.

Poroelasticity (u, e, p_f; λ, μ, α, κ)

```
f1_u(...) {
  p_f = u[uOff[2]];
  for (c = 0; c < Nc; ++c) {
    for (d = 0; d < dim; ++d) {
      f1[c*dim+d] += mu*(u_x[c*dim+d] + u_x[d*dim+c]);
      f1[c*dim+c] += lambda*u_x[d*dim+d];
    }
    f1[c*dim+c] += alpha*p_f;
  }
}

f0_e(...) {
  for (d = 0; d < dim; ++d) divu += u_x[d*dim+d];
  f0[0] = divu - u[uOff[1]];
}

f1_p(...) {
  for (d = 0; d < dim; ++d) f0[d] -= kappa*u_x[uOff_x[2]+d];
}
```

Abstractions

Equations

Functions

Adaptation

In Pylith 3, finite element solutions and functions are interchangable.

Import data from function f into FEM vector A

```
DMPlexProject(..., f, ..., A)
```

This works in any dimension.

In Pylith 3, finite element solutions and functions are interchangable.

Import data from function f into FEM vector A

PyLith's `spatialdata` can be used as f, and evaluated at any x, independent of the mesh.

In Pylith 3, finite element solutions and functions are interchangable.

Create a boundary condition C from a function f and constitutive model parameters in vector A

```
DMPlexProject(..., f, ..., A, C)
```

`spatialdata` makes boundary/initial condition independent of the mesh.

In Pylith 3, finite element solutions and functions are interchangable.

Create a boundary condition C from a function f and *boundary* model parameters in vector A,

```
DMPlexProject(..., f, ..., A, C)
```

such as time-dependent boundary displacements.

In Pylith 3, finite element solutions and functions are interchangable.

Create a rupture condition F from a function f and *boundary* model parameters in vector A,

```
DMPlexProject(..., f, ..., A, F)
```

such as rupture parameters that sequence events.

In Pylith 3, finite element solutions and functions are interchangable.

Output normal stress on fault S from a function s and *bulk* model solution in vector A

```
DMPlexProject(..., s, ..., A, S)

s(...) {
  for (c = 0; c < Nc; ++c) {
    for (d = 0; d < dim; ++d) {
      stress[c*dim+d] += mu*(u_x[c*dim+d] + u_x[d*dim+c]);
      stress[c*dim+c] += lambda*u_x[d*dim+d];
    }
    for (d = 0; d < dim; ++d) f0[c] += stress[c*dim+d]*n[d];
  }
}
```

In Pylith 3, finite element solutions and functions are interchangable.

Update PyLith state variables SV with function sv, using the old state variables and old solution as auxiliary data A,

```
DMPlexProject(..., sv, ..., A, SV)
```

Abstractions

Equations

Functions

Adaptation

In Pylith 3, you can adapt anisotropically around fault surfaces.

In Pylith 3, you can adapt anisotropically around fault surfaces.

Since we have abstracted

- initial conditions
- boundary conditions
- fault conditions
- discretizations

away from the mesh, we can easily form new problems on adapted meshes.

In Pylith 3, you can adapt anisotropically around fault surfaces.

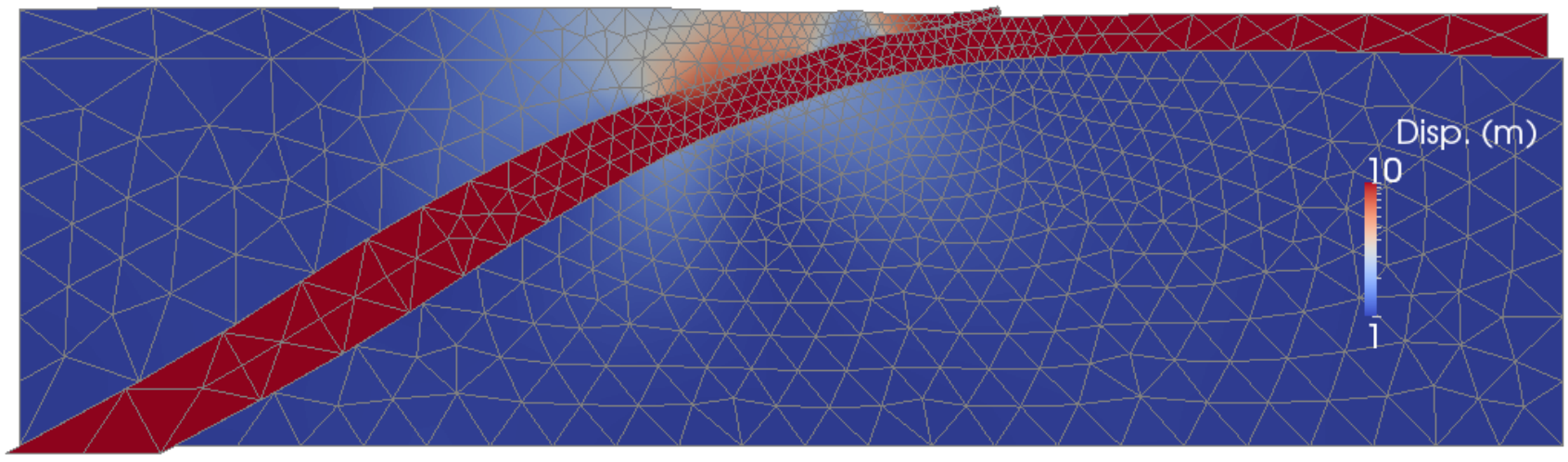
Since we have abstracted

- initial conditions
- boundary conditions
- fault conditions
- discretizations

away from the mesh, we can easily form new problems on adapted meshes.

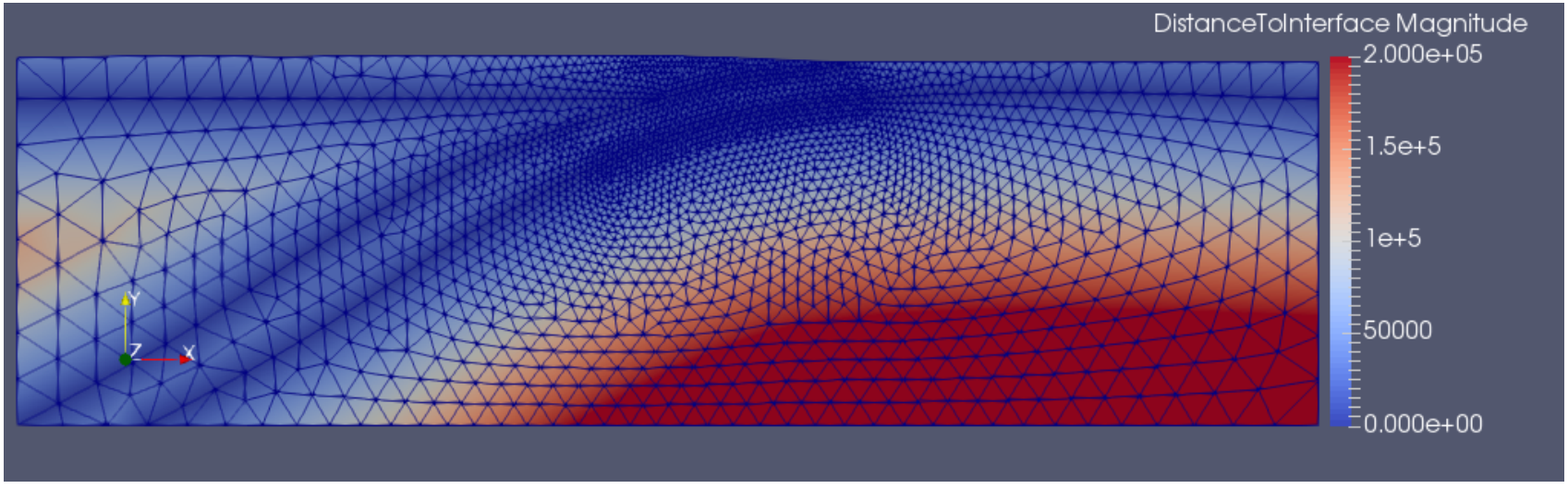
This also enables MG on the full problem.

In Pylith 3, you can adapt anisotropically around fault surfaces.



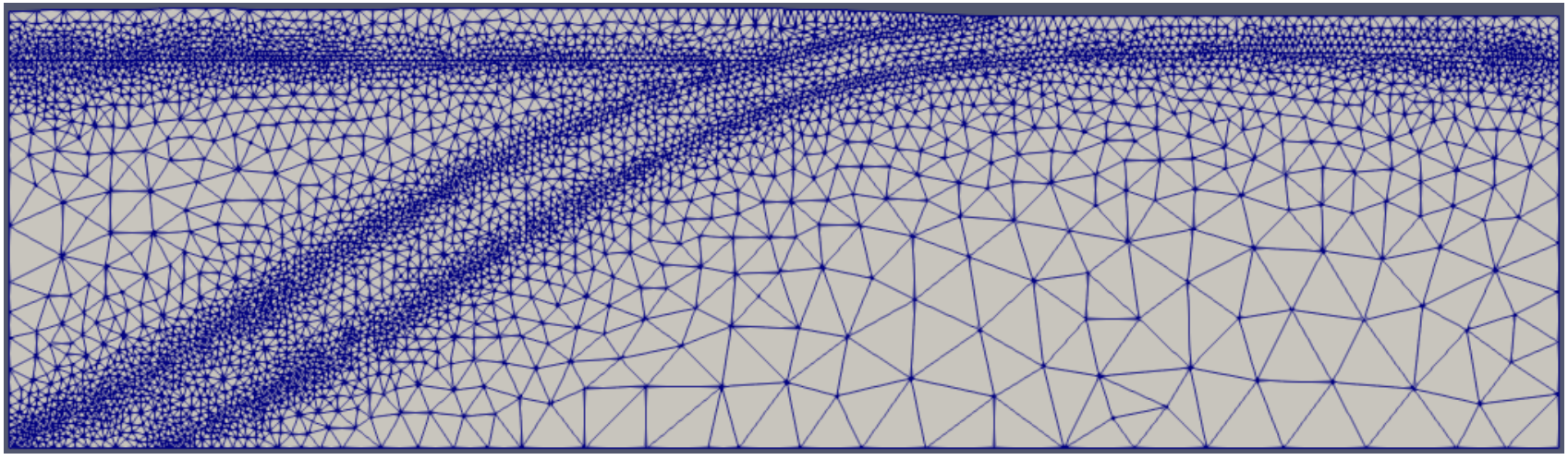
2D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



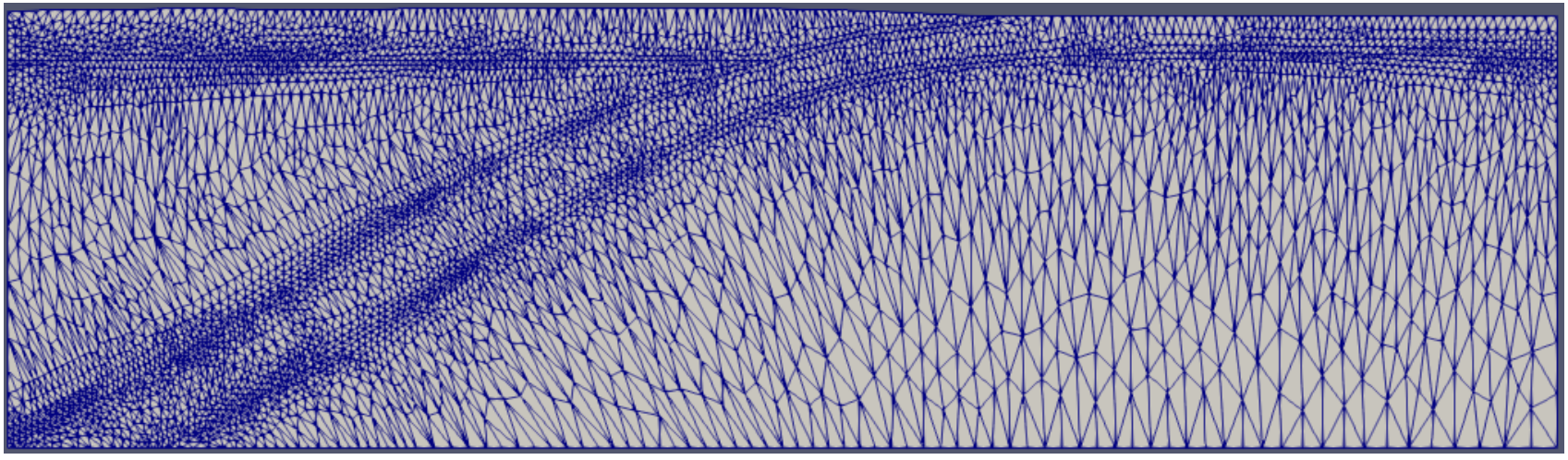
2D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



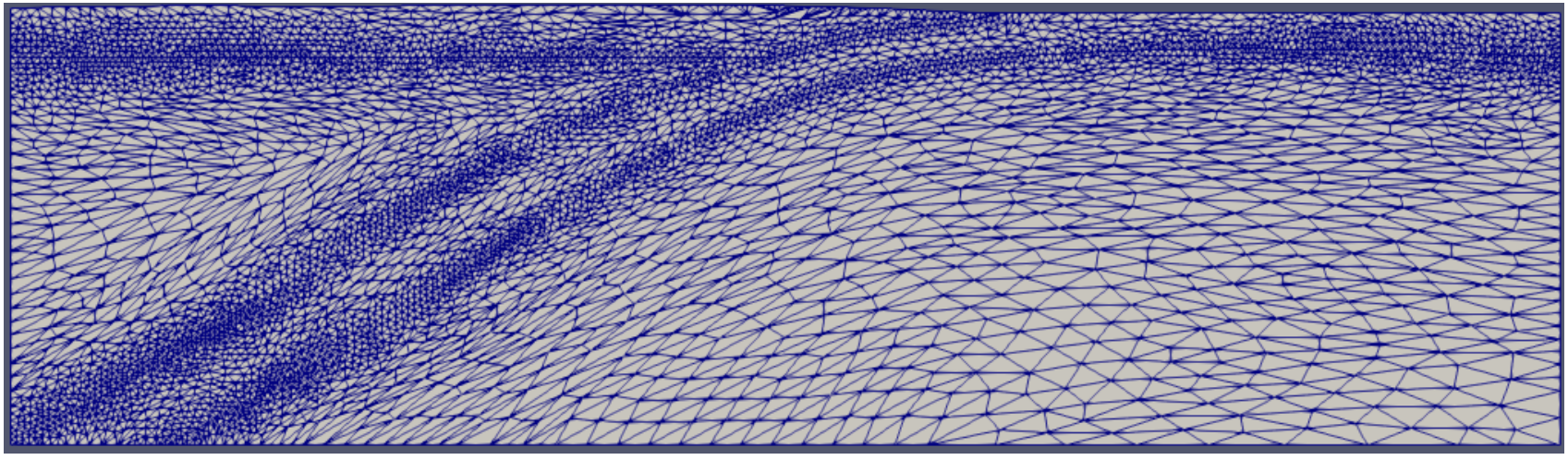
2D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



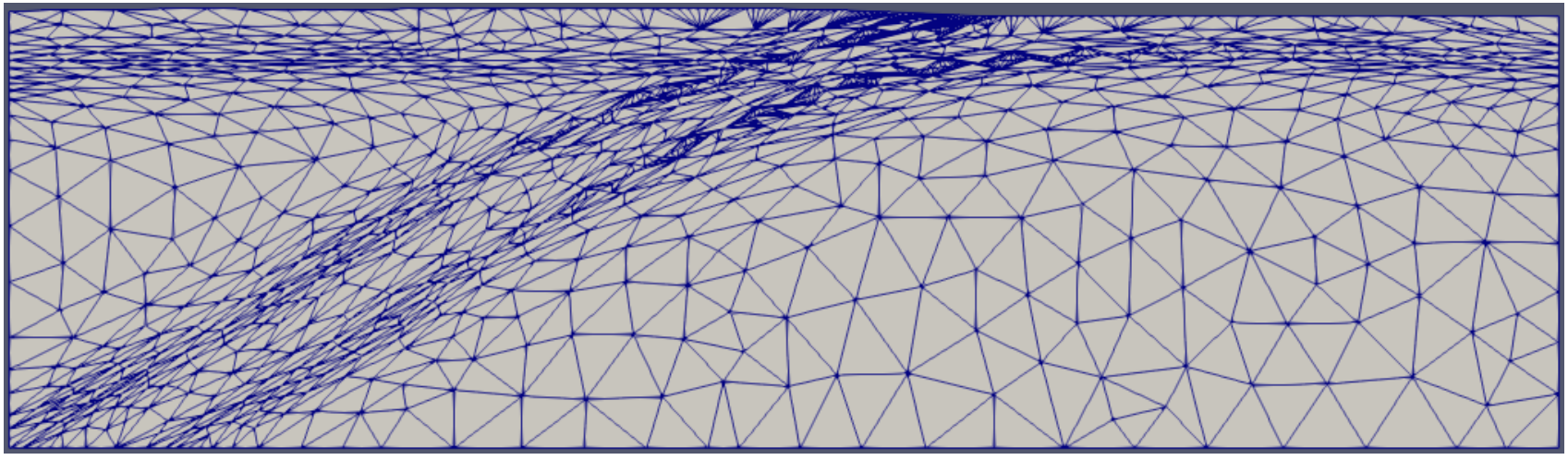
2D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



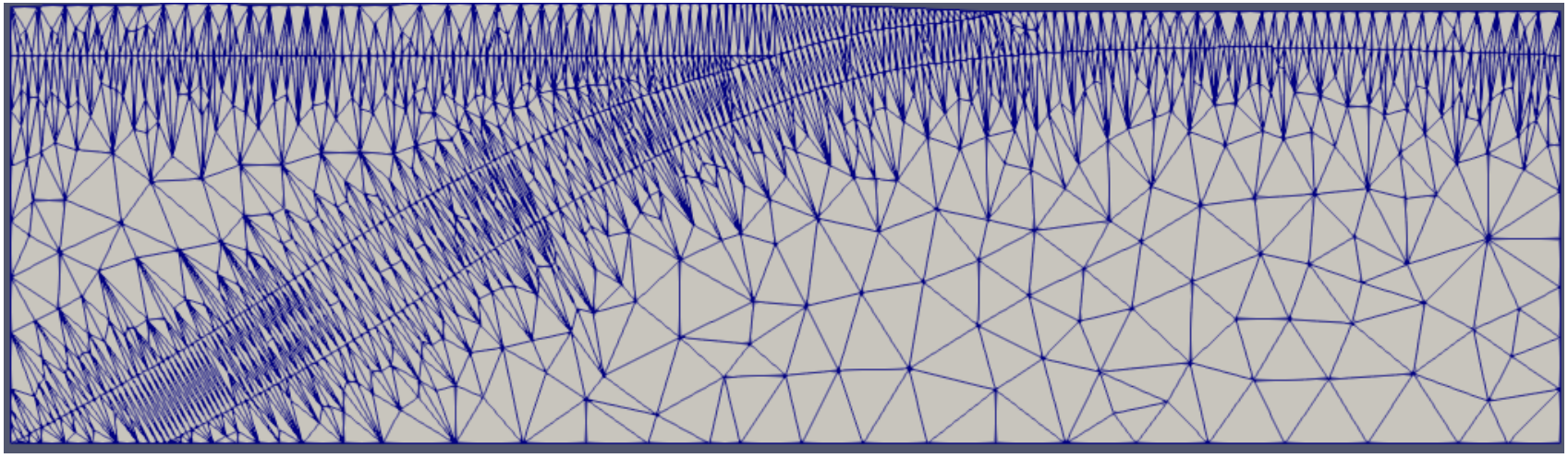
2D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



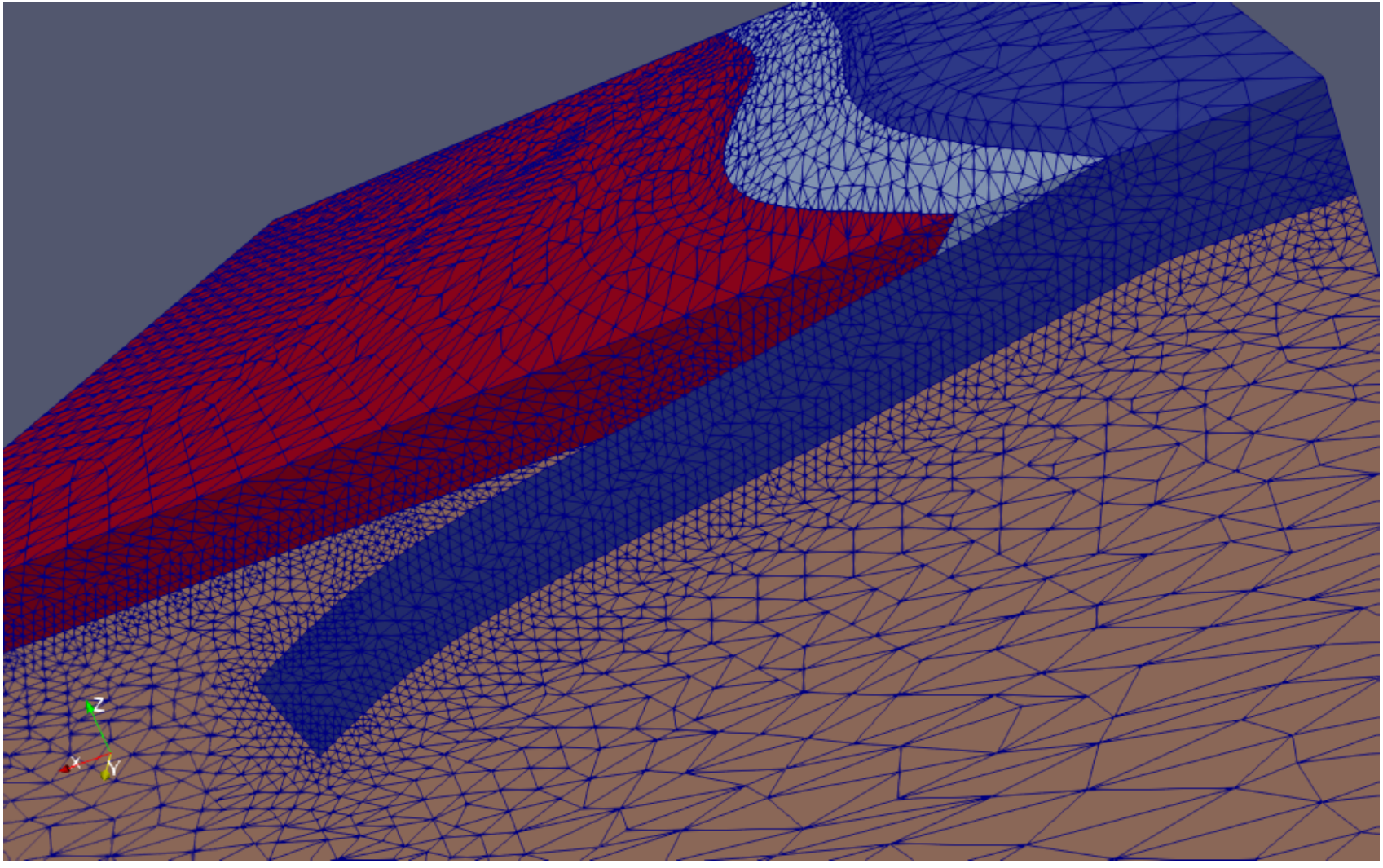
2D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



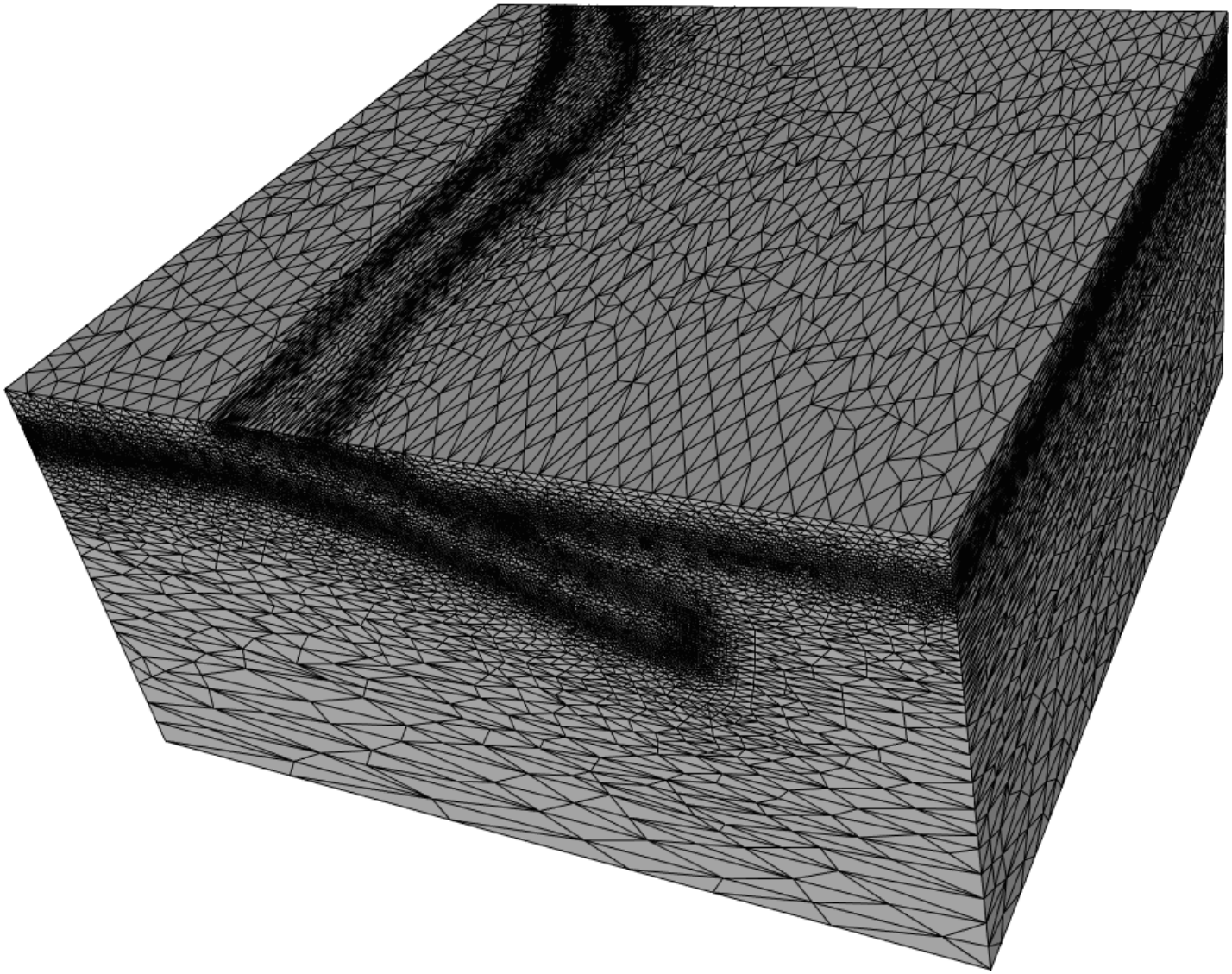
2D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



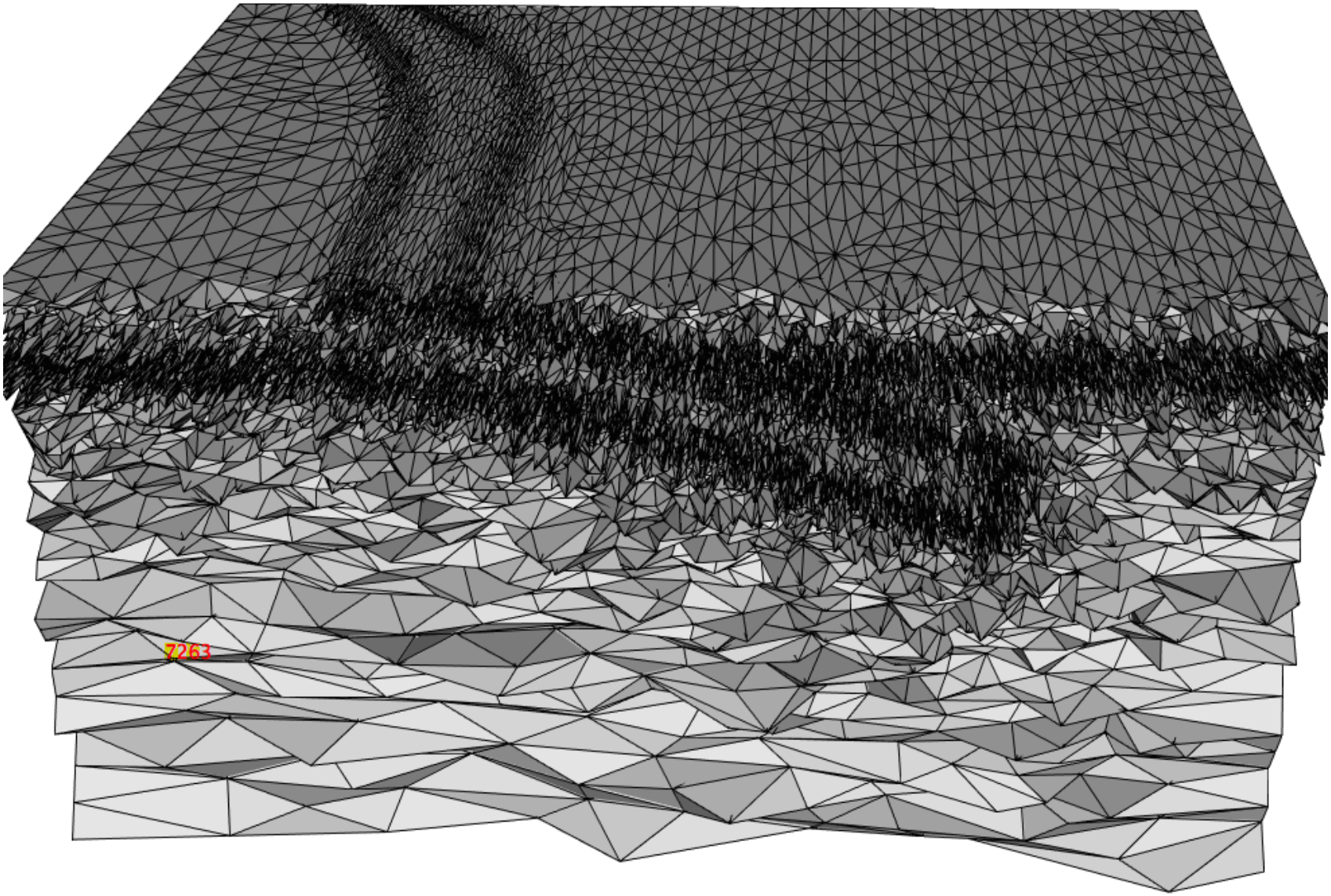
3D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



3D subduction zone

In Pylith 3, you can adapt anisotropically around fault surfaces.



3D subduction zone

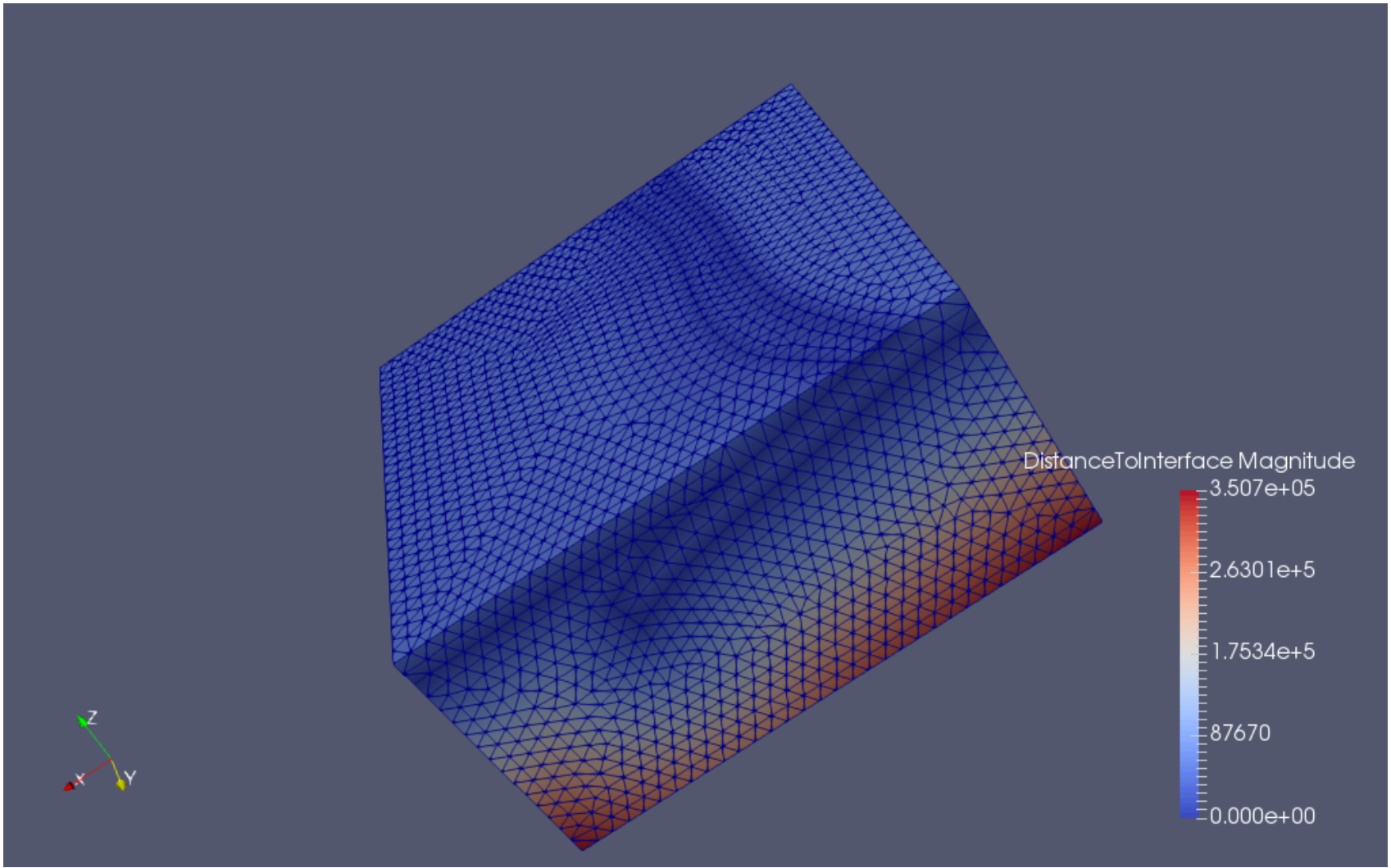
PyLith

https://github.com/geodynamics/pylith/

https://geodynamics.org/cig/software/pylith/

cig-short@geodynamics.org

In Pylith 3, you can adapt anisotropically around fault surfaces.



In Pylith 3, you can adapt anisotropically around fault surfaces.