Simulation of mixed-dimensional multiphysics problems in PorePy

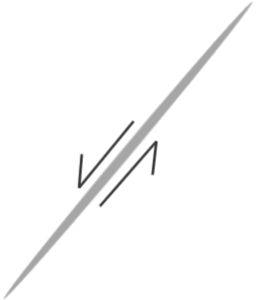
Eirik Keilegavlen

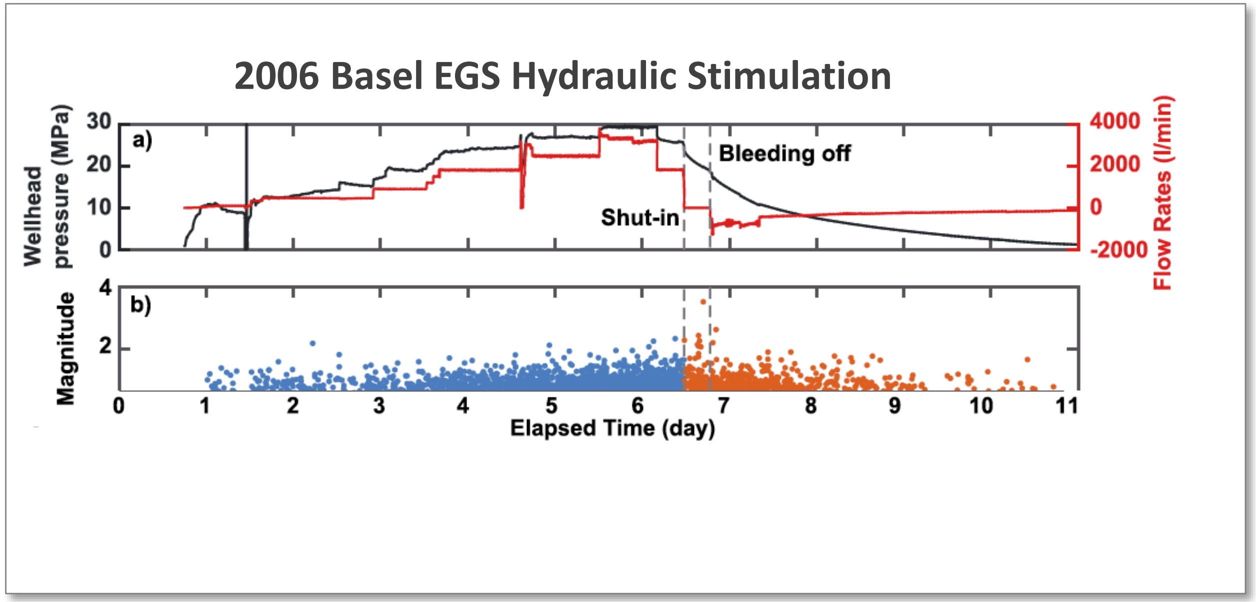
DARTS workshop

March 7, 2023

Motivation: Hydrothermal stimulation of fractures

- In hydraulic stimulation, seismicity is deliberately induced to increase permeability
- Generally, $M_L < 3.0$ (micro earthquake)
- Larger earthquakes must be avoided





Science REPORTS

Cite as: K.-H. Kim et al., Science 10.1126/science.aat6081 (2018).

Assessing whether the 2017 M_w 5.4 Pohang earthquake in South Korea was an induced event

Kwang-Hee Kim,[1]* Jin-Han Ree,[2]* YoungHee Kim,[3] Sungshil Kim,[2] Su Young Kang,[1] Wooseok Seo[1]

Science REPORTS

Cite as: F. Grigoli et al., Science 10.1126/science.aat2010 (2018).

The November 2017 M_w 5.5 Pohang earthquake: A possible case of induced seismicity in South Korea

F. Grigoli,[1]* S. Cesca,[2] A. P. Rinaldi,[1] A. Manconi,[3] J. A. López-Comino,[2] J. F. Clinton,[1] R. Westaway,[4] C. Cauzzi,[1] T. Dahm,[2,5] S. Wiemer[1]

First release: 27 April 2018 www.sciencemag.org (Page numbers not final at time of first release) 1

INDUCED SEISMICITY

Managing injection-induced seismic risks

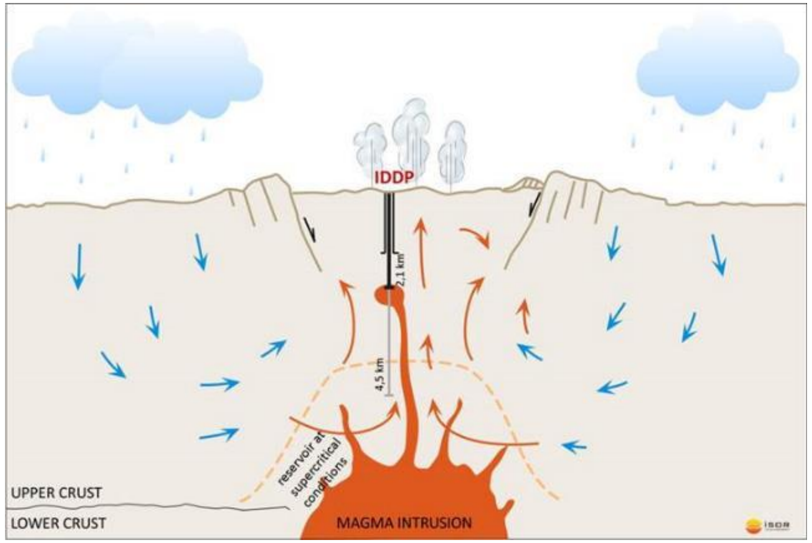
The Pohang quake shows the need for new methods to assess and manage evolving risk

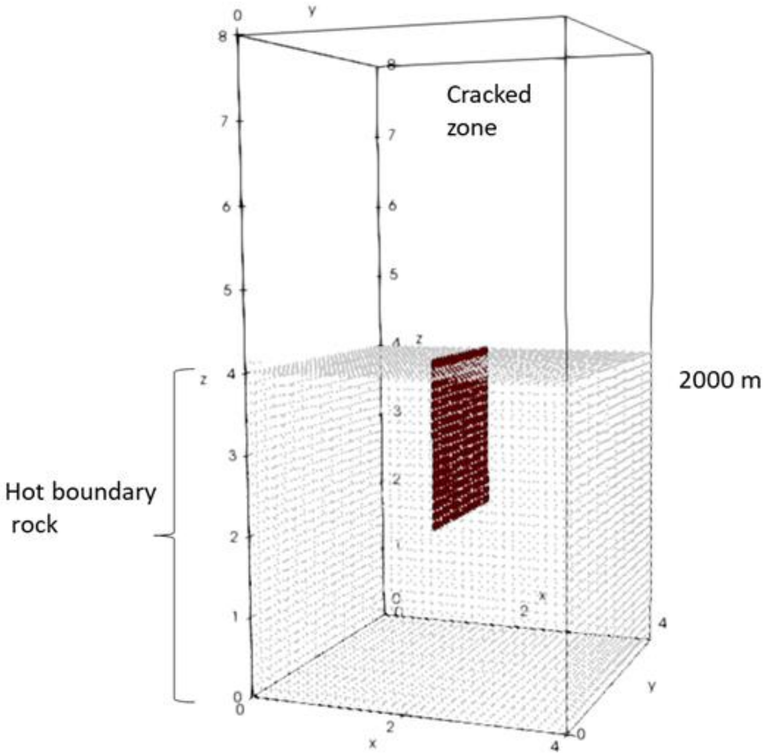
By Kang-Kun Lee, William L. Ellsworth, Domenico Giardini, John Townend, Shemin Ge, Toshihiko Shimamoto, In-Wook Yeo, Tae-Seob Kang, Junkee Rhie, Dong-Hoon Sheen, Chandong Chang, Jeong-Ung Woo, Cornelius Langenbruch

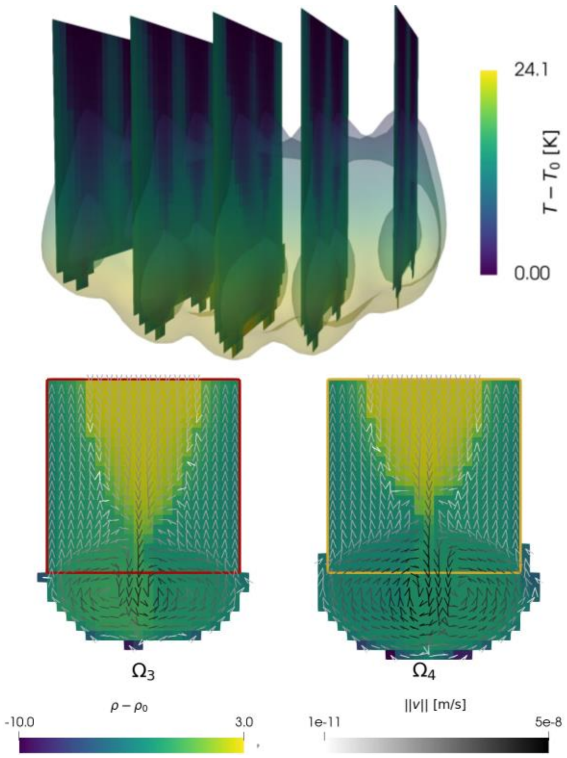
730 24 MAY 2019 • VOL 364 ISSUE 6442 sciencemag.org SCIENCE

Lee, K.-K. Yeo, I.-W., Lee, J.-Y. et al. (2019): Summary Report of the Korean Government Commission on Relations between the 2017 Pohang Earthquake and EGS Project. Korean Government Commission on the Cause of the Pohang Earthquake. http://www.gskorea.or.kr/custom/27/data/Summary_Report_on_Pohang_Earthquake_March_20_2019.pdf

Motivation: Heat transport into geothermal fields







Motivation: Multiphysics processes in fractured porous media

Processes:

- Coupled flow, heat transport, mechanical deformation
- Deformation and propagation of fractures
- (Reactive transport)
- (Multiphase flow)

Applications:

- Geothermal energy from low-permeable rocks
- CO2 storage
- General coupled processes

Development of numerical methods:

- Fracture deformation
- Phase equilibrium
- Spatial discretizations
- Linear solvers

What should PorePy be able to do?

Develop mathematical models and numerical approaches for multiphysics processes in (fractured) porous media and use the methodology for application-relevant simulations.

Target user groups

1. PhD students and researchers
2. Other (commercial?) users

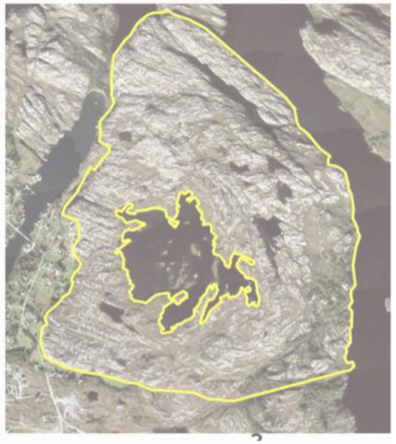
Key assumptions/requirements on users:

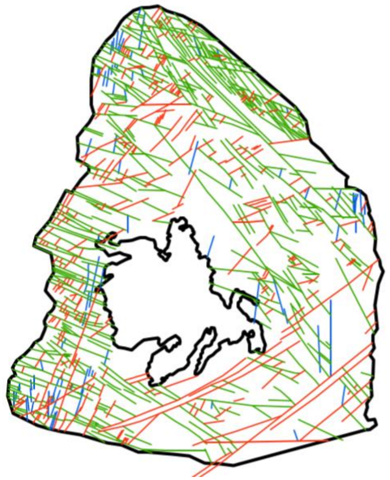
- Literacy in Python coding (need not be experts)
- Ability to use coding to set up simulations (no GUI)
- Ability to think in terms of equations

Building blocks

Meshing, data-structures and mixed-dimensional equations

Challenges in modeling and simulation in fractured porous media





Geometry:

- Individual fractures have high aspect ratios
- Fracture networks have complex geometries
- Fractures may propagate

Processes:

- Non-linear multiphysics couplings
- Heterogenous governing equations
- Parameter heterogeneity

Strong interaction between geometry and processes

Ingredients of mixed-dimensional simulations

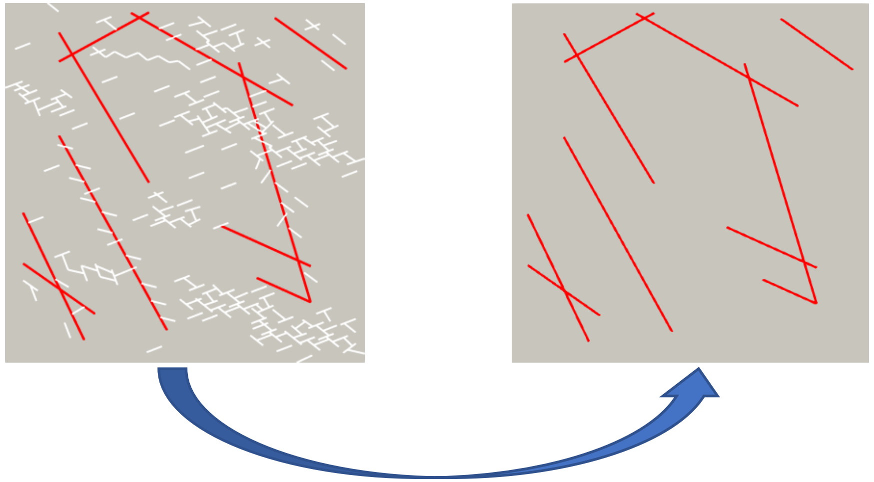
- Domain decomposition: Rock, fractures, and their intersections
- Construction of conformal meshes
- Modeling of physical processes:
    - Governing equations
    - Couplings between and within subdomains
- Discretize and solve

Mixed-dimensional geometry by domain decomposition

Consider fractures and intersections as lower-dimensional objects

Consider a subset of (large) fractures

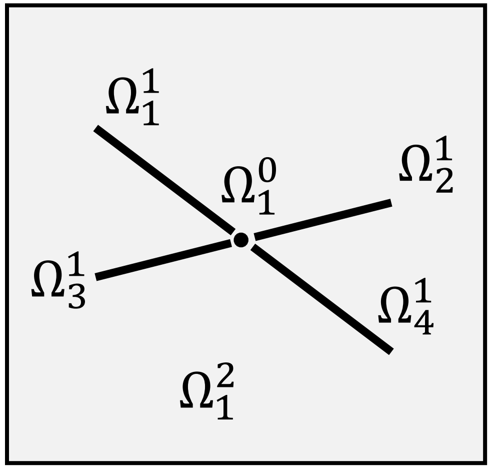
- Upscaled fractures manifest as parameter heterogeneity.



Divide geometry into:

- D-dimensional host medium
- (D-1)-dimensional fractures
- (D-2)- and (D-3)-dimensional intersections

Assign subdomains to each geometric object



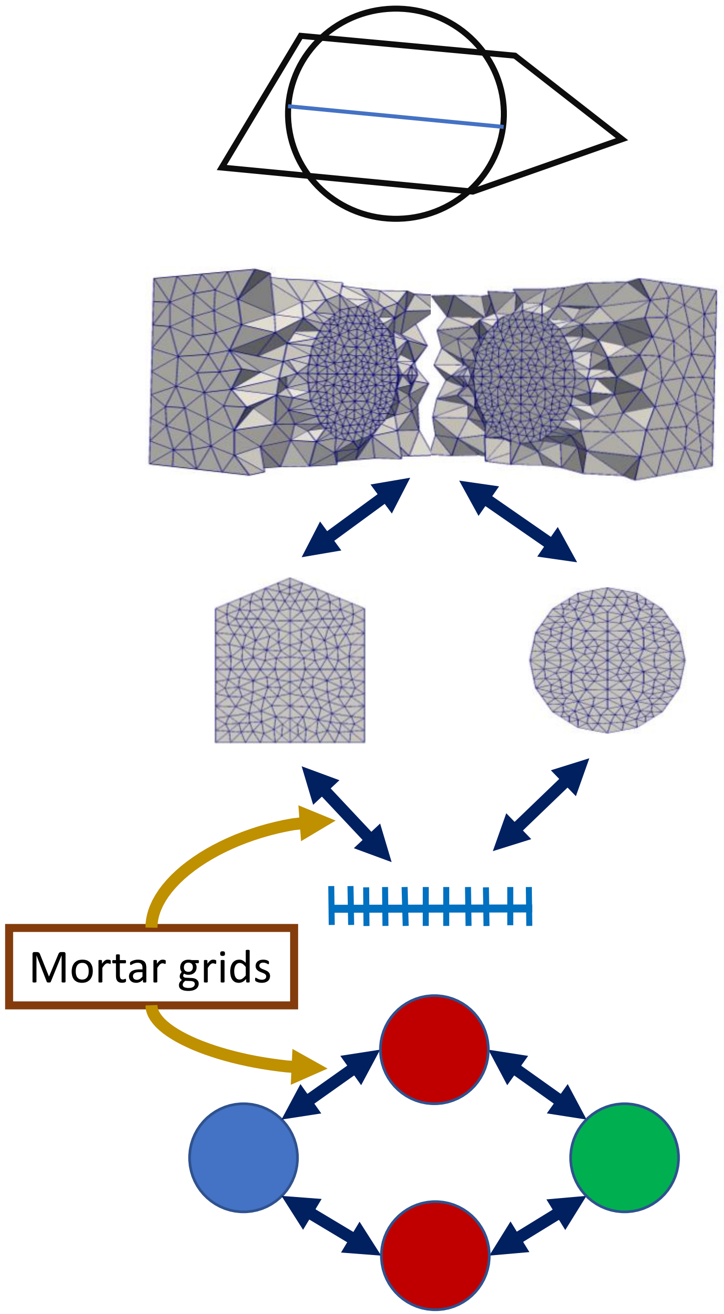
Meshing of mixed-dimensional geometries

Mesh is constrained to geometric objects of all dimensions

Mortar meshes are placed on the interface between subdomains grids; non-matching meshes are permitted

Mixed-dimensional data structure: Graph with subdomains as nodes, mortar meshes on the graph edges

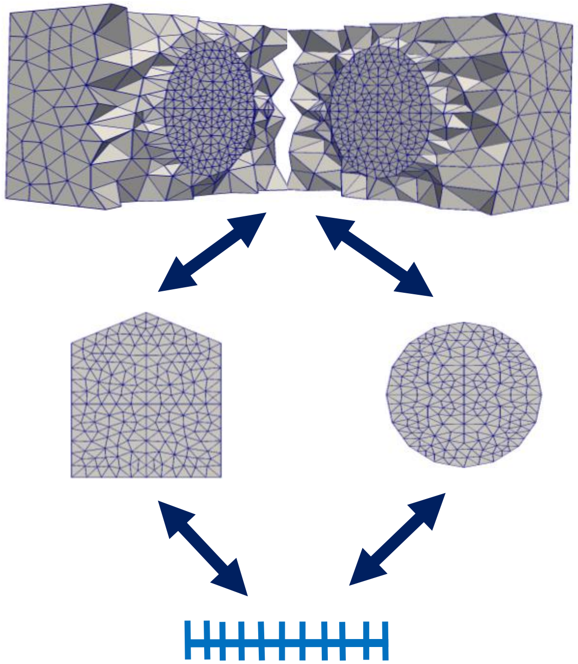
Subdomain mesh resembles that of a standard problem



Coupling of mixed-dimensional processes

Modeling principles:

1. Coupling only between subdomains with dimension gap of 1
2. Interaction between subdomains must go through interfaces
3. Equations on interfaces can only involve immediate subdomain neighbors



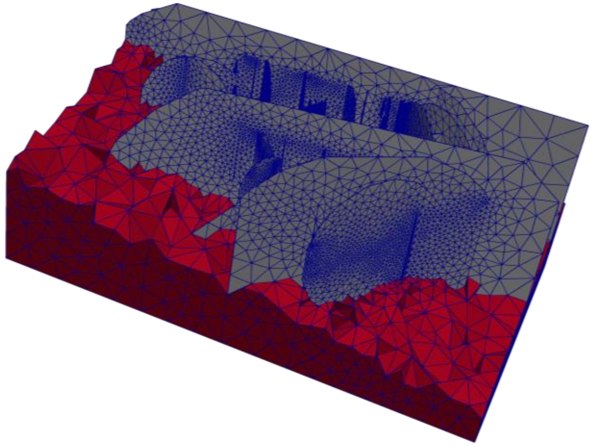
Mixed-dimensional grid is implemented to facilitate only these couplings

Benefits

- Framework has solid analytical foundations
- Subdomain models resemble fixed-dimensional problems
  - Couplings manifests as boundary conditions and generalized source terms
  - Legacy implementation of subdomain discretizations can be reused
- Interface equations and discretizations make the difference from fixed-dimensional problems

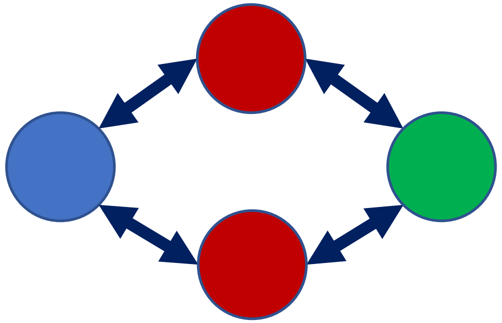
Boon, Nordbotten, Vatne: Functional analysis and exterior calculus on mixed-dimensional geometries
Annali di Matematica Pura ed Applicata, 2021

Meshing of md-geometries



```
# Define individual fractures
frac_1 = pp.Fracture3d(…)  #  give vertex coordinates
frac_2 = pp.Fracture3d(…)  #  give vertex coordinates

# Define a fracture network
fracture_network = pp.FractureNetwork3d([frac_1, frac_2, …])

# Generate a mixed-dimensional grid (mdg) via gmsh backend
mdg = fracture_network.mesh(…)  # Mesh size arguments
```

Accessing grid information



```
# Loop over subdomain grids
for sd in mdg.subdomains():
  # Get hold of subdomain data
  sd_data = mdg.subdomain_data(sd)

  # Get subdomain grid information
  sd.cell_centers
  sd.nodes
  …
```

```
# Loop over mortar grids
for intf in mdg.interfaces():
  # Get interface data
  intf_data = mdg.interface_data(intf)

  # Project to neighboring subdomains
  intf.mortar_to_primary_int()
  intf.secondary_to_mortar_avg()
  …
```

Example equations: Mixed-dimensional flow

Subdomains

Conservation (matrix, fractures, fracture intersections):

$$\nabla_d \cdot q_i^d - \sum \lambda_{i,j}^d + \psi_i^d = 0$$

$$q_j^d \cdot n_j^d = \lambda_{i,j}^{d-1}$$

Darcy flow ($d > 0$)

$$-\kappa_{i,||}^d \nabla_d p_i^d = q_i^d$$

λ^d: Flow in/out of higher-dimensional objects (source/sink)

λ^{d-1}: Flow in/out of lower-dimensional objects (boundary condition)

ψ: Standard sources and sinks

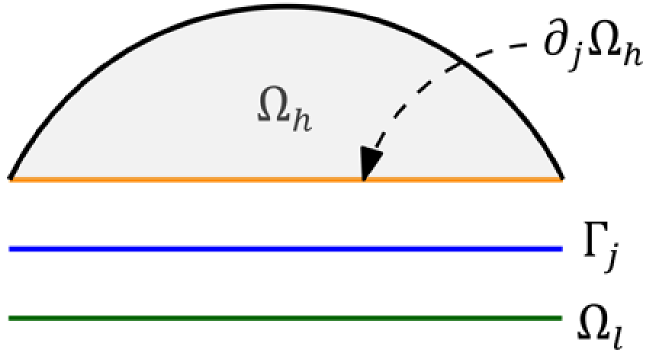
Projection operators to and from mortar grids are suppressed

Interfaces

Coupling condition

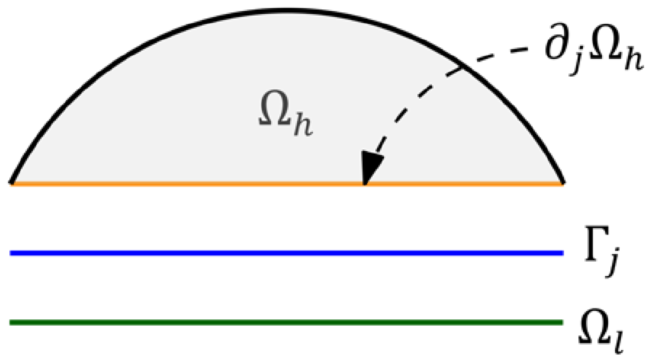
On Γ_j:

$$\lambda_{i,j}^d = \kappa_{i,\perp}^d \left(\operatorname{tr} p_j^{d+1} - p_i^d\right)$$



Nordbotten, Boon, Fumagalli, K: Unified approach to discretization of flow in fractured porous media, Comp. Geosci., 2019.

Implementation - pseudocode




```
flux = mpfa.flux * p # Flux within sd
  + mpfa.bound_flux  # BC discretization
  * mg.mortar_to_primary_int() # projection
  * intf_flux  # mortar flux

sources = source_sink  # Standard source
  + mg.mortar_to_secondary_int() # project
  * intf_flux

accumulation = div * flux + sources
```

```
# Project pressures to the mortar grids
p_h = mg.primary_to_mortar_avg() * p
p_l = mg.secondary_to_mortar_avg() * p

# Construct interface flux
intf_flux = normal_perm
  * (p_h - p_l)
  * mg.cell_volume
```

flux

$$q_i^d = -\kappa_{i,||}^d \nabla_d p_i^d$$

$$q_j^d \cdot n_j^d = \lambda_{i,j}^{d-1}$$

accumulation

$$\nabla_d \cdot q_i^d - \sum \lambda_{i,j}^d + \psi_i^d = 0$$

intf_flux

$$\lambda_{i,j}^d = \kappa_{i,\perp}^d \left(\operatorname{tr} p_j^{d+1} - p_i^d\right)$$

Defining mixed-dimensional multiphysics problems

Development of PorePy – Phase I

- Initiated as a collaborative project in early 2017.
- Initial focus:
  - Meshing of fractured domains
  - Enable simulations of flow, transport, fracture deformation
  - Philosophy: Move fast and break things
    - Mesh generation and basic (single-physics) discretizations remain in reworked form
    - Most other functionality implemented in 2017 was purged long ago
  - A principled approach to multiphysics simulations was lacking
- The code was open sourced in May 2017 (because why not?)

Development of PorePy – Phase II

- Framework for general multiphysics couplings
- Consolidation of already covered processes (flow, transport, deformation)
- Design principle: Code should reflect underlying mathematics
  - Rules for communication between subdomains and interfaces (mortar grids)
  - Clearer distinction between subdomain and interface problems
  - Equations should look similar on paper and in the code
- Gradual introduction of automatic differentiation
- Expansion of physical processes:
  - Reactive flow
  - Two-phase flow
  - Fracture propagation

Design principles

- Mixed-dimensional mesh: Collection of subdomain meshes, with projections in-between
- Finite-volume based modeling:
  - Impose conservation principles
  - Play around with constitutive laws
- Available discretization methods:
  - Two- and multipoint flux (diffusion)
  - Multipoint-stress (mechanics, poromechanics)
  - Upwinding (transport)
  - Variational inequalities (frictional contact mechanics)
- Tie everything together with automatic differentiation

Defining equations

Conservation laws

Single-physics:

- Mass
- Energy (or component transport)
- Momentum

Multiphysics:

- Mass + energy
- Mass + momentum (poromechanics)
- Mass + momentum + energy (thermo-poromechanics)

Constitutive laws (arbitrary classification)

Material laws:

- Darcy, Fourier, Hook
- Frictional contact mechanics

Fluid/rock-fluid:

- Density, viscosity
- (Relative permeability, capillary pressure)

Mechanics-related:

- Kozeny-Carman/aperture-permeability relation
- Shear dilation (fracture displacement jump -> aperture increase)

…

Technical details

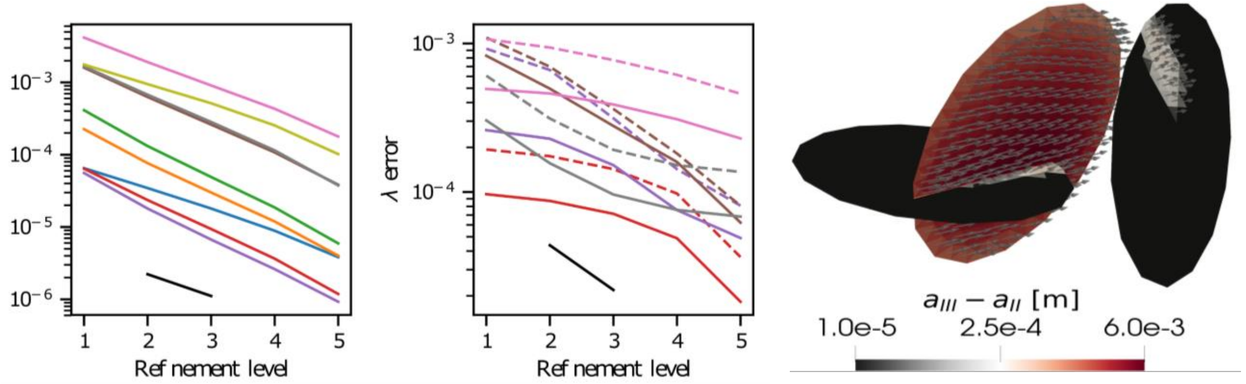
- Implementation by mixin classes (think: special type of inheritance)
  - Modularized implementation
  - Extreme flexibility in combining constitutive laws
  - Steep learning curve to navigate in the code base
  - Precision needed in problem definitions

- Two-layer Ad formulation:
  - Abstract representation of expressions as a computational graph
  - Graph is translated into numerical value by forward Ad

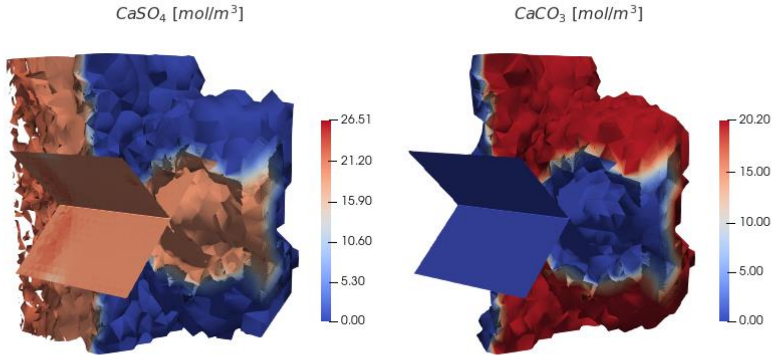
Example: Implementation of

$$\phi = \phi_0 + \frac{1}{N}(p - p_0) + \alpha(\nabla \cdot u)$$

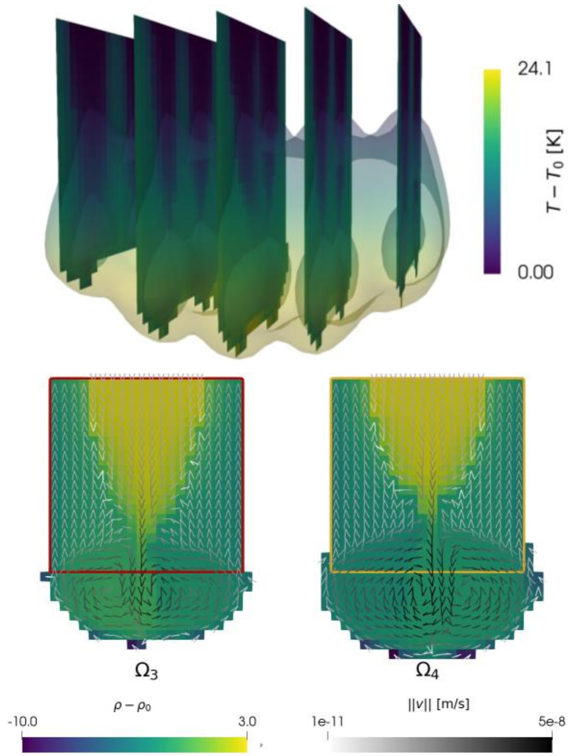
```
class PoroMechanicsPorosity():
  def matrix_porosity(self, subdomains: list[pp.Grid]) -> pp.ad.Operator:
    return (self.reference_porosity(subdomains)
      + self.porosity_change_from_pressure(subdomains)
      + self.porosity_change_from_displacement(subdomains)
    )

  def porosity_change_from_displacement(self, subdomains) -> pp.ad.Operator:
    alpha = self.biot_alpha(subdomains)
    div_u = self.dispcament_divergence(subdomains)
    return alpha * div_u

  def porosity_change_from_pressure(self, subdomains) -> pp.ad.Operators:
    …
```



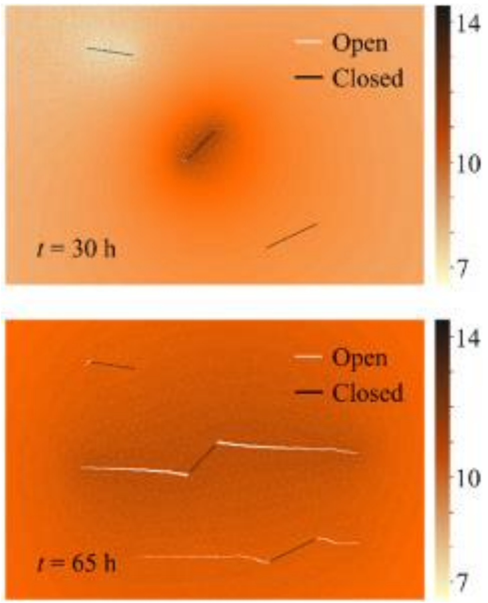
Thermo-poromechanical fracture deformation
Stefansson et al, CMAME, 2022



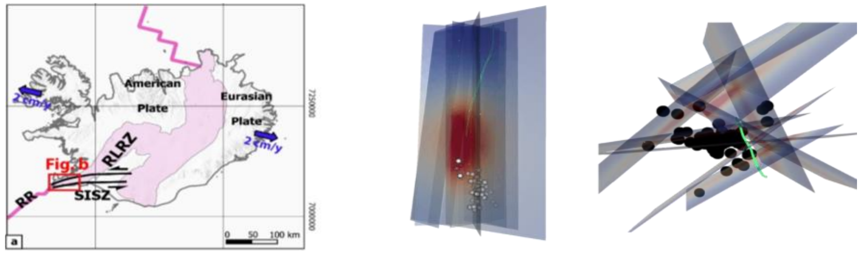
Non-isothermal reactive transport
Banshoya et al, submitted



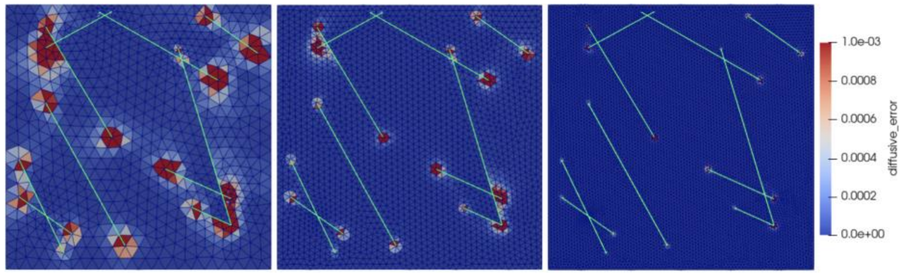
Propagation by thermal contraction
Stefansson et al, TiPM, 2022



Poromechanical fracture propagation
Dang et al, IJRMMS 2022



Field studies of injection-induced seismicity
K. Et al, Geothermics, 2021



A posteriori error estimates
Varela et al, JNM 2022

Current development mode

- Main drivers for development:
  - Inclusion of new physical processes
  - Improved numerical approaches motivated by weak points in physical modeling
  - Large-scale maintenance (both front- and backend)
- Narrowing of main usage mode (Finite volumes, mixins, AD)
  - Broadening of what can be done within that frame
  - Larger parts of the code become stable
- Most development takes place at UiB
  - Some external usage and contributions
- Code standardization:
  - Documentation
  - Tutorials
  - Test-driven development

Future directions

- Additional physics to be introduced:
  - Non-isothermal multiphase multicomponent flow
  - (Constitutive modeling for) fracture deformation and propagation
  - Tighter coupling between processes
- Enable more complex simulations:
  - More robust numerical approaches:
    - Splitting schemes for multiphysics
    - Improved spatial discretizations
  - Limit computational cost:
    - Iterative block solvers for mixed-dimensional problems
    - Flexibility while also shielding users from solver design
- Stay alive and relevant:
  - Fight code entropy
  - Contribute to getting the next project

Implementation: *https://github.com/pmgbergen/porepy*