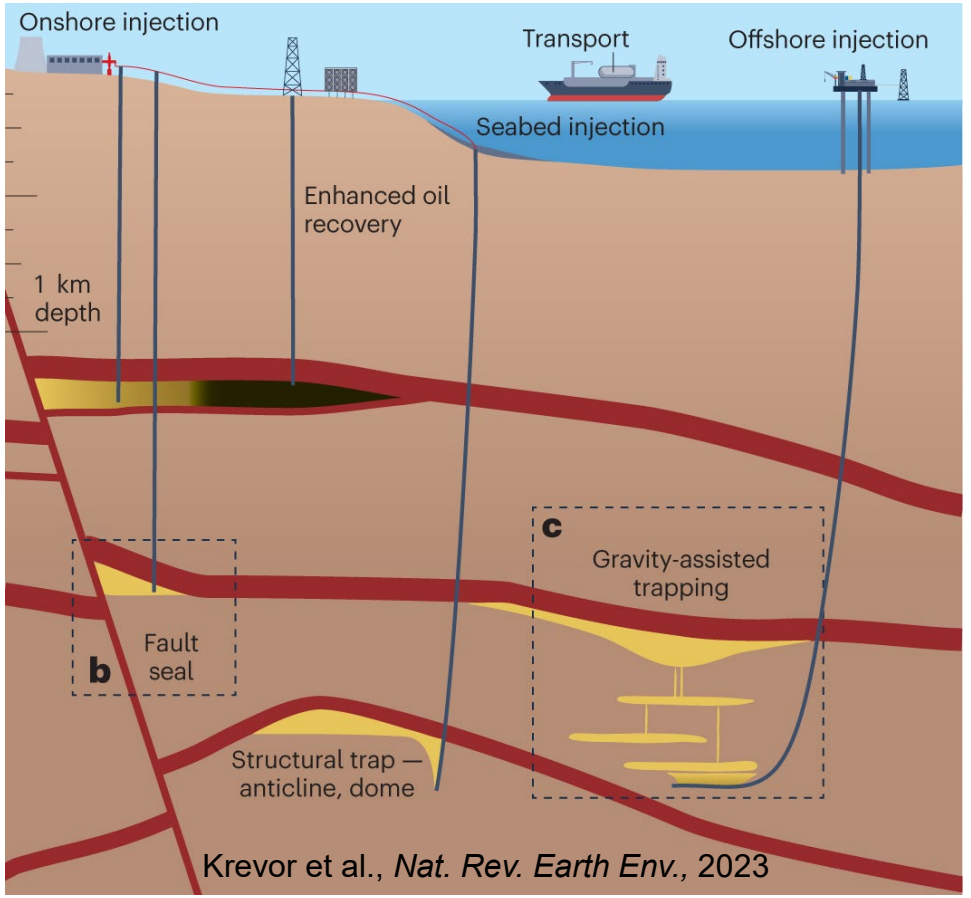
Interpolation Strategies For Complex Thermal-Multiphase-Reactive Modeling With Operator-Based Linearization

Aleks Novikov[1], Denis Voskov[1,2]

1. Delft University of Technology, Netherlands
2. Stanford University, USA

Motivation

- *Thermal-multiphase-reactive compositional modelling* is essential for feasibility and performance assessments of
  - CO2 and hydrogen storage, geothermal sites and EOR
- Modeling is challenging
  - assembly is inflexible and resource-intensive
  - necessity to work with third-party thermodynamics and geochemistry solvers
- Advanced modeling techniques can reduce complexities
  - Operator-Based Linearization (OBL)
  - Adaptive parametrization

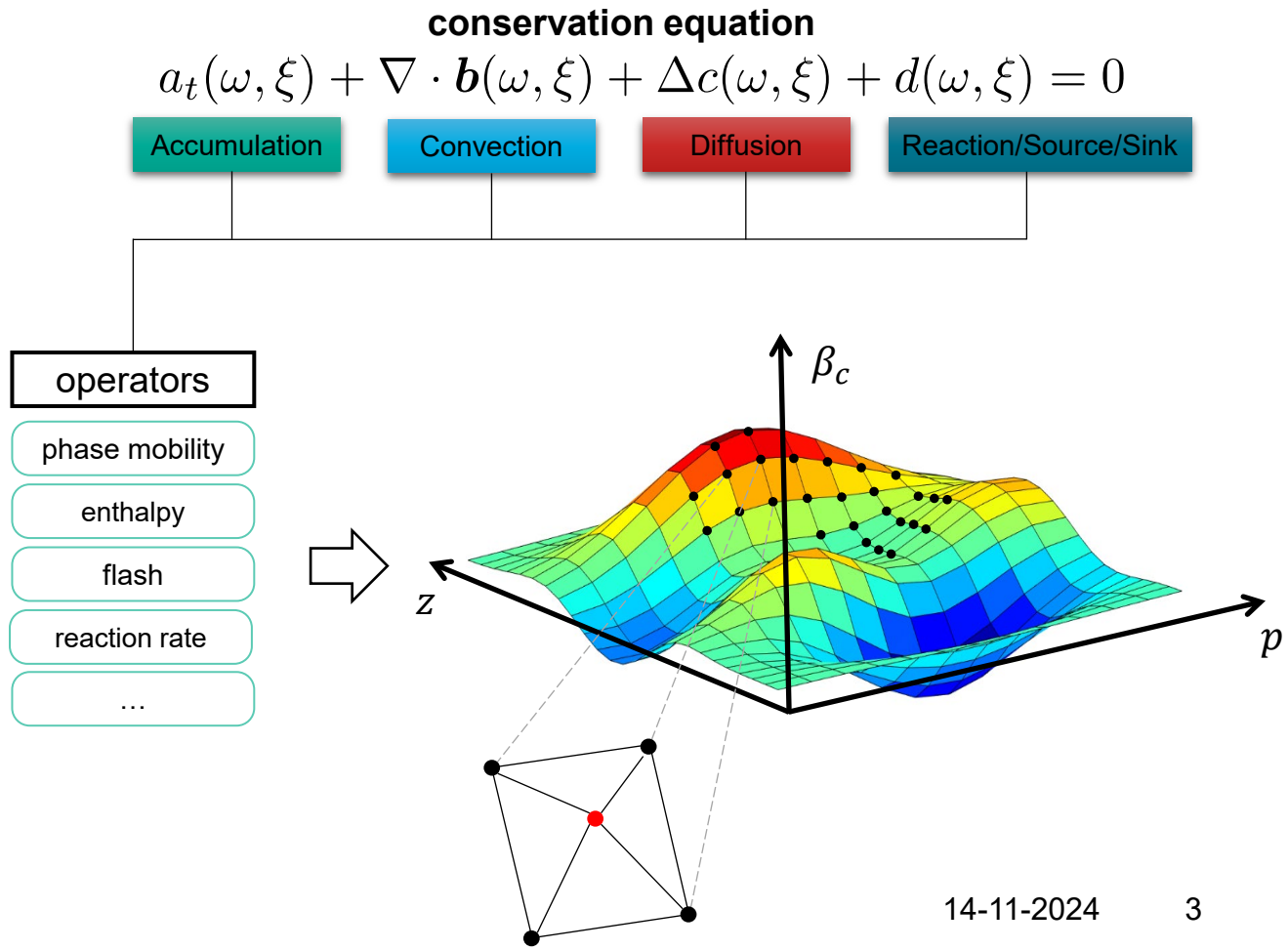


Krevor et al., *Nat. Rev. Earth Env.*, 2023

TU Delft

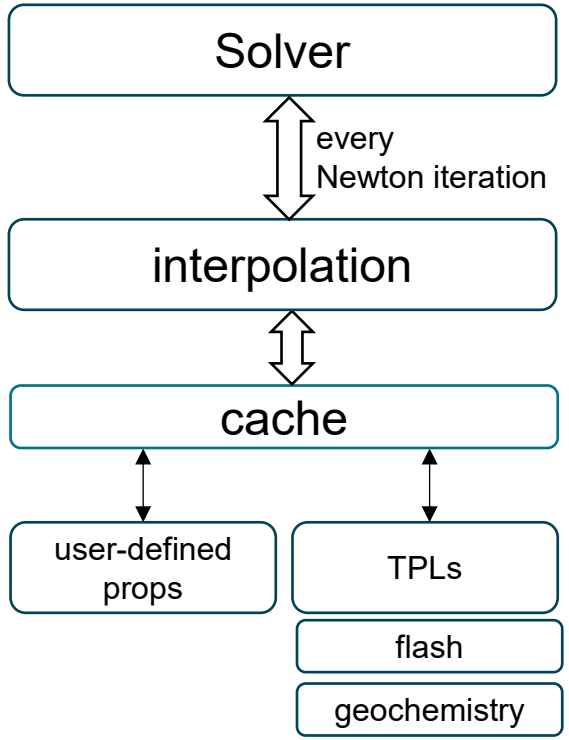
Operator-based Linearization (OBL)

- Group input properties into operators according to computational needs
- Cover state space with a structured grid
- Evaluate operators at grid nodes
- Interpolate in-between



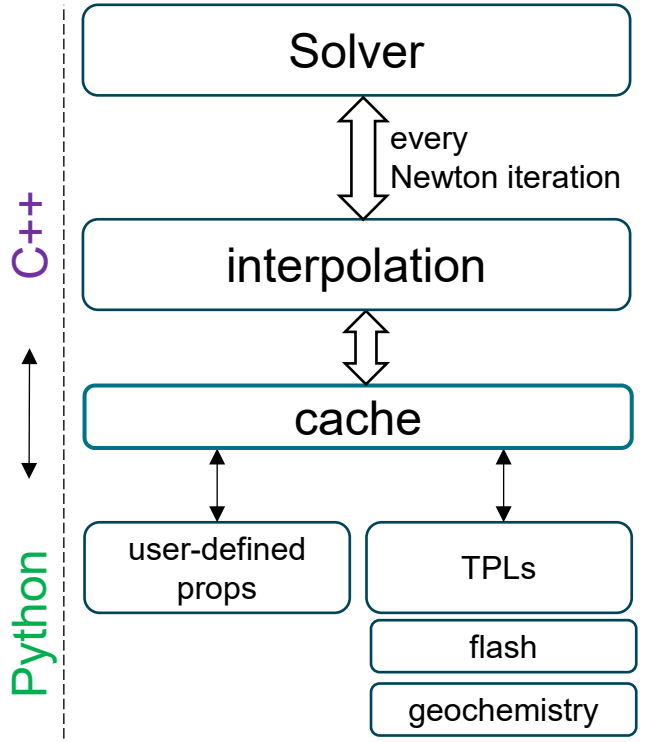
Advantages of Operator-based Linearization

- Caching of operators
  - Flexibility: works with any kind of user-defined or calculated properties
  - Efficiency: example - linear interpolation (~10 CPU cycles) vs exponent evaluation (~100 CPU cycles)
- User-friendly model input from Python
  - Convenient integration with third-party libraries
  - Easy to connect to C++ (Python C API, boost.Python, Pybind11, SWIG)
- Various Parametrization Strategies
  - Static parametrization: operators are evaluated beforehand
  - Adaptive parametrization: efficient and accurate in many dimensions



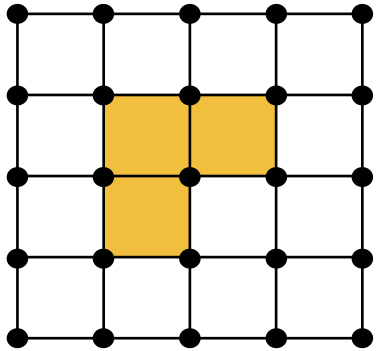
Advantages of Operator-based Linearization

- Caching of operators
  - Flexibility: works with any kind of user-defined or calculated properties
  - Efficiency: example - linear interpolation (~10 CPU cycles) vs exponent evaluation (~100 CPU cycles)
- User-friendly model input from Python
  - Convenient integration with third-party libraries
  - Easy to connect to C++ (Python C API, boost.Python, Pybind11, SWIG)
- Various Parametrization Strategies
  - Static parametrization: operators are evaluated beforehand
  - Adaptive parametrization: efficient and accurate in many dimensions

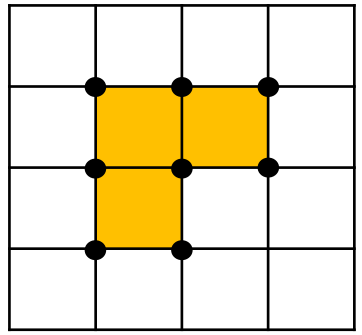


Advantages of Operator-based Linearization

- Caching of operators
  - Flexibility: works with any kind of user-defined or calculated properties
  - Efficiency: example - linear interpolation (~10 CPU cycles) vs exponent evaluation (~100 CPU cycles)
- User-friendly model input from Python
  - Convenient integration with third-party libraries
  - Easy to connect to C++ (Python C API, boost.Python, Pybind11, SWIG)
- Various Parametrization Strategies
  - Static parametrization: operators are evaluated beforehand
  - Adaptive parametrization: efficient and accurate in many dimensions



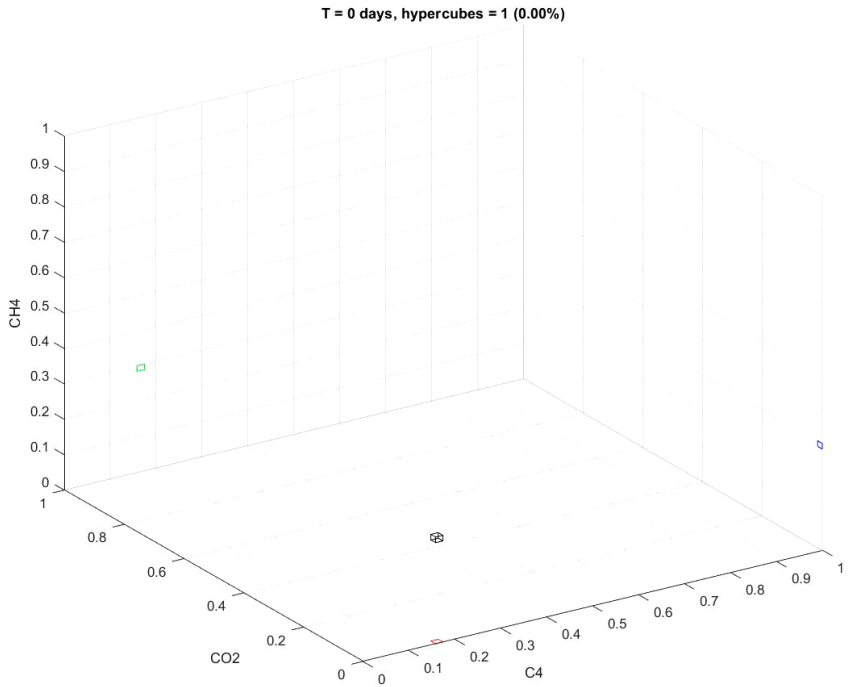
static parametrization

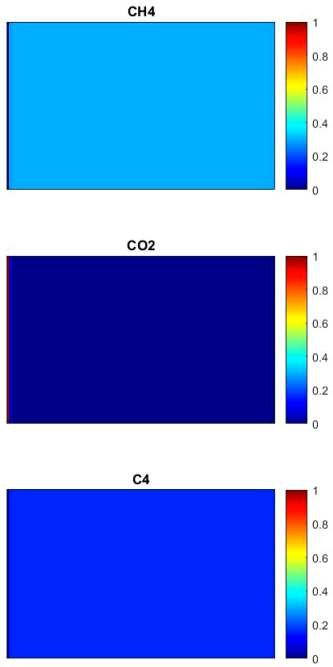


adaptive parametrization

TU Delft

Sparse occupancy of state space



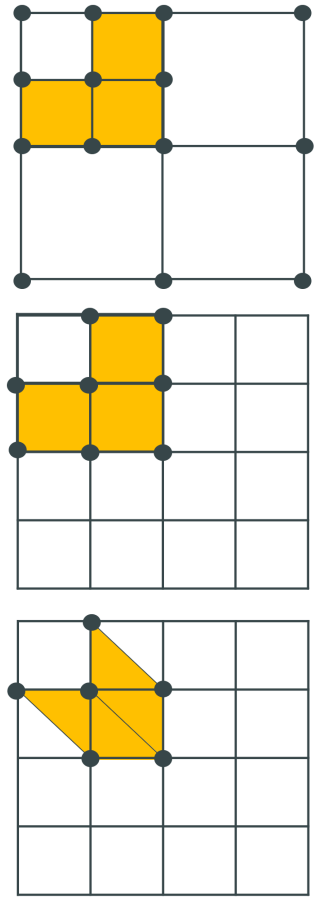


4 components: ~0.1%

6 components: ~0.001%

Interpolation strategies

- **Nested static** interpolation
  - coarse grid with nested refinement
  - refinement of active hypercubes
  - compromise between accuracy and inability to call Python
- **Multilinear** adaptive interpolation
  - $O(2^d)$ operations
  - accurate but expensive in many dimensions
- **Linear** adaptive interpolation
  - $O(d)$ operations
  - standard vs Delaunay triangulation
  - supported higher dimensions (up to 20 components)



Example 1
static nested interpolation

- liquid-vapor mixture of 4 components [CO2, C1, C4, C10] with given K-values
- Adaptive interpolation with 1000 points per axis
- Static nested interpolation
  - 10 points per axis initially
  - 10 points per axis in adaptive refinement between time steps

Comparison of adaptive and static nested interpolators

TU Delft

Example 2:
Linear interpolation for 20 component fluid

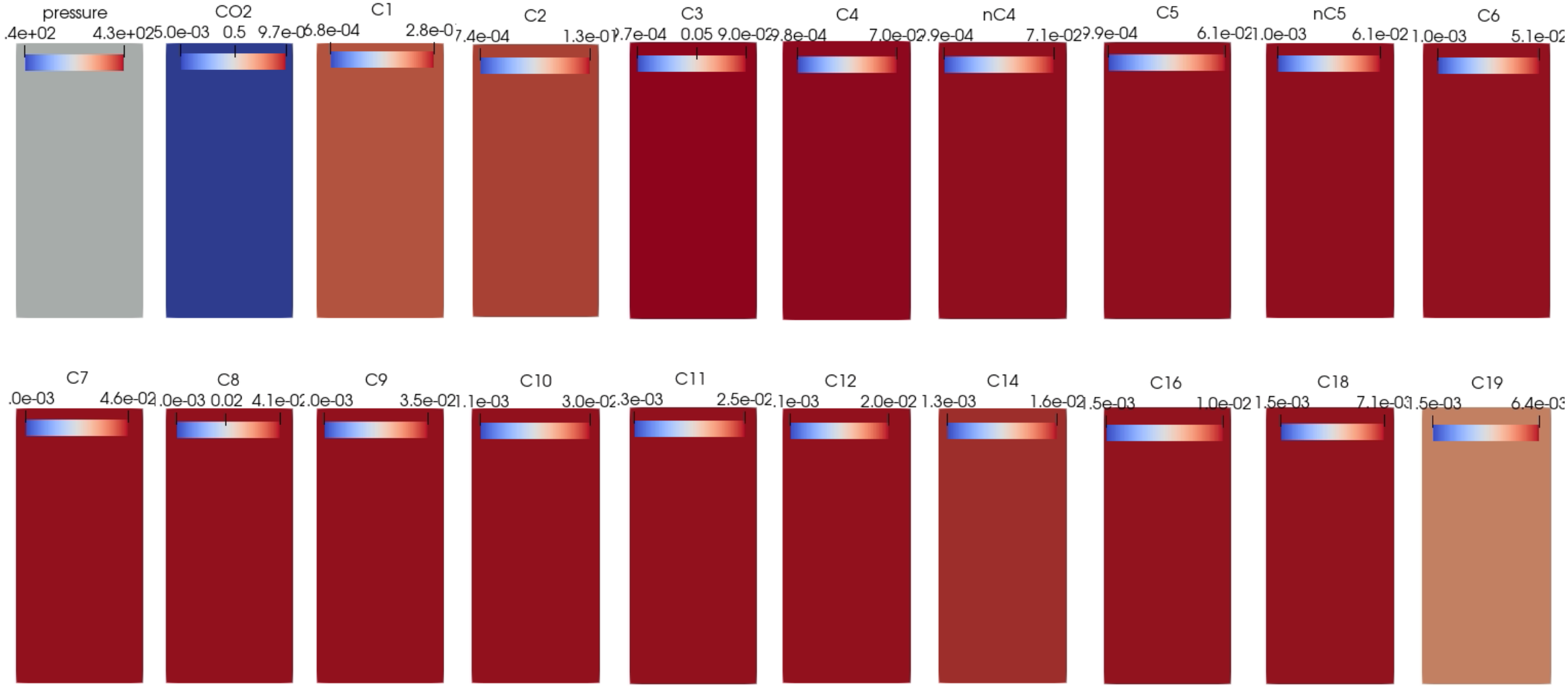
- two-phase 20 components fluid flow
  - CO2, C1, C2, C3, C4, nC4, C5, nC5, C6, C7, C8, C9, C10, C11, C12, C14, C16, C18, C19, C20
- fluid physics is defined by constant K-values
- multivariate linear interpolation in 20 dimensions, 80 points per axis
- inject pure CO2

TU Delft

Example 2:
Linear interpolation for 20 component fluid

simulation time on a single-core CPU:

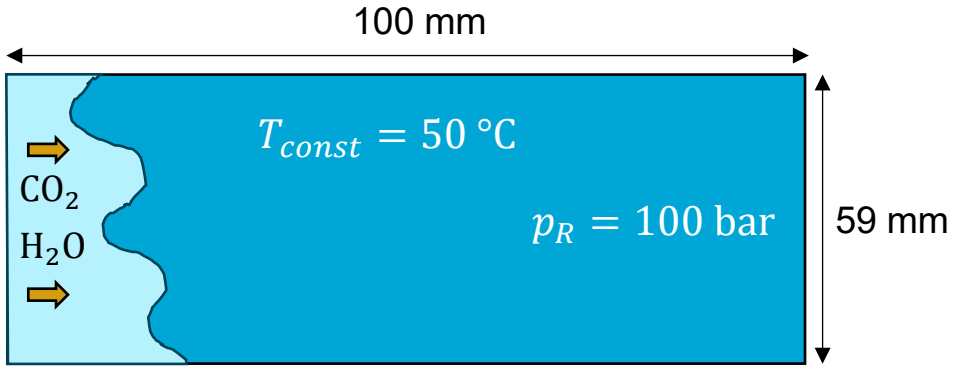
- assembly 31.9%
  - interpolation 13.6%
  - point generation 12%
- solution 67.7%



Example 3:
Calcite dissolution in CO_2 injection

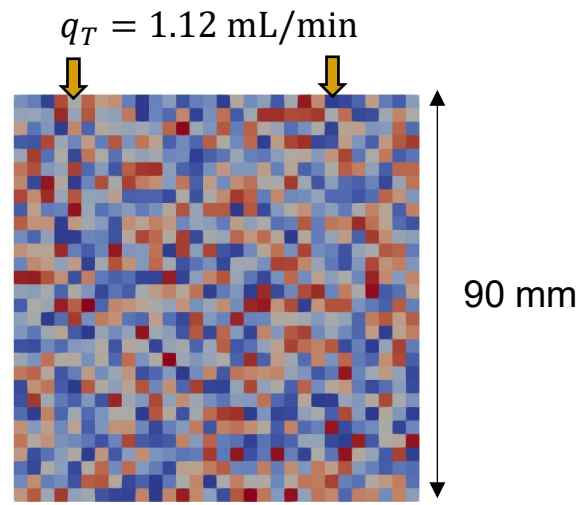
- element-based formulation of reactive flow and transport
  - isothermal two-phase flow
  - porosity-permeability relationship $k/k_0 = (\phi/\phi_0)^4$
  - fixed Corey-like phase permeabilities
- phase viscosities are calculated by CoolProp database
- geochemical speciation and equilibrium from PHREEQC
- reaction kinetics $r = r_a + r_n$
  - acidic reaction $r_a = a_{H^+} k_{ref}^{(a)} \cdot \exp\left(-\frac{E_{aa}}{R}(T^{-1} - T_{ref}^{-1})\right) \Omega(1 - \mathrm{SR}^p)^q$
  - neutral reaction $r_n = k_{ref}^{(n)} \cdot \exp\left(-\frac{E_{an}}{R}\left(T^{-1} - T_{ref}^{-1}\right)\right) \Omega(1 - \mathrm{SR}^p)^q$
  - fixed area multiplier $\Omega = \Omega_0$
  - dissolution-precipitation multiplier $(1 - \mathrm{SR}^p)^q, \quad p = q = 1$

1 Homogeneous 1D model



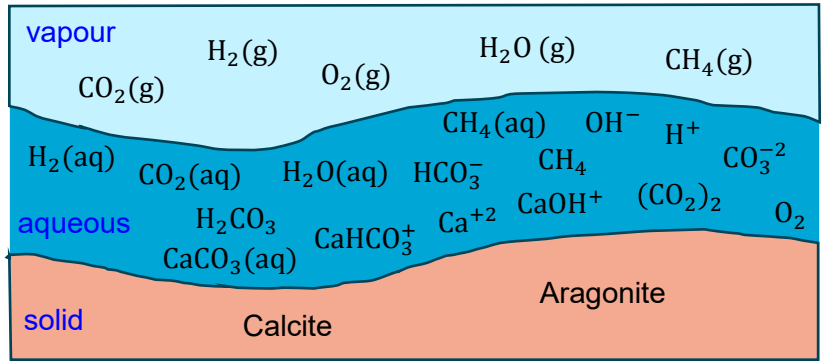
$q_L = 0.15$ mL/min

2 Heterogeneous 2D model

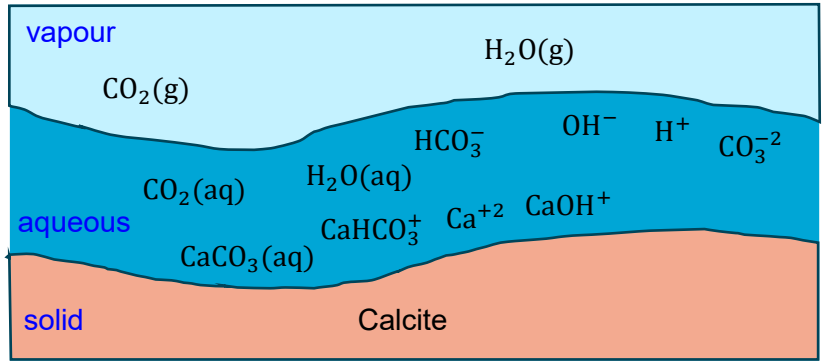


Example 3:
Calcite dissolution in CO_2 injection

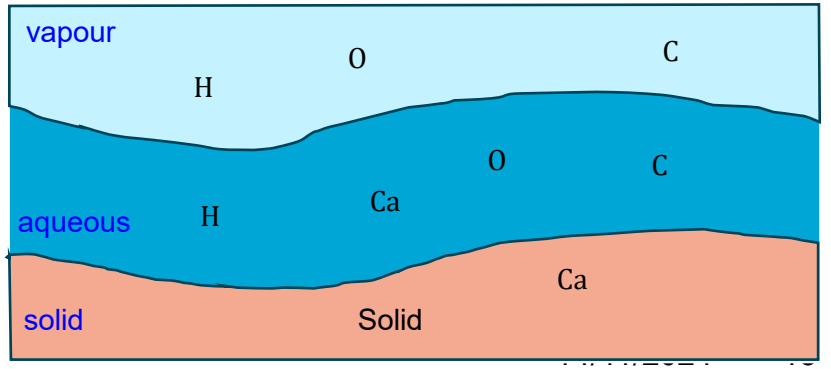
- geochemical speciation and equilibrium from PHREEQC
- simplified geochemical model for calculation of initial state
- the following classes of reactions are considered during simulation
  - oxidation-reduction reactions
  - acid-base reactions (carbonic acid, carbonate, bicarbonate)
  - complexation and speciation ($CaHCO_3^+$, $CaCO_3$(aq) complexes)
  - redox environment (methanogenesis)
  - dissolution-precipitation of calcite, aragonite
  - gas equilibria (CO_2, CH_4, H_2, O_2)



Species in geochemistry system



Species in transport system



Elements (unknowns) in transport system

TU Delft

Example 3: homogeneous 1D setup

Calcite dissolution in CO_2 injection

- OBL Interpolation time:

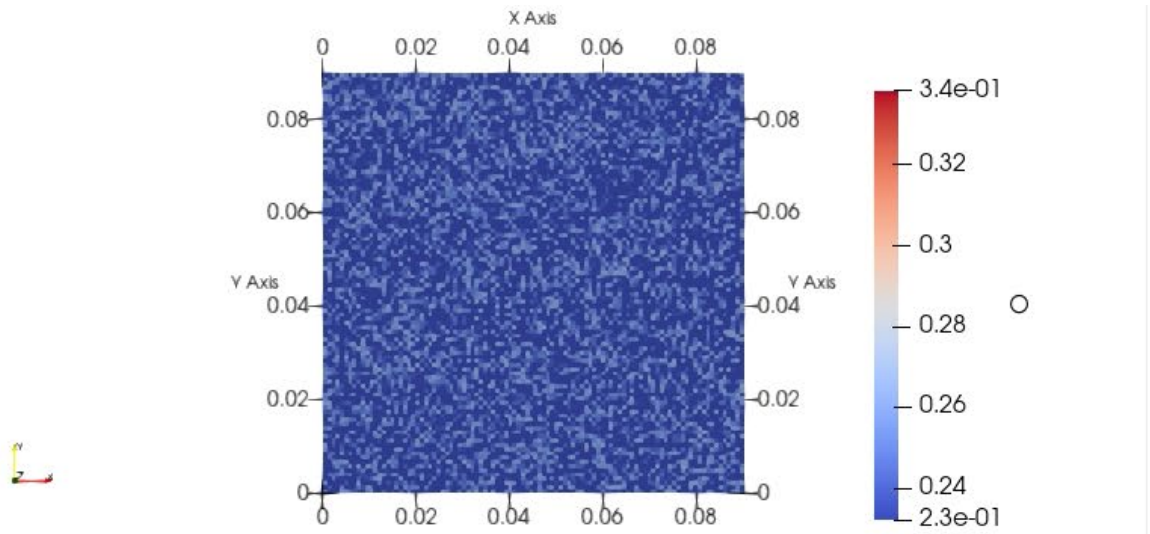
  (not-cached) 52% of simulation time

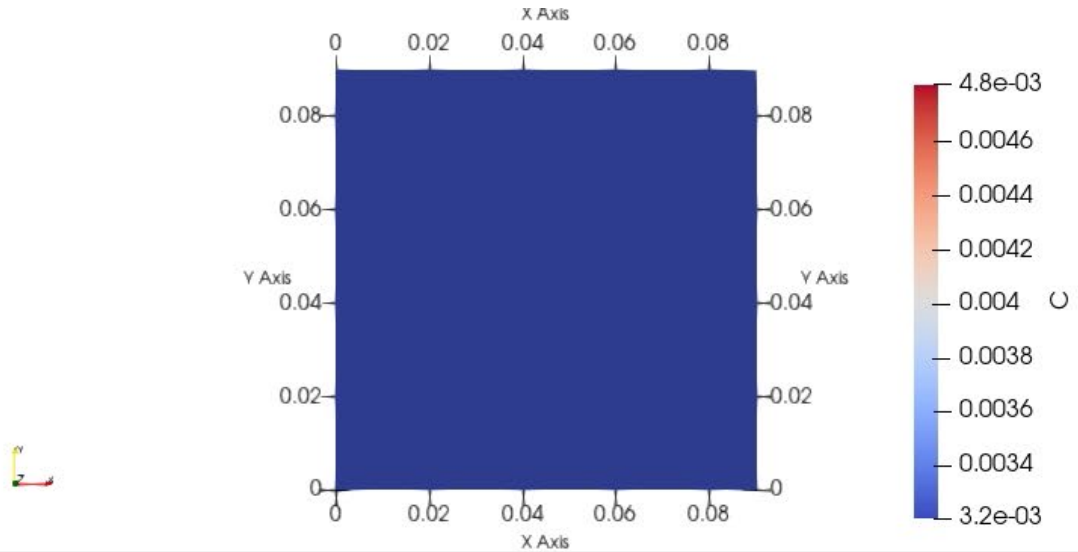
  (cached) 5% of simulation time

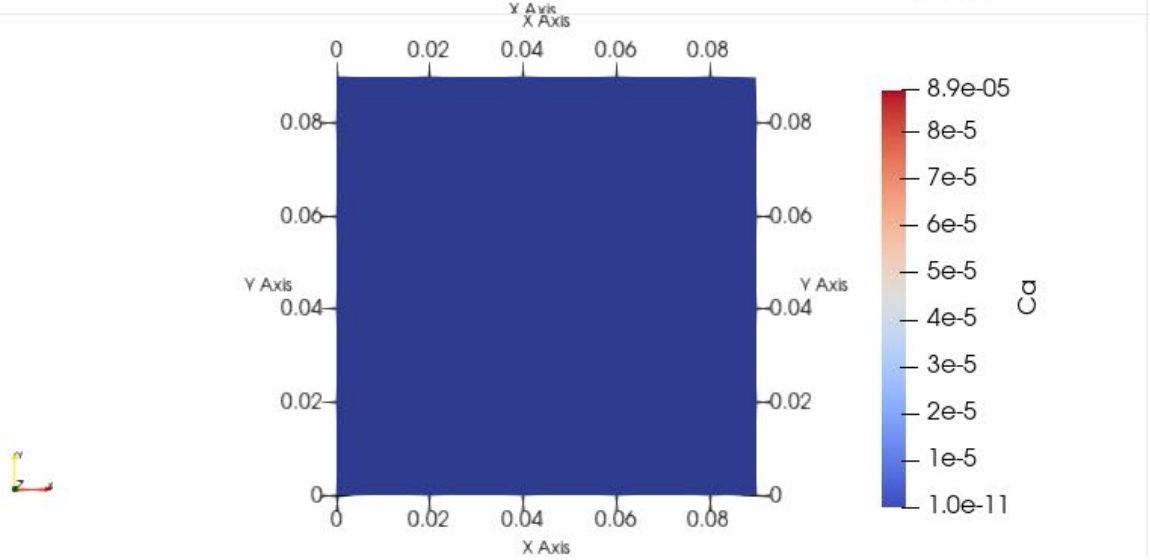
TU Delft

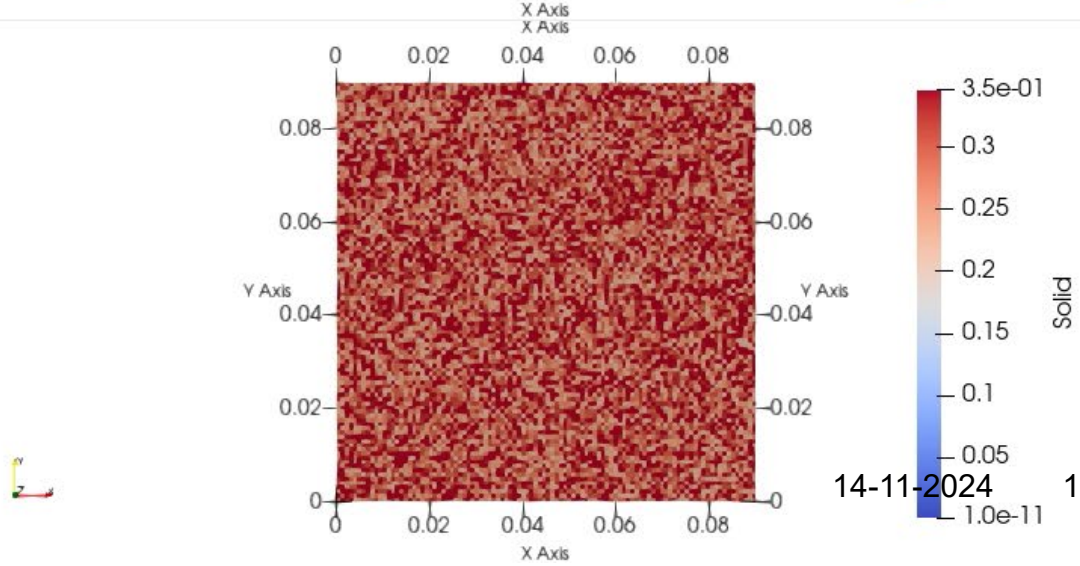
Example 3: heterogenous 2D setup

Calcite dissolution in CO_2 injection









Takeaways & Future work

- **Takeaways:**
  - OBL and adaptive parametrization simplify the assembly of thermal-multiphase-reactive systems, save simulation and development resources
  - Linear interpolation is efficient and accurate in many dimensions
  - Nested static interpolation can be accurate but not affordable in many dimensions
  - Caching is crucial for complex fluid physics, especially while coupling to third-party solvers
- **Future work:**
  - general PHREEQC-backed thermal two-phase dissolution-precipitation physics in open-darts
  - parallel adaptive interpolation in GEOS, millions-cell model modelling
  - a big room for improvement of OBL: asynchronous evaluation and interpolation, adaptive sampling strategies

Thank you!