FAIR Workflows with OPM Flow: History Matching FluidFlower and Benchmarking on SPE11

David Landa-Marbán

Computational Geosciences and Modelling

dmar@norceresearch.no

NORCE Research AS

Centre for Sustainable Subsurface Resources (CSSR)

Outline

OPM Flow selected contributions (since last OPM summit)

Overview of the **GitHub/cssr-tools** page

Motivation for **FAIR** workflows

pofff

pyopmspe11

Summary

OPM Flow selected contributions

- Consistent unit handling in OPERATE ✓ manual:bugfix — 10
  #4765 by daavid00 Member was merged on Oct 2, 2025
- BFLO[G|O|W][I|J|K][|-] summary keywords ✓ manual:new-feature — 1 — 6
  #4954 by daavid00 Member was merged on Feb 2
- BVEL[G|O|W][I|J|K][|-] summary keywords ✓ manual:enhancement — 4
  #4989 by daavid00 Member was merged on Feb 16
- Correct handling of overlaping NNC and EDITNNC ✓ manual:irrelevant — 1 — 18
  #5025 by daavid00 Member was merged 3 weeks ago
- Making possible to simulate 100 Million years ✓ manual:enhancement — 15
  #6336 by daavid00 Member was merged on Jul 17, 2025
- Support for CONV_NEW when CONV in RPTRST ✓ manual:new-feature — 36
  #6374 by daavid00 Member was merged on Aug 21, 2025
- Support for NEWTMX and NEWTMN in TUNING ✓ manual:new-feature — 9
  #6378 by daavid00 Member was merged on Sep 5, 2025
- TUNING: XXX and TRG for CNV and MBE ✓ manual:enhancement — 11
  #6413 by daavid00 Member was merged on Aug 22, 2025
- Use new lineariser for simulators with EnableBrine ✓ manual:irrelevant — 11
  #6503 by daavid00 Member was merged on Sep 29, 2025
- Keep derivatives for trans_mult ✓ manual:irrelevant — 13
  #6586 by daavid00 Member was merged on Nov 12, 2025
- Bringing back XMFCO2/YMFWAT output ✓ manual:bugfix — 1 — 7
  #6637 by daavid00 Member was merged on Dec 1, 2025
- BIOFILM effects for gas-water systems ✓ manual:new-feature — 22
  #4733 by daavid00 Member was merged on Sep 18, 2025

OPM OPEN POROUS MEDIA

OPM FLOW REFERENCE MANUAL (2025-10) Revision: Rev-0

5.3.10 BIOFILM – ACTIVATE THE BIOFILM MODEL

RUNSPEC	GRID	EDIT	PROPS	REGIONS	SOLUTION	SUMMARY	SCHEDULE

Description

The *BIOFILM* keyword activates the biofilm model for the run. Biofilm-related effects in subsurface applications such as hydrogen storage include reduced injectivity and hydrogen loss. The conceptual model includes the following mechanisms: (1) Biofilm is present in the storage site prior to injection and (2) the biofilm consumes injected H_2/CO_2, leading to clogging effects.

Note

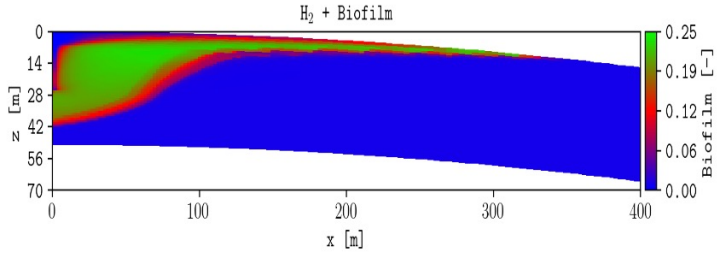
This is an OPM Flow specific keyword used to investigate biofilm effects in underground applications. The module requires that either the *CO2STORE* or *H2STORE* keywords in the *RUNSPEC* to be active.

There is no data required for this keyword and there is no terminating "/" for this keyword.

Example

```
--
--       ACTIVATE THE BIOFILM MODEL
--
BIOFILM
--
--       ACTIVATE THE H2STORE MODEL
--
WATER
GAS
H2STORE
```

The above example declares that the *BIOFILM* model is active for the run and activates the *H2STORE* model. For a complete example of this model, see *H2STORE_BIOFILM.DATA*.



The FAIR principles

Findable

- Data and metadata have unique identifiers and are easy to discover.

Accessible

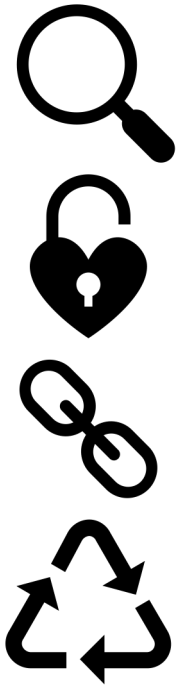
- Data can be reliably accessed using standard, open protocols (with clear access conditions).

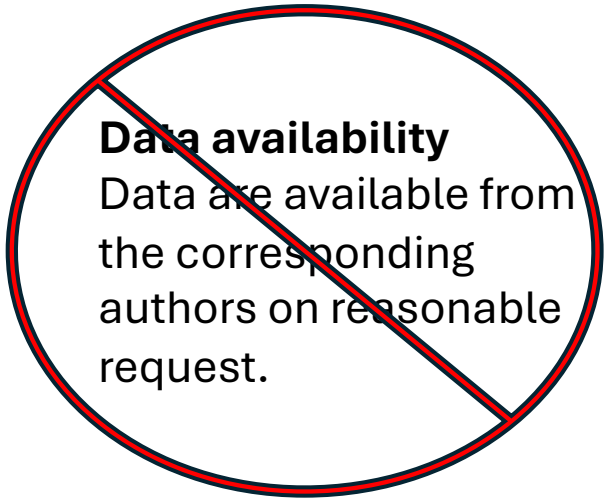
Interoperable

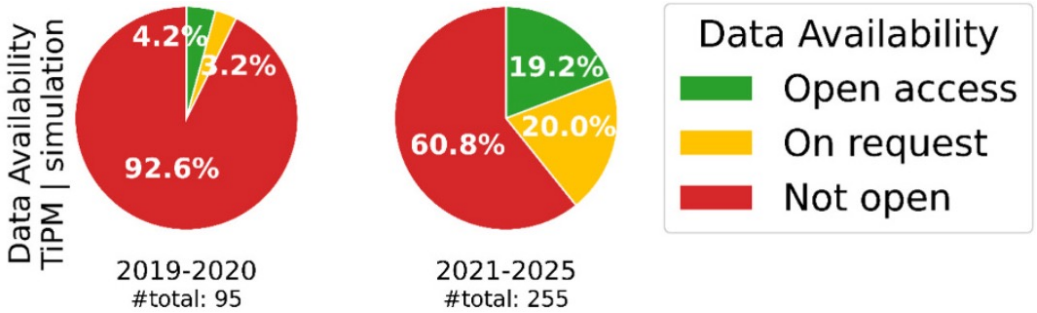
- Data use shared formats, vocabularies, and standards so they work with other data and tools.

Reusable

- Data are well described, with clear licenses and provenance, enabling reuse over time.







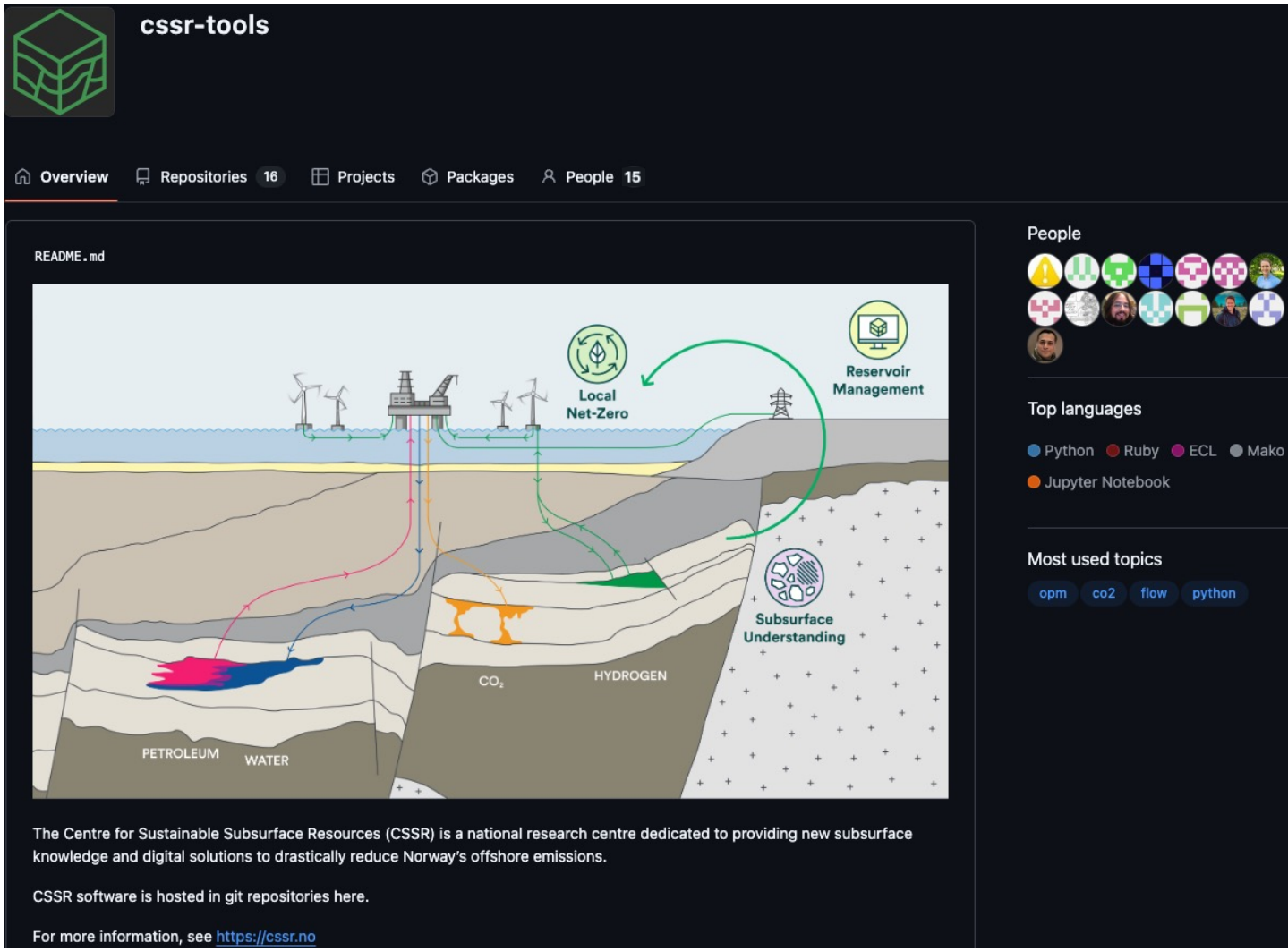
Data Availability

The simulator OPM Flow can be obtained at https://opm-project.org. Link to complete codes for the numerical studies are available in https://github.com/cssr-tools/pofff. The optimization tool Everest is available at https://github.com/equinor/ert. The Python image analysis toolbox DarSIA used for obtaining the segmentation and continuous CO2 data can be accessed via https://github.com/pmgbergen/DarSIA; corresponding runscripts analyzing the FluidFlower dataset are available at https://github.com/pmgbergen/fluidflower_benchmark_analysis. The plotting tool for the simulation results is available at https://github.com/cssr-tools/plopm.

Wilkinson, M., et al., 2016. The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 3, 160018. https://doi.org/10.1038/sdata.2016.18
Liu, N., et al. 2025. Trends in porous media laboratory imaging and open science practices. Adv. Colloid Interface Sci. 352, 103831. https://doi.org/10.1016/j.cis.2026.103831

https://github.com/cssr-tools

- CSSR started in July 2022 (until 2029)
- cssr-tools launched in February 2024
- 15 people (NORCE and UiB)
- 12 hosted repositories
- 4 forked repositories



https://github.com/orgs/cssr-tools/repositories

16 repositories — Stars

homebrew-opm	Using brew to install ResInsight and OPM Flow with mpi support on macOS	Ruby	0	6
expreccs	A Python framework using OPM Flow to simulate regional and site reservoirs for CO2 storage	C++	1	6
pycopm	Creation of OPM Flow geological models from provided input decks with options for grid refinement, grid coarsening, submodels, and transformations...	ECL	2	6
pymm	Open-source image-based framework for computational fluid dynamics on microsystems	Python	2	5
plopm	Quick generation of PNGs, GIFs, and VTKs from a OPM Flow type model	Python	2	5
SubCSeT	Screening CO2 storage potential in petroleum reservoirs on the Norwegian Continental Shelf	Jupyter ...	3	5
pyopmnearwell	A Python framework to simulate near-well dynamics using OPM Flow	Python	3	4
hyopml	Hybrid Newton package and tutorial for usage within OPM Flow reservoir simulation	Python	0	4
pofff	An open-source image-based history-matching framework for the FluidFlower benchmark study using OPM Flow	Python	1	3
ML_near_well	Runfiles for an ML near-well model and to reproduce results from the article "A machine-learned near-well model in OPM Flow". Uses pyopmne...	Mako	2	3
DrogonECMOR2024	Wind-powered reservoir management with the Drogon field	Python	1	1
.github			0	0
saltprectough2	The main files used within TOUGH2 for studying salt precipitation		0	0
pyopmspe11	A Python framework using OPM Flow for the SPE11 benchmark project	Python	13	0
PET	PET for data assimilation and optimization	Python	14	0
DarSIA	Darcy scale image analysis toolbox	Python	7	0

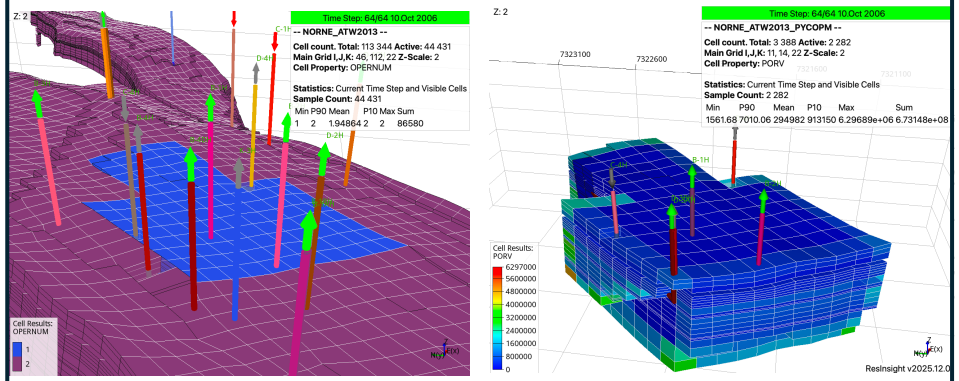
This presentation

Example of usage of pycopm: submodel extraction given an irregular polygon, where the pore volume from the outside region is added to the cell boundaries (other options for extraction are cells around a given well, as well as defined properties such as OPERNUM, FIPNUM, etc).

```
pycopm -i NORNE_ATW2013.DATA  -p 1 -v 'xypolygon [457302,7322519] [456941,7321991]
[457331,7321213] [457798,7321745] [457661,7321959] [458077,7322403] [457779.86,7322696]
[457457,7322299] [457302,7322519]'
```

Or defining the region using OPERNUM:

```
pycopm -i NORNE_ATW2013.DATA  -m all -p 1 -v 'opernum 1'
```



See the online documentation (https://cssr-tools.github.io/pycopm/) and preprint (https://arxiv.org/abs/2602.11777) for details.

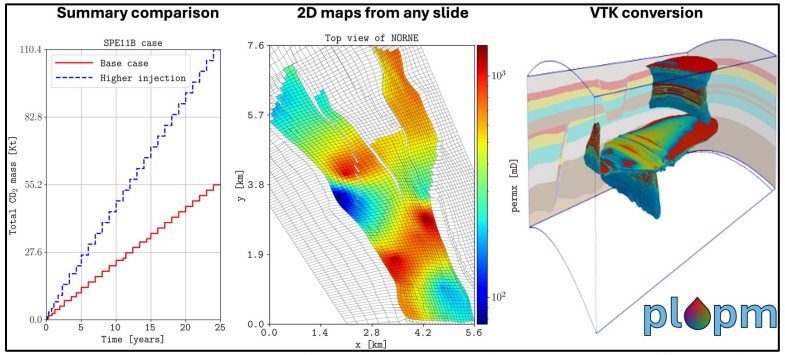
Install OPM Flow and ResInsight in macOS using brew | passing | License GPLv3

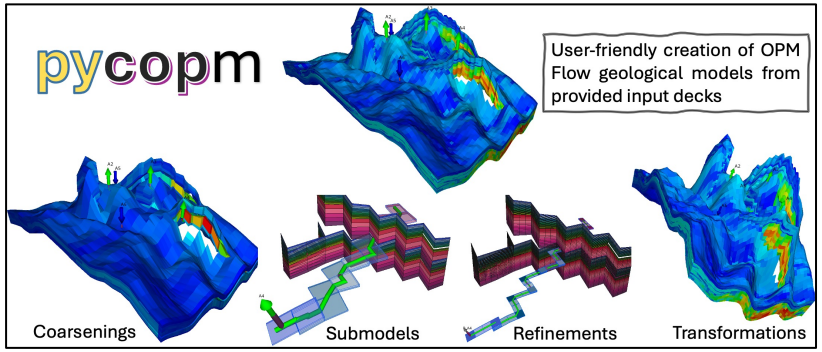
Installing ResInsight and OPM Flow with support for mpirun in macOS using brew

This repository uses *brew* to build *ResInsight* (v2025.12.0) and *OPM Flow* (v2026.02) in macOS Tahoe in GitHub Actions, showing the status and details of the build in the *Actions*. See *this script* that is run in GitHub actions.

```
brew install cssr-tools/opm/opm-simulators
brew install cssr-tools/opm/resinsight
```

After you execute the above lines, you can check if the installation of OPM Flow succeded by typing in the terminal `flow --help`, and ResInsight is launched by typing `resinsight`.





* Successfully tested with Apple M5

User- and developer-friendly features

“Python” as programming language

“GitHub” as hosting service

“Black” for code formatter/beautifier

“Pylint” for code analysis

“Pytest” for code testing

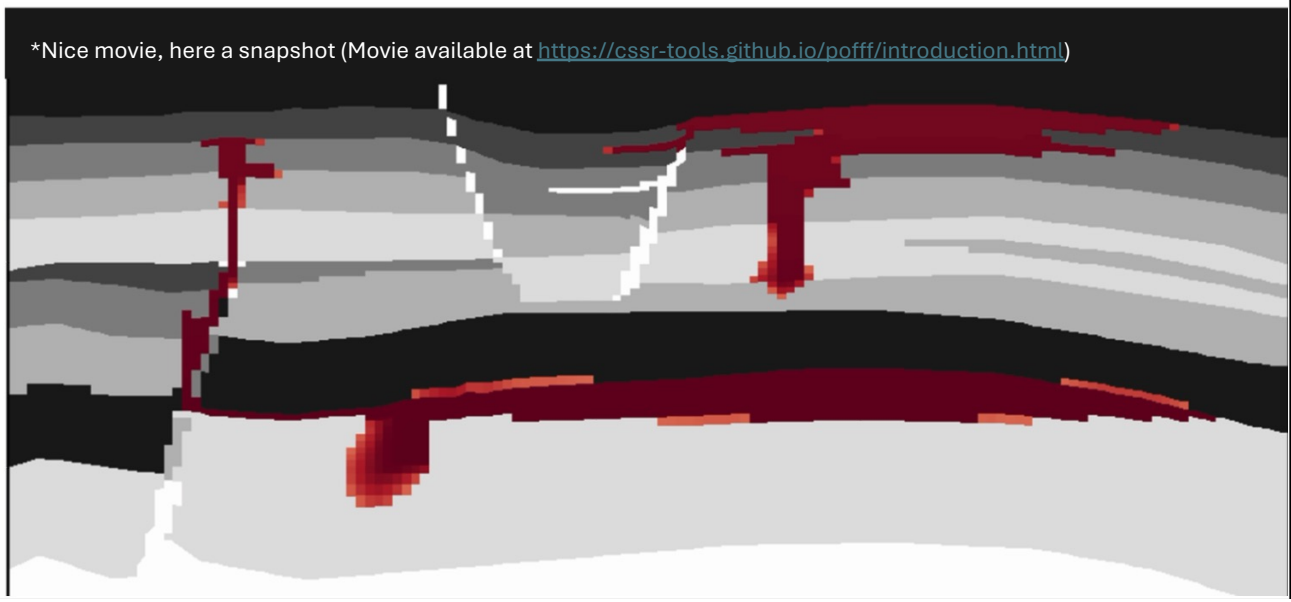
“Read the Docs” for documentation

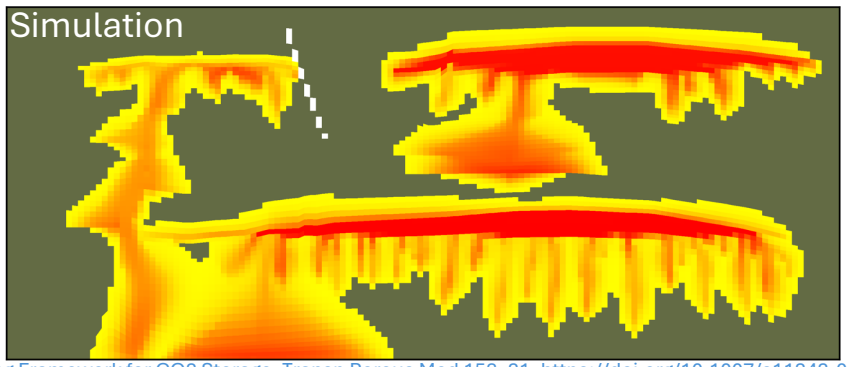
“TOML” for configuration files

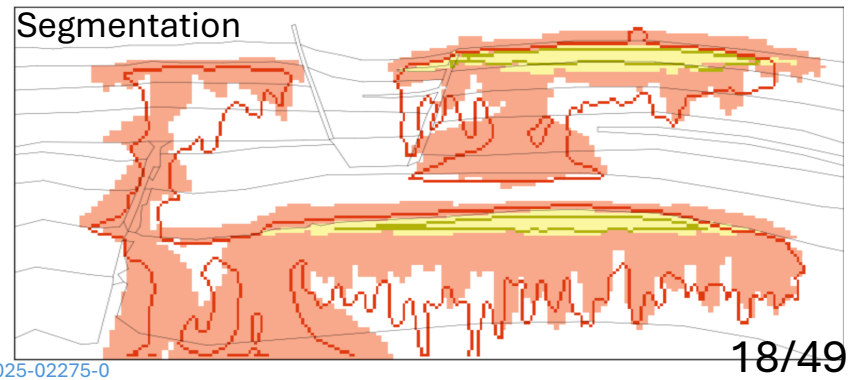
“Ruff” for linter and code formatter (from 2026)

pofff : An open-source image-based history-matching framework for the FluidFlower benchmark study using OPM Flow

https://github.com/cssr-tools/pofff



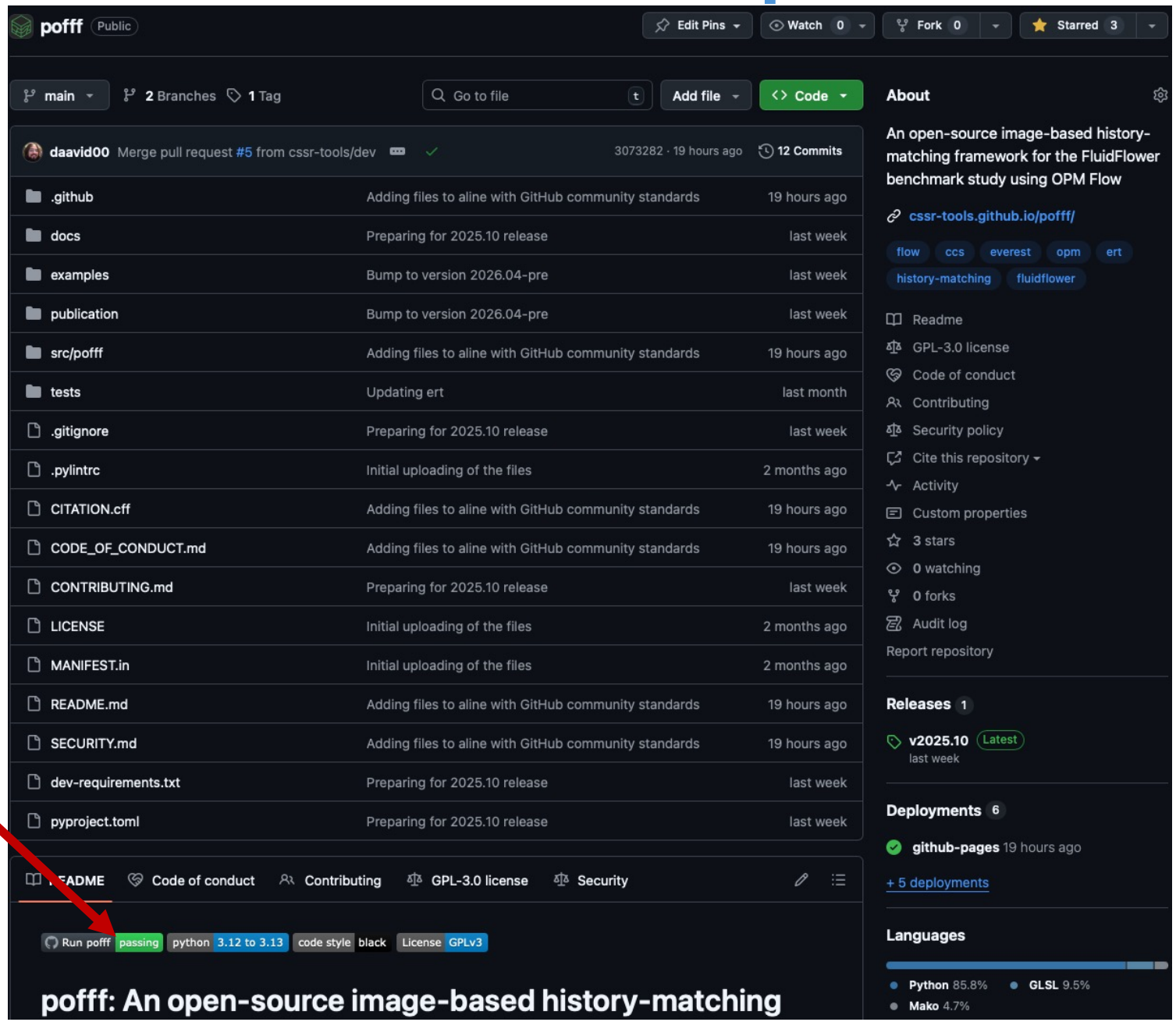


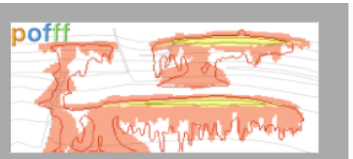


Landa-Marbán, D. et al. 2026. Performance of an Open-Source Image-Based History Matching Framework for CO2 Storage. Transp Porous Med 153, 21. https://doi.org/10.1007/s11242-025-02275-0

https://github.com/cssr-tools/pofff

CI.yml helps users to know if something is broken and additional details on the installation





Search docs

Concept

User-friendly open-source image-based history-matching framework for the FluidFlower benchmark study using OPM Flow.

Overview

The current implementation supports the following executable with the argument options:

```
pofff -i name_of_input_file.toml
```

where

-i

The base name of the toml configuration file, ('input.toml' by default).

-o

The base name of the output folder ('output' by default).

-f

Use 'all' to generate all benchmark figures, 'basic' to not generate the Wasserstain distance plot (it is slow), and 'none' for no figures ('basic' by default).

-m

Run a 'single' simulation, generate only the input 'files', generate only the csv 'data', 'everest', 'ert', 'fair', or 'none' ('none' is useful to generate only figures in combination to '-f') ('single' by default).

-t

Times in hours separated by commas to evaluate the metrics ('0.25' by default).

-e

Experimental data to history match, valid options are C1 to C5 ('C2' by default).

-s

The minimum saturation above which gaseous CO2 is considered for the segmentation ('1e-2' by default).

-c

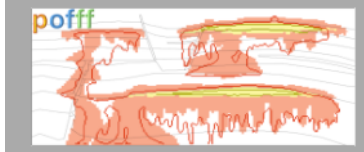
The minimum concentration above which CO2 is considered to be dissolved for the segmentation ('1e-1' by default).

-u

Use the precomputed Wasserstein distance matrix values for minimum concentration of 1e-1 and 5e-2 (minimum saturation of 1e-2) to speed up the computations ('1' by default; set to '0' to compute all).

```
# pofff -i everest_iter_3.toml -o everest_iter_3 -m everest -t 24,48,72,96,120
# Set the full path to the flow executable and flags
flow="flow --newton-min-iterations=1 --solver-max-restarts=20 --time-step-control=newtoniterationcount --solver-growth-factor=1.6 --linear-solver=cpr_trueimpes

# Set the model parameters
grid="corner-point" # Type of grid (cartesian, tensor, or corner-point)
thickness="final" # Thickness maps (measured 'initial', 'final', or a real positive value)
mult_thickness=1 # Thickness multiplier (a real positive value)
x=[140] # If cartesian, number of x cells [-]; otherwise, variable array of x-refinement
z=[7,5,5,5,5,5,5,8,10,9,5] # If cartesian, number of z cells [-]; if tensor, variable array of z-refinement; if corner-point, fix array of z-refinement (18 entries
temperature=[20, 20] # Temperature bottom and top rig [C]
pressure=104900 # Pressure at the datum [Pa]
diffusion=[1e-9, 2e-8] # Diffusion (in liquid and gas) [m^2/s]
sources=[[0.9, 0.3], [1.7, 0.7]] # Source positions: x and z coordinates [m], source 1 to 2

# Schedule: 1) injection time [s], 2) time step size to write results [s], 3) injection rate [kg/s] (source1), and 4) injection rate [kg/s] (source2)
inj=[[8100, 8100, 3E-7, 0, '1e-2 3e-4 1e-20 1e-20 1.6 0.2 0.65 1.1'],
[10200, 10200, 3E-7, 3E-7, '1e-2 1e-4 1e-20 1e-20 1.6 0.2 0.65 1.1'],
[68100, 68100, 0, 0, '1e-2 1e-3 1e-20 1e-20 1.6 0.2 0.65 1.1'],
[345600, 86400, 0, 0, '1e-2 1e-2 1e-20 1e-20 1.6 0.2 0.65 1.1']]

# Facie Properties
facie1={"permx1"=62000,"permz1"=62000,"poro1"=0.37,"disperc1"=0,"swi1"=0.3425,"sni1"=0.25,"pen1"=1900,"nkrw1"=2,"nkrn1"=2,"npe1"=2,"thre1"=5e-2,"npnt1"=100}
facie2={"permx2"=152000,"permz2"=152000,"poro2"=0.38,"disperc2"=0,"swi2"=0.32,"sni2"=0.2,"pen2"=950,"nkrw2"=2,"nkrn2"=2,"npe2"=2,"thre2"=5e-2,"npnt2"=100}
facie3={"permx3"=428000,"permz3"=428000,"poro3"=0.40,"disperc3"=0,"swi3"=0.293,"sni3"=0.077,"pen3"=185,"nkrw3"=2,"nkrn3"=2,"npe3"=2,"thre3"=5e-2,"npnt3"=100}
facie4={"permx4"=1120000,"permz4"=1120000,"poro4"=0.39,"disperc4"=0,"swi4"=0.275,"sni4"=0.06,"pen4"=175,"nkrw4"=2,"nkrn4"=2,"npe4"=2,"thre4"=5e-2,"npnt4"=100}
facie5={"permx5"=2120000,"permz5"=2120000,"poro5"=0.39,"disperc5"=0,"swi5"=0.23,"sni5"=0.01,"pen5"=170,"nkrw5"=2,"nkrn5"=2,"npe5"=2,"thre5"=5e-2,"npnt5"=100}
facie6={"permx6"=2500000,"permz6"=2500000,"poro6"=0.42,"disperc6"=0,"swi6"=0.1742,"sni6"=0,"pen6"=163.2,"nkrw6"=2,"nkrn6"=2,"npe6"=2,"thre6"=5e-2,"npnt6"=100}

# Set the saturation functions
krw="(max(0, (sw - swi) / (1 - swi))) ** nkrw"            #Wetting rel perm saturation function [-]
krn="(max(0, (1 - sw - sni) / (1 - sni))) ** nkrn"        #Non-wetting rel perm saturation function [-]
cap="pen * ((sw-swi) / (1-swi)) ** (-(1.0 / npen))"       #Capillary pressure saturation function [Pa]

# Everest
min_realizations_success = 0 # Minimum number of simulations that must have succeeded for the simulation to be regarded as a success
max_function_evaluations = 200000 # Maximum number of simulations
rng = 7 # Set a specific seed for reproducibility; a value of 0 means no seed (rng for everest and random_seed for ert)
cores = 50 # Maximum number of simulations running in parallel
maxtime = 3600 # Maximum runtime in seconds of a realization; a value of 0 means unlimited runtime
delete = true # Delete large files?
monotonic = true # Only consider monotonic values, e.g, increasing entry pressure with decreasing sand size
popsize = 200 # Population size
perm5 = [2200e3, 1120e3, 2500e3, 20]
swi5 =  [  0.23,   0.17,   0.27,  4]
sni5 =  [  0.01,   0.01,   0.06,  6]
pen5 =  [   170,    164,    175, 10]
```

Example of TOML configuration file



Search docs

Installation

The following steps work installing the dependencies in Linux via apt-get or in macOS using brew or macports. While using package managers such as Anaconda, Miniforge, or Mamba might work, these are not tested. We will update the documentation when Python 3.14 is supported (e.g., the ert Python package is not yet available via pip install in Python 3.14).

Python package

To install the **pofff** executable from the development version:

```
pip install git+https://github.com/cssr-tools/pofff.git
```

If you are interested in a specific version (e.g., v2025.10) or in modifying the source code, then you can clone the repository and install the Python requirements in a virtual environment with the following commands:

```
# Clone the repo
git clone https://github.com/cssr-tools/pofff.git
# Get inside the folder
cd pofff
# For a specific version (e.g., v2025.10), or skip this step (i.e., edge version)
git checkout v2025.10
# Create virtual environment (to specific Python, python3.12 -m venv vpofff)
python3 -m venv vpofff
# Activate virtual environment
source vpofff/bin/activate
# Upgrade pip, setuptools, and wheel
pip install --upgrade pip setuptools wheel
# Install the pofff package
pip install -e .
# For contributions/testing/linting, install the dev-requirements
pip install -r dev-requirements.txt
```

Tip

Typing **git tag -l** writes all available specific versions.

OPM Flow

You also need to install:

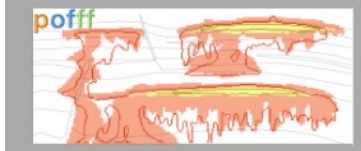
- OPM Flow (https://opm-project.org, Release 2025.10 or current master branches)

Tip

See the CI.yml script for installation of OPM Flow (binary packages), LaTeX (optional) libraries, and the pofff package in Ubuntu 24.04 and Python 3.12.

Installation is simple as:

pip install git+https://github.com/cssr-tools/pofff.git



Search docs

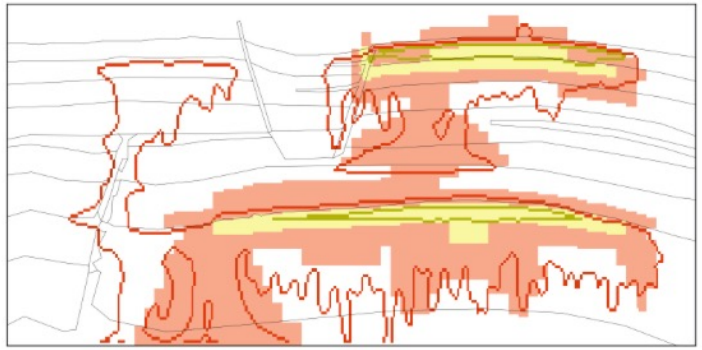
Examples

Hello world

The examples folder contains a few configuration files with low grid resolution and shorter simulation times (for initial testing of the framework). For example, by executing:

```
pofff -i single.toml -t 24,48,72
```

The following is the figure *map_24h.png*. You can compare your example results to this figure to evaluate if your example ran correctly:

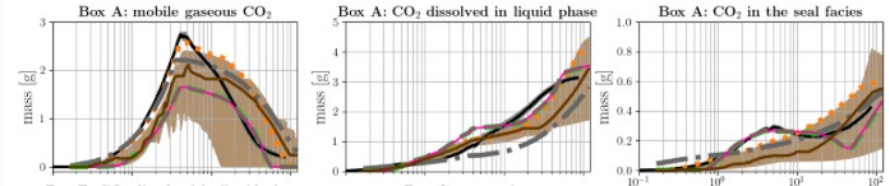


Adding your results

The publication folder contains the configuration files used for the results in the pofff paper (see publication for details in the steps to reproduce the figures in the paper). For example, running inside that folder:

```
pofff -i results.toml -o YOURS -m single -t 24,48,72,96,120 -f all
```

generates the figures in the paper in addition to the new simulation labeled as the name of the outpur folder ("YOURS" in this case), e.g., for the compare_all_time_series.png:



Having a simple Hello world in repositories help for quick testing of the installation.

Observations

Five different experimental runs

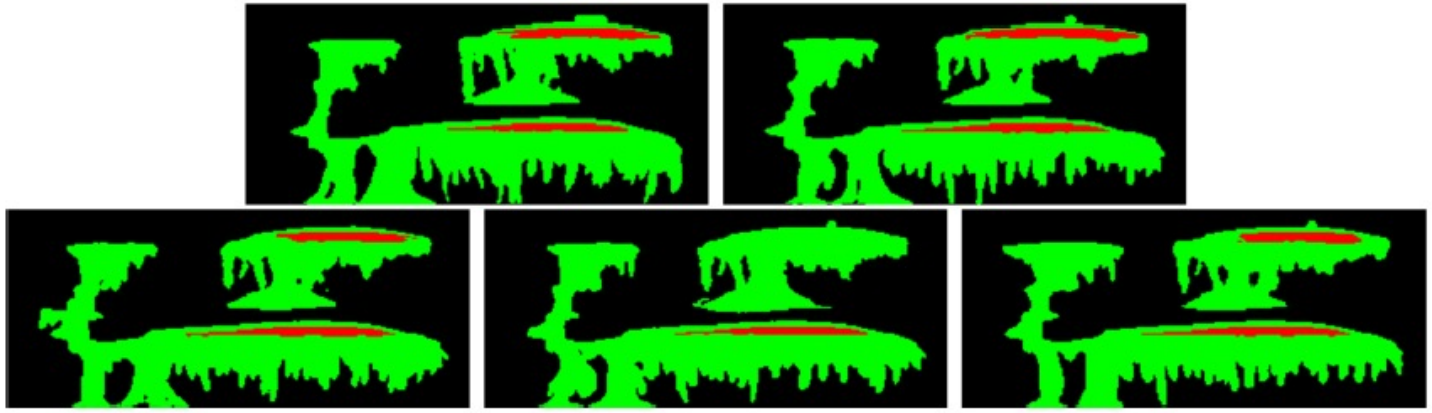


Fig. 25 Segmentation data after 24 h for five experimental runs. Black, green and red indicate pure water, water with dissolved CO_2 and gas, respectively

Difficult to quantify the CO_2 mass from the images, only distinguished gas and dissolved CO_2

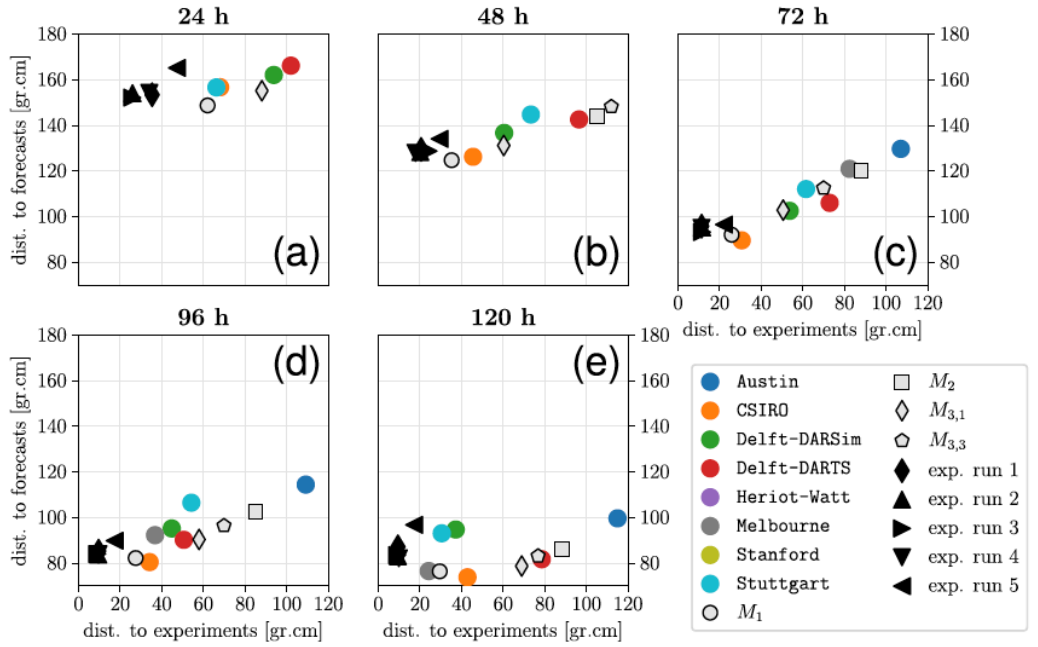
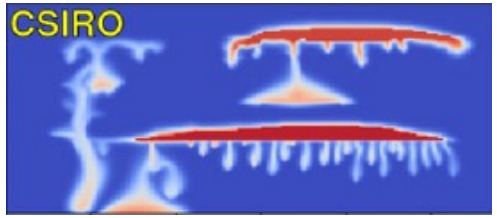
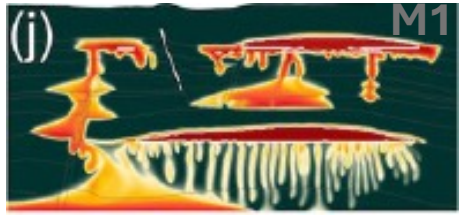


Fig. 18 Wasserstein distances to experiments and forecasts (simulations). Colored circles show forecasts by IBS groups, and results with calibrated Models 1–3 are presented with light gray markers. In each subplot, the vertical axis shows the mean distance between a given datapoint and the forecasts (considering the IBS participants only), while the horizontal axis shows the mean distance between a given datapoint and the experiments. Markers not present fall outside of the axes limits. **a** 24 h. **b** 48 h. **c** 72 h. **d** 96 h. **e** 120 h

Goal: finding model parameters that results in the OPM simulations to be the closest to the experimental data with respect to **CSIRO** and **M1** results.

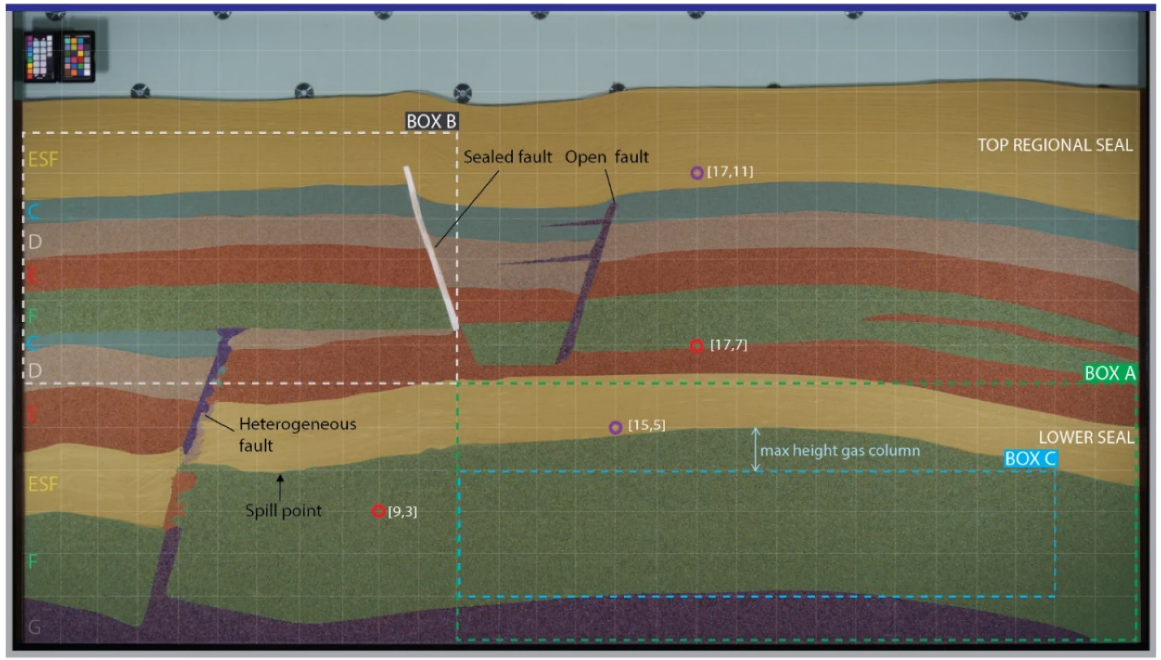




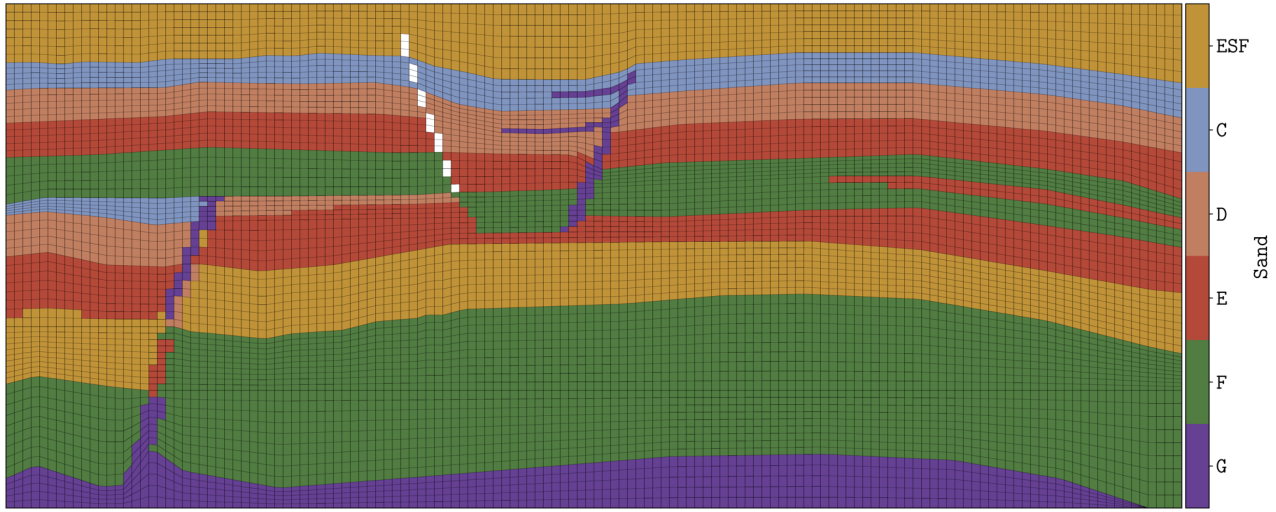
? OPM

Flemish, B., et al., 2024. The FluidFlower Validation Benchmark Study for the Storage of CO2 . Transp. Porous Media. https:// doi. org/ 10.1007/ s11242- 023- 01977-78
Saló.-Salgado, L., et al., 2024. Direct Comparison of Numerical Simulations and Experiments of CO2 Injection and Migration in Geologic Media: Value of Local Data and Forecasting Capability. Transp. Porous Media. https:// doi.org/ 10.1007/ s11242- 023- 01972-y

Spatial characterization



Fernø, M.A. et al. Room-Scale CO2 Injections in a Physical Reservoir Model with Faults. Transp. Porous Media 151(5), 913–937 (2024). https://doi.org/10.1007/s11242-023-02013-4

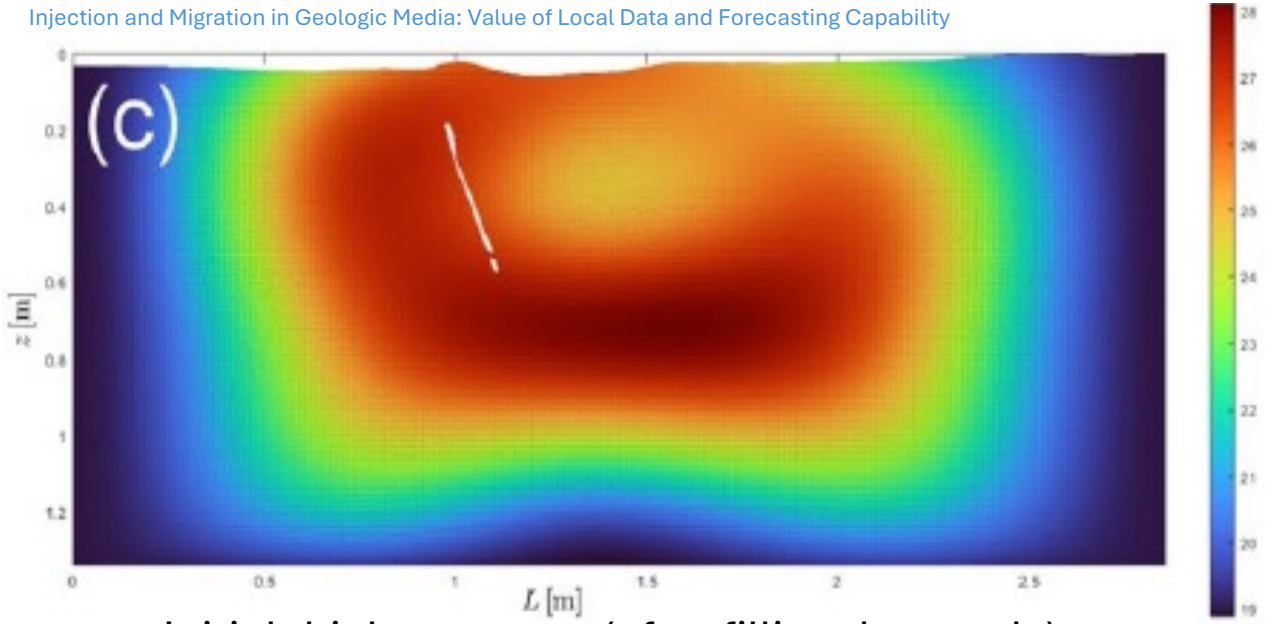


Landa-Marbán, D. et al. 2026. Performance of an Open-Source Image-Based History Matching Framework for CO2 Storage. Transp Porous Med 153, 21. https://doi.org/10.1007/s11242-025-02275-0

- It is key to have a grid following the main horizons
- The grid has a size of 2cm x 2cm(ish)
- Using eight CPUs for OPM Flow in a Mac, the simulation time is **LESS THAN TWO MINUTES!!!!** (good since HM requires running many simulations).

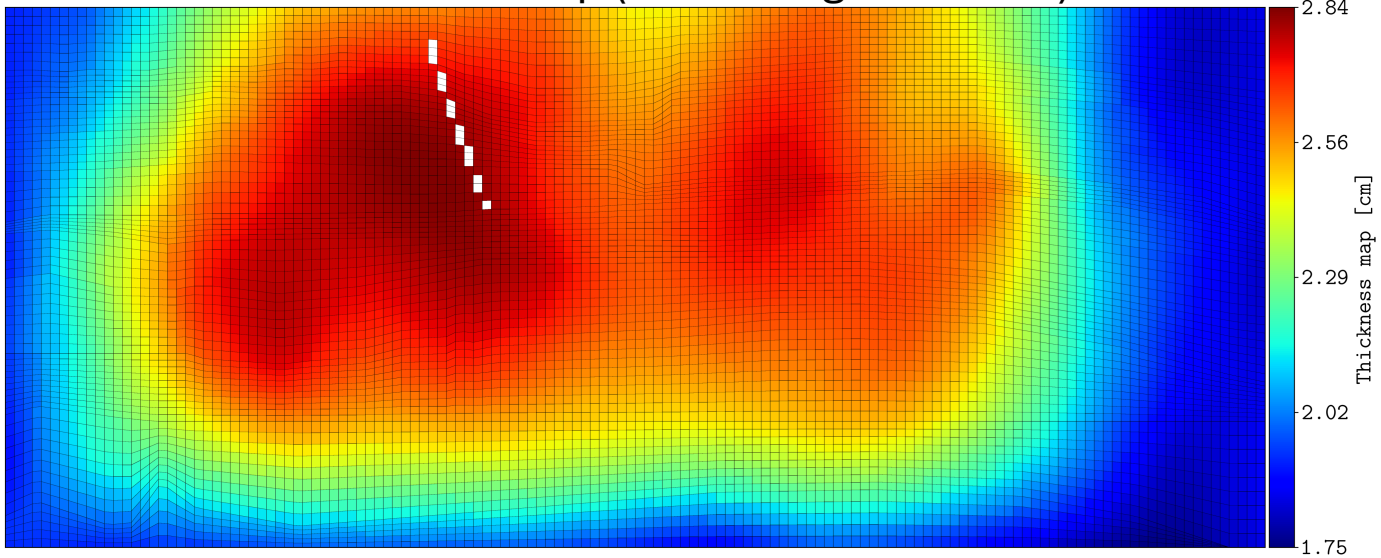
Thickness map

Saló-Salgado et al. 2024. Direct Comparison of Numerical Simulations and Experiments of CO2 Injection and Migration in Geologic Media: Value of Local Data and Forecasting Capability



Initial thickness map (after filling the sands)

Nordbotten, J. M., Fernø, M., Flemisch, B., Juanes, R., & Jørgensen, M. (2022). Final Benchmark Description: FluidFlower International Benchmark Study. Zenodo. https://doi.org/10.5281/zenodo.6807102



Final thickness map (at the end of all experiment)

- Large uncertainty in the thickness map, i.e., pore volume.

Landa-Marbán, D. et al. 2026. Performance of an Open-Source Image-Based History Matching Framework for CO2 Storage. Transp Porous Med 153, 21. https://doi.org/10.1007/s11242-025-02275-0

History matching tools

https://everest.readthedocs.io/en/latest/theory.html

README | Code of conduct | Contributing | GPL-3.0 license | Security

+ 66 contributors

Deployments 500+

github-pages 4 months ago

+ more deployments

Languages

Python 99.1% Other 0.9%

ert

Build and test passing | python 3.11 | 3.12 | 3.13 | Style passing | Type checking passing | codecov 91% | License GPLv3

ert - Ensemble based Reservoir Tool - is designed for running ensembles of dynamical models such as reservoir models, in order to do sensitivity analysis and data assimilation. ert supports data assimilation using the Ensemble Smoother (ES) and Ensemble Smoother with Multiple Data Assimilation (ES-MDA).

Installation

```
pip install ert
ert --help
```

or, for the latest development version:

```
pip install git+https://github.com/equinor/ert.git@main
ert --help
```

For examples and help with configuration, see the ert Documentation.

Everest™

EVEREST

The primary goal of the Everest tool is to find *optimal* well planning and production strategies by utilizing an ensemble of reservoir models (e.g., an ensemble of geologically-consistent models). This will enable robust decisions about drilling schedule and well placement, in order to achieve results of significant practical value.

```
pip install ert[everest]
```

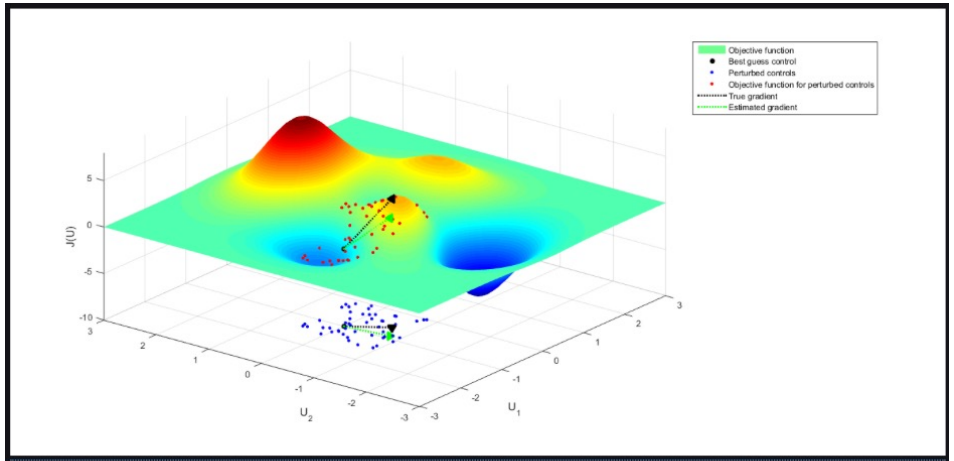


Fig. 1 The ensemble-based optimization method (EnOpt).

Results depending on the initial guess and random seed, i.e., luck is a factor.

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html

scipy.optimize.

differential_evolution

```
differential_evolution(func, bounds, args=(),
strategy='best1bin', maxiter=1000, popsize=15, tol=0.01,
mutation=(0.5, 1), recombination=0.7, rng=None,
callback=None, disp=False, polish=True,
init='latinhypercube', atol=0, updating='immediate',
workers=1, constraints=(), x0=None, *, integrality=None,
vectorized=False, seed=None)
```

[source]

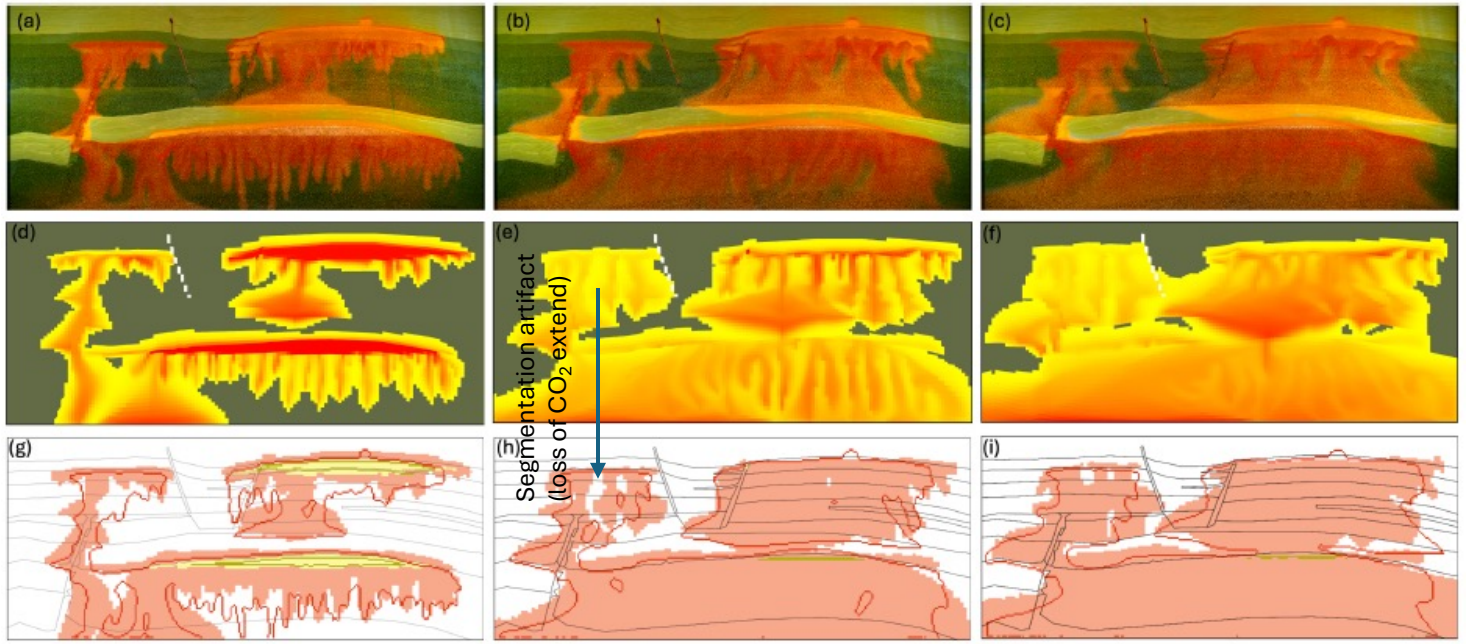
Finds the global minimum of a multivariate function.

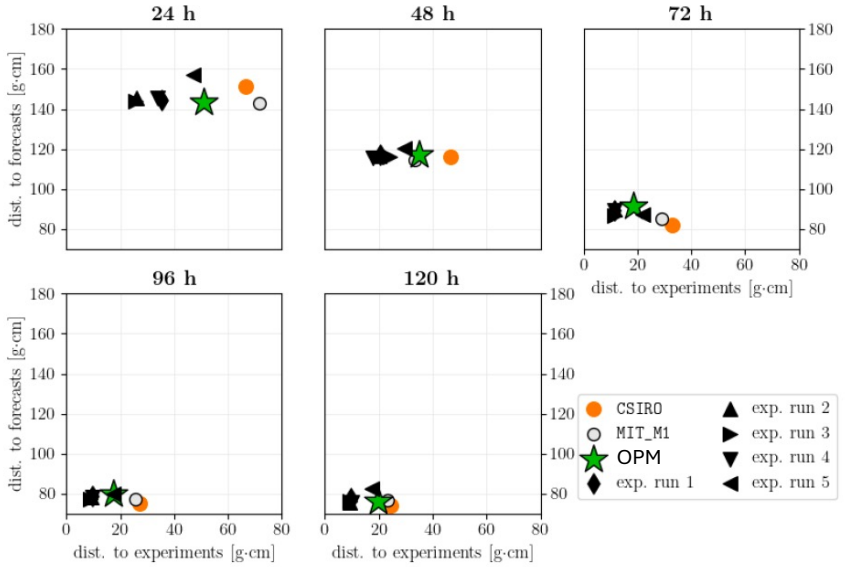
The differential evolution method [1] is stochastic in nature. It does not use gradient methods to find the minimum, and can search large areas of candidate space, but often requires larger numbers of function evaluations than conventional gradient-based techniques.

The algorithm is due to Storn and Price [2].

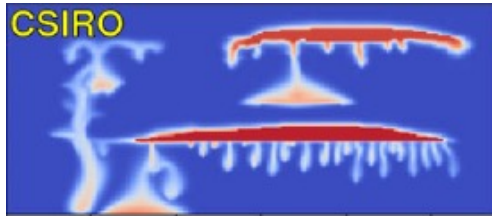
- More about EVEREST in the coming presentation of Eduardo Barros.

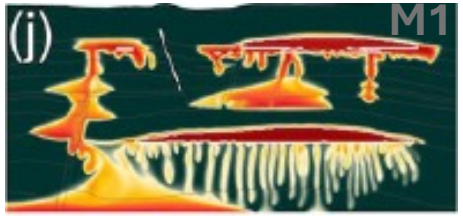
Results





- Results depend on the selected threshold to segment the simulations (1 kg/m^3 used in the benchmark).
- Ongoing research focus on generating "continuous" maps of CO_2 mass directly from experimental images, to enable more robust comparison with simulation results.
- You could give it a try to find better parameters to get closer results to the experiments using pofff.





Flemish, B., et al., 2024. The FluidFlower Validation Benchmark Study for the Storage of CO2 . Transp. Porous Media. https:// doi. org/ 10.1007/ s11242- 023- 01977-78
Saló.-Salgado, L., et al., 2024. Direct Comparison of Numerical Simulations and Experiments of CO2 Injection and Migration in Geologic Media: Value of Local Data and Forecasting Capability. Transp. Porous Media. https:// doi.org/ 10.1007/ s11242- 023- 01972-y
Landa-Marbán, D. et al., 2026. Performance of an Open-Source Image-Based History Matching Framework for CO2 Storage. Transp Porous Med 153, 21. https://doi.org/10.1007/s11242-025-02275-0

Contents lists available at ScienceDirect

International Journal of Greenhouse Gas Control

journal homepage: www.elsevier.com/locate/ijggc

Benchmarking CO_2 storage simulations: Results from the 11th Society of Petroleum Engineers Comparative Solution Project

Jan M. Nordbotten [a,b,*], Martin A. Fernø [b,c], Bernd Flemisch [d], Anthony R. Kovscek [e], Knut-Andreas Lie [f], Jakub W. Both [a], Olav Møyner [f], Tor Harald Sandve [b], Etienne Ahusborde [g], Sebastian Bauer [h], Zhangxing Chen [i,j], Holger Class [d], Chaojie Di [i], Didier Ding [l], David Element [k], Eric Flauraud [l], Jacques Franc [e], Firdovsi Gasanzade [h], Yousef Ghomian [m], Marie Ann Giddins [n], Christopher Green [o], Bruno R.B. Fernandes [p], George Hadjisotiriou [q], Glenn Hammond [r], Hai Huang [s], Dickson Kachuma [t], Michel Kern [u,v], Timo Koch [d,w], Prasanna Krishnamurthy [x], Kjetil Olsen Lye [f], David Landa-Marbán [b], Michael Nole [r,y], Paolo Orsini [z], Nicolas Ruby [m], Pablo Salinas [z], Mohammad Sayyafzadeh [o], Jakob Torben [f], Adam Turner [k], Denis V. Voskov [e,q], Kai Wendel [d], AbdAllah A. Youssef [aa]

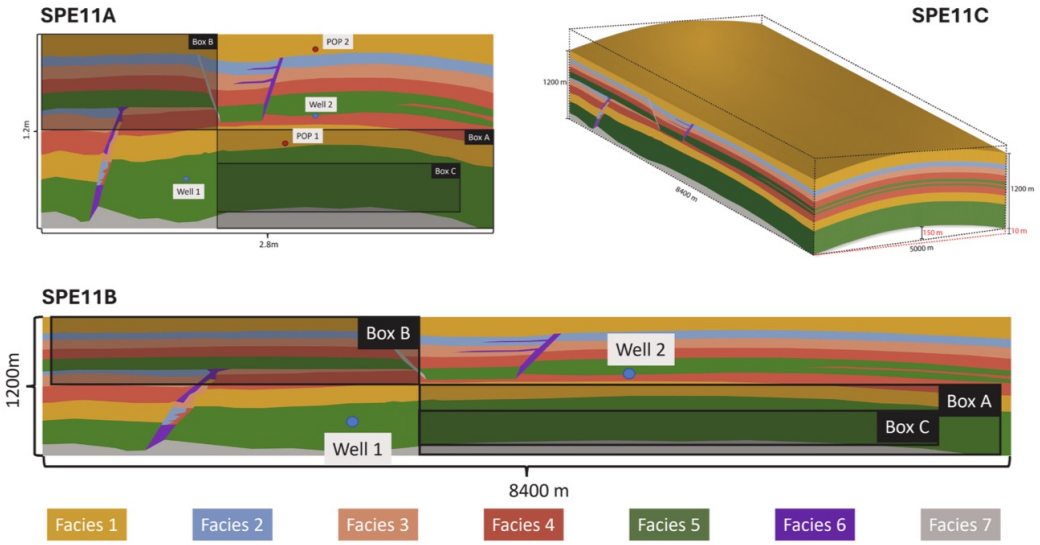


Fig. 2. The SPE11 subcases are derived from the same geometry but are set respectively as a 2D experiment at the laboratory scale and atmospheric conditions (SPE11A), as a 2D transect at field scale and reservoir conditions (SPE11B), and as a synthetic 3D storage reservoir at field scale and reservoir conditions (SPE11C).

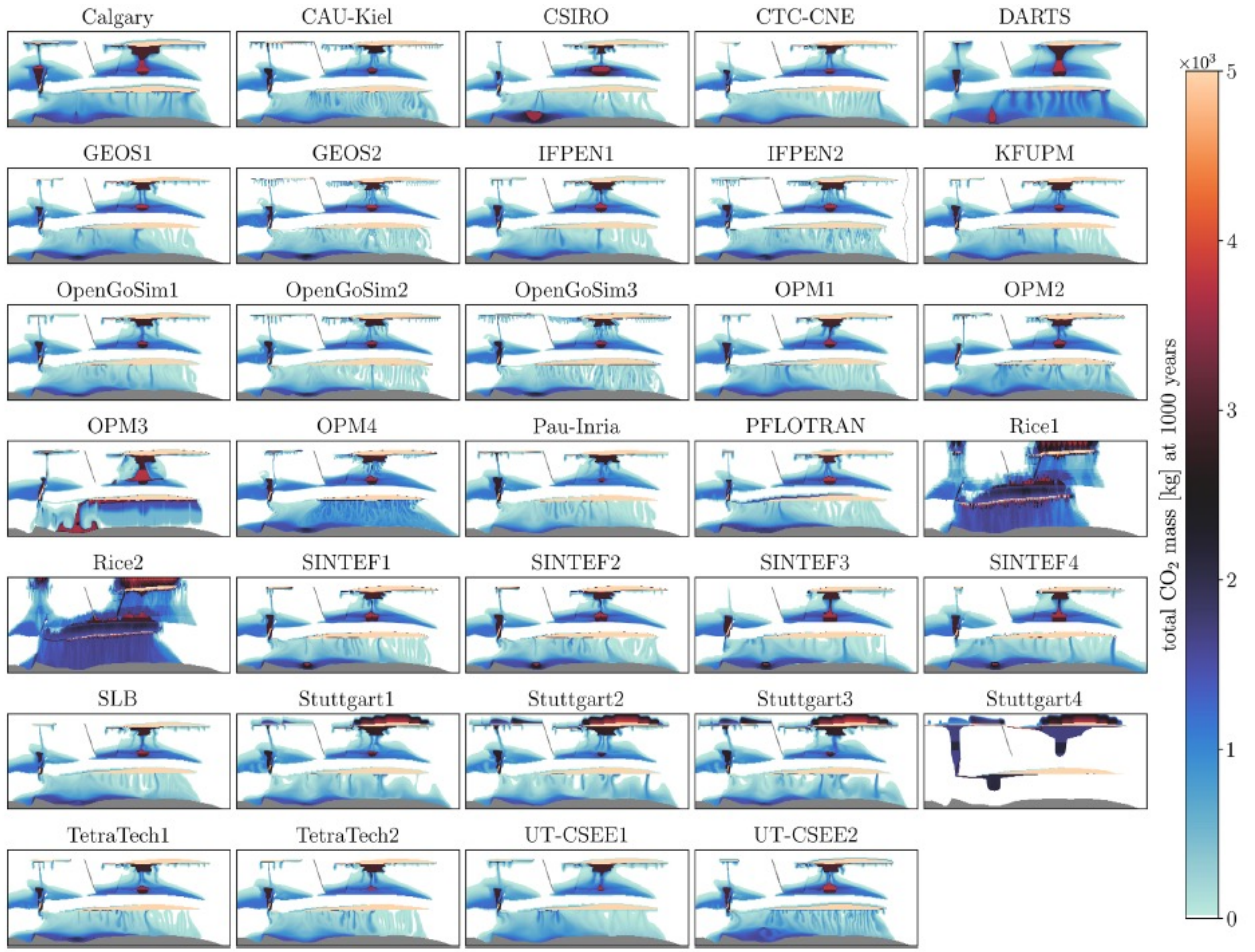


Fig. 5. An overview of the SPE11B submissions, in terms of distribution of CO_2 at the end of the simulation period. Note that the figures are based on the submitted data, thus for the higher resolution simulations, some details are lost relative to the underlying simulation data. The grey zone at the bottom of each figure is the impermeable facies.

The OPM team submitted four results for SPE11A, four for SPE11B, and four for SPE11C, with all required reporting data i.e., including performance-spatial maps (the only missing column is the a posteori error estimate).

All submitted data can be generated using pyopmspe11 (a FAIR submission).

pyopmspe11: A Python framework using OPM Flow for the SPE11 benchmark project

Python GLSL Mako Submitted 28 June 2024 • Published 30 January 2025

JOSS
The Journal of Open Source Software

pyopmspe11: A Python framework using OPM Flow for the SPE11 benchmark project

David Landa-Marbán [1¶] **and Tor H. Sandve** [1]

1 NORCE Norwegian Research Centre AS, Bergen, Norway ¶ Corresponding author

DOI: 10.21105/joss.07357

Software
- Review
- Repository
- Archive

Editor: Rachel Wegener
Reviewers:
- @MatthewFlamm
- @gassmoeller

Submitted: 28 June 2024
Published: 30 January 2025

Summary

The imperative to achieve climate change goals and the increasing worldwide demand for energy have made geological carbon storage (GCS) technology more relevant today. Since utilizing computational models is essential for planning large-scale GCS projects, it is crucial to benchmark simulation tools to enhance confidence in their results. Inspired by a recent validation study for laboratory-scale CO_2 storage (Flemisch et al., 2024), a new comparative solution project (CSP) was launched to simulate both lab- and field-scale CO_2 storage (Nordbotten et al., 2024). This project is called the 11th Society of Petroleum Engineers CSP, and we refer to it as the SPE11 benchmark. The main objective for the SPE11 benchmark is to provide a common platform and reference case for numerical simulation of GCS. A community effort was run by the "Early Access Team" to create utility scripts and input files for popular simulators to make participation more accessible. As part of the "Early Access Team", we have developed and made open the `pyopmspe11` tool which facilitates reproducible solutions to the SPE11 benchmark. This tool serves as a common starting point for developing and testing new GCS simulation technology. Due to its user-friendly functionality (e.g., generation of different types of grids at different grid resolutions, flexibility to choose different rock and fluid properties, flexibility to define well/source locations and schedule for operations), it is expected that its impact will extend far beyond the initial benchmark study (e.g., studies focusing on grid refinement, upscaling/coarsening approaches, numerical solvers, optimization/history matching techniques).

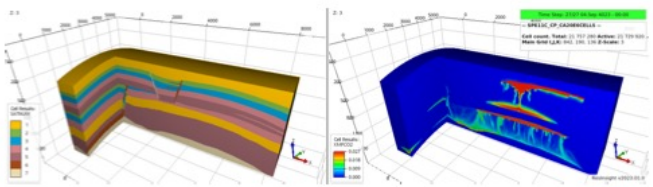


Figure 1: Generated model by the configuration file `spe11c_cp_ca20e6cells.txt` in the examples folder (the model corresponds to the SPE11C Case using a corner-point grid with 21729920 active cells).

Software repository
Paper review
Download paper
Software archive

Review
Editor: @rwegener2 (all papers)
Reviewers: @MatthewFlamm (all reviews), @gassmoeller (all reviews)

Authors
David Landa-Marbán (0000-0002-3343-1005), Tor H. Sandve (0000-0002-3267-8276)

Citation
Landa-Marbán et al., (2025). pyopmspe11: A Python framework using OPM Flow for the SPE11 benchmark project. Journal of Open Source Software, 10(105), 7357, https://doi.org/10.21105/joss.07357
Copy citation string · Copy BibTeX

Tags
CO$_2$ OPM Flow CSP11 SPE11

Altmetrics
1

Markdown badge
JOSS 10.21105/joss.07357

License
Authors of JOSS papers retain copyright.

OPM / pyopmspe11 Public

Notifications | Fork 13 | Star 18

Code | Issues 7 | Pull requests | Actions | Projects | Security and quality | Insights

main | 1 Branch | 4 Tags | Go to file | Code

daavid00 Merge pull request #145 from daavid00/dev — 28990f9 · 2 days ago — 264 Commits

.github	Modernizing data.py and convergence.py	2 days ago
benchmark	Examples using plopm to plot benchmark data	3 months ago
convergence	Modernizing data.py and convergence.py	2 days ago
dockers	updating the dockerfiles	last year
docs	Modernizing data.py and convergence.py	2 days ago
examples	Examples using plopm to plot benchmark data	3 months ago
paper	Fix ISSNs	last year
src/pyopmspe11	Modernizing data.py and convergence.py	2 days ago
tests	Improving test format and adding ruff	4 days ago
.gitignore	Modernizing plotting.py and retiring -s (hidding pyt...	4 days ago
CITATION.cff	JOSS paper Markdown and citation	last year
CODE_OF_CONDUCT.md	Adding files to aline with GitHub community standa...	4 months ago
CONTRIBUTING.md	Modernizing data.py and convergence.py	2 days ago
LICENSE	Adding the source files	3 years ago
MANIFEST.in	Retiring resdata, introducing pofff, and OPM Flow ...	5 months ago
README.md	Modernizing data.py and convergence.py	2 days ago
SECURITY.md	Adding files to aline with GitHub community standa...	4 months ago
dev-requirements.txt	Improving test format and adding ruff	4 days ago
pyproject.toml	Bump to 2026.04-pre	5 months ago

README | Code of conduct | Contributing | MIT license | Security

Run pyopmspe11 passing | python 3.11 to 3.14 | code style black | license MIT | JOSS 10.21105/joss.07357

pyopmspe11: A Python framework using OPM Flow for the SPE11 benchmark project

About

A Python framework using OPM Flow for the SPE11 benchmark project

OPM.github.io/pyopmspe11/

flow | co2 | opm | csp11 | spe11

Readme
MIT license
Code of conduct
Contributing
Security policy
Cite this repository
Activity
Custom properties
18 stars
3 watching
13 forks
Report repository

Releases 4

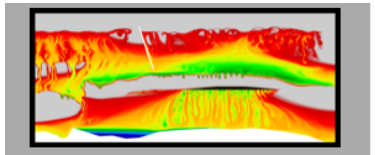
v2025.10 Latest
on Nov 3, 2025

+ 3 releases

Contributors 7

Languages

Python 61.3% | GLSL 19.1% | Mako 14.6% | TeX 4.6% | Dockerfile 0.4%



Search docs

- Introduction
- Installation
- Configuration file
- Examples
- Benchmark
- Convergence
  - Full domain
  - Lower domain
- pyopmspe11 Python API
- Output folder
- Contributing
- Related
- About pyopmspe11

Convergence

This section describes how to run the cases in the paper:

- Landa-Marbán, D., Lie, K.-A., Lye, K. O., Møyner, O., Rasmussen, A. F., and T. H. Sandve (2026). Exploring Convergence and Its Limits in Case B of the 11th SPE Comparative Solution Project. SPE J. https://doi.org/10.2118/231853-PA.

To run the cases in the terminal, inside the convergence folder:

```
python3 convergence.py
```

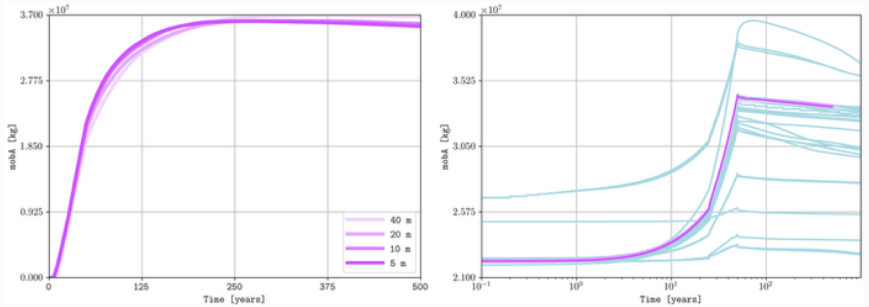
plopm is used to generate the figures, which can be installed by executing in the terminal:

```
pip install git+https://github.com/cssr-tools/plopm.git
```

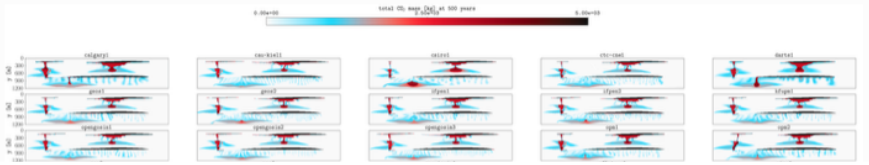
The cases are run with 8 cpus, then you can remove or increase this in the spe11b.mako template.

Full domain

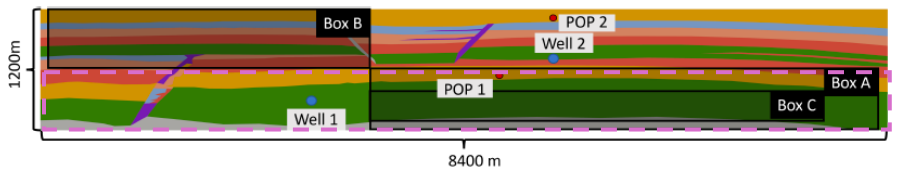
These cases are run in the grid sizes of 40, 20, 10, and 5 meters. To simulate additional refinements, edit the 'sizes' variable in convergence.py.



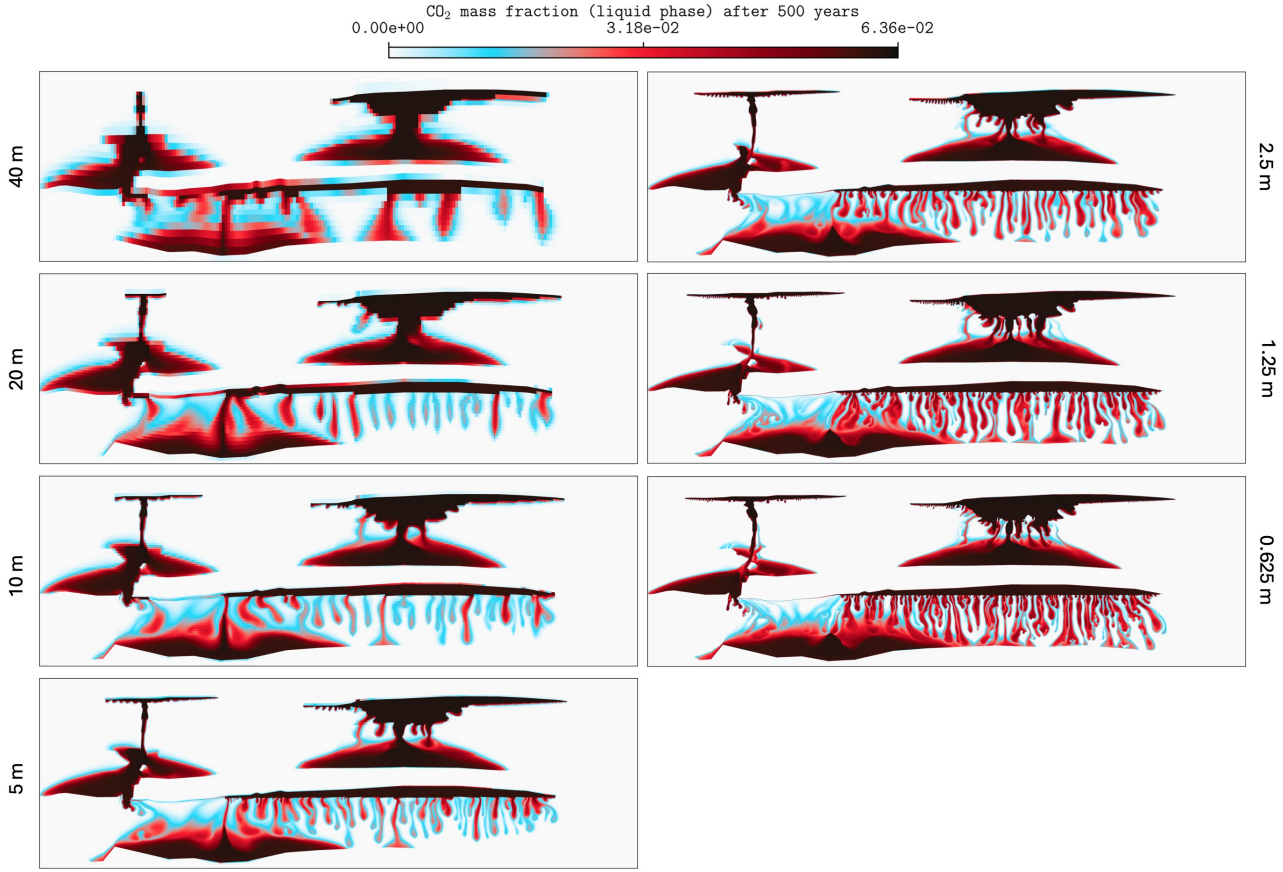
(Left) mobile CO2 in box A for the four different refinements and (right) adding all participants (time axis in log scale).



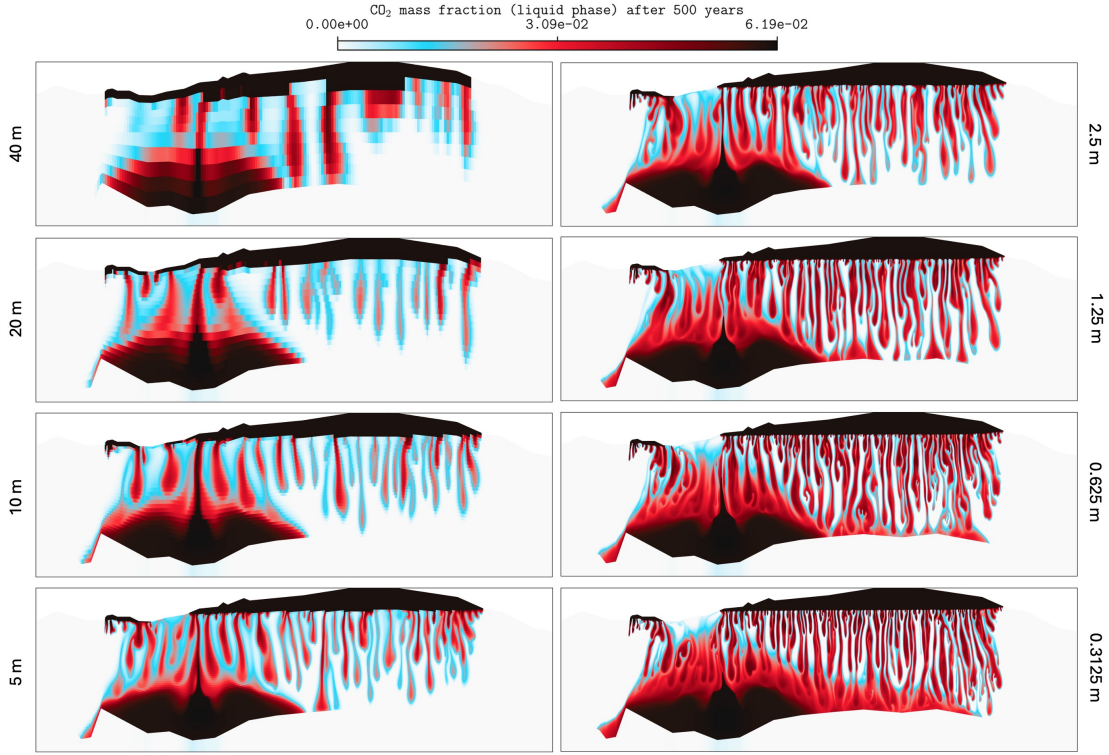
Convergence study



Full domain



Localized domain

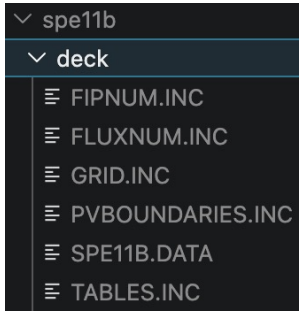


* New flag in pyopmspe11 to generate this lower part "-n lower"

Landa-Marbán et al. 2026. Exploring Convergence and Its Limits in Case B of the 11th SPE Comparative Solution Project. SPE J. 31. https://doi.org/10.2118/231853-PA

Improvements since the benchmark

- Remove OPM Flow dependency to only generate the deck files.
- Decreasing the number and size of generating files, by using '*' for repeated values and EQUALREG




```
FIPNUM
10 7 28*6 52*7 2*10 7
```

```
EQUALREG
PORO     1.000000E-01 1 F /
PORO     2.000000E-01 2 F /
PORO     2.000000E-01 3 F /
PORO     2.000000E-01 4 F /
PORO     2.500000E-01 5 F /
PORO     3.500000E-01 6 F /
PORO     1.000000E-06 7 F /
PERMX    1.013200E-01 1 F /
```

- Option to generate all files in the output folder using the "-f 0" flag, i.e., no deck/flow/data/figures subfolders.
- Using alive_progress for the terminal status.

```
Deck: Generating the input files, please wait.
|                                    | 4814/4814 [100%] in 0.2s (21829.07/s)
```

- Assisted by Copilot, improving the code in visualization.py and data.py (maximum score 5)

Comparison with Previous Version

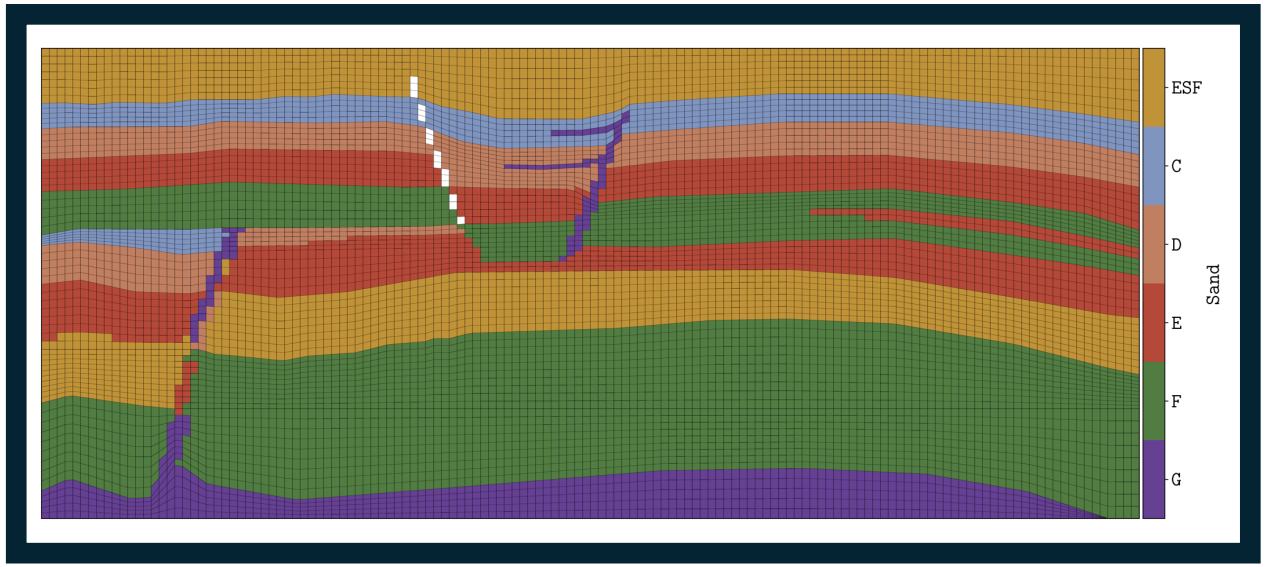
Dimension	Before	Now
Readability	2.0	3.5
Maintainability	2.0	3.5
Correctness	4.0	4.0
Complexity	2.0	2.5
Testability	1.5	2.5
Performance	4.0	4.5
Consistency	3.0	4.0

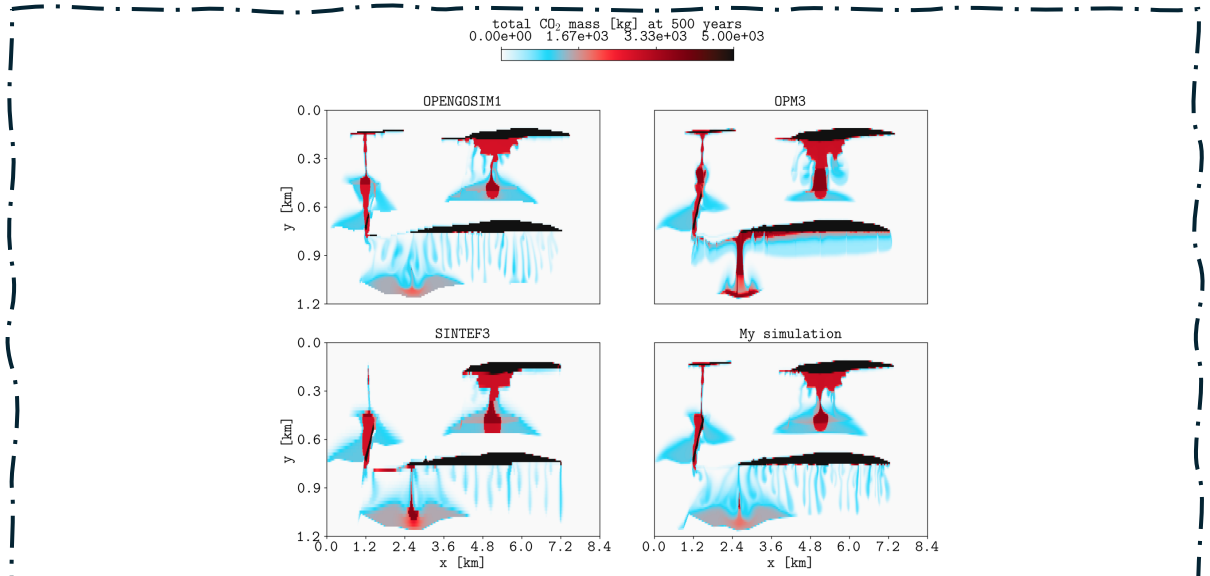
How much score your code?

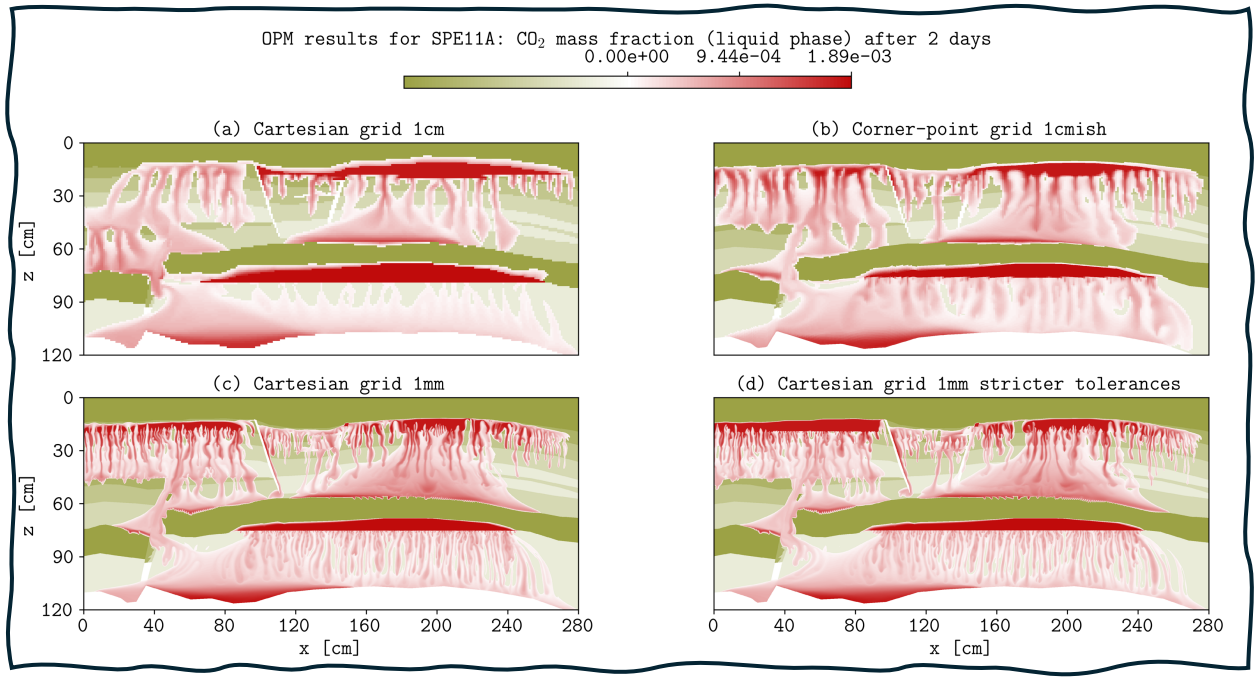
- In general, the code performance to generate the deck and postprocess the results has been significantly improved. For example, for the SPE11B case with the reporting data resolution:

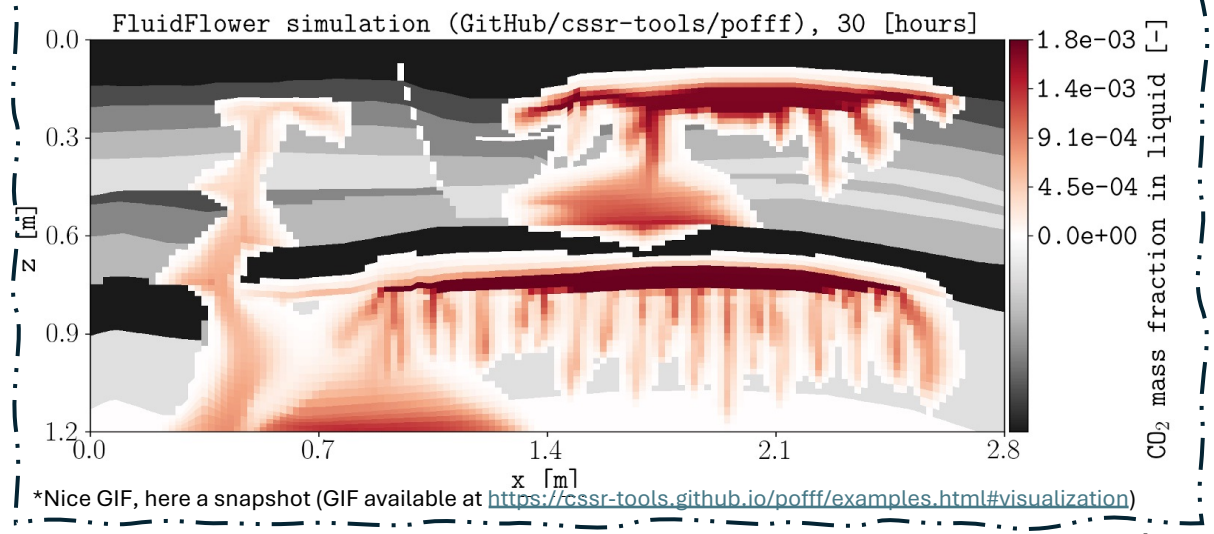
	Before	Now
Deck	1m21s	0m28s
Data	24m22s	9m34s

Gallery









*Nice GIF, here a snapshot (GIF available at https://cssr-tools.github.io/pofff/examples.html#visualization)

Gallery (behind the scenes, towards FAIR)

curl -O https://github.com/cssr-tools/pofff/blob/main/examples/input.toml

pofff -i input.toml -o figure3 -f none

plopm -i figure3/FIGURE3 -c '101;64;147 81;124;66 181;73;57 193;127;97 127;148;191 193;147;56' -cticks '[G, F, E, D, C, ESF]' -v 'pvtnum - 1 - satnum' -grid 'black,1e-2' -remove 1,1,0,1 -d 20,15 -o figure3 -save figure3a -f 20 -clabel 'Sand'

curl -L -O https://raw.githubusercontent.com/OPM/pyopmspe11/refs/heads/main/benchmark/spe11b/r2_cp_10mish.toml
curl -L -O https://darus.uni-stuttgart.de/api/access/datafile/375739
curl -L -O https://darus.uni-stuttgart.de/api/access/datafile/375754
curl -L -O https://darus.uni-stuttgart.de/api/access/datafile/375723

pyopmspe11 -i r2_cp_10mish.toml -o r2_cp_10mish -m all -g all -r 840,1,120 -t 5 -w 0.1

plopm -i "spe11b/opengosim1/spe11b_spatial_map_500y spe11b/opm3/spe11b_spatial_map_500y spe11b/sintef3/spe11b_spatial_map_500y PR/spe11b_spatial_map_500y" -csv "1,2,9" -subfigs 2,2 -delax 1 -suptitle 0 -z 0 -cbsfax 0.35,0.97,0.3,0.02 -yunits km -xunits km -yformat .1f -f 20 -xformat .1f -cnum 4 -xlnum 8 -cformat .2e -d 14,10 -t "OPENGOSIM1 OPM3 SINTEF3 My simulation" -clabel 'total CO$_2$ mass [kg] at 500 years' -c cet_CET_CBTL1_r -b '[0,5e3]'

curl -O https://raw.githubusercontent.com/OPM/pyopmspe11/refs/heads/main/benchmark/spe11a/r3_cp_1cmish_capmax2500Pa.txt
curl -O https://raw.githubusercontent.com/OPM/pyopmspe11/refs/heads/main/benchmark/spe11a/r3_cp_1cmish_capmax2500Pa.txt
curl -O https://raw.githubusercontent.com/OPM/pyopmspe11/refs/heads/main/benchmark/spe11a/r4_Cart_1mm_capmax2500Pa.txt
curl -O https://raw.githubusercontent.com/OPM/pyopmspe11/refs/heads/main/benchmark/spe11a/r5_Cart_1mm_capmax2500Pa_strictol.txt

pyopmspe11 -i r2_Cart_1cm_capmax2500Pa.txt -o r2_Cart_1cm_capmax2500Pa -t 1 -r 280,1,120 -w 0.16
pyopmspe11 -i r3_cp_1cmish_capmax2500Pa.txt -o r3_cp_1cmish_capmax2500Pa -t 1 -r 280,1,120 -w 0.16
pyopmspe11 -i r4_Cart_1mm_capmax2500Pa.txt -o r4_Cart_1mm_capmax2500Pa -t 1 -r 280,1,120 -w 0.16
pyopmspe11 -i r5_Cart_1mm_capmax2500Pa_strictol.txt -o r5_Cart_1mm_capmax2500Pa_strictol -t 1 -r 280,1,120 -w 0.16

plopm -v xco2l -r 53 -mask satnum -maskthr 7e-5 -i 'r2_Cart_1cm_capmax2500Pa/flow/R2_CART_1CM_CAPMAX2500PA r3_cp_1cmish_capmax2500Pa/flow/R3_CP_1CMISH_CAPMAX2500PA r4_Cart_1mm_capmax2500Pa/flow/R4_CART_1MM_CAPMAX2500PA r5_Cart_1mm_capmax2500Pa_strictol/flow/R5_CART_1MM_CAPMAX2500PA_STRICTOL' -cnum 3 -xlnum 8 -clabel 'OPM results for SPE11A: CO$_2$ mass fraction (liquid phase) after 2 days' -d 16,6.5 -t "(a) Cartesian grid 1cm (b) Corner-point grid 1cmish (c) Cartesian grid 1mm (d) Cartesian grid 1mm stricter tolerances" -yunits cm -xunits cm -yformat .0f -xformat .0f -f 16 -save figure4 -cformat .2e -suptitle 0 -subfigs 2,2 -cbsfax 0.35,0.97,0.3,0.02 -delax 1 -c '#9ca245 #9da347 #9fa44a #a0a64d #a2a750 #a3a953 #a5aa56 #a6ac59 #a8ad5c #a9af5f #abb062 #adb164 #aeb367 #b0b46a #b1b66d #b3b770 #b4b973 #b6ba76 #b7bc79 #b9bd7c #babf7f #bcc082 #bec184 #bfc387 #c1c48a #c2c68d #c4c790 #c5c993 #c7ca96 #c8cc99 #cacd9c #cbcf9f #cdd0a2 #cfd1a4 #d0d3a7 #d2d4aa #d3d6ad #d5d7b0 #d6d9b3 #d8dab6 #d9dcb9 #dbddbc #dcdfbf #dee0c1 #e0e1c4 #e1e3c7 #e3e4ca #e4e6cd #e6e7d0 #e7e9d3 #e9ead6 #eaecd9 #eceddc #edefdf #eff0e1 #f1f1e4 #f2f3e7 #f4f4ea #f5f6ed #f7f7f0 #f8f9f3 #fafaf6 #fbfcf9 #fdfdfc #ffffff #fefbfb #fdf7f7 #fcf3f3 #fbefef #faebeb #f9e7e7 #f8e3e3 #f7e0e0 #f6dcdc #f5d8d8 #f4d4d4 #f3d0d0 #f2cccc #f1c8c8 #f0c5c5 #efc1c1 #eebdbd #edb9b9 #ecb5b5 #ebb1b1 #eaadad #e9aaaa #e8a6a6 #e7a2a2 #e69e9e #e59a9a #e49696 #e39292 #e28f8f #e18b8b #e08787 #df8383 #de7f7f #dd7b7b #dc7777 #db7474 #da7070 #d96c6c #d86868 #d76464 #d66060 #d55c5c #d45959 #d35555 #d25151 #d14d4d #d04949 #cf4545 #ce4141 #cd3e3e #cc3a3a #cb3636 #ca3232 #c92e2e #c82a2a #c72626 #c62323 #c51f1f #c41b1b #c31717 #c21313 #c10f0f #c00b0b'

curl -O https://github.com/cssr-tools/pofff/blob/main/publication/appendixb.toml

sed -i.bak "s|8100, 8100|8100, 300|g" appendixb.toml
sed -i.bak "s|\[10200, 10200, 3E-7, 3E-7\]|[10200, 300, 3E-7, 3E-7],\n[3300, 300, 0, 0]|g" appendixb.toml
sed -i.bak "s|68100, 68100|64800, 3600|g" appendixb.toml
sed -i.bak "s|86400|21600|g" appendixb.toml

pofff -i appendixb.toml -o gif -m single -c '5e-2' -f none

plopm -v xco2l -i 'pofff+plopm/POFFF+PLOPM' -d 16,5 -mask satnum -m gif -dpi 1000 -f 20 -loop 1 -cformat .1e -cbsfax 0.30,0.01,0.4,0.02 -interval 437.5 -maskthr 1e-5 -tunits h -cnum 5 -clabel 'CO$_2$ mass fraction in liquid [-]' -t 'FluidFlower simulation (GitHub/cssr-tools/pofff),'

Summary and future work

- ✓ **github/cssr-tools** hosts Python-based user-friendly software solutions (e.g., pyopmspe11, pofff, plopm), aiming to follow the **FAIR** data principles.

How FAIR are your published results?

- ✓ **Continuously updating** the installation dependencies, documentation, and working examples is key for **"keeping alive"** open-source repositories.
- ➢ Assisted by Copilot, keep modernizing/improving the code in pyopmspe11 (deck generation), as well as other repositories in **github/cssr-tools**.
- ➢ Issues and contributions are more than welcome 🙂.

Funding acknowledgement

HPC Simulation Software for the Gigatonne Storage Challenge project [project 622059]

Center for Sustainable Subsurface Resources (CSSR) [project 331841]

https://github.com/cssr-tools

Centre for Sustainable Subsurface Resources

NORCE

```
cssr-tool -i input -o output
```

OPM OPEN POROUS MEDIA

OpenFOAM®

python

gmsh

equinor/ert

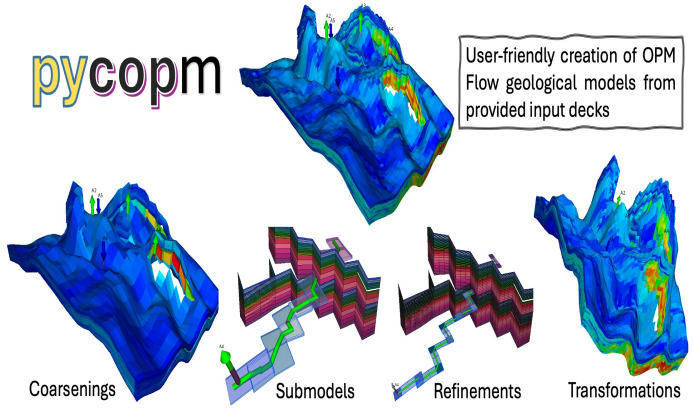
ERT - Ensemble based Reservoir Tool - is designed for running ensembles of dynamical models such as reservoir models, in...

equinor

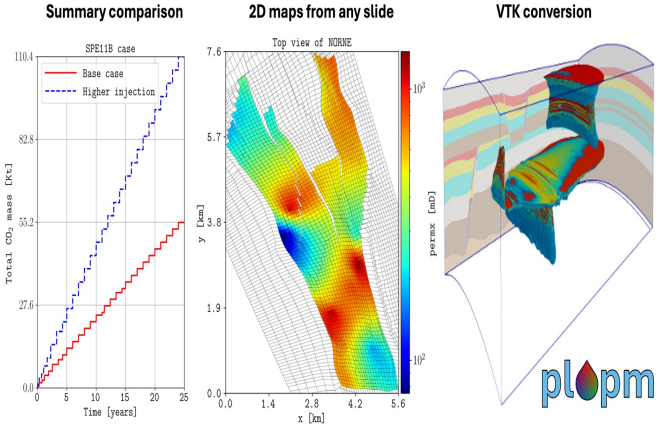
Python-Ensemble-Toolbox /PET

PET for data assimilation and optimization

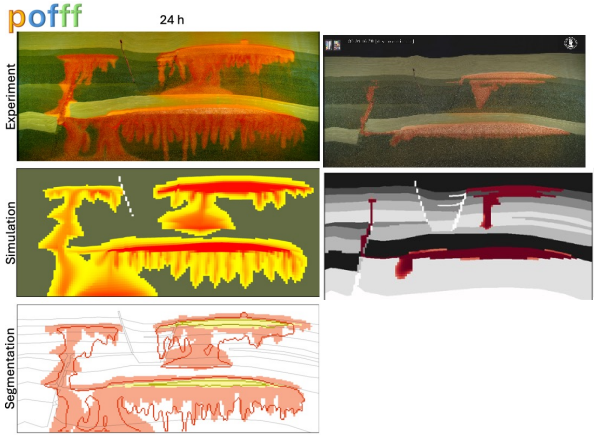
pycopm



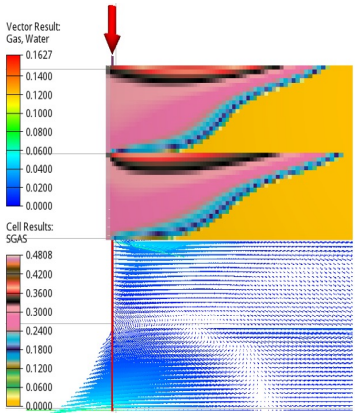
plopm



pofff



pyopmnearwell



pyopmnearwell

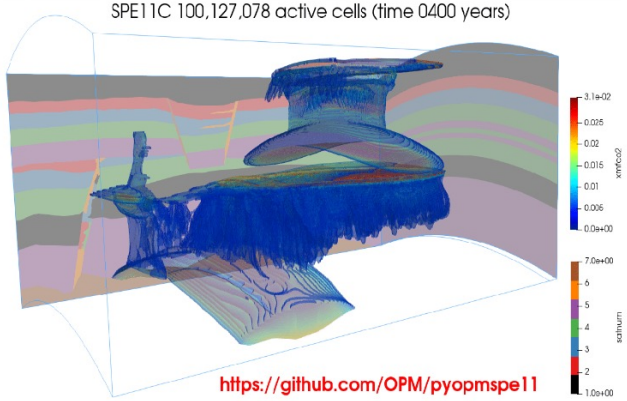
Simplified and flexible testing framework for near-well simulations via a configuration file using the OPM Flow simulator.

H_2store
CO_2store
Saltprec
CO_2EOR
Foam

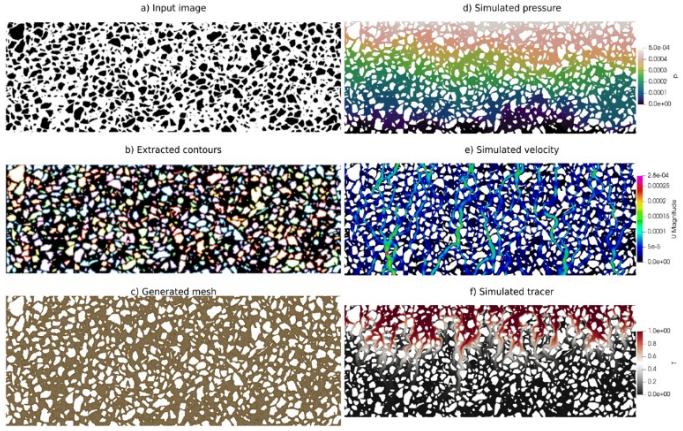
Radial grids
Core samples
Box grids

Well scheduling

pyopmspe11



pymm



expreccs