Automated inversion for porosity in shallow reservoirs using a pseudo-transient adjoint solver for non-linear hydro-mechanical equations

Georg Reuber, Lukas Holbach, Ludovic Räss

DFG Deutsche Forschungsgemeinschaft
German Research Foundation

Hydro-mechanics

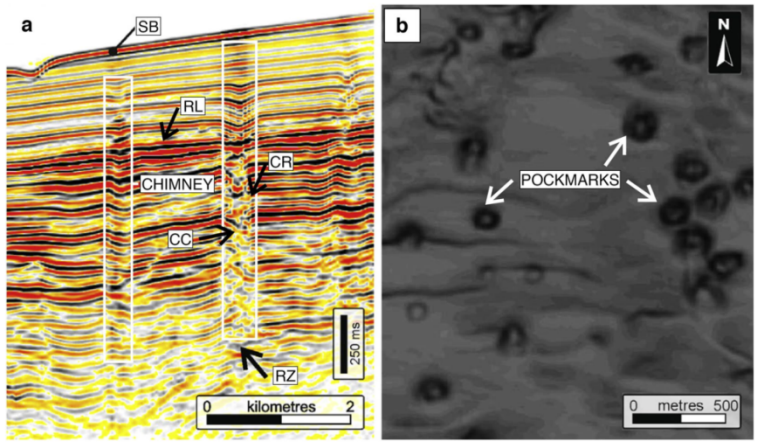
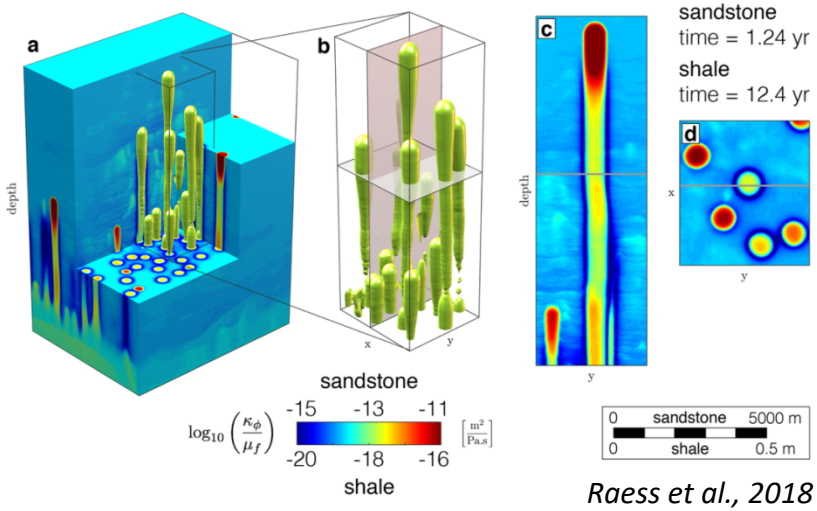


Figure 1. Seismic expression of chimneys and pockmarks. Figure modified from Cartwright and colleagues[11,22]. (**a**) Vertical seismic profile through fluid migration pathways from offshore Namibia. SB = seabed, RL = reflective horizontal sedimentary layer, CR = chimney downward-bending compacted rim, CC = chimney core, RZ = root zone and diffuse base of the chimney. (**b**) Horizontal slice through a group of chimneys displaying the typical circular craters or pockmarks.



Raess et al., 2018

I. Can we invert for porosity in reservoir systems?

- Can we automatically invert for pointwise porosity and therefore the geometry of the chimney?

II. Solving adjoint systems with explicit pseudo-transient (PT) solvers (on GPU?)

Hydro-mechanics

2-phase flow in geology is the **combination** of **Stokes** (rock) and **Darcy** flow (fluid – water/melt), while **coupling** the two with **porosity** (void space filled with fluid).

$$\operatorname{div} \mathbf{u}_s = -\frac{p - p_f}{\eta(1-\Phi)},$$

Stokes, mass conservation + coupling term

$$\operatorname{div} \mathbf{u}_f = \frac{p - p_f}{\eta(1-\Phi)},$$

Darcy, mass conservation + coupling term

$$\operatorname{div} \boldsymbol{\sigma} = \rho g \mathbf{e}_d,$$

Stokes, momentum conservation

$$\mathbf{u}_f = -\kappa\left(\nabla p_f + \rho_f g \mathbf{e}_d\right),$$

Darcy , momentum conservation

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{u}_s, p) = 2\mu_s \dot{\boldsymbol{\varepsilon}}(\mathbf{u}_s) - p\mathbf{I},$$

$$\rho = \rho_f \Phi + \rho_s(1-\Phi),$$

$$\eta = \frac{\mu_s}{\Phi} C\left(1 + \frac{1}{2}\left(\frac{1}{R} - 1\right)\left(1 + \tanh\left(-\frac{p - p_f}{\lambda}\right)\right)\right),$$

$$\kappa = \kappa_0 \mu_f^{-1} \left(\frac{\Phi}{\Phi_0}\right)^n,$$

Constitutive relations:

- Nonlinear in bulk viscosity (etas)

Reuber et al., 2020 (submitted)

Hydro-mechanics

2-phase flow in geology is the **combination** of **Stokes** (rock) and **Darcy** flow (fluid – water/melt), while **coupling** the two with **porosity** (void space filled with fluid).

$$\operatorname{div}\mathbf{u}_s = -\frac{p - p_f}{\eta(1-\Phi)},$$

Stokes, mass conservation + coupling term

$$\operatorname{div}\mathbf{u}_f = \frac{p -$$

oupling term

$$\operatorname{div}\boldsymbol{\sigma} = \rho g \mathbf{e}_d$$

ation

$$\mathbf{u}_f = -\kappa\left(\nabla$$

tion

Assumptions in this work:

- Permeability scales with porosity
- We neglect advection of porosity (solid velocity) in the channel limit

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{u}_s, p) = 2\mu_s\dot{\boldsymbol{\varepsilon}}(\mathbf{u}_s) - p\mathbf{I},$$

$$\rho = \rho_f\Phi + \rho_s(1-\Phi),$$

$$\eta = \frac{\mu_s}{\Phi}C\left(1 + \frac{1}{2}\left(\frac{1}{R} - 1\right)\left(1 + \tanh\left(-\frac{p - p_f}{\lambda}\right)\right)\right),$$

$$\kappa = \kappa_0\mu_f^{-1}\left(\frac{\Phi}{\Phi_0}\right)^n,$$

Constitutive relations:

- Nonlinear in bulk viscosity (etas)

Reuber et al., 2020 (submitted)

Inversion

We seek to **minimize** the misfit between **observed** and **calculated vertical fluid velocity**:

$$\mathcal{J}(\Phi) := \frac{1}{2}\int_{\Omega_{\text{obs}}} \left((\mathbf{u}_f - \mathbf{u}_f^{\text{obs}})\cdot \boldsymbol{\nu}\right)^2 \, d\mathbf{x},$$

By taking **variations** of the **Lagrangian function** (e.g. Tröltzsch 2010) one can calculate the **derivative of a cost function with respect to the design parameter**:

$$\mathcal{G}(\Phi) = \mathbf{v}_s \cdot (\rho_f - \rho_s) g \mathbf{e}_d - n\,\kappa_0\,\mu_f^{-1}\frac{\Phi^{n-1}}{\Phi_0^n}\mathbf{v}_f \cdot \left(\rho_f g \mathbf{e}_d + \nabla p_f\right) - \frac{(q+q_f)(p-p_f)}{\eta\Phi(1-\Phi)^2},$$

where,

$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{v}_s, q) = \mathbf{0},$$

$$-\operatorname{div}\mathbf{v}_s = (q+q_f)\frac{\eta + p\,\psi(\Phi, p, p_f)}{\eta^2(1-\Phi)},$$

$$\mathbf{v}_f = \begin{cases} -\nabla q_f + \mathbf{e}_d(\mathbf{u}_f - \mathbf{u}_f^{\text{obs}})\cdot\mathbf{e}_d & \text{on } \Omega_{\text{obs}} \\ -\nabla q_f & \text{on } \Omega \setminus \Omega_{\text{obs}} \end{cases},$$

$$-\operatorname{div}(\kappa\mathbf{v}_f) = (q+q_f)\frac{\eta - p_f\,\psi(\Phi, p, p_f)}{\eta^2(1-\Phi)},$$

is the **adjoint system** of equations

Reuber et al., 2020 (submitted)

Inversion

We seek to **minimize** the misfit between **observed** and **calculated vertical fluid velocity**:

$$\mathcal{J}(\Phi) := \frac{1}{2} \int_{\Omega_{\text{obs}}} \left((\mathbf{u}_f - \mathbf{u}_f^{\text{obs}}) \cdot \boldsymbol{\nu} \right)^2 \, d\mathbf{x},$$

By taking **variati** can calculate the **derivative of a c**

$$\mathcal{G}(\Phi) = \mathbf{v}_s \cdot ($$

$$\frac{(q + q_f)(p - p_f)}{\eta \Phi (1 - \Phi)^2},$$

where, $\operatorname{div} \boldsymbol{\sigma}$

Adjoint method:

- Pointwise gradient with only one additional linear solve

Gradient descent:

$$\Phi^{(i+1)} = \Phi^{(i)} - \alpha \, \mathcal{G}(\Phi^{(i)}),$$

$$-\operatorname{div} \mathbf{v}_s = (q + q_f) \frac{\eta - p_f \psi(\Phi, p, p_f)}{\eta^2 (1 - \Phi)},$$

$$\mathbf{v}_f = \begin{cases} -\nabla q_f + \mathbf{e}_d (\mathbf{u}_f - \mathbf{u}_f^{\text{obs}}) \cdot \mathbf{e}_d & \text{on } \Omega_{\text{obs}} \\ -\nabla q_f & \text{on } \Omega \setminus \Omega_{\text{obs}} \end{cases},$$

$$-\operatorname{div} (\kappa \mathbf{v}_f) = (q + q_f) \frac{\eta - p_f \psi(\Phi, p, p_f)}{\eta^2 (1 - \Phi)},$$

is the **adjoint system** of equations

Reuber et al., 2020 (submitted)

PT solver

Frankel, 1950, Otter, 1966, […], Räss et al., 2018

On the example of nonlinear diffusion:

$$\frac{\partial H}{\partial t} = -\frac{\partial}{\partial x}\left(H^n \frac{\partial H}{\partial x}\right)$$

$$\frac{\partial H}{\partial \tau} = -\frac{\partial}{\partial x}\left(H^n \frac{\partial H}{\partial x}\right)$$

$$\partial\tau = \frac{1}{\frac{1}{dx H^n 2.1} + \frac{1}{dt}}$$

→ **pseudo timestepping** to converge

Matlab main function **example**:

```
function [H,evol,res] = ForwardSolve(H,n,dx,dt,niter,epsi)
  evol=[];
  i = 1;
  for iter=1:niter
    Dx          = H.^n;
    Dxa         = (Dx(1:end-1)+Dx(2:end))/2;
    dtau        = 1/(1/(min(dx*dx)/max(Dx)/2.1) + 1/dt);
    qx          = -Dxa.*diff(H)/dx;
    dHdt        = -diff(qx)/dx;
    H(2:end-1) = H(2:end-1) + dtau*dHdt;
    H([1,end]) = [1,0.5];
    end
end
```

PT solver

Frankel, 1950, Otter, 1966, […], Räss et al., 2018

On the example of nonlinear diffusion:

$$\frac{\partial H}{\partial t} = -\frac{\partial}{\partial x}\left(H^n \frac{\partial H}{\partial x}\right)$$

$$\frac{\partial H}{\partial \tau} = -\frac{\partial}{\partial}$$

$$\partial\tau = \frac{}{dx}$$

+ Matrix free method (explicit FD)
  → perfect for accelerator architecture (GPUs)
+ Solver for any (non) linear system
+ Easy to develop

- Many more iterations (versus many more FLOPS)
- How to find the pseudo-timestep?

Matlab main fu

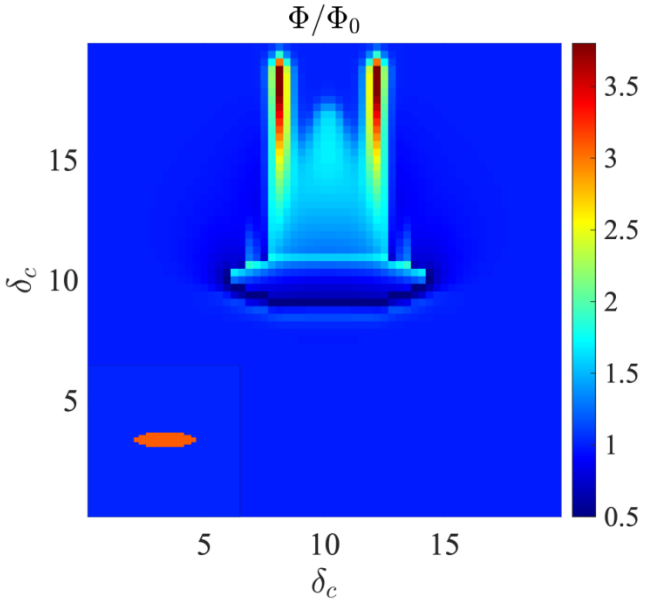
```
dx,dt,niter,epsi)
```

```
2;
/2.1) + 1/dt);
```

```
9       dHdt        = -diff(qx)/dx;
10      H(2:end-1)  = H(2:end-1) + dtau*dHdt;
11      H([1,end])  = [1,0.5];
12      end
13  end
14
```

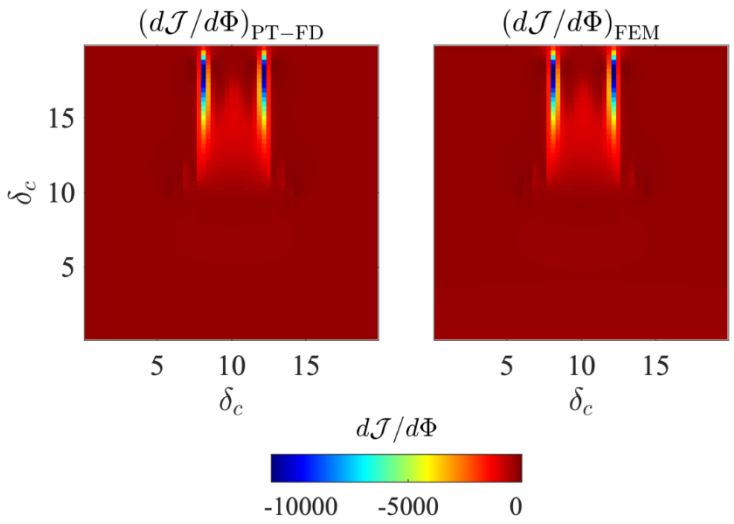
Benchmark

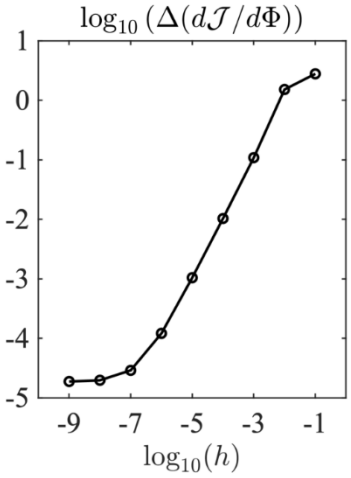
Comparison of the gradient with **direct finite difference**, a FEniCS **FEM** python code and the **pseudo-transient** matlab code (64 x 64 elements/cells/parameters):



Very good agreement among the methods

The synthetic porosity field is created by solving the transient forward problem for some time



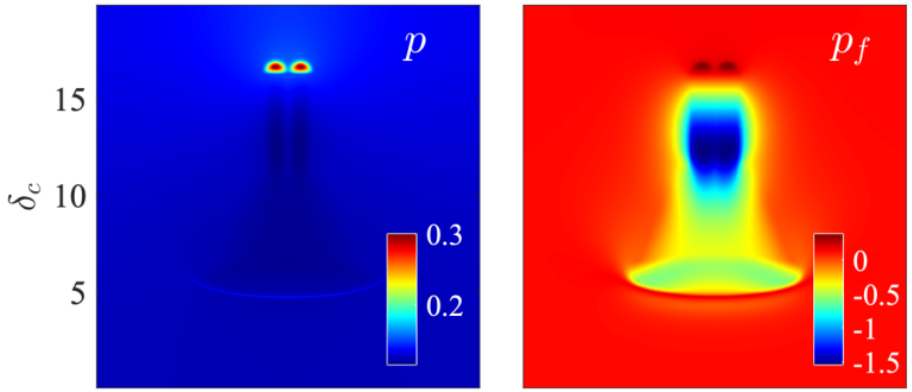


Reuber et al., 2020 (submitted)

Results

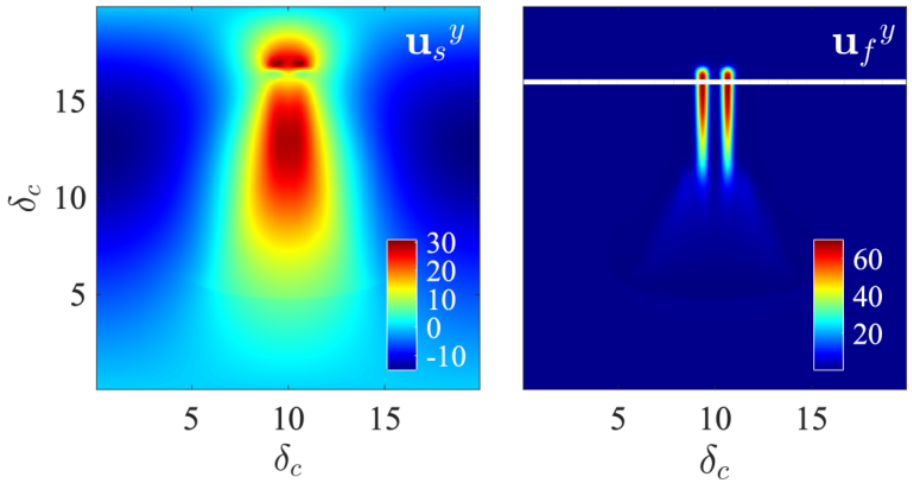
Forward problem result (1024 x 1024 cells) used as **synthetic measurements**:

Total pressure



Fluid pressure

Vertical solid velocity



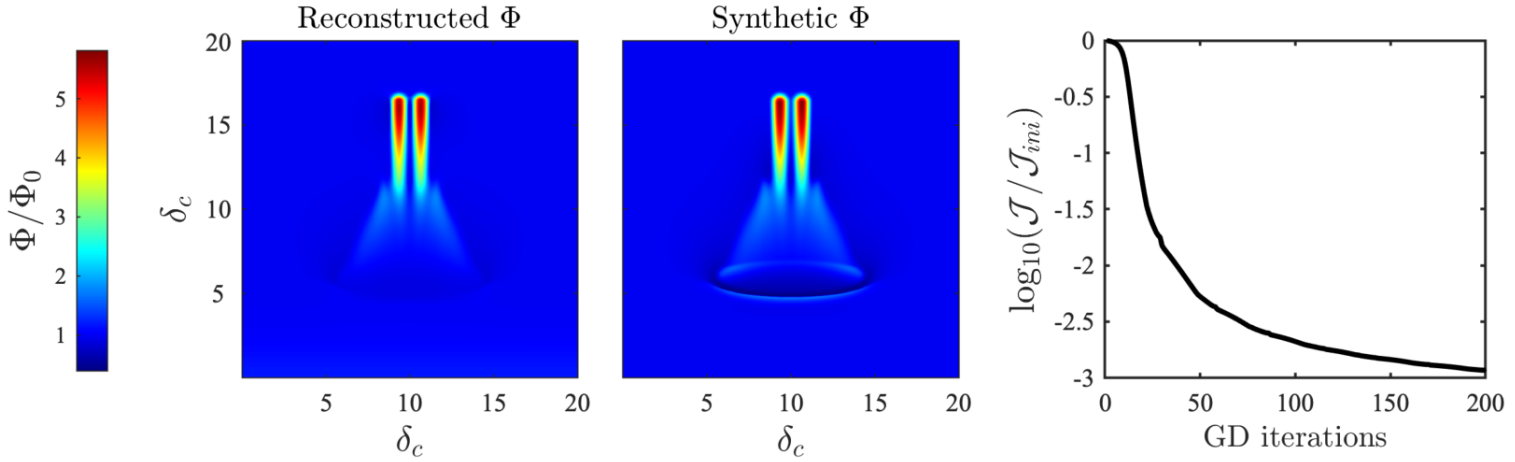
Vertical fluid velocity

Reuber et al., 2020 (submitted)

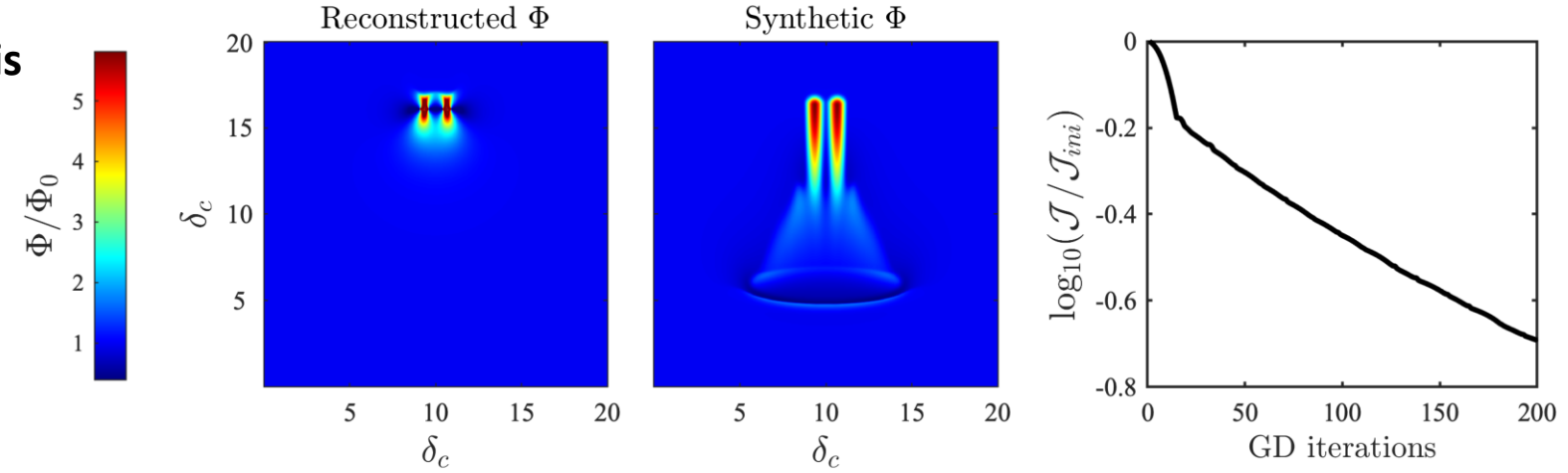
Results

We try to **recover** the synthetic porosity field from an initial **homogeneous porosity field** (1024 x 1024 cells/parameters):

Measurement is available everywhere: Very good reconstruction



Measurement is available along a surface: Very localized sensitivity/ reconstruction



Outlook

1. Julia/CUDA MPI 3D pseudo-transient solver (ParallelStencil and ImplicitGlobalGrid Julia package by Sam Omlin) - **HPC**
2. Transient inverse problem - **improved reconstruction**
3. Regularization (H^1 Tikhonov) – **improved convergence**

2D slice through a 3D porosity inversion (256x256x256 grid points) in Julia on GPU