Adjoint-based inversion for porosity in shallow reservoirs using pseudo-transient solvers for non-linear hydro-mechanical processes

Georg S. Reuber[a,b,*], Lukas Holbach[c], Ludovic Räss[d,e,f]

[a] *Johannes Gutenberg University Mainz, Institute of Geosciences, Mainz, Germany*
[b] *Max Planck Graduate Center, Mainz, Germany*
[c] *Johannes Gutenberg University Mainz, Institute of Mathematics, Mainz, Germany*
[d] *Stanford University, Geophysics Department, Stanford CA, USA*
[e] *Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Zurich, Switzerland*
[f] *Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland*

Abstract

Porous flow is of major importance in the shallow subsurface, since it directly impacts on reservoir-scale processes such as waste fluid sequestration or oil and gas exploration. Coupled and non-linear hydro-mechanical processes describe the motion of a low-viscous fluid interacting with a higher viscous porous rock matrix. This two-phase flow may trigger the initiation of solitary waves of porosity, further developing into vertical high-porosity pipes or chimneys. These preferred fluid escape features may lead to localised and fast vertical flow pathways potentially problematic in the case of for instance CO_2 sequestration. Constraining the porosity and the non-linearly related permeability distribution in such environments is a major challenge. Although seismic imaging methods accurately localise the high-porosity chimneys in the inverted wave-speed field, the conversion to porosity is not straightforward. We develop an inversion framework to reconstruct the unknown porosity field using relevant observable quantities such as subsurface fluid fluxes. We introduce the adjoint framework for the two-phase flow equations, which allows for efficient calculations of the pointwise gradients of the flow solution with respect to the porosity. We then use the gradients in a gradient descent method to invert for the pointwise porosity. We solve the forward and the adjoint equations using an iterative matrix-free pseudo-transient approach with the finite difference method. The proposed parallel solving procedure executes optimally on the latest many-core hardware accelerators such as GPUs. Numerical results show that an inversion for porosity is challenging if data is sparse since the porosity is very locally sensitive to the fluid flux. We introduce the concepts of sensitivity kernels as employed in seismology for the set of two-phase equations and suggest this approach as a standard for future studies.

Keywords: Two-phase flow, Adjoint gradients, Inversion, Pseudo-transient solver, GPU accelerators

1. Introduction

The two-phase flow equations describe the composite motion of two viscous fluids that can be arbitrarily distinct in their behaviour. Geological applications of two-phase flow dynamics are for instance the motion of magma (linear viscosity of $1-10^3$ Pa.s) in the pore space of host rocks (non-linear viscosity of $10^{18}-10^{30}$ Pa.s) [e.g. 1, 2, 3, 4, 5, 6, 7], or buoyant fluids in the pore space of shallow sedimentary rock stacks leading to the formation of fluid escape pipes or chimneys [e.g. 8, 9, 10, 11, 12, 13]. Thus, solving the two-phase flow equations is crucial if one is to understand and predict the behaviour of the pore-fluid interaction with the porous matrix deformation, i.e. capturing the hydro-mechanical processes and interactions. [14] has shown that the pore fluid, which is less dense than the host rock, can potentially create a pathway through the host rock breaching toward the seafloor. The buoyant pore fluid generates a continuous and localised force on the porous matrix that undergoes creep, opening and closing pore space at rates

*Corresponding author
Email address: reuber@uni-mainz.de (Georg S. Reuber)

proportional to the matrix's bulk decompaction and compaction viscosity, respectively [3, 15, 16, 14]. The resulting wave-like propagation of an elevated porosity anomaly – also referred to as a porosity wave – is sensitive to the difference in decompaction and compaction viscosity values. This asymmetry leads to the formation of elongated high-porosity chimneys responsible for creating vertical fluid escape pathways in the subsurface. An interesting problem is locating these channels and determining their porosity and permeability in order to constrain the prediction of potential fluid migrations. Finding these channels can be achieved by seismic imaging. However, this method is neither able to determine the channels' porosity, nor the viscosity and density of the pore fluid, which can be interesting for resources exploration (oil and gas) or for monitoring CO_2 sequestration operations. An automatic inversion approach can help to constrain the porosity as well as various material parameters of the fluid and the porous matrix.

Our first aim is to develop an inversion framework for the system of non-linear hydro-mechanical equations [15, 14]. Instead of performing many forward simulations or a direct parameter search [e.g. 17, 18], we compute the gradient of a given physical variable (here, the fluid flux and solid velocity) with respect to the porosity at any point in the model domain. As a result we get a new porosity field as update, unraveling information about the channel's geometry as well as its porosity. To make this approach tractable, we develop an adjoint framework, both in the strong and weak form, to efficiently calculate the gradients. This adjoint approach has been applied in many fields in geosciences, for instance to Stokes problems [e.g. 19, 20, 21, 22] or to the wave equation [e.g. 23, 24].

Our second aim is to test whether iterative matrix-free pseudo-transient (PT) solvers [e.g. 25] can be used to solve the adjoint equations. These solvers rely on iterating on physically-motivated numerical transient terms to simultaneously converge both the linear and the non-linear problem. Second-order methods prevent a non-scalable growth of the iteration count [26] with increase in numerical grid resolution. The method has the advantage of producing short and concise codes that do not rely on matrix-vector operations. The resulting codes exhibit a close to optimal scalability on many-core accelerators such as graphical processing units (GPUs) and multi-GPU implementations, which we will demonstrate. One limitation of this approach is the solvers' sensitivity to the iterative time-step criterion and the tuning of the second-order damping coefficient. We provide the methodological foundation and the proof of concept that a pseudo-transient approach is viable to solve the two-phase adjoint equations providing a matrix-free and totally scalable inversion procedure. We present a finite-difference based (FD) pseudo-transient solver and verify it using a FEniCS developed [27] finite element (FE) code. We limited the calculations to two dimensions (2-D) in space for augmented methodological conciseness.

Our third aim is to introduce the concept of sensitivity kernels for the non-linear hydro-mechanical equations in analogy to recent developments in the field of computational seismology with application to the wave equation [e.g. 23]. These kernels only require a small modification to the adjoint equation to enable the calculation of the gradient of the integral mean of the forward solution with respect to an input material parameter, here the pointwise porosity. They reveal the 'resolution' or 'sensitivity' of the current forward solution (in a chosen subdomain) to the input parameters, providing an additional tool to highlight the robustness of interpretations made based on the forward solution. Owing to the low numerical cost of the evaluation of the adjoint equation (there is only one additional linear solve) these kernels can be calculated after every forward simulation, i.e. for each time-step. In contrast to the sensitivity kernels derived from the linear wave equation, the ones we present in the two-phase framework are neither invariant to the evolution of the forward solution nor to the initial conditions, owing to the non-linear nature of the underlying equation system.

Finally, the Appendix contains the proof-of-concept study of the adjoint derivation and solution, including the calculation of the sensitivity kernels, with a PT-FD and a FE FEniCS solver presented for the 1-D non-linear diffusion equation. The codes produced for the Appendix, which illustrate the methods for this particular example, can be accessed from the Bitbucket repository: `https://bitbucket.org/hydromechanics_adjoint_pt/hm_adj_pt/src/master/`. The GPU-based routines can be accessed upon request to the authors.

2. The governing equations and the numerical methods

2.1. The two-phase flow equations

We describe the steady flow solution of a viscous fluid in an incompressible viscously deforming porous medium on an open and bounded rectangular domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, by the following system of partial differential equations (see [15] for derivation):

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

$$\operatorname{div} \mathbf{u}_f = \frac{p - p_f}{\eta(1-\Phi)}, \tag{2}$$

$$\operatorname{div} \boldsymbol{\sigma} = \rho g \mathbf{e}_d, \tag{3}$$

$$\mathbf{u}_f = -\kappa \left(\nabla p_f + \rho_f g \mathbf{e}_d \right), \tag{4}$$

representing the mass (1) - (2) and the momentum (3) - (4) balance equations for the solid and the fluid, respectively. The deviatoric stress tensor $\boldsymbol{\sigma}$ is given by (5), ρ is the bulk density, g the gravitational acceleration, $\mathbf{e}_d$ the d-th unit vector in $\mathbb{R}^d$, $\mathbf{u}_s$ is the solid velocity, p is the total pressure, p_f is the fluid pressure, η is the bulk viscosity of the two-phase system, Φ is the porosity, $\mathbf{u}_f$ is the fluid flux (or velocity), κ is the effective permeability, ρ_f is the fluid density, and ρ_s is the solid density. A set of constitutive relationships closes the system of equations:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{u}_s, p) = 2\mu_s \dot{\boldsymbol{\varepsilon}}(\mathbf{u}_s) - p\mathbf{I}, \tag{5}$$

$$\rho = \rho_f \Phi + \rho_s (1 - \Phi), \tag{6}$$

$$\eta = \frac{\mu_s}{\Phi} C \left(1 + \frac{1}{2} \left(\frac{1}{R} - 1 \right) \left(1 + \tanh\left(-\frac{p - p_f}{\lambda} \right) \right) \right), \tag{7}$$

$$\kappa = \kappa_0 \mu_f^{-1} \left(\frac{\Phi}{\Phi_0} \right)^n, \tag{8}$$

where μ_s is the solid shear viscosity, $\dot{\boldsymbol{\varepsilon}}(\mathbf{u}_s) = \left(\nabla \mathbf{u}_s + \nabla \mathbf{u}_s^T \right)/2$ is the deviatoric strain rate tensor, $\mathbf{I}$ is the identity matrix, C is the shear viscosity ratio, R is the compaction strength ratio for the bulk viscosity, λ is the effective pressure transition zone, κ_0 is the constant reference permeability, μ_f is the fluid viscosity, Φ_0 is the constant background porosity, and $n = 3$ is the permeability power-law exponent. Notably, the two-phase equations (1) - (4) are non-linear since the bulk viscosity η depends on the effective pressure and the porosity. An evolution equation for the porosity finalizes the system of two-phase equations:

$$\frac{\partial \Phi}{\partial t} = (1 - \Phi) \operatorname{div} \mathbf{u}_s \, . \tag{9}$$

In this work, we only consider a steady solution of the two-phase flow problem after a given number of time-steps, i.e. we neglect the porosity evolution and advection. The forward problem would be transient and thus computationally more expensive, and the adjoint problem would therefore be transient backward in time. Our aim is to describe the general framework, provide the set of two-phase adjoint equations, and investigate the performance of the overall inversion framework, leaving the transient problem for future studies. Further technical details regarding the two-phase flow equations with application to spontaneous fluid flow localisation can be found in [25].

To close the system of partial differential equations, we define boundary conditions on the boundary $\Gamma = \Gamma_{\text{sides}} \cup \Gamma_{\text{top}} \cup \Gamma_{\text{bottom}}$ of Ω. We employ a free slip condition and no normal flow on the entire boundary for the Stokes problem, using no normal flux on the sides and constant pressure on the top and bottom for the Darcy flow problem:

$$\mathbf{u}_s \cdot \boldsymbol{\nu} = 0 \quad \text{on } \Gamma, \tag{10}$$

$$\mathbf{T}(\boldsymbol{\sigma}\boldsymbol{\nu}) = \mathbf{0} \quad \text{on } \Gamma, \tag{11}$$

$$\mathbf{u}_f \cdot \boldsymbol{\nu} = 0 \quad \text{on } \Gamma_{\text{sides}}, \tag{12}$$

$$p_f = p_0 \quad \text{on } \Gamma_{\text{top}}, \tag{13}$$

$$p_f = p_1 \quad \text{on } \Gamma_{\text{bottom}}, \tag{14}$$

where $\boldsymbol{\nu}$ denotes the outward normal vector, $\mathbf{T} = \mathbf{I} - \boldsymbol{\nu}\boldsymbol{\nu}^T$ the tangential projection, and p_0, $p_1 \in \mathbb{R}$ are constants. The primary and constant input variables are summarised in Table 1.

Next, we derive the weak form of the two-phase system of equations. We will use this weak formulation to derive the adjoint equations in Section 2.3. The first step is to multiply equations (1) - (4) with sufficiently smooth test functions q, q_f, $\mathbf{v}_s$ and $\mathbf{v}_f$ respectively. Here, $\mathbf{v}_s$ needs to satisfy (10) and q_f needs to satisfy the homogeneous versions of (13) and (14) (i.e. $q_f = 0$ on $\Gamma_{\text{top}} \cup \Gamma_{\text{bottom}}$), whereas the other boundary conditions are natural. We integrate the equations over the domain Ω followed by integration by parts and the addition of the equations. Defining the operators

$$\begin{aligned} a(\mathbf{u}_s, p, \mathbf{u}_f, p_f; \mathbf{v}_s, q, \mathbf{v}_f, q_f) = &\int_\Omega 2\mu_s \dot{\boldsymbol{\varepsilon}}(\mathbf{u}_s) : \dot{\boldsymbol{\varepsilon}}(\mathbf{v}_s)\, d\mathbf{x} - \int_\Omega p \operatorname{div} \mathbf{v}_s\, d\mathbf{x} \\ &- \int_\Omega q \operatorname{div} \mathbf{u}_s\, d\mathbf{x} - \int_\Omega q \frac{p - p_f}{\eta(1-\Phi)}\, d\mathbf{x} \\ &- \int_\Omega \mathbf{u}_f \cdot \mathbf{v}_f\, d\mathbf{x} - \int_\Omega \kappa \mathbf{v}_f \cdot \nabla p_f\, d\mathbf{x} \\ &- \int_\Omega \nabla q_f \cdot \mathbf{u}_f\, d\mathbf{x} - \int_\Omega q_f \frac{p - p_f}{\eta(1-\Phi)}\, d\mathbf{x} \end{aligned} \tag{15}$$

and

$$\ell(\mathbf{v}_s, q, \mathbf{v}_f, q_f) = -\int_\Omega \mathbf{v}_s \cdot \rho g \mathbf{e}_d\, d\mathbf{x} + \int_\Omega \kappa \mathbf{v}_f \cdot \rho_f g \mathbf{e}_d\, d\mathbf{x}, \tag{16}$$

the weak form of (1) - (4) is therefore: Find $\mathbf{z} = (\mathbf{u}_s, p, \mathbf{u}_f, p_f)$ such that

$$a(\mathbf{z}, \mathbf{w}) = \ell(\mathbf{w}) \tag{17}$$

for all test functions $\mathbf{w} = (\mathbf{v}_s, q, \mathbf{v}_f, q_f)$.

2.2. The inverse problem

Because the porosity in fluid-saturated subsurface regions may not be a direct observable, our goal is to infer the porosity from field observations such as fluid fluxes or matrix deformation (solid velocities).

We formulate the inverse problem as an infinite-dimensional constrained least-squares optimisation problem of the form

$$\min_{\Phi} \mathcal{J}(\Phi) \tag{18}$$

with

$$\mathcal{J}(\Phi) := \frac{1}{2} \int_{\Omega_{\text{obs}}} \left((\mathbf{u}_f - \mathbf{u}_f^{\text{obs}}) \cdot \boldsymbol{\nu} \right)^2 d\mathbf{x}, \tag{19}$$

where $\mathbf{u}_f^{\text{obs}}$ denotes the fluid fluxes measured on $\Omega_{\text{obs}} \subset \Omega$ and $\mathbf{u}_f$ is the solution of the forward problem (17) corresponding to the parameter field Φ.

There are several methods to minimise the cost functional, for instance quasi-Newton methods that include the computation (or an approximation) of the Hessian, or a simple gradient descent (GD), which will be sufficient for the studies we present here:

$$\Phi^{(i+1)} = \Phi^{(i)} - \alpha\, \mathcal{G}(\Phi^{(i)}), \tag{20}$$

where i denotes the iteration index, α is a line-search parameter that we can choose according to a simple bisection or an Armijo line-search condition [28], and $\mathcal{G}$ denotes the gradient of the cost functional $\mathcal{J}$. All gradient-based methods require efficient computation of $\mathcal{G}$, which we will address now.

2.3. The adjoint equations

The adjoint method is an efficient way to compute gradients of the cost function $\mathcal{J}$ and can be derived in multiple ways. One way is to first discretise the equations and then apply the chain rule multiple times to obtain the final set of local derivatives [e.g. 19, 29]. However, since we want to compare numerical solutions with different discretisation approaches, for consistency, it is necessary to first derive the equations in the infinite-dimensional setting and to then discretise the equations.

Thus, we use the formal Lagrangian approach on function spaces [see e.g. 30, 31], which involves the definition of the Lagrangian functional:

$$\mathcal{L}(\mathbf{z}, \mathbf{w}, \Phi) := \mathcal{J}(\Phi) + a(\mathbf{z}, \mathbf{w}) - \ell(\mathbf{w}). \tag{21}$$

In this method, $\mathbf{z}$, $\mathbf{w}$, and Φ are treated as independent variables, and the test functions serve as Lagrange multipliers that become the adjoint variables. In the next step, we compute the variations of the Lagrangian functional with respect to i) the Lagrange multipliers $\mathbf{w}$, which recovers the weak form of the forward problem, ii) the forward solution $\mathbf{z}$, which yields the weak form of the adjoint equation, and iii) the design parameter (in our case the porosity Φ), which yields the weak form of the gradient of the cost function depending on the solution of the forward and adjoint equations. In the Appendix, we apply the formal Lagrangian approach to a 1-D non-linear diffusion equation so as to outline the workflow in a concise and simple example.

Taking variations of the Lagrangian with respect to the forward solution variables $\mathbf{z} = (\mathbf{u}_s, p, \mathbf{u}_f, p_f)$ in direction $\hat{\mathbf{z}} = (\hat{\mathbf{u}}_s, \hat{p}, \hat{\mathbf{u}}_f, \hat{p}_f)$ and requiring them to vanish for all $\hat{\mathbf{z}}$ yields the weak form of the adjoint equation: Find $(\mathbf{v}_s, q, \mathbf{v}_f, q_f)$ such that

$$\begin{aligned} 0 \stackrel{!}{=} \mathcal{L}_{\mathbf{z}}(\mathbf{z}, \mathbf{w}, \Phi)\,\hat{\mathbf{z}} = &\int_{\Omega_{\text{obs}}} \hat{\mathbf{u}}_f \cdot \nu\,(\mathbf{u}_f - \mathbf{u}_f^{\text{obs}}) \cdot \nu \, d\mathbf{x} + \int_\Omega 2\mu_s \dot{\boldsymbol{\varepsilon}}(\hat{\mathbf{u}}_s) : \dot{\boldsymbol{\varepsilon}}(\mathbf{v}_s)\, d\mathbf{x} - \int_\Omega \hat{p}\, \operatorname{div} \mathbf{v}_s \, d\mathbf{x} \\ &- \int_\Omega q \operatorname{div} \hat{\mathbf{u}}_s \, d\mathbf{x} - \int_\Omega q \frac{(\hat{p} - \hat{p}_f)\eta + (p\hat{p} + p_f\hat{p}_f)\,\psi(\Phi, p, p_f)}{\eta^2(1-\Phi)}\, d\mathbf{x} \\ &- \int_\Omega \hat{\mathbf{u}}_f \cdot \mathbf{v}_f \, d\mathbf{x} - \int_\Omega \kappa \mathbf{v}_f \cdot \nabla \hat{p}_f \, d\mathbf{x} \\ &- \int_\Omega \nabla q_f \cdot \hat{\mathbf{u}}_f \, d\mathbf{x} - \int_\Omega q_f \frac{(\hat{p} - \hat{p}_f)\eta + (p\hat{p} + p_f\hat{p}_f)\,\psi(\Phi, p, p_f)}{\eta^2(1-\Phi)}\, d\mathbf{x} \end{aligned} \tag{22}$$

for all $(\hat{\mathbf{u}}_s, \hat{p}, \hat{\mathbf{u}}_f, \hat{p}_f)$, where $(\mathbf{u}_s, p, \mathbf{u}_f, p_f)$ denotes the solution of the forward problem (17) corresponding to Φ and

$$\psi(\Phi, p, p_f) := \frac{\mu_s C}{\Phi 2\lambda}\left(\frac{1}{R} - 1\right)\left(1 - \tanh^2\left(-\frac{p - p_f}{\lambda}\right)\right). \tag{23}$$

We emphasise that the adjoint equation (22) is linear and can therefore be efficiently solved numerically. Further, the first term in this equation depends on the explicit form of the cost function $\mathcal{J}$. Thus, for different data types considered in Section 4, the only change in the cost functional and thus the adjoint equation is either the domain over which one needs to integrate, or replacing fluid fluxes with solid velocities. The strong form of the adjoint equation is therefore also affected, as we will explain at the end of this section.

Differentiating $\mathcal{L}$ with respect to the design variable, porosity, yields the weak form of the gradient $\mathcal{G}$ of $\mathcal{J}$ in direction $\hat{\Phi}$, evaluated at Φ:

$$\begin{aligned} \mathcal{L}_\Phi(\mathbf{z}, \mathbf{w}, \Phi)\,\hat{\Phi} = &\int_\Omega \hat{\Phi}\, \mathbf{v}_s \cdot \left(\rho_f - \rho_s\right) g \mathbf{e}_d \, d\mathbf{x} - \int_\Omega \hat{\Phi}\, n\, \kappa_0\, \mu_f^{-1} \frac{\Phi^{n-1}}{\Phi_0^n} \mathbf{v}_f \cdot \rho_f g \mathbf{e}_d \, d\mathbf{x} \\ &- \int_\Omega \hat{\Phi}\, n\, \kappa_0\, \mu_f^{-1} \frac{\Phi^{n-1}}{\Phi_0^n} \mathbf{v}_f \cdot \nabla p_f \, d\mathbf{x} - \int_\Omega \hat{\Phi}\, \frac{(q + q_f)(p - p_f)}{\eta \Phi (1-\Phi)^2}\, d\mathbf{x}. \end{aligned} \tag{24}$$

Here, one must insert the corresponding solutions $(\mathbf{u}_s, p, \mathbf{u}_f, p_f)$ and $(\mathbf{v}_s, q, \mathbf{v}_f, q_f)$ of the forward problem (17) and the adjoint problem (22), respectively.

As demonstrated, the Lagrangian approach naturally provides a complete set of equations to compute the gradient of the cost function with respect to some design variable. Since FE discretisations are based on the weak form of the equation, one can take advantage of equations (17), (22) and (24), choose appropriate shape functions and solve the systems to retrieve the gradient in a straightforward way. On the other hand, other discretisation techniques, for instance FD methods, require the strong form of the equation that can be obtained by isolating the test functions through integration by parts. Thus, the strong form of the adjoint equation is:

$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{v}_s, q) = \mathbf{0}, \tag{25}$$

$$-\operatorname{div}\mathbf{v}_s = (q + q_f)\frac{\eta + p\,\psi(\Phi, p, p_f)}{\eta^2(1-\Phi)}, \tag{26}$$

$$\mathbf{v}_f = \begin{cases} -\nabla q_f + \mathbf{e}_d(\mathbf{u}_f - \mathbf{u}_f^{\text{obs}}) \cdot \mathbf{e}_d & \text{on } \Omega_{\text{obs}} \\ -\nabla q_f & \text{on } \Omega \setminus \Omega_{\text{obs}} \end{cases}, \tag{27}$$

$$-\operatorname{div}(\kappa\mathbf{v}_f) = (q + q_f)\frac{\eta - p_f\,\psi(\Phi, p, p_f)}{\eta^2(1-\Phi)}, \tag{28}$$

with boundary conditions defined as:

$$\mathbf{v}_s \cdot \boldsymbol{\nu} = 0 \quad \text{on } \Gamma, \tag{29}$$

$$\mathbf{T}(\boldsymbol{\sigma}(\mathbf{v}_s, q)\,\boldsymbol{\nu}) = \mathbf{0} \quad \text{on } \Gamma, \tag{30}$$

$$\kappa\mathbf{v}_f \cdot \boldsymbol{\nu} = 0 \quad \text{on } \Gamma_{\text{sides}}, \tag{31}$$

$$q_f = 0 \quad \text{on } \Gamma_{\text{top}} \cup \Gamma_{\text{bottom}}. \tag{32}$$

If vertical solid velocities instead of fluid fluxes are measured on Ω_{obs}, equations (25) and (27) are replaced by

$$\operatorname{div}\boldsymbol{\sigma}(\mathbf{v}_s, q) = \begin{cases} \mathbf{e}_d(\mathbf{u}_s - \mathbf{u}_s^{\text{obs}}) \cdot \mathbf{e}_d & \text{on } \Omega_{\text{obs}}, \\ \mathbf{0} & \text{on } \Omega \setminus \Omega_{\text{obs}}, \end{cases} \tag{25b}$$

$$\mathbf{v}_f = -\nabla q_f. \tag{27b}$$

Independent of the data type used, the strong form of the gradient $\mathcal{G}$ of the cost functional $\mathcal{J}$ evaluated at Φ is:

$$\mathcal{G}(\Phi) = \mathbf{v}_s \cdot (\rho_f - \rho_s) g \mathbf{e}_d - n\,\kappa_0\,\mu_f^{-1}\frac{\Phi^{n-1}}{\Phi_0^n}\mathbf{v}_f \cdot \left(\rho_f g \mathbf{e}_d + \nabla p_f\right) - \frac{(q + q_f)(p - p_f)}{\eta\Phi(1-\Phi)^2}, \tag{33}$$

where $(\mathbf{u}_s, p, \mathbf{u}_f, p_f)$ and $(\mathbf{v}_s, q, \mathbf{v}_f, q_f)$ are the corresponding solutions of the forward equations (1) - (14) and adjoint system (25) - (32), respectively.

Adjoint-based gradient evaluations have the advantage that the computational cost is independent of the number of parameters and thus offer the possibility to invert for a parameter field such as porosity, which is known to strongly vary within an exploration area. However, gradient-based inversion methods have the general weakness that they cannot explore the full space of a non-convex cost function but may only converge to a local minimum, which statistical methods can overcome at significantly higher computational costs. Here, we focus on the deterministic gradient-based inversion that could be embedded in a statistical framework in the future [e.g. 32, 33, 34].

2.4. The numerical implementation

We discretise the presented system of equations (Sections 2.1 – 2.3) over a finite domain Ω in order to solve them numerically. We rely on two discretisation methods: finite differences and finite elements. For each method, we employ a different solver type. Our main contribution builds on the FD discretisation on a regular Cartesian staggered grid using a matrix-free iterative solver we hereafter refer to as pseudo-transient (PT). This alternative and classical approach is particularly well suited for parallel computations due to the related low memory footprint, the high data locality and the straightforward parallelisation on both shared and distributed memory machines [see e.g. 35, 25,

and references therein]. We extend the framework that solves the forward two-phase hydro-mechanical problem introduced in [14, 25] to solve the adjoint problem in an analogous way. We benchmark the PT-FD-based calculations against a FEniCS-based [27] FE workflow relying on matrix-based solvers. We present a specific implementation of a 1-D non-linear diffusion equation to showcase the MATLAB-based [36] PT-FD solution procedure and provide the FEniCS-based FE verification in the supplementary material to this paper.

The results we present are produced using the matrix-free iterative PT-FD approach that, to our best knowledge, has not yet been applied to solve an adjoint equation.

2.4.1. The pseudo-transient finite difference approach

PT methods [as discussed in 25, 37] can be used to solve linear equation systems that may result from steady-state, elliptic equations or from the linearisation of a non-linear equation. The two-phase equations, as described, represent such a system, since they build an elliptic non-linear system of equations. The essence of the PT approach lies in rearranging the targeted equations to a residual form and augmenting them by introducing physically-motivated transient terms. Such transient terms may, for instance, represent inertia or bulk compressibility in the Stokes momentum and mass balance equations, respectively [see e.g. 25, 37, 38, for further details]. The main advantage is the ability to iterate toward the steady state using these transient terms. We discretise the strong form of the forward and adjoint equations using an FD scheme, which minimises the necessary memory usage because we use a matrix-free approach, explicitly iterating within the pseudo-time. For instance, rearranging the momentum equation (3) in the residual form yields

$$\operatorname{div}\boldsymbol{\sigma} - (0, \rho g)^T = \mathbf{r}\,, \tag{34}$$

where $\mathbf{r}$ is the residual for the momentum conservation. The solution is considered converged once the absolute error reaches the desired accuracy threshold ε, i.e. $\|\,\mathbf{r}\,\| \leq \varepsilon$, retrieving an implicit solution of the original equations. We introduce the PT term

$$\frac{\partial \mathbf{u}_s}{\partial \tau} = \mathbf{r}\,, \tag{35}$$

that allows us to iterate on $\mathbf{u_s}$ using a straightforward update rule. We choose the discrete pseudo-time-step $\Delta\tau$ based on the current physical quantities, mesh size, and degree of non-linearity. This definition may represent one limitation of the PT solver. We specifically use the following iterative pseudo-time-steps for the velocities, total pressure and the fluid pressure equations, respectively:

$$\Delta\tau_{\mathbf{u}} = \frac{\min(\Delta x, \Delta y)^2}{\mu_s(1+\beta)4.1u_{\mathrm{sc}}}, \tag{36}$$

$$\Delta\tau_p = \frac{4.1\mu_s(1+\beta)}{\max(n_x, n_y)p_{\mathrm{sc}}}, \tag{37}$$

$$\Delta\tau_{p_f} = \frac{\min(\Delta x, \Delta y)^2}{4.1\kappa p_{f_{\mathrm{sc}}}}, \tag{38}$$

where Δx and Δy represent the discretisation-dependent mesh size, n_x and n_y the corresponding number of nodes, β represents some numerical divergence to balance the divergence-free implementation during the transient phase, u_{sc}, p_{sc}, and $p_{f_{\mathrm{sc}}}$ are variable scaling terms limiting the momentum, total pressure and fluid pressure iterative time-steps, respectively, and the factor 4.1 is the 2-D explicit diffusion CFL limiter.

We employ second-order Richardson pseudo-transient iterations [26] in order to ensure that the PT iteration count scales as close as possible to order N, where N represents the total number of grid points. The second-order PT forward (synthetic) solver scales to the order of $N^{1.3}$ for the specific two-phase equations in 3-D [35]. The introduced damping term arising from the second-order method sums up a significant portion of the previous rate of change to the current rate of change at every PT iteration with a factor close to 1.0 [26, 25].

PT-FD solvers have significant potential for two reasons. They are less complex, and therefore faster and more versatile to develop than implicit direct (or iterative) types of FE or FD solvers. An equal ease of development can only be achieved if one uses high-order libraries, such as FEniCS, which may reduce the control of the possible amount of optimisation. PT-FD solvers unleash their full potential when targeting the current and future high-performance and supercomputing parallel hardware accelerators, such as GPUs.

We show that the PT-FD method is a viable approach to solving the adjoint equation, paving the way for a high-performance parallel GPU-based framework implementation. We will now demonstrate the robustness of the proposed FD discretisation. More detailed information about the forward solution of the two-phase equations using a PT scheme can be found in [25].

2.4.2. The finite element method

In the finite element method (FEM) [39], one seeks a solution to the weak form (17) of the underlying system of partial differential equations, including all necessary boundary conditions and constitutive relationships. The numerical solution of the weak form requires an appropriate choice of test functions, for instance piecewise polynomials of arbitrary order. An advantage of the FE method is that the computation is carried out on a reference element of finite size in an arbitrary coordinate system, which allows for a fairly simple implementation of complex meshes compared to FD-based methods. However, this comes with the additional implementation effort of non-trivial numbering schemes and transformation functions. FEniCS [27] is a high-level software tool that offers a Python interface to lower-level parallel solver libraries such as PETSc [40] while, to a given extent, taking care of the underlying parallelisation, numbering schemes, meshing, and implementation of various test function types. One can develop an FE solver for a system of partial differential equations with comparatively little effort.

We use a standard Q2P1 element (also known as Taylor-Hood element [41]) for the Stokes equations (1 and 3), a zeroth-order Raviart-Thomas element [42] for the Darcy fluid fluxes (4) and a first-order Lagrange element for the fluid pressure (2) in our FEniCS-based benchmarking workflow.

2.5. GPU acceleration

Among the motivations that drive the development of the adjoint workflow in a PT-FD framework, the current availability and continuous development of many-core computing hardware are prominent. GPUs are one type of massively parallel computing accelerators that populate most modern supercomputers and are necessary to tackle computing at the exascale. Although GPUs feature an inherent potential in accelerating serial calculations, the speed-up may not be achieved in a straightforward way. Besides some successful GPU-based implementations there is also a growing frustration in various scientific communities regarding the current movement toward GPU-accelerated machines.

In this context, our study may provide some insights into concise, robust and efficient algorithms to tackle forward and inverse workflows, taking full advantage of the intrinsic fine-level parallelism unique to many-core hardware such as GPUs. To unleash the optimal computing performance of GPUs, algorithms should feature high locality, an *as low as possible* memory footprint, and a high potential of parallelism, and should execute in a very local way. Matrix-free iterative FD solvers fulfil most of these requirements and are therefore optimal candidates to build massively parallel solvers to optimally execute on GPU-accelerated machines.

Building on the successful implementation of coupled multi-physics forward solvers [14, 25, 37, 38], we propose to follow a similar path in implementing the adjoint solver of the two-phase equations. We assign a GPU thread to every grid point of the FD domain using the identical mapping as for the forward solver. The adjoint solution can be retrieved using a similar second-order iteration as in the forward solver.

The 2-D inversion workflow we present serves as proof of concept for the future step: the porting of the GPU-based inversion procedure to 3-D multi-GPU solvers.

3. Verification

To demonstrate the PT-FD solver's robustness, we compare its forward results for a reference model to the solution obtained with the FEM using FEniCS. We perform the calculations and report the simulation results in non-dimensional units. The reference model domain has a length and height of 20 characteristic compaction lengths δ_c

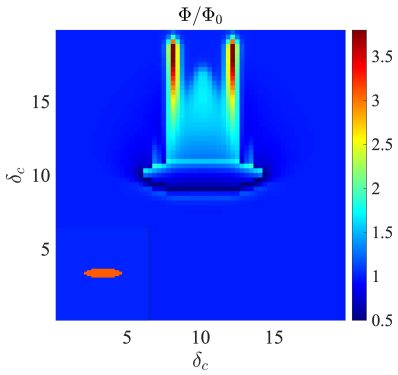

Figure 1: Normalized synthetic porosity field Φ/Φ_0 used for the comparison of the steady-state PT-FD and FE implementations. The porosity field is produced by evolving the time-dependent PT-FD forward code for 210 time-steps, starting with the initial configuration depicted in the sub-figure at the bottom left. The ellipsoidal initial fluid inclusion has a normalised porosity of three times the background reference porosity value $\Phi_0 = 0.01$.

Parameter	non-dimensional value
ρ_s	2
ρ_f	1
g	1
κ_0	1
ϕ_0	0.01
n	3
μ_s	0.001
μ_f	1
C	10
R	100, 200 or 400
λ	0.01
p_0	0
p_1	0

Table 1: Primary physical constant input parameters in non-dimensional units.

[1, 43]. We discretise the domain using 64 FD cells or 64 elements (FEM) in both x-direction and y-direction. To generate a realistic porosity field for the model, we run the PT-FD solver for 210 time-steps using an initial physical $\Delta t = 10^{-5}$ within the porosity evolution equations until the high-porosity channels propagate close to the surface (Figure 1). For this comparison benchmark, we employ a value of $R = 100$. The obtained porosity field is then used to compare the results of the PT-FD and the FEM implementations of the steady-state forward two-phase equations. The related non-dimensional physical parameters appear in Table 1.

We compare the horizontal x and vertical y fluid fluxes (Figure 2) obtained with the PT-FD and FE method. The reported x and y solid velocities (Figure 3) as well as the total and fluid pressure (Figure 4) agree well among the PT-FD method and the FEM.

To verify the adjoint-based gradients (33) in the PT-FD framework, we compare them with the gradients obtained using forward finite differences of the cost function, i.e. with

$$(d\mathcal{J}/d\Phi)_{\mathrm{FD}} \approx \frac{\mathcal{J}(\Phi + h\,\hat{\Phi}) - \mathcal{J}(\Phi)}{h}, \tag{39}$$

where $\hat{\Phi}$ is a randomly chosen perturbation field and h a small step size. In the case presented here, $\hat{\Phi}$ represents a perturbation of a randomly chosen single porosity point. We report the convergence of the FD gradient toward the

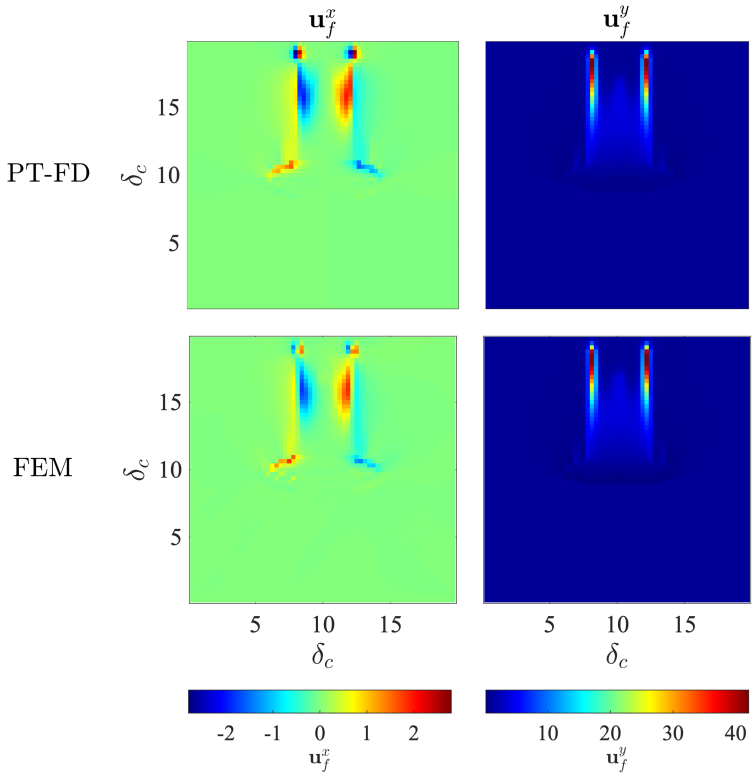

Figure 2: A Comparison of the non-dimensional fluid flux components for the PT-FD and FEM solvers. The left-hand and right-hand columns stand for the *x*-direction and *y*-direction, respectively. The top and bottom rows depict the PT-FD and FEniCS-based FEM results, respectively.

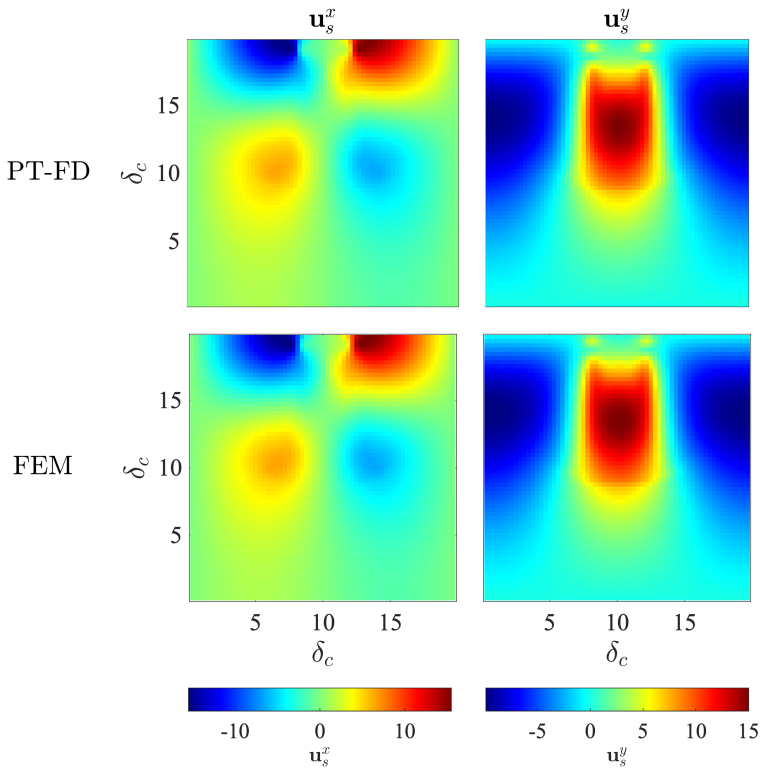

Figure 3: A comparison of the non-dimensional solid velocity components for the PT-FD and FEM solvers. The left-hand and right-hand columns stand for the *x*-direction and *y*-direction, respectively. The top and bottom rows depict the PT-FD and FEniCS-based FEM results, respectively.

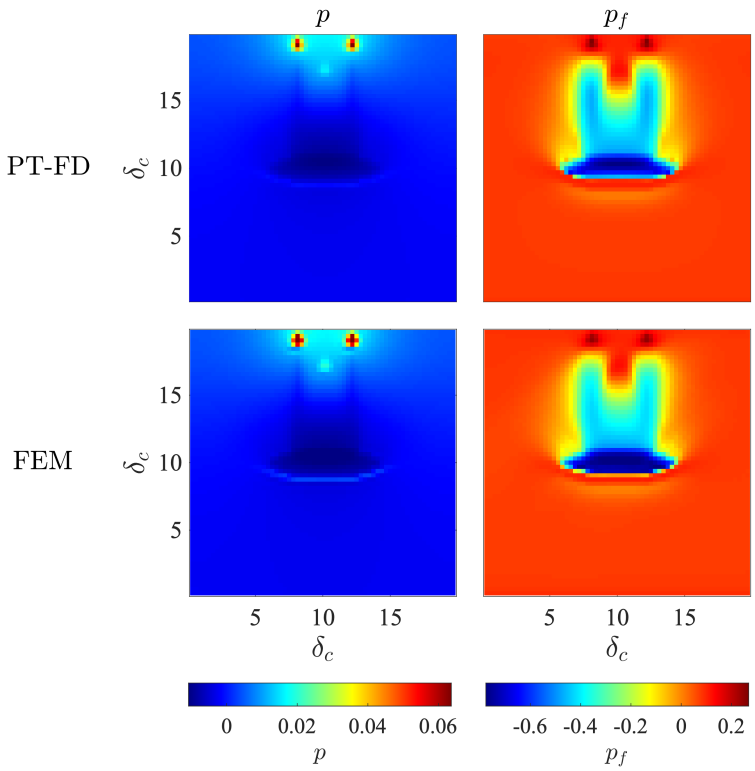

Figure 4: A comparison of the non-dimensional total and fluid pressure for the PT-FD and FEM solvers. The left-hand and right-hand columns stand for the total and fluid pressure respectively. The top and bottom rows depict the PT-FD and FEniCS-based FEM results respectively.

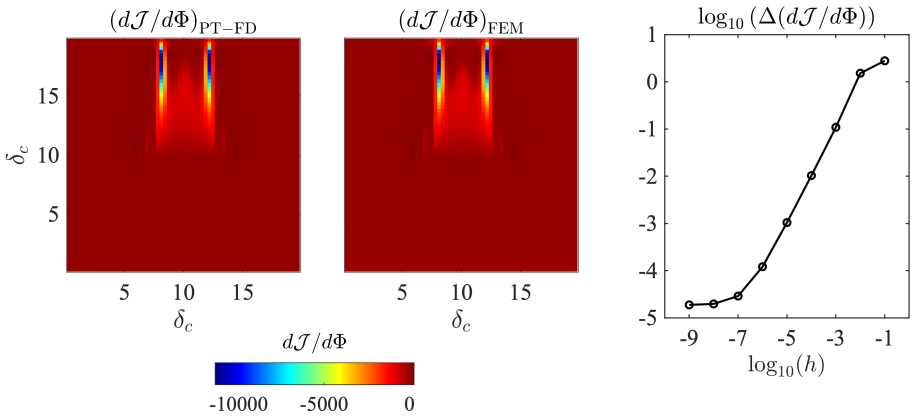

Figure 5: Left-hand column: PT-FD pointwise adjoint-based gradient of the cost function with respect to the porosity (equation (33)) using a homogeneous initial guess. Middle column: The same gradient, but calculated using the FEniCS FEM solver. Right-hand column: Convergence of the FD approximation of the gradient toward the PT-FD adjoint-based gradient in a specific direction $\hat{\Phi}$. The gradients agree well for sufficiently small perturbation parameter h.

adjoint-based PT-FD gradient (Figure 5 – right-hand panel), illustrating the relative error between these quantities:

$$\Delta(d\mathcal{J}/d\Phi) = \frac{(d\mathcal{J}/d\Phi)_{\text{PT-FD}}\hat{\Phi} - (d\mathcal{J}/d\Phi)_{\text{FD}}}{\|(d\mathcal{J}/d\Phi)_{\text{PT-FD}}\hat{\Phi}\|^2}. \tag{40}$$

Further, the pointwise adjoint-based derivatives of the cost function obtained with the PT-FD and the FEM approaches are consistent (Figure 5 – left-hand and the centre panel, respectively). In both cases, we use the vertical fluid flux corresponding to the porosity field depicted in Figure 1 at each point in the domain as synthetic data $\mathbf{u}_f^{\text{obs}}$ and evaluate the gradients at a homogeneous porosity field with a porosity of 1 %.

The time needed for one non-linear forward sequential solve run on the same machine is about 2.2 seconds for the MATLAB PT-FD solver and about 127 seconds for the FEniCS FE solver, executed from a Jupyter notebook [44] running within a Docker virtual machine [45].

4. Results and discussion

We now present the performance of a simple gradient descent inversion for various observables using the PT-FD approach. We seek to reconstruct the porosity field related to the synthetic forward results depicted in Figure 6. For benchmarking purposes we assume the synthetic data to be noise-free throughout this study. We further discuss the sensitivity of the solid velocities and fluid fluxes to changes of the porosity. The porosity field used as the 'true' parameter of the steady-state inversions is obtained in the same way as described in the beginning of Section 3, only using a higher resolution. In all the simulations, the initial guess for the porosity field is chosen to be constant and equal to 1% in the entire domain.

The numerical grid resolution contains 1024×1024 grid points in 2-D. The targeted high resolution is only achievable by relying on an HPC approach. We used a GPU-accelerated version of the dynamical cores of the PT-FD inversion procedure, namely the forward and adjoint solvers. These solvers are written in CUDA-C and run on a single Nvidia GPU. In future work we plan to employ a straightforward MPI-based multi-GPU implementation using a similar approach as in [14, 25, 35] in order to port the inversion workflow for 3-D geometries.

4.1. Vertical fluid flux in the entire domain

Here, we use the vertical fluid flux ($\mathbf{u}_f{}^y$) in the entire domain as synthetic data (the lower right-hand panel in Figure 6). Although it is unlikely that this amount of data will ever be available in a practical geological setting, we use it here to show the local sensitivity of the inversion to the anomaly. Figure 7 depicts the results of this inversion.

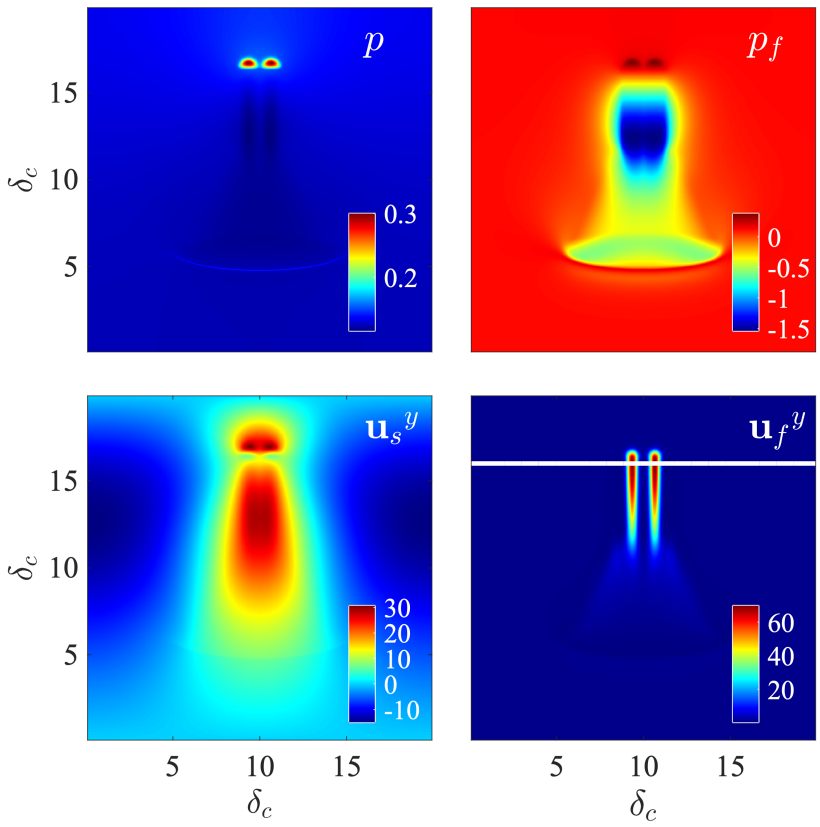

Figure 6: High-resolution 2-D forward numerical simulation results obtained with the PT-FD method on a regular Cartesian grid of 1024×1024 grid points. From left to right: the upper panels depict the total p and fluid p_f pressure respectively; the lower panels depict the vertical (y) solid velocity $\mathbf{u}_s{}^y$ and the fluid flux $\mathbf{u}_f{}^y$. The horizontal line in the $\mathbf{u}_f{}^y$ panel represents the observation horizon used for one of the inversions.

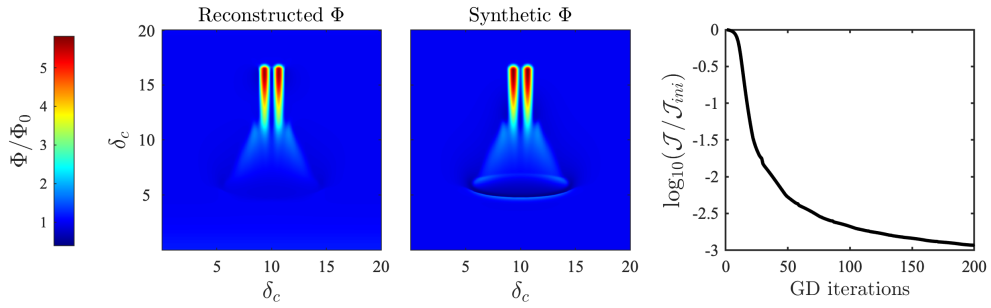

Figure 7: Inversion for the pointwise porosity using the vertical fluid flux in the entire domain as data. The left-hand panel shows the reconstructed porosity field after the inversion is converged, using a homogeneous initial guess. The centre panel depicts the 'true' porosity field that was used to create the observations. The right-hand panel shows the convergence history of the gradient descent algorithm. The value of the cost function was reduced by three orders of magnitude within 200 GD iterations.

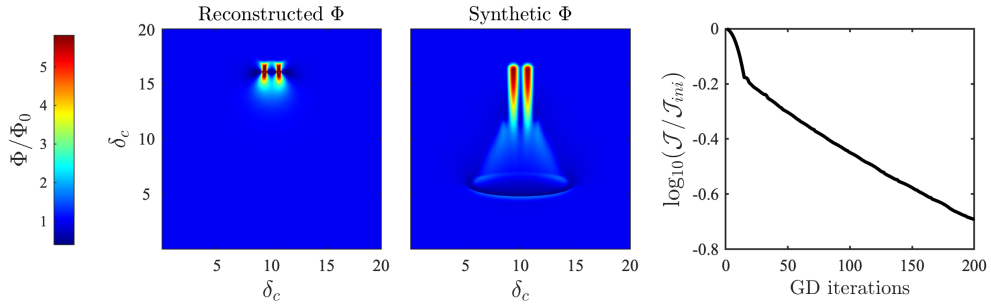

Figure 8: Inversion for the pointwise porosity using the vertical fluid flux along a horizontal line representing the surface as data. The left-hand panel shows the reconstructed porosity field after the inversion is converged, using a homogeneous initial guess. The centre panel depicts the 'true' porosity field that was used to create the observations. The right-hand panel shows the convergence history of the gradient descent algorithm. The value of the cost function could be reduced by half an order of magnitude within 200 GD iterations.

The porosity field can be reproduced consistently in the high-porosity channels that propagate toward the surface. The residual porosity in the initial anomaly is not recovered in all detail by the inversion. We stop the inversion after 200 gradient descent (GD) iterations, which is sufficient to reduce the cost function by three orders of magnitude and to reach a convergence plateau.

4.2. Vertical fluid flux along a line

A more realistic scenario is that the vertical fluid flux is known at a given horizon close to the subsurface. In case of reservoir geology, this could be done by a measurement of the vertical fluid flow at the ocean bottom above a chimney anomaly. We realise this numerically by collecting the synthetic data along a line, depicted as a white horizontal line in Figure 6 (the lower right-hand panel). We choose the line to be within the upper part of the numerical domain and as sufficiently far away from any possible boundaries to avoid boundary effects. We summarise the inversion result in Figure 8. Only the porosity very close to the available observable can be recovered, converging toward the correct porosity value. This highlights the extremely local sensitivity of the fluid flux to the porosity in the two-phase equations. Thus, a change of porosity anywhere else in the domain does not affect the fluid flux at the measurement positions, which is a very different outcome to what one may expect from the Stokes behaviour. We limit the inversion to 200 GD iterations, which reduces the cost function value by half an order of magnitude. Considering that the channelised porosity propagates in a truncated wave-like motion, this observation could be used to determine what the current magnitude of a porosity wave at the surface is.

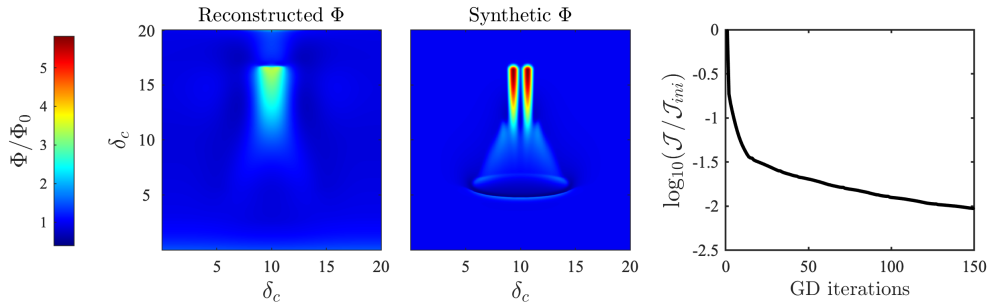

Figure 9: Inversion for the pointwise porosity using the vertical solid velocity in the entire domain as data. The left-hand panel shows the reconstructed porosity field after the inversion is converged, using a homogeneous initial guess. The centre panel depicts the 'true' porosity field that was used to create the observations. The right-hand panel shows the convergence history of the gradient descent algorithm. The value of the cost function was reduced by two orders of magnitude within 150 GD iterations. The solid velocity is less sensitive to the actual chimneys but shows a more diffusive sensitivity produced by the pure density contrast.

4.3. Vertical solid velocity in the entire domain

Alternatively, one can take the solid material displacement $\mathbf{u}_s{}^y$ as an observable (the lower left-hand panel in Figure 6). To generally test this approach, we first assume that the measurement is taken in the entire numerical domain. This simulation reports the sensitivity of the Stokes velocity to the porosity. The results of the inversion are summarised in Figure 9. The inversion creates one large area with slightly increased porosity in the centre of the domain. This decreases the cost function by two orders of magnitude within 150 GD iterations. The recovery is correlated with the actual anomaly, which is located in the centre and depicts a highly non-local sensitivity. The pattern reminds more of the pattern of a Rayleigh-Taylor instability density inversion. A Rayleigh-Taylor instability occurs if a lower-density fluid penetrates a higher-density fluid [e.g. 46]. A possible explanation for this observation is the fact that the Stokes velocity is mostly sensitive to the de-facto density difference rather than the local porosity.

4.4. Sparse observation of vertical fluid flux

To highlight the locality of the vertical fluid flux's sensitivity (the lower right-hand panel in Figure 6) to the porosity, we test the recovery pattern with a randomly initialised correlated Gaussian noise measurement distribution, shown in the small embedded figure in the left panel of Figure 10. This scenario will most likely never be available in geological settings, but could potentially be relevant in material science applications. In the latter case, it may stand to a measure of the density of sensors around the block of porous material. The results of the inversion are summarised in Figure 10. The correct porosity field can only be recovered in areas where observations are available, confirming the highly local sensitivity in the previous results. The cost function is reduced by two orders of magnitude after 60 GD iterations.

4.5. Sensitivity kernels

Instead of calculating the gradient of the cost function with respect to the parameter, one can also calculate the derivative of the forward problem solution with respect to the parameter. We will refer to this quantity as a sensitivity kernel, inspired from seismology [23] or glacial isostatic adjustment modelling [47, 48]. The resulting field indicates in which regions of the domain and by how much a change of the parameter influences the simulation solution. For the specific problem we target, the kernels unveil areas where a change in porosity affects the fluid flux and solid velocity in the entire domain. The kernels can be computed following the same workflow as described in Section 2.3, with the only change that the cost function is defined as

$$\hat{\mathcal{J}}(\Phi) = \int_{\tilde{\Omega}} z \, d\mathbf{x}, \tag{41}$$

where z is one component of the solution of the forward problem corresponding to Φ, defined on a subset $\tilde{\Omega} \subset \Omega$ of the domain. The derivative $\partial\hat{\mathcal{J}}(\Phi)/\partial\Phi$ for an arbitrary porosity field Φ can be calculated using the adjoint framework

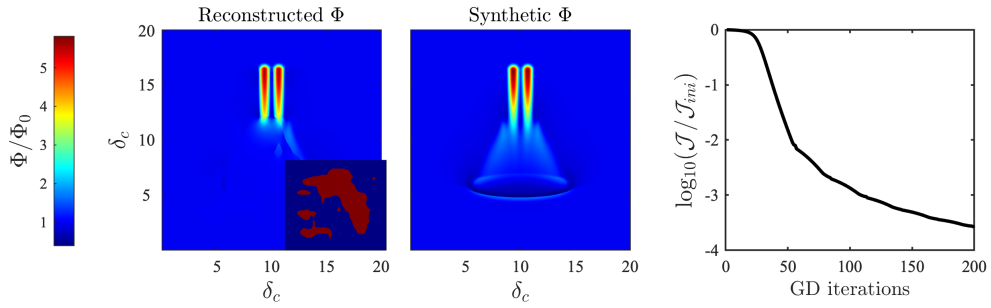

Figure 10: Inversion for the pointwise porosity using the vertical fluid flux at randomised measurement locations as data, emulating sparse observations. The left-hand panel shows the reconstructed porosity field after the inversion is converged, using a homogeneous initial guess. The small embedded figure depicts the measurement points in red. The centre panel shows the 'true' porosity field that was used to create the observations. The right-hand panel shows the convergence history of the gradient descent algorithm. The value of the cost function could be reduced by two orders of magnitude within 60 GD iterations. These results highlights the very local sensitivity of the fluid flux to the porosity.

and represents the sensitivity kernel. These kernels inform about the physical response of the current configuration to a change in the porosity. Figure 11 depicts the sensitivity kernels evaluated at the porosity field Φ used as the synthetic parameter in the inversion. We calculate the sensitivity of the fluid flux and the solid velocity along the observation line also used in Section 4.2 (Figure 6) with respect to the pointwise porosity in the entire domain.

The sensitivities of the vertical and horizontal fluid flux to the porosity are very localised in the chimneys. Measurements along the line do not 'feel' the large scale porosity structure but are only influenced in the vicinity of their position. The sensitivity of the vertical solid velocity is localised toward the tip of the chimneys, further away from the measurement line. The horizontal solid velocity suggests a very diffusive sensitivity in the entire domain. This reveals that the solid velocity is not driving the particular channel propagation but only 'feels' a broader density-driven uplift of parts of the domain. A possible interpretation is that a deviation in the solid velocities from data may be caused by a broader porosity field, while a deviation in the fluid fluxes could be retrieved by very local changes in the porosity field. These results suggest that the hydrological part of the coupled system of equations triggers the highly localised chimneys observed in nature.

These sensitivity kernels are not required in the inverse framework, but can be employed – even if only forward models are performed – as resolution proxy. With the adjoint method the calculation of the kernels is very cheap (there is only one additional linear solve) and one could calculate them after every forward solve (time-step) to trace the sensitivity of the desired observable through time. This opens interesting possibilities to support interpretations in geodynamic modelling, as the kernels highlight whether the interpretation area is in fact sensitive to a given parameter or structure. However, it is very important to note that, compared to for instance the linear wave equation, where the kernels remain the same during a forward simulation, the kernels for the non-linear problem are a function of both the input and the evolution of the forward solution. In the Appendix, we provide a very explicit and simplified example of how the kernels can support interpretations on the 1-D non-linear diffusion equation.

Notably, in the case of the relatively high-resolution 2-D simulations presented, the sharp gradients in the viscosity at the tip of the channels affect the convergence rate of the adjoint solver. To overcome this limitation we apply minor smoothing on the effective adjoint bulk viscosity field in every inverse iteration. We could not recognise any significant effect of this smoothing on the solution while significantly improving the convergence rate. Further investigation of an enhanced physical solution to overcome this limitation is left for future works. We suppose that the effective bulk viscosity function used in the adjoint solve must be sufficiently smooth, because it acts like a source term in the mass-balance equation of the two-phase system.

4.6. The scaling of the iterative method

We evaluate the scalability of the inversion framework based on the iterative PT approach. We vary the number of grid points from 64×64 to 1024×1024 in the x-direction and y-direction. We run the non-linear forward PT-FD solver until the channels' tips reach 85% of the y domain extent and employ a value of $R = 200$. We report the iteration count evolution for various numerical grid resolutions both for the adjoint and the forward solvers repeatedly called in the

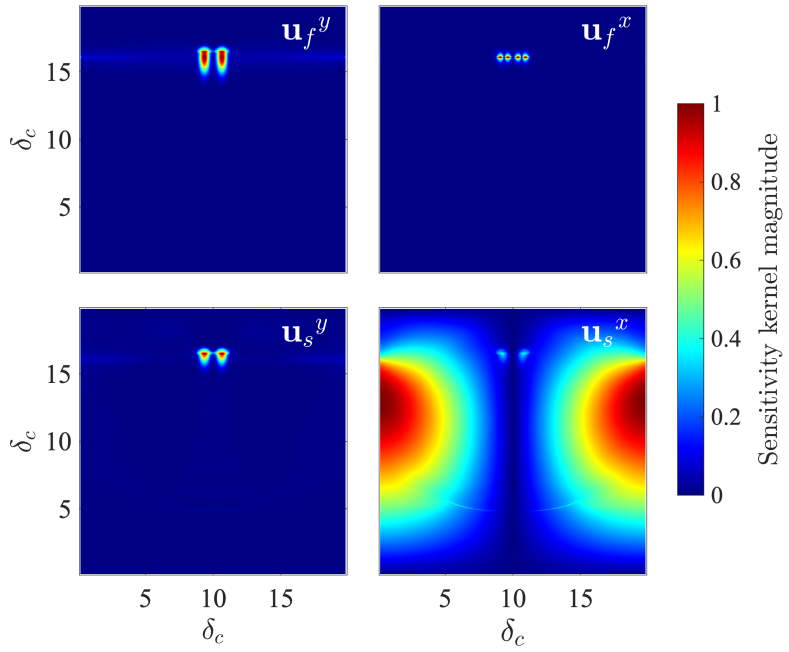

Figure 11: Sensitivity kernels of the fluid and solid velocities with respect to the porosity taking observations in the interior of the domain (excluding boundary nodes). We show the absolute value of the kernel normalised for each kernel separately. The first row depicts the sensitivity of the vertical and horizontal fluid flux to the porosity respectively. The second row depicts the sensitivity of the vertical and horizontal solid velocity to the porosity, respectively.

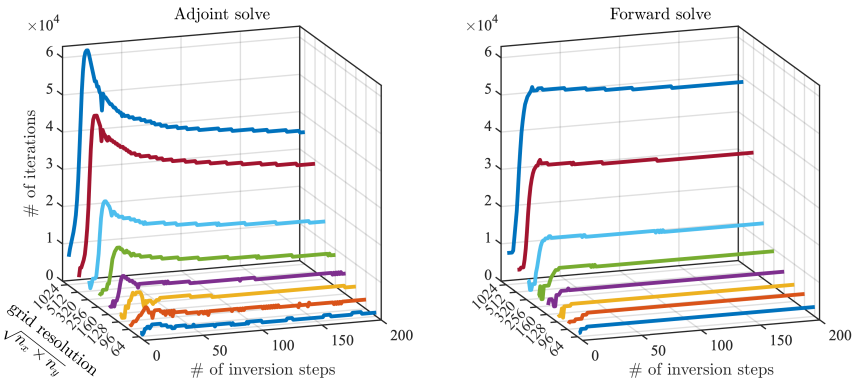

Figure 12: Iteration count evolution plot for the adjoint (left) and the forward (right) solvers for various numerical grid resolutions ranging from 64^2 to 1024^2. The x-axis represents the number of inversion steps within the gradient descent algorithm (here, fixed to 200). The y-axis informs about the numerical grid resolution. The vertical axis reports the number of transient iterations required to solve the problem until the precision threshold of 2×10^{-5} in double precision is reached.

Grid resolution	solver time [sec] / 1000 iterations	
$n_x \times n_y$	Adjoint	Forward
64×64	0.0675	0.1226
96×96	0.0679	0.1117
128×128	0.0649	0.1121
160×160	0.0676	0.1123
256×256	0.0912	0.1374
320×320	0.1231	0.1722
512×512	0.2550	0.3198
1024×1024	0.8550	0.9817

Table 2: Time to perform 1000 PT iterations for both the GPU-based adjoint and the forward solver called within the inversion procedure.

gradient descent routine. The adjoint solver necessitates an increased number of iterations during the first inversion steps. The iteration count further stabilises toward a lower count, continuously decreasing along with the inversion step count (Figure 12). The forward solver iteration count evolution remains fairly stable after the initial increase in the required number of iterations to evaluate the forward solution using the current guess (Figure 12). At the highest tested resolution (1024×1024 grid points), the iteration count of the adjoint solver peaks at 6×10^4 while the forward solver requires about 5×10^4 iterations.

We further analyse how the average global iteration count $\left(N_{\text{ops}}^{\text{avg}}\right)$ for the entire inversion steps evolves with an increase in the numerical grid resolution for both the adjoint and the forward solver (Figure 13). We extract the average iteration count by summing up the number of iterations at each inversion step, dividing it by the total number of inversion steps (here 200), and multiplying it by the total number of grid points N. We do this for each targeted numerical resolution and report it as symbols of the iteration count in a log-log plot; red circles and blue diamonds for the adjoint and forward solver, respectively (Figure 13). We perform the data fit using a line to inform about the scaling order of the method. The average number of operations includes the total number of grid points N as each data point is accessed at least once per iteration in our algorithm. The 2-D PT-FD adjoint and forward solvers scale $O(N^{1.6})$ and $O(N^{1.5})$, respectively, over the entire inversion procedure. Classical references such as multi-grid methods and FFT algorithms characteristically scale $O(N)$ and $O(N \log(N))$, respectively. Lately, [25, 35] showed that the PT-FD solver applied to the hydro-mechanical equations in 3-D scales $O(N^{1.3})$.

Recent studies [25, 38, 37] reported the hardware utilisation efficiency as effective memory throughput (MTP_{eff}) metric due to the memory-bounded nature of the algorithms. We decided not to follow this path here, since the 2-D nature of the current calculations do not saturate the GPUs' memory bandwidth and are thus not as optimal as the 3-D results reported in for instance [25]. Instead, we report the wall-time in seconds needed for the solver to perform 1000 PT iterations at all tested grid resolutions (Table 2). We obtained these results on a Nvidia Tesla V100 Nvlink accelerator featuring 32 GB of internal memory and close to 1 TB/s on-chip memory bandwidth.

5. Conclusions and outlook

We have investigated the general performance and characteristics of an inversion for porosity from hydro-mechanical principles. We derived the adjoint equation system for the two-phase flow equations to perform a gradient-based inversion for porosity. We minimised the misfit between the observed and the calculated vertical fluid fluxes or solid velocities. Technically, we discretised the forward, adjoint, and gradient solvers using a pseudo-transient finite difference scheme that was verified against a finite element framework. Our results unveiled a highly local sensitivity of the vertical fluid fluxes to the porosity. We achieved an almost perfect reconstruction of the porosity field in case the fluid flux was given in the entire domain. However, we only accurately recovered the local porosity if the vertical fluid flux information was sparse. When we used the vertical solid velocity as data, the recovered porosity showed a more 'diffuse' distribution, making it impossible to image the individual channels. This suggests that the hydrological part of the two-phase system mainly explains the localised porosity distribution. Further, we showed that the

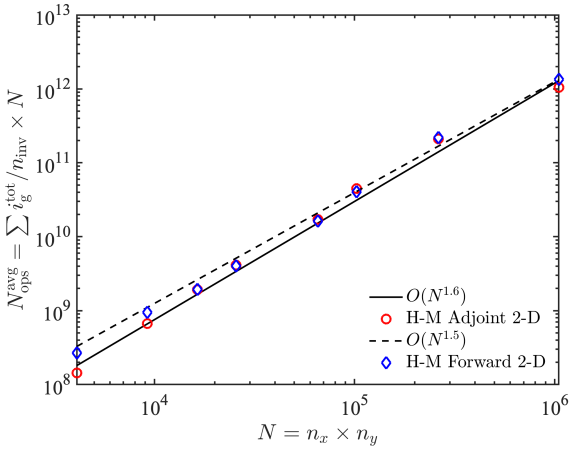

Figure 13: Scaling of the iterative hydro-mechanical (H-M) adjoint inversion procedure as function of total number of grid points N for the adjoint (H-M Adjoint 2-D) and the forward (H-M Forward 2-D) solvers. The y-axis represents the average number of operations $\left(N_{\mathrm{ops}}^{\mathrm{avg}}\right)$ defined as the sum of the overall iteration count over the entire inversion steps for every grid resolution $\left(\sum i_{\mathrm{g}}^{\mathrm{tot}}\right)$ divided by the number of iteration steps (n_{inv}) and multiplied by the total number of grid points N.

PT solvers provide a very efficient framework to solve the system of coupled partial differential equations making perfect use of the latest hardware accelerators, such as GPUs. One complete inverse iteration for the 1024×1024 2-D resolution requires one non-linear forward solve, one linear adjoint solve and one gradient evaluation. The wall-time for this inverse iteration is about 10 minutes on one Nvidia Tesla V100 Nvlink GPU using a (non-optimal) MATLAB implementation of the inversion framework calling CUDA-C sub-routines for the forward, adjoint, and gradient solvers.

The local sensitivity of the vertical fluid flux to the porosity may be promising as an imaging tool in case the time-dependant porosity evolution is solved together with considering the full time history in the inversion. This will require a non-linear forward-in-time and a linear backward-in-time adjoint solution at each time-step. We plan to address these steps in future studies and may expect more informative kernels similar to the full waveform kernels derived from the transient wave equation. The proposed framework adds a missing piece to reservoir-scale inverse problems, since the wave equation is able to image wave-speed anomalies but the interpretation of such anomalies is up to parametrisations. The approach we presented shows how to perform hydro-mechanically consistent inversions for porosity based on flux or velocity constraints. An interesting future step will be to combine both inversions. We may then for instance be able to retrieve the initial conditions of the porosity that led to the observed chimney formation via the inversion procedure.

On the computing side, we ported our 2-D inversion framework to 3-D on multi-GPUs [49] using the extremely scalable Julia ImplicitGlobalGrid framework [50], allowing for both prototyping and deploying large-scale production runs on supercomputers [51, 52]. This work will soon be available as an open source Julia package.

Acknowledgments

The authors thank Martin Hanke, Georg Stadler and Alexander Minakov for fruitful discussions. GSR acknowledges support from the DFG grant KA3367/4. LR acknowledges support from the Swiss National Science Foundation's Early Postdoc.Mobility Fellowship 178075 and computational resources from the Swiss Geocomputing Centre. We thank two anonymous reviewers for their constructive comments.

Author contributions

GSR: Conceptualization, Methodology, Software, Writing - Original Draft, Visualization, Investigation, Formal analysis, Project administration. **LH:** Conceptualization, Methodology, Formal analysis, Writing - Original Draft.

LR: Conceptualization, Methodology, Software, Visualization, Writing - Original Draft, Supervision.

References

[1] D. McKenzie, The generation and compaction of partially molten rock, Journal of Petrology 25 (3) (1984) 713–765.

[2] D. R. Scott, D. J. Stevenson, Magma solitons, Geophysical Research Letters 11 (11) (1984) 1161–1164. doi:10.1029/GL011i011p01161.

[3] J. Connolly, Y. Podladchikov, Decompaction weakening and channeling instability in ductile porous media: Implications for asthenospheric melt segregation, Journal of Geophysical Research: Solid Earth 112 (B10) (2007).

[4] J. F. Rudge, D. Bercovici, M. Spiegelman, Disequilibrium melting of a two phase multicomponent mantle, Geophysical Journal International 184 (2) (2011) 699–718. doi:10.1111/j.1365-246X.2010.04870.x.

[5] Z. Cai, D. Bercovici, Two-phase damage models of magma-fracturing, Earth and Planetary Science Letters 368 (2013) 1–8. doi:10.1016/j.epsl.2013.02.023.

[6] R. F. Katz, S. M. Weatherley, Consequences of mantle heterogeneity for melt extraction at mid-ocean ridges, Earth and Planetary Science Letters 335-336 (2012) 226–237. doi:10.1016/j.epsl.2012.04.042.

[7] T. Keller, D. A. May, B. J. P. Kaus, Numerical modelling of magma dynamics coupled to tectonic deformation of lithosphere and crust, Geophysical Journal International 195 (3) (2013) 1406–1442. doi:10.1093/gji/ggt306.

[8] A. Judd, M. Hovland, Seabed Fluid Flow, Cambridge University Press, Cambridge, 2007. doi:10.1017/CBO9780511535918.

[9] L. Cathles, Z. Su, D. Chen, The physics of gas chimney and pockmark formation, with implications for assessment of seafloor hazards and gas sequestration, Marine and Petroleum Geology 27 (1) (2010) 82–91. doi:10.1016/j.marpetgeo.2009.09.010.

[10] J. Cartwright, C. Santamarina, Seismic characteristics of fluid escape pipes in sedimentary basins: implications for pipe genesis, Marine and Petroleum Geology 65 (2015) 126–140.

[11] A. Plaza-Faverola, S. Bünz, J. Mienert, Repeated fluid expulsion through sub-seabed chimneys offshore Norway in response to glacial cycles, Earth and Planetary Science Letters 305 (3) (2011) 297–308.

[12] H. Løseth, L. Wensaas, B. Arntsen, N.-M. Hanken, C. Basire, K. Graue, 1000 m long gas blow-out pipes, Marine and Petroleum Geology 28 (5) (2011) 1047–1060.

[13] H. Løseth, M. Gading, L. Wensaas, Hydrocarbon leakage interpreted on seismic data, Marine and Petroleum Geology 26 (7) (2009) 1304–1319.

[14] L. Räss, N. S. Simon, Y. Y. Podladchikov, Spontaneous formation of fluid escape pipes from subsurface reservoirs, Scientific reports 8 (1) (2018) 11116.

[15] V. M. Yarushina, Y. Y. Podladchikov, (De) compaction of porous viscoelastoplastic media: Model formulation, Journal of Geophysical Research: Solid Earth 120 (6) (2015) 4146–4170.

[16] R. Y. Makhnenko, Y. Y. Podladchikov, Experimental Poroviscoelasticity of Common Sedimentary Rocks, Journal of Geophysical Research: Solid Earth (2018) 1–18doi:10.1029/2018JB015685.

[17] A. A. Popov, S. V. Sobolev, M. D. Zoback, Modeling evolution of the san andreas fault system in northern and central california, Geochemistry, Geophysics, Geosystems 13 (8) (2012).

[18] T. V. Gerya, F. Meilick, Geodynamic regimes of subduction under an active margin: effects of rheological weakening by fluids and melts, Journal of Metamorphic Geology 29 (1) (2011) 7–31.

[19] G. S. Reuber, A. A. Popov, B. J. Kaus, Deriving scaling laws in geodynamics using adjoint gradients, Tectonophysics 746 (2018) 352–363.

[20] G. Stadler, M. Gurnis, C. Burstedde, L. C. Wilcox, L. Alisic, O. Ghattas, The dynamics of plate tectonics and mantle flow: From local to global scales, Science 329 (5995) (2010) 1033–1038.

[21] G. S. Reuber, B. J. P. Kaus, A. A. Popov, T. S. Baumann, Unraveling the physics of the Yellowstone magmatic system using geodynamic simulations, Frontiers in Earth Science 6 (2018) 117.

[22] S. Ghelichkhan, H.-P. Bunge, The compressible adjoint equations in geodynamics: derivation and numerical assessment, GEM-International Journal on Geomathematics 7 (1) (2016) 1–30.

[23] J. Tromp, C. Tape, Q. Liu, Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels, Geophysical Journal International 160 (1) (2005) 195–216.

[24] A. Fichtner, H.-P. Bunge, H. Igel, The adjoint method in seismology: I. theory, Physics of the Earth and Planetary Interiors 157 (1-2) (2006) 86–104.

[25] L. Räss, T. Duretz, Y. Podladchikov, Resolving hydromechanical coupling in two and three dimensions: spontaneous channelling of porous fluids owing to decompaction weakening, Geophysical Journal International 218 (3) (2019) 1591–1616.

[26] S. P. Frankel, Convergence rates of iterative treatments of partial differential equations, Mathematical Tables and Other Aids to Computation 4 (30) (1950) 65–75.

[27] A. Logg, K.-A. Mardal, G. Wells, Automated solution of differential equations by the finite element method: The FEniCS book, Vol. 84, Springer Science & Business Media, 2012.

[28] L. Armijo, Minimization of functions having Lipschitz continuous first partial derivatives, Pacific Journal of mathematics 16 (1) (1966) 1–3.

[29] M. B. Giles, N. A. Pierce, An introduction to the adjoint approach to design, Flow, turbulence and combustion 65 (3-4) (2000) 393–415.

[30] F. Tröltzsch, Optimal Control of Partial Differential Equations: Theory, Methods and Applications, Vol. 112, American Mathematical Society Graduate Studies in Mathematics, 2010.

[31] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, Springer Science + Business Media B.V., 2009.

[32] A. Fichtner, S. Simutė, Hamiltonian Monte Carlo inversion of seismic sources in complex media, Journal of Geophysical Research: Solid Earth 123 (4) (2018) 2984–2999.

[33] T. Bui-Thanh, O. Ghattas, J. Martin, G. Stadler, A computational framework for infinite-dimensional Bayesian inverse problems Part I: The linearized case, with application to global seismic inversion, SIAM Journal on Scientific Computing 35 (6) (2013) A2494–A2523.

[34] J. Martin, L. C. Wilcox, C. Burstedde, O. Ghattas, A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion, SIAM Journal on Scientific Computing 34 (3) (2012) A1460–A1487.
[35] L. Räss, S. Omlin, Y. Y. Podladchikov, Resolving Spontaneous Nonlinear Multi-Physics Flow Localization in 3-D: Tackling Hardware Limit, GTC Silicon Valley - 2019 (2019).
URL https://developer.nvidia.com/gtc/2019/video/S9368
[36] MATLAB, version 8.4.0 (R2014B), The MathWorks Inc., Natick, Massachusetts, 2014.
[37] T. Duretz, L. Räss, Y. Podladchikov, S. Schmalholz, Resolving thermomechanical coupling in two and three dimensions: spontaneous strain localization owing to shear heating, Geophysical Journal International 216 (1) (2019) 365–379. doi:10.1093/gji/ggy434.
[38] L. Räss, A. Licul, F. Herman, Y. Y. Podladchikov, J. Suckale, Modelling thermomechanical ice deformation using a GPU-based implicit pseudo-transient method (FastICE v1.0), Geoscientific Model Development Discussions 2019 (2019) 1–34. doi:10.5194/gmd-2019-249.
[39] R. Courant, et al., Variational methods for the solution of problems of equilibrium and vibrations, Verlag nicht ermittelbar, 1943.
[40] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al., PETSc users manual (2019).
[41] C. Taylor, P. Hood, A numerical solution of the Navier-Stokes equations using the finite element technique, Computers & Fluids 1 (1) (1973) 73–100.
[42] P.-A. Raviart, J. Thomas, Primal hybrid finite element methods for 2nd order elliptic equations, Mathematics of computation 31 (138) (1977) 391–413.
[43] J. A. D. Connolly, Y. Y. Podladchikov, An analytical solution for solitary porosity waves: dynamic permeability and fluidization of nonlinear viscous and viscoplastic rock, Geofluids 15 (1-2) (2014) 269–292. doi:10.1111/gfl.12110.
[44] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, et al., Jupyter Notebooks-a publishing format for reproducible computational workflows., in: ELPUB, 2016, pp. 87–90.
[45] D. Merkel, Docker: lightweight linux containers for consistent development and deployment, Linux Journal 2014 (239) (2014) 2.
[46] G. I. Taylor, The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. i, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 201 (1065) (1950) 192–196.
[47] O. Crawford, D. Al-Attar, J. Tromp, J. X. Mitrovica, J. Austermann, H. C. Lau, Quantifying the sensitivity of post-glacial sea level change to laterally varying viscosity, Geophysical journal international 214 (2) (2018) 1324–1363.
[48] D. Al-Attar, J. Tromp, Sensitivity kernels for viscoelastic loading based on adjoint methods, Geophysical Journal International 196 (1) (2014) 34–77.
[49] L. Räss, G. S. Reuber, S. Omlin, Nonlinear Multi-Physics 3-D Solver: from CUDA C + MPI to Julia, JuliaCon 2020, online everywhere on Earth (2019).
URL https://www.youtube.com/watch?v=1t1AKnnGRqA
[50] S. Omlin, L. Räss, Y. Y. Podladchikov, Implicitglobalgrid.jl, https://github.com/eth-cscs/ImplicitGlobalGrid.jl (2019).
[51] L. Räss, S. Omlin, Y. Y. Podladchikov, Nonlinear Multi-Physics 3-D Solver: from CUDA C + MPI to Julia, JuliaCon 2019, Baltimore MD, USA (2019).
URL https://www.youtube.com/watch?v=b90qqbYJ58Q
[52] S. Omlin, L. Räss, G. Kwasniewski, B. Malvoisin, Y. Y. Podladchikov, Nonlinear Multi-Physics 3-D Solver: from CUDA C + MPI to Julia, JuliaCon 2020, online everywhere on Earth (2019).
URL https://www.youtube.com/watch?v=vPsfZUqI4_0

6. Appendix

6.1. Non-linear diffusion

We will now outline the workflow of deriving the adjoint equation with the formal Lagrangian approach for a simple 1-D non-linear diffusion example on the interval $[a, b] \subset \mathbb{R}$ and will then discretise the resulting equations in various ways. In particular, we will consider a non-linear steady-state diffusion equation with Dirichlet boundary conditions:

$$-\frac{d}{dx}\left(-D(u)\,u'\right) = 0 \quad \text{on } (a,b), \tag{42}$$

$$u(a) = u_a, \tag{43}$$

$$u(b) = u_b, \tag{44}$$

where u_a, $u_b > 0$. The diffusion coefficient D is non-linear in u:

$$D(u) = u^n, \tag{45}$$

where n is some spatially varying power-law exponent. Multiplying equation (42) by the negative of a test function $v \in H_0^1(a,b)$ and integrating by parts yields the weak form of the non-linear diffusion equation:
Find $u \in U := \{u \in H^1(a,b) \mid u(a) = u_a,\, u(b) = u_b\}$ such that

$$\int_a^b u^n u' v'\, dx = 0 \tag{46}$$

for all test functions $v \in H_0^1(a,b)$.

We seek to reconstruct the power-law exponent function n from observations u_{obs}, which we assume to be available in the entire interval (a,b) for simplicity. We formulate this inverse problem as an infinite-dimensional constrained non-linear least-squares optimisation problem, i.e. we seek to minimise the cost function

$$\mathcal{J}(n) = \frac{1}{2}\int_a^b (u - u_{\text{obs}})^2\, dx, \tag{47}$$

where $u = u(n)$ denotes the solution of the forward problem (46) corresponding to the parameter field n.

To obtain the gradient $d\mathcal{J}/dn$ in the formal Lagrangian approach, the first step is to define the so-called Lagrangian functional, which is the sum of the cost function and the weak form of the forward problem:

$$\mathcal{L}(u,v,n) = \frac{1}{2}\int_a^b (u - u_{\text{obs}})^2\, dx + \int_a^b u^n u' v'\, dx. \tag{48}$$

In this approach, the variables u, v, and n are assumed to be independent of one another. By taking variations with respect to v and requiring them to vanish for all directions $\hat{v} \in H_0^1(a,b)$, one simply recovers the weak form of the forward problem. On the other hand, differentiating $\mathcal{L}$ with respect to u in a direction $\hat{u} \in H_0^1(a,b)$ and setting it to zero yields the weak form of the adjoint equation:
Find $v \in H_0^1(a,b)$ such that

$$\mathcal{L}_u(u,v,n)\hat{u} = \int_a^b (u - u_{\text{obs}})\hat{u}\, dx + \int_a^b v'\left(u^n \hat{u}' + n u^{n-1} u' \hat{u}\right) dx = 0 \tag{49}$$

for all test functions $\hat{u} \in H_0^1(a,b)$, where u is the solution of the forward problem corresponding to the power-law exponent n. Note that the adjoint equation (49) is linear in contrast to the forward problem.

The remaining variation yields the weak form of the gradient $d\mathcal{J}/dn$:

$$\mathcal{L}_n(u,v,n)\hat{n} = \int_a^b \hat{n}\log(u) u^n u' v'\, dx, \tag{50}$$

where u is the solution of the forward problem corresponding to n and v is the corresponding solution of the adjoint equation. This set of weak form equations (46), (49), and (50) can be readily discretised using a finite element method. The finite difference discretisation requires the strong form of the adjoint and gradient equation, which can be derived by isolating the test functions and, if necessary, integrating by parts:

$$n u^{n-1} u' v' - \frac{d}{dx}\left(u^n v'\right) = u_{\text{obs}} - u \quad \text{on } (a,b), \tag{51}$$

$$v(a) = v(b) = 0, \tag{52}$$

$$\frac{d\mathcal{J}}{dn} = \log(u) u^n u' v', \tag{53}$$

where u is the solution of the forward problem corresponding to n and v is the corresponding solution of the adjoint equation.

For our test simulations, we choose $[a,b] = [-10,10]$ and the Dirichlet boundary values $u_a = 1$, $u_b = 0.5$. Synthetic data for the inversion is created with the constant power-law exponent $n \equiv 3$.

We discretise the equations both with a finite element method using FEniCS and with a MATLAB implementation of the PT-FD scheme. For the FEM implementation we use 64 linear Lagrange elements both for the forward and the adjoint equations. In the PT-FD case, the domain is discretised by 64 cells and the quantity u is initialised as constant and equal to 1.0 in the entire domain. The initial guess for the inversion using the method of steepest descent is chosen to be constant $n \equiv 1$. Since the gradient approximation is similar with both discretisations, we only show numerical results obtained with the inversion using the PT-FD approach (Figure 14).

Following the previously reported approach (Section 4.5), we present the sensitivity kernels for the 1-D non-linear diffusion equation. The new function that we seek the sensitivity of reads

$$\hat{\mathcal{J}}(n) = \int_a^b u\, dx. \tag{54}$$

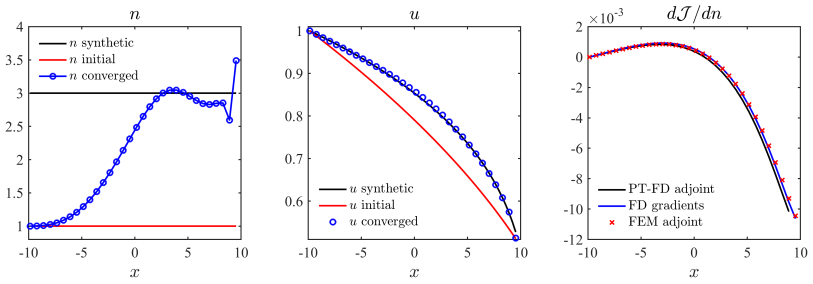

Figure 14: Inversion for the power-law exponent n function in the 1-D non-linear diffusion problem solved with the PT-FD method. The left-hand panel shows the synthetic, initial and inferred power-law exponent n. The middle panel depicts the quantity u in its initial, synthetic and converged state. The right-hand panel shows the comparison of the gradients for three different discretisations: an adjoint-based PT-FD, a direct FD approximation to verify the results, and an adjoint-based FEM. All three methods are in good agreement.

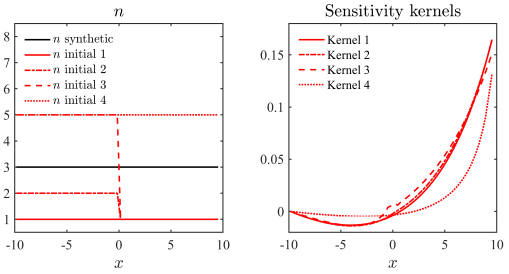

Figure 15: Sensitivity kernels for the 1-D non-linear diffusion equation. The left-hand panel shows four initial guesses for n, including a step function located in the centre of the domain and a homogeneous guess for n. The right panel depicts the resulting sensitivity kernels, considering u at every node.

Following the derivation described earlier, the only difference is the right-hand-side of equation (51):

$$nu^{n-1}u'v' - \frac{d}{dx}\left(u^n v'\right) = -1 \quad \text{on } (a,b). \tag{55}$$

Evaluating the gradient equation (53) subject to $\hat{\mathcal{J}}$ portrays how significantly sensitive the solution quantity u is to the power-law exponent n. This implies that any interpretation made on the value of n where the kernel is zero is not significant since u does not depend on n in that particular location.

We report an explicit example calculating the kernel of n for u given in the entire domain (Figure 15). For all four guesses of n, the quantity u only weakly depends on the value of n in the left half of the domain, but is significantly influenced by the choice of n in the right half of the domain. For an inversion, we now expect the reconstruction of n in the right half of the domain to be more reliable than for the quantities n in the left half. The inversions for the four initial guesses (kernel 1-4 in Figure 15) do in fact converge to the synthetic value $n = 3$ in the right half of the domain, but only slightly modify the initial value of n in the left half (left column in Figure 14). The sensitivity kernels yield an explanation for this behaviour; in the right half of the domain, the parameter n significantly influenced the outcome of forward simulation results. In the left part, however, the 'resolution' of the equation is fairly limited, since the behaviour of u is mainly controlled by the Dirichlet boundary condition and not by the value of n. This is why the kernel for n is close to zero on the left half, i.e. a change in n may not affect u.

However, the discussed underlying equation is non-linear, and the kernel is neither invariant to the evolution of the forward solution nor to the boundary conditions. In fact, a change in the Dirichlet boundary conditions may result in a qualitatively different kernel structure that needs to be interpreted for this particular setup. Nonetheless, since the computation of the kernel is very cheap (there is only one additional linear solve), one could still evaluate the kernel

for each forward solve (time-step). Doing so may support the interpretation of the modelling unveiling which area of interest the equation is in fact sensitive to.

The code used for the simulations in the Appendix can be accessed from the Bitbucket repository: `https://bitbucket.org/hydromechanics_adjoint_pt/hm_adj_pt/src/master/`