Computational Geosciences
https://doi.org/10.1007/s10596-024-10320-y

ORIGINAL PAPER

Application of deep learning reduced-order modeling for single-phase flow in faulted porous media

Enrico Ballini[1] · Luca Formaggia[1] · Alessio Fumagalli[1] · Anna Scotti[1] · Paolo Zunino[1]

Received: 21 March 2024 / Accepted: 8 August 2024

Abstract
Our research is positioned within the framework of subsurface resource utilization for sustainable economies. We concentrate on modeling the underground single-phase fluid flow affected by geological faults using numerical simulations. The study of such flows is characterized by strong uncertainites in the data defing the problem due to the difficulty of taking precise measurements in the subsoil. We aim to demonstrate the feasibility of a reduced order model that is both reliable and computationally efficient, thereby facilitating the incorporation of uncertainties. We account for the uncertainities of the properties of the rock and the geometry of the fault. The latter is achieved by using a radial basis function mesh deformation method. This approach benefits from a mixed-dimensional framework to model the rock matrix and faults as n and $n-1$ dimensional domains, allowing for non-conforming meshes. Our primary focus is on a reduced-order model capable of reproducing flow variables across the entire domain. We utilize the Deep Learning Reduced Order Model (DL-ROM), a nonintrusive neural network-based technique, and we compare it against the traditional Proper Orthogonal Decomposition (POD) method across various scenarios. The most relevant contributions of this work are: the proof of concept of the use of neural network for reduced order models for subsoil flow, dealing with non-affine problems and mixed dimensional domain. Additionally, we generalize an existing mesh deformation method for discontinuous deformation maps. Our analysis highlights the capability of reduced order model, highlighting DL-ROM's capacity to expedite complex analyses with promising accuracy and efficiency, making multi-query analyses with various quantities of interest affordable.

Keywords Porous media · Faults · Reduced order modeling · Proper orthogonal decomposition · Deep learning

1 Introduction

Nowadays, the practice of injecting fluids into the subsoil is gaining prominence not only for the production of fossil fuels but also for the storage of carbon dioxide and for the strategic storage of thermal energy and the exploitation of geothermal resources. These applications are essential in our quest for sustainable and renewable energy solutions [1]. Injection of fluid into the subsoil alters the local equilibrium, changing the stress field, and possibly causing fault reactivation. Furthermore, leakage phenomena from storage reservoirs must be predicted [2]. These phenomena have to be assessed taking into account the uncertainties related to the subsoil, both those concerning the physical property of the rock, e.g. the porosity or the permeability, and uncertainties about the geometry, such as the exact position of the fault or the relative displacement of the two sides.

Due to this lack of knowledge, we need to execute multi-query applications such as sensitivity analysis, parameter identification, or uncertainty quantification, which require many evaluations of the discrete model for different scenarios. The evaluation of each scenario can be computationally demanding; therefore, these analyses can be a burden. Reduced Order Modeling (ROM) techniques come into play to provide a surrogate model that is both reliable and fast

✉ Enrico Ballini
enrico.ballini@polimi.it

Luca Formaggia
luca.formaggia@polimi.it

Alessio Fumagalli
alessio.fumagalli@polimi.it

Anna Scotti
anna.scotti@polimi.it

Paolo Zunino
paolo.zunino@polimi.it

[1] MOX, Department of Mathematics, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

Published online: 28 August 2024

to evaluate, making such intensive computation feasible. Reduced order models can be created using linear reduction techniques [3–6] with methods such as Proper Orthogonal Decomposition (POD) [3–5], greedy algorithm [3, 4], empirical interpolation method [3, 7], dynamic mode decomposition [8, 9]. Moreover, the methods can be adopted in a multi-fidelity context [10] or in dynamic adaptation [11]. Some other works investigate the combination of POD for finding spatial basis functions and neural networks to evolve the modal coefficients [12–14]. The primary purpose of employing data-driven ROM empowered by neural networks, as an alternative to traditional approaches, is the built-in non-intrusiveness and the ability to overcome the limitation of using a linear combination of basis functions by introducing a nonlinear trial manifold. This is achieved using autoencoders, which offer advantages in highly nonlinear or advection-dominated problems [15, 16]. When determining the latent space using autoencoders, the temporal evolution of the latent solution can be accomplished using various methods, including recurrent neural networks [17], feedforward neural networks [18–23], and self-attention neural networks [24]. It is also feasible to uncover the latent dynamics and compute a quantity of interest from it, as demonstrated in [25].

In this paper, we focus on the problem of incompressible, non-reactive, single-phase flow in a faulted porous medium, considering both the physical and geometrical variability of the data. For the latter, we choose to account for the geometrical changes of the fault configuration by deforming the computational grid using a method based on algebraic equations. We apply a new data-driven model order reduction technique based on deep feedforward neural networks, called the Deep Learning Reduced Order Model (DL-ROM) [18, 22]. This method is intrinsically non-intrusive and naturally capable of efficiently dealing with nonaffine parameterizations, such as the one used for the changing geometry of the faults. We compare DL-ROM with the well-established POD method [3, 4], using several test cases on the problem of flows in fractured porous media with deformable geometry.

We organize the development and assessment of the proposed methodology as follows. In Section 2 we present the mathematical model that governs single-phase flow in a porous medium with particular attention to the treatment of the coupling of subdomains of different dimensions. The discretization of this model is described in Section 3. Section 4 regards the reduced-order modeling techniques, with an introduction of both the POD and DL-ROM methods and a discussion of the nonaffine parameterization of our problem. In Section 5, we present our methodology for deforming the geometry while maintaining a non-conforming mesh at the interfaces of subdomains. In Section 6, we set up three different test cases to assess the main features that define the properties of the methods, such as offline and online time and solution error compared to the full-order model. Finally, in Section 7 we show an example of two possible applications of the proposed model order reduction technique.

2 Mathematical model

We present the mathematical model of an incompressible, non-reactive, single-phase flow in a porous medium, showing the treatment of the faults. The reader can find a complete list of symbols used in this document in Appendix A.

2.1 The continuous model

Let us consider $\Omega \subset \mathbb{R}^D$, with $D = 2$ or 3, a sufficiently regular domain that represents a porous medium, with outer boundary $\partial\Omega$ with outward normal υ. We assume that there exists a partition of $\partial\Omega$ into two measurable parts $\partial_p\Omega$ and $\partial_q\Omega$, such that $\overline{\partial\Omega} = \overline{\partial_p\Omega} \cup \overline{\partial_q\Omega}$ and $\partial_p^{\circ}\Omega \cap \partial_q^{\circ}\Omega = \emptyset$, with $|\partial_p\Omega| \neq 0$. We consider a Darcy model for flow in a saturated porous medium, where the Darcy velocity q, in [m s^{-1}], and the fluid pressure p, in [Pa], satisfy the following system of partial differential equations consisting of the Darcy law and the mass balance [26], with associated boundary conditions.

$$\begin{cases} q + K\nabla p = 0 \\ \nabla \cdot q = f \end{cases} \quad \text{in } \Omega, \qquad \begin{cases} p = \overline{p} & \text{on } \partial_p\Omega, \\ q \cdot \upsilon = \overline{q} & \text{on } \partial_q\Omega, \end{cases} \tag{1}$$

The data and parameters of the model are the permeability of the rock matrix scaled by dynamic viscosity K, in [m^{3}s/kg], a scalar forcing term f, in [s^{-1}] (representing a source or a sink), and the boundary data $\overline{p}$ and $\overline{q}$, in [Pa] and [m s^{-1}], respectively. We assume that K is a bounded, symmetric, positive-definite tensor.

In this work, we consider the primal formulation associated with Eq. 1, where the only variable is the pressure p, and the Darcy velocity may be reconstructed using Darcy's law. It reads

$$-\nabla \cdot (K\nabla p) = f \quad \text{in } \Omega, \qquad \begin{cases} p = \overline{p} & \text{on } \partial_p\Omega, \\ K\frac{\partial p}{\partial \upsilon} = -\overline{q} & \text{on } \partial_q\Omega, \end{cases} \tag{2}$$

where $\frac{\partial p}{\partial \upsilon} = \nabla p \cdot \upsilon$. Under suitable regularity assumptions on the domain shape, forcing, and boundary terms, it is well known that the problem is well posed and admits a unique weak solution p in $H^1(\Omega)$ as long as $\partial_p\Omega$ has a non-null measure, see [27]. If $|\partial_p\Omega| = 0$, p is known up to a constant.

2.2 Flow in faulted porous medium

Fractures or faults might be present in our domain of interest; these are regions with one dimension, the aperture, that

may be orders of magnitude smaller than the lateral extension. For this reason, we adopt a model reduction strategy that considers fractures as objects of codimension 1. Consequently, we need to write a model for the flow inside the fracture by appropriately averaging the Darcy model in the Eq. 1. In the reduced model, the aperture becomes a parameter. The model applies to fractures and faults, yet because of the target application, we will use only the term faults in the sequel.

Faults might have permeabilities very different compared to the surrounding porous media. When they are more permeable than the surrounding medium, they form a preferential flow path. In contrast, they block the flow and may create compartments in the porous medium when they have a lower permeability. We are interested in both cases: for this reason, we consider the mixed-dimensional model already used, among others, in [28–32], which accounts for both situations.

For the convenience of the readers, we introduce the mathematical model when only one fault, γ, is present, as shown in Fig. 1; the intersection between faults will be treated later. Moreover, we assume that the fault is planar with a unitary normal υ_γ and an aperture $\epsilon > 0$. The variables in the porous medium, Ω, are the same as before, while in the fault we consider the following reduced variables,

$$q_\gamma(x) = \int_{\epsilon(x)} (I - \upsilon_\gamma \otimes \upsilon_\gamma) q \quad \text{and} \quad p_\gamma(x) = \frac{1}{\epsilon(x)} \int_{\epsilon(x)} p.$$

Here, q_γ is an integrated tangential flux in $[\mathrm{m}^2\,\mathrm{s}^{-1}]$, and p_γ an averaged pressure in [Pa].

The normal υ_γ allows us to uniquely define the positive and negative side of the fault, as shown in Fig. 1. On each side, we introduce an additional interface, called γ^+ or γ^-, where we define a new variable λ^+ or λ^- that represents the flux exchange between the fault and the porous medium, both measured in $[\mathrm{m}\,\mathrm{s}^{-1}]$. We will simplify the notation by indicating $\lambda = (\lambda^+, \lambda^-)$.

We note that now the domain Ω does not include the fault, which is, in fact, an internal boundary. Thus, the boundary of Ω can be partitioned into two open sets: $\partial_{ex}\Omega$ and $\partial_{in}\Omega$.

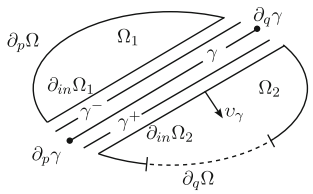


Fig. 1 Generic domain divided in Ω_1 and Ω_2 by a fault, γ, with normal υ_γ. The coupling fluxes, λ^+ and λ^-, are defined at the additional interfaces, γ^+ and γ^-. The parts of the external boundary with the Neumann conditions are indicated by $\partial_q\Omega$ and $\partial_q\gamma$, while the external boundaries with the Dirichlet conditions are denoted by $\partial_p\Omega$ and $\partial_p\gamma$. The internal boundaries facing the fault are called $\partial_{in}\Omega_1$ and $\partial_{in}\Omega_2$

The first indicates the external boundary, $\partial_{ex}\Omega \cap \Gamma = \emptyset$, and the second indicates the part of the boundary of Ω facing the fault. Similarly, the boundary of γ is subdivided into two (possibly empty) disjoint parts: $\partial_{ex}\gamma$ and $\partial_{in}\gamma$, the former being the portion of the boundary of γ in contact with $\overline{\partial_{ex}\Omega}$, while $\partial_{in}\gamma$ is the boundary of γ internal to Ω, characterized by $\overline{\partial\Omega} \cap \overline{\partial_{in}\gamma} = \emptyset$. The external boundary $\partial_{ex}\gamma$ is again divided into two disjoint parts, possibly empty, $\partial_p\gamma$ and $\partial_q\gamma$ where we impose pressure and fluxes, respectively. Finally, we define $\hat{\upsilon}_\gamma$ as the unit normal outward of $\partial\gamma$ (tangent to γ).

According to the mixed-dimensional model described in the cited literature, we can write the primal formulation for the coupled problem where the unknowns are p in Ω, p_γ in γ and (λ^+, λ^-) in $\gamma^+ \times \gamma^-$. We have

$$-\nabla\cdot K\nabla p = f, \text{ in } \Omega, \quad -\nabla\cdot\epsilon K_\tau \nabla p_\gamma + \lambda^- - \lambda^+ = f_\gamma, \text{ in } \gamma, \tag{3a}$$

$$\begin{cases} \epsilon\lambda^+ - 2K_n\left(p\big|_{\gamma^+} - p_\gamma\right) = 0, & \text{on } \gamma^+, \\ \epsilon\lambda^- - 2K_n\left(p_\gamma - p\big|_{\gamma^-}\right) = 0, & \text{on } \gamma^-, \end{cases} \tag{3b}$$

where Eq. 3a represents the mass balances in, respectively, Ω and γ, whereas Eq. 3b is the constitutive law for the fluxes λ. K_τ and K_n are the in-plane and normal permeability of the fault, scaled by the dynamic viscosity, respectively, both expressed in $[\mathrm{m}^3\mathrm{s/kg}]$. Furthermore, f_γ is the averaged source or sink term in the fault, in $[\mathrm{s}^{-1}]$. Velocities can be reconstructed by Darcy's law, which for the fracture is expressed as $q_\gamma = -\epsilon K_\tau \nabla p_\gamma$. Note that the differential operators in γ are defined with respect to a local coordinate system, but we retain the same notation for simplicity. Here, with $|_{\gamma^+}$ and $|_{\gamma^-}$, we indicate the trace operators on the respective sides of the fault.

System Eqs. 3a with 3b is complemented by the following boundary conditions

$$\begin{aligned} &p = \overline{p} \quad \text{on } \partial_p\Omega, \qquad K\frac{\partial p}{\partial \upsilon} = -\overline{q}, \quad \text{on } \partial_q\Omega, \\ &p_\gamma = \overline{p_\gamma} \quad \text{on } \partial_p\gamma, \qquad \epsilon K_\tau \frac{\partial p_\gamma}{\partial \upsilon_\gamma} = -\overline{q_\gamma} \quad \text{on } \partial_q\gamma, \\ &\qquad\qquad \frac{\partial p_\gamma}{\partial \upsilon_\gamma} = 0 \quad \text{on } \partial_{in}\gamma. \end{aligned} \tag{3c}$$

The last relation, known as the tip condition, imposes a null flux.

2.3 Intersections

We focus here on the intersection of 1D faults because it is the only kind of intersection that we consider in this paper.

Different intersection topologies can exist, such as X-, T-, or Y-shaped intersections; see Fig. 2 for an example of a Y-shaped intersection. All types of intersection are treated similarly, as explained below. The intersection point is represented as a 0D domain where the mass conservation equation and the coupling conditions must be solved. The 0D mass conservation is:

$$\sum_{i=0}^{n_{br}} \lambda_{\gamma_i} = 0,$$

where n_{br} is the number of intersecting branches, λ_{γ_i} [m s^{-1}] are the fluxes exchanged between the branches through the intersection point, ι: $\lambda_{\gamma_i} = q_{\gamma_i}|_\iota \cdot \hat{\upsilon}_{\gamma_i}$, where $\hat{\upsilon}_{\gamma_i}$ is the outwards unit vector aligned with the i-th fault. Regarding the coupling condition, we can use a model considering pressure jumps as:

$$\epsilon^2 \lambda_\gamma + 2K_\iota \left(p_\iota - p_\gamma|_\iota\right) = 0,$$

where K_ι [m^{3}s/kg] is a representative value of permeability at the intersection, p_ι [Pa] is the pressure of the intersection. The well-posedness of the problem of flows in faulted porous media, including intersections, for a different, yet equivalent, formulation has been studied in [33, 34].

3 Discretization

We discretize problem (3) with a cell-centered finite volume method [35, 36]. The degrees of freedom of the primary variables p, p_γ, and λ are located in the center of the cells that discretize the domain and fractures. Numerical fluxes at the cell boundary are computed with the multipoint flux approximation (MPFA) [37–40], in particular, the so-called MPFA-O method first proposed in [37] implemented in the Porepy library [41]. We stress that our objective is to solve the problem for different values of the geometrical and physical parameters, respectively μ_{geom} and μ_{phy}. We set $\mu = (\mu_{geom}, \mu_{phy}) \in \Theta \subset \mathbb{R}^e$, where Θ is the space of parameters, and e the total number of parameters. Variations

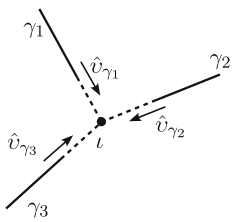


Fig. 2 Y-shaped intersection. The three faults, γ_1, γ_2, γ_3, cross each other at the point ι. $\hat{\upsilon}_{\gamma_i}$ are the outward unit vectors aligned with the respective fault

in geometric parameters lead to domain deformations. We detail the issues related to domain discretization in Section 5.

For a given grid, the discretization of the full-order model (3a) leads to a perturbed saddle point linear system of the form

$$\begin{bmatrix} A & B_1^T \\ B_2 & -C \end{bmatrix} \begin{bmatrix} \hat{\mathfrak{p}} \\ \boldsymbol{\lambda} \end{bmatrix} = b_N, \tag{4}$$

where $A = \begin{bmatrix} A_p & 0 \\ 0 & A_{p_\gamma} \end{bmatrix}$ is a semi-definite positive matrix containing the discretization of the fluxes in Eq. 3a, precisely A_p discretizes the fluxes in Ω and A_{p_γ} discretizes the fluxes in γ. The terms B_1 and B_2 derive from the coupling conditions, involving the fluxes λ, while C is the mass matrix that appears in the discretization of the constitutive law (3b). We note that Eq. 5 satisfies the assumptions for well-posedness; for example, see [27] Sect. 5.5. The unknown $\hat{\mathfrak{p}} = [\mathfrak{p}^T, \mathfrak{p}_\gamma^T]^T$ contains the degrees of freedom $\mathfrak{p}$ and $\mathfrak{p}_\gamma$ related to p and p_γ, while $\boldsymbol{\lambda}$ contains the degrees of freedom of λ. The previous system can be rewritten in the following compact form.

$$A_N u_N = b_N. \tag{5}$$

Here, $A_N \in \mathbb{R}^{N\times N}$, $u_N \in \mathcal{V}_N = \text{col}(A_N) \subset \mathbb{R}^N$ is the unknown vector consisting of all degrees of freedom $u_N = [\hat{\mathfrak{p}}^T, \boldsymbol{\lambda}^T]^T$, $b_N \in \mathbb{R}^N$, N is the number of degrees of freedom of the full order model. All these quantities depend on μ, which is not explicitly indicated to simplify the notation.

4 Model order reduction

For several applications, it is necessary to repeatedly query a model with various input values to perform tasks such as sensitivity analysis, parameter estimation through inverse problem solving, or uncertainty quantification. When the model is complex and computationally demanding, managing multiple queries becomes impractical. Therefore, the development of a fast and reliable surrogate model is crucial.

We address this problem by approximating the *solution manifold*, $\mathcal{S} := \{u_N\}_{\mu\in\Theta}$, of the flow problem in faulted porous media (3) through a reduced model that takes as input the physical and geometric parameters μ_{phy}, μ_{geom}, and returns an estimate of the corresponding variables p, p_γ, λ. The key idea under the model order reduction procedure is that a few main patterns characterize the parametric dependence of a problem described by many degrees of freedom (d.o.f.). Thus, we exploit these patterns to formulate a smaller problem that is easier to solve. Then we reconstruct an approximation of the desired solution using a linear or nonlinear map $\mathcal{M}$ from the reduced space to the full-order space.

Model order reduction can be model-based or data-driven [3, 5, 42]. The former uses the model equations in their differential or discrete form to derive a reduced version. It has the property of being intrusive, which means that we need to manipulate the equations and, consequently, modify the solver code to implement the reduced model. In the latter case, instead, the reduced model is built using a collection of data through which the dynamics of the system can be inferred; the procedure is non-intrusive because it is equations and software-agnostic.

In the following sections, we apply two reduced-order model techniques to the problem of flows in a porous medium with faults. The first is the Proper Orthogonal Decomposition (POD), which is model-based and linear. The second technique, called DL-ROM, is data-driven and approximates the nonlinear map $\mathcal{M}$ using neural networks.

4.1 Proper orthogonal decomposition

In this case, the map $\mathcal{M}$ is linear, and it is represented by an orthogonal matrix $\Phi \in \mathbb{R}^{N\times n}$ called *transition matrix*, with $n < N$. A slightly different definition is considered in Section 4.2, where a non-linear map replaces the linear map, Φ. The matrix Φ contains information about the basis functions that represent the full-order solutions. Typical ways to compute it are the singular value decomposition (SVD) and the greedy algorithm [3, 4]. We now briefly recall the SVD process. A number n_s, $n < n_s < N$, of full-order model solutions, called snapshots, are calculated and collected in the snapshot matrix,

$$S = \left[u^{(1)}|\dots|u^{(n_s)}\right]. \tag{6}$$

Then, the singular value decomposition factorizes S as $S = U\Sigma V^H$, where $U \in \mathbb{R}^{N\times N}$ is a unitary matrix whose columns are the left singular vectors, $\Sigma \in \mathbb{R}^{N\times n_s}$, is a rectangular diagonal matrix containing the singular values in decreasing order, $V \in \mathbb{R}^{n_s\times n_s}$ is a unitary matrix whose columns are the right singular vectors. Φ will be formed by the first n columns of U associated with the highest singular values $\Phi = U_{tr}$, where $[U_{tr}]_{ij} = [U]_{ij}$, $i = 1,\dots,N$, $j = 1,\dots,n$. This choice of Φ is optimal in the sense that it minimizes the reconstruction error in the snapshots:

$$\sum_{i=1}^{n_s} \|u^{(i)} - \Phi\Phi^\top u^{(i)}\|_2^2 = \min_{W\in\mathbb{R}^{N\times n}} \sum_{i=1}^{n_s} \|u^{(i)} - WW^\top u^{(i)}\|_2^2.$$

The proof follows the Eckart-Young theorem [43], originally discovered by Schmidt in the continuous framework [44]; see [3] for a detailed proof. The previous equation tells us that the POD is an optimal linear reduced-order modeling approach in the sense described previously.

Given the basis functions stored in the columns of the transition matrix Φ, we define the *reconstructed solution*, $\tilde{u}_N$, belonging to the full-order space $\mathcal{V}_N$, obtained from the reduced basis solution:

$$\tilde{u}_N = \Phi u_n. \tag{7}$$

where $u_n \in \mathcal{V}_n \subset \mathbb{R}^n$ are the *reduced coefficients*, solution of the reduced problem.

A possible strategy to retrieve u_n is to make the residual orthogonal to the linear subspace defined by the column of Φ, i.e., to compute a Galerkin projection into the linear subspace $S_n = \mathrm{col}(\Phi)$,

$$\Phi^\top (b_N - A_N\Phi u_n) = 0,$$

which yields

$$\Phi^\top A_N\Phi u_n = \Phi^\top b_N.$$

Defining $A_n = \Phi^\top A_N\Phi u_n$, and $b_n = \Phi^\top b_N$, we have the following reduced problem:

$$A_n u_n = b_n. \tag{8}$$

After the reduced problem has been solved, $\tilde{u}_N$ is retrieved from Eq. 7.

Since our problem features multiple coupled variables, an alternative formulation involves applying the SVD separately to each variable. In this way, we replace the transition matrix with a block-diagonal one. Considering a problem with two generic physical variables c and d we have:

$$S_c = \left[u_c^{(1)}|\dots|u_c^{(n_s)}\right] = U^c\Sigma^c(V^c)^H$$
$$S_d = \left[u_d^{(1)}|\dots|u_d^{(n_s)}\right] = U^d\Sigma^d(V^d)^H.$$

In addition, the truncation is done independently, producing the matrices U^c_{tr} and U^d_{tr} whose elements are: $U^c_{tr,ij} = U^c_{ij}$, $i = 1,\dots,N$, $j = 1,\dots,n_c$ and $U^d_{tr,ik} = U^d_{ik}$, $i = 1,\dots,N$, $k = 1,\dots,n_d$. The transition matrix is a block diagonal matrix of the truncated left singular vectors matrices:

$$\Phi = \begin{bmatrix} U^c_{tr} & 0 \\ 0 & U^d_{tr} \end{bmatrix}.$$

We refer to this approach as block-POD. Considering the unknowns of our problem, p, p_γ, λ, the matrix takes the form:

$$\Phi = \begin{bmatrix} U^p_{tr} & 0 & 0 \\ 0 & U^{p_\gamma}_{tr} & 0 \\ 0 & 0 & U^\lambda_{tr} \end{bmatrix}.$$

Here, we choose the number of modes in each left-singular vector to be the same for each physical variable $n_p = n_{p_\gamma} = n_\lambda$.

A problem is affine if the discrete operator can be rewritten as a linear combination where only the coefficient depends on μ. Taking into account the full-order matrix, A_N, the affine parametric dependence implies that:

$$A_N = \sum_{q=1}^{Q} \theta_q(\mu) A_N^q,$$

where $\theta_q(\mu)$ are scalar functions, and A_N^q are constant matrices, so they are computed once and for all, independently of μ. The reduced order matrix, A_n, is thus expressed by the following sum:

$$A_n = \Phi^\top \sum_{q=1}^{Q} \theta_q(\mu) A_N^q \Phi = \sum_{q=1}^{Q} \theta_q(\mu) \Phi^\top A_N^q \Phi = \sum_{q=1}^{Q} \theta_q(\mu) A_n^q. \tag{9}$$

POD is particularly efficient when the affine parametric dependence is satisfied because the calculation of A_n for each new value of the parameters μ does not require reassembling the matrix A_N but using Eq. 9 with the precomputed A_n^q. However, as we will see in Section 5 the changes in geometry related to the sliding of a fault imply a non-affine problem.

4.2 Deep learning-ROM

We restrict ourselves to the core ideas of the DL-ROM approach, and the reader is invited to read [18–23] for more details. In this case, the parameter-to-solution map $\mathcal{M}$ is approximated employing a neural network that naturally accounts for the nonlinear structure of the solution manifold. The reduction procedure is divided into two parts.

(i) First, we perform a dimensionality reduction step that identifies a low-dimensional latent space obtained from a nonlinear mapping of the full-order space to $\mathbb{R}^n$. With little abuse of notation, here n denotes the dimension of the reduced space, which can be different from the corresponding space obtained using the POD method. The mappings from the full-order space to the reduced-order space and back, named $\Psi' : \mathcal{S} \to \mathcal{V}_n$ and $\Psi : \mathcal{V}_n \to \mathcal{V}_N$ respectively, are approximated here by feedforward neural networks. Here, $\mathcal{V}_n$ denotes the reduced-order space obtained with the encoder. More precisely, the approximation of Ψ' is called the *encoder* neural network, while Ψ is called the *decoder*. Their combination is called *autoencoder*; see Fig. 3 for a graphical representation of the adopted neural networks. This approach defines a nonlinear trial manifold because the d.o.f. in the latent space is mapped to the full-order space by the nonlinear function provided by the neural network. Note that neural networks work in discrete spaces; in particular, the encoder input is the discrete solution u_N on a given mesh, and the decoder outputs the reconstructed solution on the same mesh.

(ii) Once the reduced space has been identified, we have to surrogate the operator that solves the problem in the reduced space. This is done again using a data-driven approach. Specifically, a third neural network, named φ, is trained to provide a representation of the solution in the reduced space for any admissible value of the parameters. This approach is feasible due to the low dimensionality of the input and output spaces. In essence, we employ a deep feedforward neural network called *reduced map nerwork*, to describe the map, φ, from the parameter space to the reduced solution space: $\varphi : \Theta \to \mathcal{V}_n$. Therefore, the reduced solution is $u_n = \varphi(\mu)$.

In conclusion, combining steps (i) and (ii) of this procedure, the resulting approximation of the parameter-to-solution map is the following composition: $\Psi(\varphi(\cdot))$, so the reconstructed solution is

$$\tilde{u}_N = \Psi(u_n) = \Psi(\varphi(\mu)).$$

In [22], it has been observed that this method enjoys some optimality properties in terms of the reducibility of the solution manifold. Given the manifold $\mathcal{S}$, the authors of [45] define the nonlinear counterpart of Kolmogorov n-width, $\delta_n(\mathcal{S})$, as the worst reconstruction error using the best encoder and decoder maps Ψ and Ψ', respectively:

$$\delta_n(\mathcal{S}) = \inf_{\substack{\Psi' \in \mathcal{C}(\mathcal{S}, \mathcal{V}_n) \\ \Psi \in \mathcal{C}(\mathcal{V}_n, \mathcal{V}_N)}} \sup_{u \in \mathcal{S}} \|u - \Psi(\Psi'(u))\|.$$

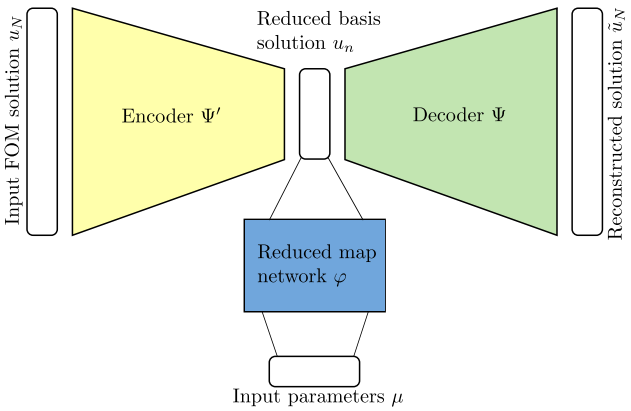


Fig. 3 DL-ROM scheme. The encoder, Ψ', takes the FOM solution, u_N, and returns the reduced basis solution, u_n. The decoder Ψ reproduces an approximate solution, called reconstructed solution $\tilde{u}_N$, with the only knowledge of the reduced solution. The reduced map network, φ, approximates u_n as a function of the given parameter μ

Then, following [22], the *minimal latent dimension* of $\mathcal{S}$, denoted as $n_{\min}(\mathcal{S})$, is defined as the smallest n for which $\delta_n(\mathcal{S}) = 0$. Under the hypothesis that the map $\mu \to u_N$ is continuous and injective and Θ has a non-empty interior, it is shown in [22] that $n_{\min}(\mathcal{S}) = e$. In other words, there exists an autoencoder, with the bottleneck width equal to the size of the parameter space, capable of achieving a null reconstruction error.

The encoder and decoder process the discrete solution as a singular vector containing all physical components p, p_γ, λ.

4.2.1 Training

Training the neural network can follow the stages reported in Section 4.2. In the dimensionality reduction step, the encoder and decoder networks Ψ' and Ψ must be trained. We associate with this task the following loss function:

$$\ell_1 = \frac{1}{N}\|u_N(\mu_i) - \Psi(\Psi'(u_N(\mu_i)))\|_2^2. \tag{10}$$

This is an unsupervised learning task in which the autoencoder $\Psi \circ \Psi'$ is trained using a sequence of unlabeled data representing randomly sampled points in the solution manifold. The second step is to train the reduced map network to learn the map φ. This is done by minimizing the following loss function:

$$\ell_2 = \frac{1}{n}\|\Psi'(u_N(\mu_i)) - \varphi(\mu_i)\|_2^2. \tag{11}$$

In this case, we use a sequence of labeled data $[\mu_i, \Psi'(u_N(\mu_i)]$, representing the parameter-to-reduced state map at a suitable number of points in the parameter space.

Two options are available to train the entire network $\Phi \circ \varphi$. One is to proceed as previously described, minimizing the function ℓ_1 first and then the function ℓ_2. This method has the advantage of reducing the computational complexity of training because the parameters of the networks Ψ' and Ψ are optimized independently of those of φ. The second training approach consists of optimizing all networks together, by minimizing the total loss function defined as

$$\mathscr{L} = \alpha\ell_1 + \beta\ell_2, \tag{12}$$

where $\alpha > 0 \in \mathbb{R}$, $\beta > 0 \in \mathbb{R}$, are user-defined hyperparameters. We remark that the encoder is used only during the offline stage, see Section 4.3. The query of the network for a new value of the parameter, μ, does not require the evaluation of the encoder. Indeed, the final reduced model is composed of only the reduced map network and the decoder. It appears that two-stage training is not beneficial, as suggested by [18]. Training the three neural networks together is advantageous in terms of both accuracy and speed.

4.3 Offline-Online stages POD DL-ROM

Model order reduction strategies are typically divided into two stages: the *offline* phase and the *online* phase. The offline phase encompasses all operations that do not need to be repeated when querying the reduced model with a new parameter value, denoted as μ. This phase includes data generation, which is common to both the POD and DL-ROM methods, see Algorithm 1. Specifically, for POD, this phase also involves calculating the SVD followed by the assembling of the transition matrix Φ. In cases of affine parametric dependence, it implies computing A_n^q. In our case, the latter step is omitted as we deal with non-affine problems (see Section 5 for more details). For DL-ROM, the offline phase ends with neural network training. In particular, the autoencoder, composed of Ψ' and Ψ, is trained simultaneously with the reduced map network φ. We note that in the linear case the map $\mathcal{M}$ from the reduced space to the full-order space is represented by the matrix Φ, while in the non-linear case it is represented by the dencoder Ψ.

Although the offline phase can be time intensive, it is a one-time process. Once completed, the online phase begins. For POD, this includes the assembly of A_n, solving the reduced system, and reconstructing the solution, see Algorithm 2. In contrast, the online phase of DL-ROM simply involves the forward evaluation of neural networks φ and Ψ.

Algorithm 1 Offline: create the dataset and the reduced model

POD	**DL-ROM**
Generate data	**Generate data**
for $i = 1, \dots, n_s$:	for $i = 1, \dots, n_s$:
assemble $A_N(\mu_i), b_N(\mu_i)$	assemble $A_N(\mu_i), b_N(\mu_i)$
solve $A_N(\mu_i)u_N^{(i)} = b_N(\mu_i)$	solve $A_N(\mu_i)u_N^{(i)} = b_N(\mu_i)$
Create the reduced model	**Create the reduced model**
compute SVD	train Ψ', Ψ, φ
assemble Φ	

5 Mesh deformation

We change the domain by deforming a reference computational grid to account for geometric uncertainties. This strategy has the advantage of keeping the same grid topology and having a fixed number of degrees of freedom. As a con-

Algorithm 2 Online: deploy the reduced model

POD	DL-ROM
1: **Input:** μ	1: **Input:** μ
2: assemble $A_N(\mu)$, $b_N(\mu)$	2: compute $\tilde{u}_N = \Psi(\varphi(\mu))$
3: compute $A_n = \Phi^T A_N \Phi$, $b_n = \Phi^T b_N$	
4: solve $A_n u_n = b_n$	
5: reconstruct $\tilde{u}_N = \Phi u_n$	

sequence, all snapshots have the same size. Moreover, the ordering of the d.o.f. is kept, so they remain associated with the same physical region since we consider small geometric deformations. The general procedure presented in the following can also work for large deformations in both the 2D and 3D cases at the price of additional mesh handling complexity. We adopt a technique based on radial basis functions (RBF) to deform the mesh, and we show first the standard approach [46–48], then its adaptation to our specific problem of geometries containing sliding faults.

In the standard approach, we seek to interpolate each component of the displacement function, $s(x) \in \mathbb{R}^D$, D being the physical dimension, using a linear combination of a given radial basis function, $g(d)$, where $d = d(x_1, x_2) = \|x_1 - x_2\|$ is the distance between the two points. Defining $g^*(x_1, x_2) = g(d(x_1, x_2))$, we have the following.

$$s(x) = \sum_{j=1}^{l} g^*(x, x_{c_j})\zeta_j, \tag{13}$$

where $\zeta_j \in \mathbb{R}^D$, are the unknown coefficients and x_{c_j} are the so-called control points. We use the following radial basis function: $g(d) = d/0.2$. To find ζ_j, we need to apply the constraints formed by selecting a number l of relevant points of which we have information about their displacement. For instance, we can choose the nodes on the fixed borders of the domain or the nodes belonging to elements whose geometry is uncertain. These points are called control points, x_{c_j}, $j = 1, \ldots, l$, and that is where we enforce a known displacement, $\overline{s}_j$:

$$s(x_{c_j}) = \overline{s}_j. \tag{14}$$

Evaluating Eq. 13 at the control points, we have a linear system for each component, $m = 1, \ldots, D$, of the vector displacement, s:

$$G z_m = \sigma_m, \tag{15}$$

where $G \in \mathbb{R}^{l \times l}$, $[G]_{i,j} = g(d(x_{c_i}, x_{c_j}))$, $z_m \in \mathbb{R}^l$ is the unknown vector where $[z_m]_j = [\zeta_m]_j$. The right-hand side $\sigma_m \in \mathbb{R}^l$ contains the known displacement as a function of the geometrical parameters $[\sigma_m]_j = [\overline{s}_j]_m(\mu_{geom})$.

We aim to let the two sides of the fault slide independently, so the control points must be positioned on both sides to avoid any undesirable deformation of the fault. Some control points inevitably become very close or even coincident, causing G to be ill-conditioned or singular. Moreover, to make the process more practical, instead of the displacement constraint, we would like to impose a sliding constraint on x_{c_j} on some specific surfaces, for example, the fault surface or the boundary of the domain; see Fig. 4.

We now show an improvement of the standard mesh deformation method to meet our requirements. Taking a generic surface, S_i, which could be, for example, a fault or a boundary face, with normal ν_i and two (one in 2D) non-parallel tangent unit vectors t_i and b_i, the sliding constraint can be described by two conditions: the non-penetration condition:

$$s(x_{c_j}) \cdot \nu_i = 0, \tag{16}$$

and by the no-tangential contribution condition:

$$\begin{aligned} \zeta_j \cdot t_i &= 0, \\ \zeta_j \cdot b_i &= 0, \end{aligned} \tag{17}$$

where ζ_j is the coefficient associated with x_{c_j} on the sliding surface. Equation 16 sets the control points on a sliding surface free to move except in the normal direction. Equation 17 implies that the tangential displacement of the control points

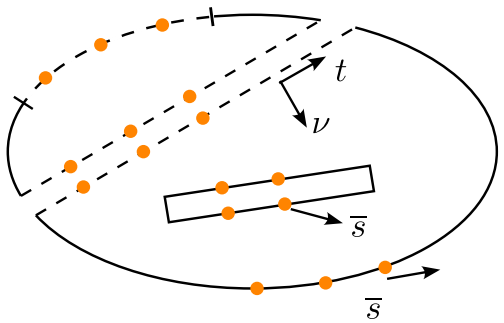


Fig. 4 Generic faulted domain with control points, in orange. The dashed lines represent sliding surfaces, one is on $\partial_{ex}\Omega$, and the control points therein belong to C_s, the other two represent the boundaries $\partial_{in}\Omega$ facing a fault, the control points on those lines are included in the set C_{sf}. The gap between the two faces of the fault was added for graphical reasons only, and it is actually absent, so the control points on the fault may be coincident. Other control points in C_d are placed on the fracture where a rigid displacement, $\overline{s}$, is applied. The remaining control points in C_{df} are on the boundary of Ω where a displacement $\overline{s}$ is enforced

of the fault does not affect the displacement of all the other points. However, the displacement of the control points on the fault is influenced by all the other control points.

To address the issue of coincident points and add the sliding condition, let us define an index set C of x_{c_j} and divide it into four subsets: C_d containing the control points not on the fault where displacement is enforced, C_{df} containing the control point on the fault where displacement is enforced, C_s and C_{sf} containing the sliding control points, respectively, not in the fault and in the fault. Moreover, we call $\overline{C}_s$ the set of surfaces where sliding conditions are applied. We introduce the *side function* of a point, $\beta(x, \nu)$, with respect to the sliding fault, γ, with normal ν:

$$\beta(x, \nu) = \frac{\text{sign}\left((x - x_{ref}) \cdot \nu\right) + 1}{2},$$

$x_{ref} \in \gamma$ is a reference point. We define the influence function, $\mathcal{I}$, as a function that hides the points that are not on the same side of the fault. An expression could be (for the sake of simplicity, we restrict the discussion to the case of a single sliding fault, with normal ν_2):

$$\mathcal{I}(x, \beta(x, \nu_2), \beta(x_{c_j}, \nu_2), \nu_2) =$$

$$\begin{cases} 1 & \text{if } x_{c_j} \in C_{ds} \\ NXOR(\beta(x, \nu_2), \beta(x_{c_j}, \nu_2)) & \text{if } x_{c_j} \in C_{sf} \text{ and } x \in \partial_{in}\Omega \\ \frac{|(x - x_{ref}) \cdot \nu_2|}{\|(x - x_{ref})\|} NXOR(\beta(x, \nu_2), \beta(x_{c_j}, \nu_2)) & \text{if } x_{c_j} \in C_{sf} \text{ and } x \in \Omega, \end{cases}$$

where $C_{ds} = C_d \cup C_s$ and $C_f = C_{df} \cup C_{sf}$ and *NXOR* is the combination of the logical operations "not" and "xor", $NXOR(a, b) = \neg(a \oplus b)$. Introducing $\mathcal{I}$ to Eq. 13 and the displacement and sliding constraints, setting $g^\dagger(x, x_{c_j}, \nu_2) = \mathcal{I}(x, \beta(x, \nu_2), \beta(x_{c_j}, \nu_2), \nu_2) g^*(x, x_{c_j}, \nu_2)$ the system to be solved to deform the mesh becomes:

$$\begin{cases} s(x) = \sum_{j=1}^{l} g^\dagger(x, x_{c_j}, \nu_2)\zeta_j, & \\ s(x_{c_j}) \cdot \nu_i = 0, & x_{c_j} \in C_s \cup C_{sf},\ S_i \in \overline{C}_s \\ \zeta_j \cdot t_i = 0, & \\ \zeta_j \cdot b_i = 0, & x_{c_j} \in C_s \cup C_{sf} \text{ s.t. } x_{c_j} \text{ on } S_i \in \overline{C_s} \end{cases}$$

For example, for a 2D case with one sliding surface, labelled as 1, and one sliding fault, labelled as 2, the system becomes:

$$\begin{bmatrix} G & 0 \\ 0 & G \\ G\nu_{1x} & G\nu_{1y} \\ G\nu_{2x} & G\nu_{2y} \\ H_1 t_{1x} & H_1 t_{1y} \\ H_2 t_{2x} & H_2 t_{2y} \end{bmatrix} \begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} \overline{s}_x \\ \overline{s}_y \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

with $[G]_{ij} = g^\dagger(x_{c_i}, x_{c_j}, \nu_2)$ for $x_{c_i} \in C_d \cup C_{df}$ and $x_{c_j} \in C$, and

$$[H_1]_{ij} = \begin{cases} 1 & \text{if } x_{c_i}, x_{c_j} \in C_s, \\ 0 & \text{else.} \end{cases} \qquad [H_2]_{ij} = \begin{cases} 1 & \text{if } x_{c_i}, x_{c_j} \in C_{sf}, \\ 0 & \text{else.} \end{cases}$$

We aim now to demonstrate that the relative sliding of the two sides of a fault introduces a non-affine problem, thereby reducing the efficiency of the POD reduced order modeling technique, as discussed in Section 4.1. From Eq. 3, the pressure on the internal boundaries of Ω facing the fault γ, see Fig. 1, can be rewritten as:

$$p|_{\partial_{in}\Omega_2} = p_\gamma + \frac{\epsilon}{2K_n}\lambda^+. \tag{18}$$

Let r be the combination of the maps $r_1 : \partial_{in}\Omega_2 \to \gamma^+$ and $r_2 : \gamma^+ \to \gamma$ map from $\partial_{in}\Omega_2$ to γ, so $r = r_1 \circ r_2$. A generic point x_γ on the fault corresponds to $x_\gamma = r(x_{\partial_{in}\Omega_2}; \mu)$ where r depends on the geometric parameterization governed by μ_{geom}. The map r could be non-affine, which already spoils the problem's affinity. Introducing r in Eq. 18 to explicitly write the link between the pressures in the two subdomains, we have:

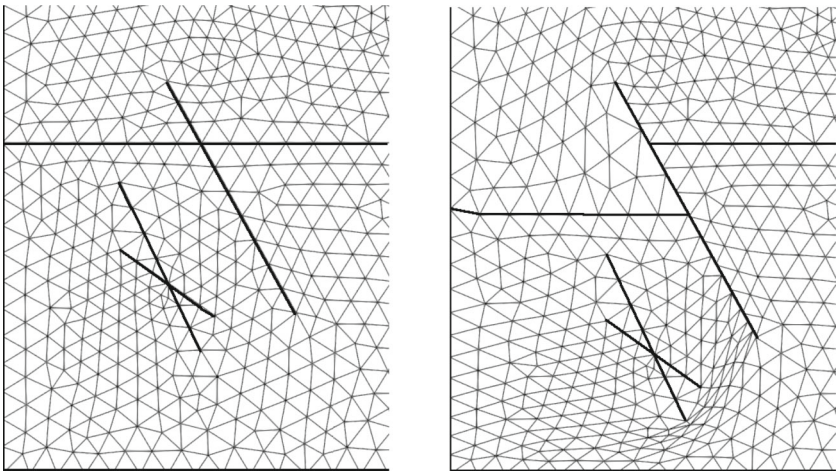
$$p(x_{\partial_{in}\Omega_2}) = p_\gamma(r(x_{\partial_{in}\Omega_2}; \mu)) + \frac{\epsilon}{2K_n}\lambda_1(r(x_{\partial_{in}\Omega_2}; \mu)),$$

where the dependence on μ cannot be separated, even if r were an affine map. In the weak formulation, this implies that the calculation of the pressure on the internal boundaries contains terms that cannot be separated from the parameters, so the problem is not affine (Fig. 5).

6 Numerical validation

The reduced order model techniques described above have been applied to four 2D and 3D test cases with different specifics. The first test is a simple problem with a 2D domain in which uncertainties are associated with the permeability of the rock and the throw of the fault. The second case is the three-dimensional extension of the first one. The third case is a complex fracture network with varying boundary conditions. In the last one, we consider the more realistic case of a strongly heterogeneous permeability field where pressure has a more complex distribution.

In all the case studies we assess the quality of the reduced model by measuring the maximum, minimum, and averaged



(a) original mesh (b) deformed mesh

Fig. 5 Mesh deformation. Zoom of the bottom-left corner of the domain of the third test case, see Section 6.3. A rigid displacement is imposed to the left side of the fault and to the two small intersecting fractures. The left boundary is a sliding surface where sliding conditions are applied, on the bottom one, instead, a null displacement is enforced

relative norm of the error, defined as:

$$
\begin{aligned}
e_{max} &= \max_j(\|e(\mu_j)\|_2/\|u_N(\mu_j)\|_2),\\
e_{min} &= \min_j(\|e(\mu_j)\|_2/\|u_N(\mu_j)\|_2),\\
e_{ave} &= \frac{1}{J}\sum_{j=1}^{J}\|e(\mu_j)\|_2/\|u_N(\mu_j)\|_2,
\end{aligned}
\tag{19}
$$

where J is the size of the test dataset and $e(\mu)$ is the difference between the FOM and ROM solution, $e(\mu) = u_N(\mu) - \tilde{u}_N(\mu)$.

6.1 Case 1 - setup

As a first test case, we studied a unit 2D square domain cut by a fault whose aperture is $\varepsilon = 10^{-3}$ tilted by an angle of 60° from the horizontal, Fig. 6. There are two layers separated by a caprock, and they are identified by different values of permeabilities K_1, K_2, K_3. An injection point and a production point are placed, respectively, at the bottom-left and top-right corners, where we enforce counterbalanced fluid fluxes. We impose homogeneous Neumann boundary conditions at all boundaries except for the bottom-left corner, where the pressure is set to 1.

The parameters of the problem are the three values of permeability, $K_1 \in [10^{-2}, 10^{-1}]$, $K_2 \in [10^2, 10^3]$, $K_3 \in [10^{-4}, 10^{-3}]$, of the three layers, the permeability $K_4 \in [10^{-4}, 10^{-3}]$, of the fault such that $K_\tau = K_4$, $K_n = 2K_4/\varepsilon$, and the height $h \in [0, 0.07]$, of the right horizons: a small displacement is allowed along the fault direction. The maximum displacement value $h = 0.07$ is less than the thickness of the caprock equal to 0.1. This setting implies a pressure distribution with three main regions: high pressure layer at the bottom, low pressure layer at the top, and high-pressure gradient in the caprock that horizontally separates the domain, as we can see in Fig. 9(a) that represents a typical snapshot of the test data set. Furthermore, based on the specific values of the permeabilities, there may be a jump in pressure across the two faces of the fault.

Subsoil permeabilities may vary widely, even by several orders of magnitude, so it is convenient to express them as exponential functions and sample the exponent. Therefore, the permeabilities are written in the following form: $K_i = e^{\eta_i}, i = 1, \ldots, 4$ and the parameter vector consists of the exponents of the permeabilities and the height of the horizons: $\mu = (\eta_1, \eta_2, \eta_3, \eta_4, h)$.

The data set is made up of 1000 snapshots whose parameters are randomly sampled from a uniform distribution (the same strategy is adopted for the other tests). It is divided into a training dataset (800 snapshots), a validation dataset (100 snapshots), and a test dataset (100 snapshots). These data sets are used in both the POD approach (Sect. 6.1.1) and the DL-ROM approach (see Section 6.1.2).

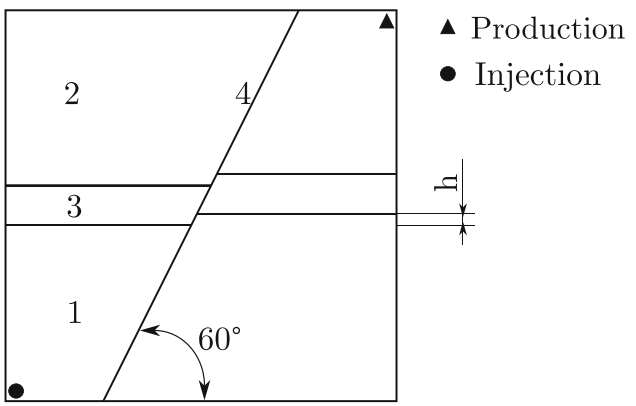


Fig. 6 Case study 1. Geometry. The fault cuts the domain from the bottom to the top with an angle of 60°. The right layers can slide along the fault. The production is in the top right corner and the injection is on the bottom-left corner

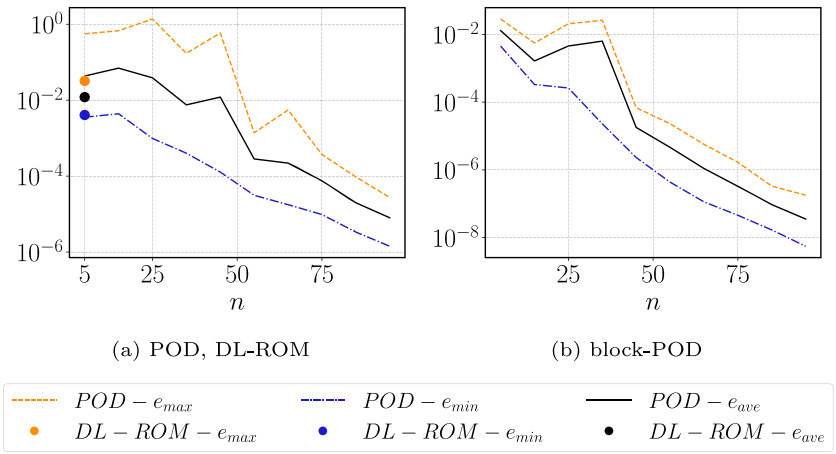


Fig. 7 Case study 1. e_{max}, e_{min}, e_{ave}, respectively, maximum, minimum and average errors versus the size of the reduced space, n, of the reduced model obtained with (a) standard POD and (b) block-POD. The points on the panel (a) represents the value errors for the DL-ROM, described in Section 6.1.2

Note that the training data set may be small compared to the complexity of the problem and the number of parameters. A comparison with a uniform grid sampling reveals that we are sampling the parameters with fewer than 4 points for each of the 5 axes of the space Θ. This is because we aim to apply the reduced-order modeling techniques to real scenarios, where a single snapshot generation is extremely expensive, making the generation of very large datasets infeasible, and therefore the test cases are designed accordingly.

6.1.1 Case 1 - POD

Both POD and block-POD (Section 4.1) are shown. After the offline phase, the quality of the reduced model is assessed using the test data set. Each parameter of the test data set is used to generate the reduced solution and then the reconstructed solution, which is compared with the solution of the full-order model to compute the errors defined in Eq. 19. We evaluated the errors for different sizes of the reduced space, $\mathcal{V}_n$, which corresponds to different truncations of the transition matrix, as shown in Fig. 7. The block POD error shown in Fig. 7(b) decreases faster than the standard POD; we justify this trend by examining the decay of the singular values of each variable in Fig. 8. We observe a faster decay in the singular values of the POD block snapshot matrices computed for p_γ and λ compared to those related to p. The trend of the latter resembles that of the complete discrete vector u_N. Then, decoupling the variables takes advantage of the faster decay of the approximations for p_γ and λ, which otherwise would not have an effect in the fully coupled approach.

Some oscillations are observed in the trend of the error; nevertheless, it decreases to negligible levels taking a sufficient number of modes.

Offline time encompasses the generation of the training data set and all computation related to the SVD decomposition required to obtain the transition matrix. Indicatively, data generation takes 4 min 30 s on an Intel i5-1135G7 and a fraction of a second to finalize the offline phase. The online phase includes the construction of the matrix that assembles the linear system, its solution, and the reconstruction of the solution. Its time is assessed at 30 points of the test data set. The main statistics are shown in Fig. 10, as well as the times for the FOM and DL-ROM. As in the other tests, similar results are obtained with the block-POD. Indeed, in both the POD and block-POD methods, the online time is dominated by the assembly of the full order model matrix A, a common step for both methods. Consequently, for the sake of simplicity, the timing results of block-POD will not be reported. The square is delimited by the first and third quartiles, whereas the whiskers extend from the box by 1.5x the interquartile range. The yellow line represents the median. The number of modes $n = 45$ is chosen because the related error is similar to the error obtained with the DL-ROM, so we can compare the timing for the same error. The consequent compression rate is $\frac{n}{N_h} = 4.3\%$.

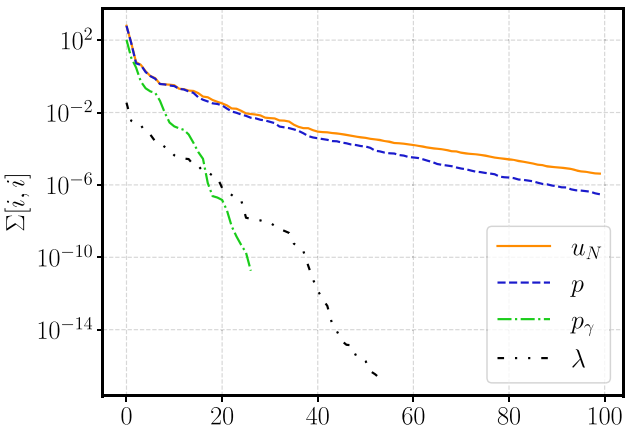


Fig. 8 Case study 1. The decay of singular values of the snapshot matrix of the fully coupled POD approach for u_N, compared to the block-POD case applied to each variable p_γ, λ and p independently

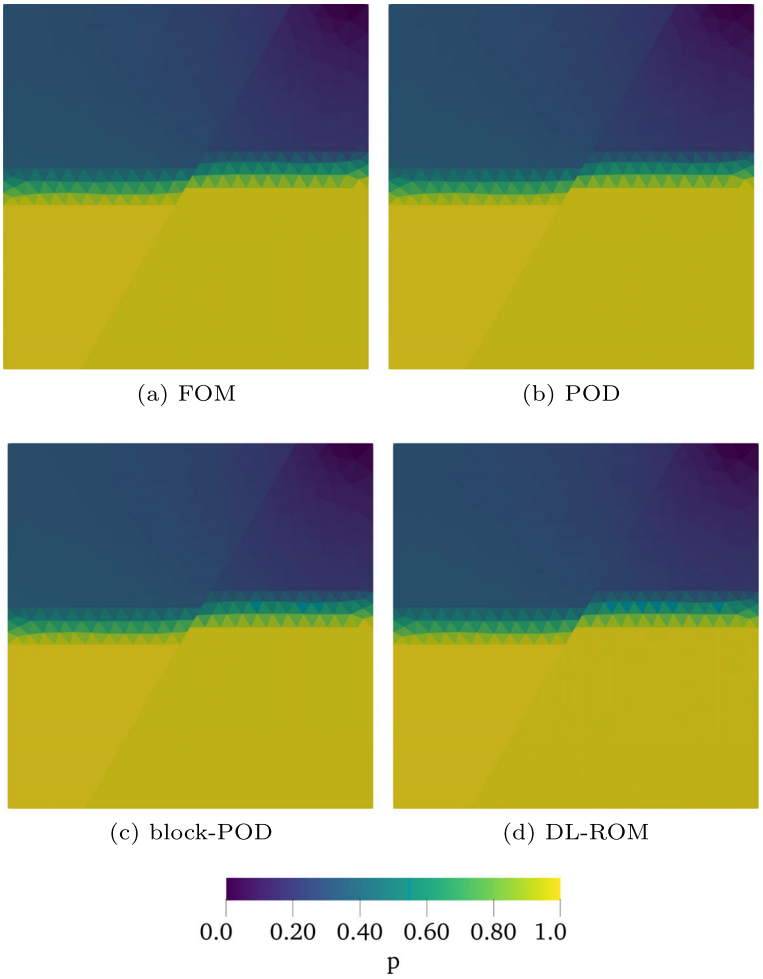


Fig. 9 Case study 1. Reconstructed solutions for a specific value of the parameter μ. (a) FOM. (b) POD. (c) block-POD (d) DL-ROM

6.1.2 Case 1 - DL-ROM

When not specified differently, we use the following architectures and settings in each test case: the encoder is made of a fully connected network with one hidden layer, the decoder architecture is symmetric with respect to that of the encoder, and the reduced map network is made of a fully connected network with one hidden layer. As a nonlinear activation function, ρ, we use, for all networks, the PReLU function: $\text{PReLU}(x) = \max(0, x) + a \min(0, x)$, where a is a trainable weight. The weights of the layers m are initialized from the uniform distribution $\mathcal{U}\left(-\sqrt{1/\text{size}(m-1)}, \sqrt{1/\text{size}(m-1)}\right)$.

The data used to train, validate, and test neural networks are the same as those used for the POD approach.

The input of the encoder is normalized to [0, 1]. Since the ratio of the physical variables may differ by some order of magnitude, we decided to normalize each discrete physical variable taking individually their maximum and minimum values over the training dataset.

Details of the neural networks of this test case are summarized in Table 1. With the use of neural networks, in this

Table 1 Case study 1. Architecture of the networks that make up the reduced model. "fc" stands for fully connected layer, tw is the number of trainable weights

	layer	# tw	# tw tot
Encoder	fc, 1045, 181, PReLU fc, 181, 5, PReLU	190 238	382 622
Decoder	fc, 5, 181, PReLU fc, 181, 1045, PReLU	191 277	
Reduced map network	fc, 5, 100, PReLU fc, 100, 5, PReLU	1 107	

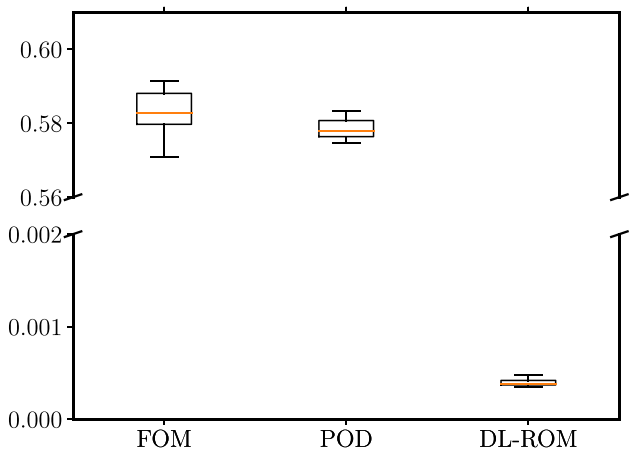


Fig. 10 Case study 1. Evaluation times in seconds of the full-order model, the reduced model obtained through POD, and the reduced model obtained through the DL-ROM. Evaluations of reduced models are timed on 30 points of the test dataset

case, we reach a compression rate of $\frac{n}{N} = 0.5\%$, which is almost ten times higher than that of the POD.

The network is trained for 4000 epochs with Adam optimizer [49], using the hyperparameters suggested in the cited reference and a minibatch size of 32. The learning rate is initialized to 10^{-3} and is reduced by a factor of 0.6 every 500 epochs. Once the networks are trained, the reconstruction errors are: $e_{max} = 3.2\%$, $e_{min} = 0.41\%$, and $e_{ave} = 1.2\%$, they are depicted in Fig. 7 (a) as points. The distance between the maximum and minimum values is small, which means that the network is accurate evenly throughout the parameter space.

The data generation and the training of the neural networks constitute the offline phase; the latter takes about 40 minutes to complete the 4000 epochs. The online phase is represented by the evaluation of the reduced map network and the decoder; this time it is evaluated on the Nvidia MX330 graphic card and is shown in Fig. 10. We can observe that, despite a slightly longer offline phase, the online is almost three orders of magnitude faster than the POD case. In fact, the online time savings (normalized difference of times) with respect to the FOM model is 99.14%. We want to highlight that the DL-ROM allows for the efficient exploitation of both CPU and GPU hardware, while the POD does not run efficiently on GPUs.

A visualization of the reconstructed solution compared to the full-order model is illustrated in Fig. 9. We can observe qualitatively that the reconstructed solutions of both POD and DL-ROM are close to the FOM solution in the entire domain.

6.2 Case 2 - setup

The second test is a 3D simulation that resembles the first in most of its features. The main differences are the domain (Fig. 11) which is a unit cube whose normal section to z corresponds to the domain pictured in Fig. 6, and the position of the injection and production points that are at two opposite corners of the cube. We chose this test case to study the effects of adding the third physical dimension without adding many other factors.

The datasets are made similarly to the previous test case: the parameters are sampled from a random distribution, 800 snapshots constitute the training dataset, 100 snapshots are used for the validation dataset and 100 for the test dataset.

6.2.1 Case 2 - POD

Figure 12 shows the errors defined by Eq. 19 with respect to the size of the reduced space. We notice a slightly slower decrease of the error with respect to test case 1 and that, for this scenario, the block POD proves to have higher performance than the standard one. The online time depends on the number of modes used; to facilitate the comparison with the other methodology, we report in Fig. 14 the online time for the standard POD for $n = 18$ which entails a test error similar to that of the DL-ROM. The related compression rate is equal to $\frac{n}{N} = 0.2\%$. Due to the great number of d.o.f., the online time reaches a high value of about 15 seconds, Fig. 14, it is mainly due to the computations for deforming the mesh and the rediscretization of the problem due to its non-affine nature.

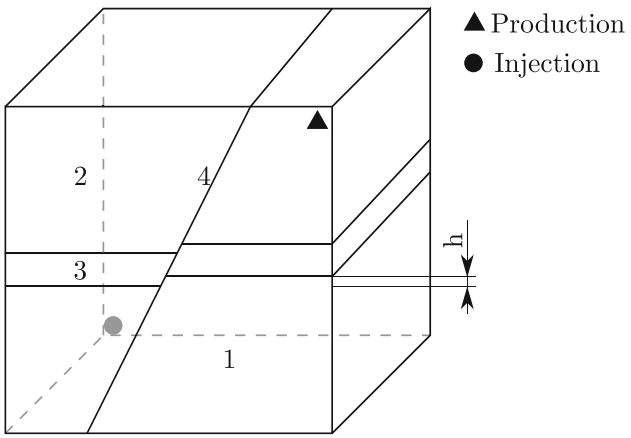
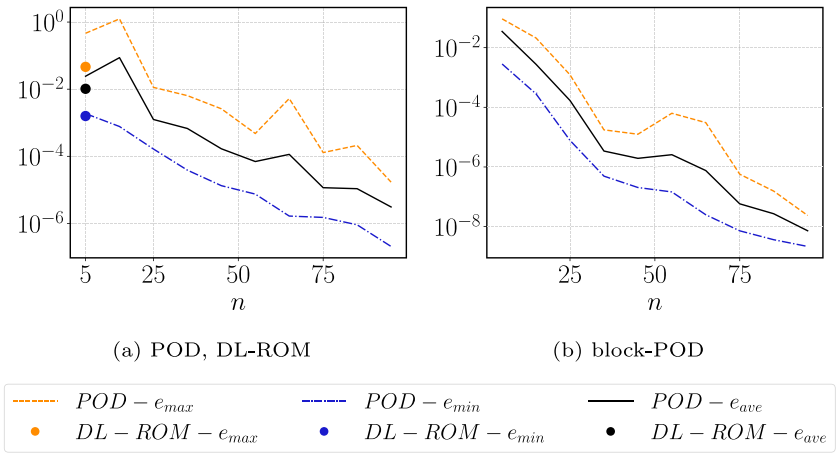


Fig. 11 Case study 2. The domain of this test corresponds to the geometry of case 1 extruded along the z-direction. Production and injection are placed at the two opposite corners



(a) POD, DL-ROM (b) block-POD

Fig. 12 Case study 2. e_{max}, e_{min}, e_{ave}, respectively, maximum, minimum, and average error versus the size of the reduced space, n, of the reduced model obtained with (a) standard POD, (b) block-POD. The points on panel (a) represent the value errors for the DL-ROM, described in Section 6.2.2

6.2.2 Case 2 - DL-ROM

The specific architectures are summarized in Table 2. We can infer that the compression rate is equal to $\frac{n}{N} = 0.06\%$, which is ten times higher than the POD. The hyperparameters of the training are the same as in Case 1, except for the number of epochs that is higher, 6000 instead of 4000 (about 3 hours). The errors in the test data set are: $e_{max} = 4.6\%$, $e_{min} = 0.16\%$, $e_{ave} = 1.02\%$. They are similar to the values of case 1, suggesting that the DL-ROM works equally in a 2D and 3D scenario. Figure 13 shows a snapshot of the test data set. As in Section 6.1, we can see that, qualitatively, all the solutions resemble the FOM one at each point of the domain. The online time is reported in Fig. 14, despite the large increase in degrees of freedom of the problem, the online time has only doubled with respect to case 1. The consequent time saving is equal to 99.97%.

6.3 Case 3 - setup

The third test case resembles the complex fracture network benchmark in [50]. The 2D square domain includes intersecting permeable and impermeable fractures, as shown in Fig. 15, whose detailed description can be found in the reference. Compared to the reference, we further increase the complexity by setting two horizontal layers whose rock permeability is equal to 10^{-2} at the bottom and 10^{2} at the top. A Dirichlet boundary condition for the pressure is applied to all the boundaries such that an average flow is generated from one side to the opposite. After a manual sensitivity analysis, we set the two most relevant features as uncertain parameters: the first controls the boundary condition creating high variation among the snapshots, see Fig. 17, while the second controls the geometry, making the problem non-affine. A sinusoidal variation of the pressure at the boundary, p_b, is applied:

$$p_b(\omega) = p_1 \left(1 - \sin\left(\frac{\omega - \omega_0}{2}\right)\right) + p_2 \left(\sin\left(\frac{\omega - \omega_0}{2}\right)\right),$$

where p_1 and p_2 are the maximum and minimum values, $\omega = \arctan(y/x)$, x and y are the coordinate of the boundary points. The first parameter is the reference angle ω_0 which can be subject to a variation of 90°, Fig. 17 shows snapshots for three different values of ω_0. The second parameter controls the position of the horizon on the left of fault num-

Table 2 Case 2. Architecture of the networks that make up the reduced model. "fc" stands for fully connected layer, tw is the number of trainable weights

	layer	# tw	# tw tot
Encoder	fc, 8706, 1708, PReLU fc, 1708, 5, PReLU	14 880 103	29 770 013
Decoder	fc, 5, 1708, PReLU fc, 1708, 8706, PReLU	14 888 803	
Reduced map network	fc, 5, 100, PReLU fc, 100, 5, PReLU	1107	

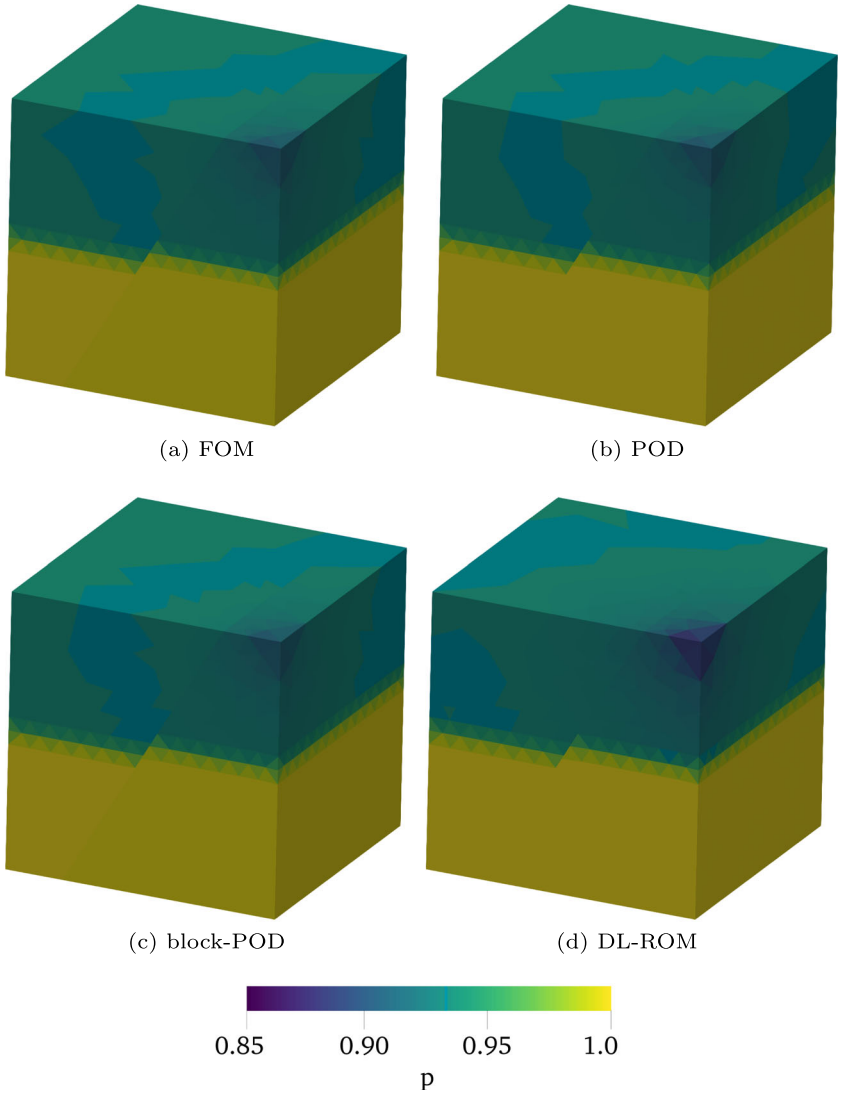


Fig. 13 Case study 2. Reconstructed solutions for a specific value of the parameter μ

ber 3, displacing rigidly also fractures number 1 and 2. The high variation of the boundary conditions and the presence of blocking fractures imply strong nonlinearities in the solution manifold that may raise issues in reducing the order of the model. Generally speaking, the results of this case show a larger gap between the performance of the two reduced-order model methods in favor of the DL-ROM one.

6.3.1 Case 3 - POD

For this test, only the standard POD is investigated. The error versus the size of the reduced space (Fig. 16) shows that several modes are required to reach a low error (Fig. 18). The online time is depicted in Fig. 19 where a number of $n = 44$ modes is used. The strong nonlinearities of the problem manifest in a compression rate higher than in the previous cases $\frac{n}{N} = 1.85\%$. The five-number summary of the online time is shown in Fig. 19.

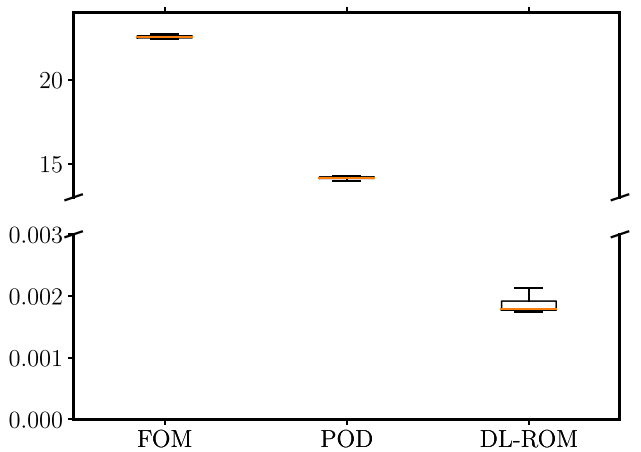


Fig. 14 Case study 2. Evaluation times in seconds of the full-order model, reduced model obtained through POD, and reduced model obtained through the DL-ROM

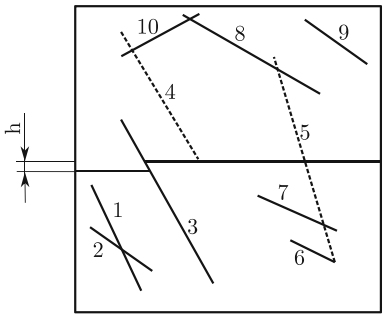


Fig. 15 Case study 3. The domain contains 9 fractures and one fault (number 3). Fractures 4 and 5 are blocking fractures, while all others have high permeability

6.3.2 Case 3 - DL-ROM

The architectures are described in Table 3, in particular, the reduced space size is equal to $n = 2$, so the compression rate results in $\frac{n}{N} = 0.08\%$. The training requires 5000 epochs (about 74 minutes) to reach a steady low value of the loss function. The errors are fairly low: $e_{max} = 6.2\%$, $e_{min} = 0.78\%$, $e_{ave} = 2.3\%$ (Fig. 16), although the ratio between the minimum and maximum values is higher than the one in case 1 and case 2. The online time can be inferred from Fig. 19, the time savings are equal to 99.86%.

6.4 Case 4

In this study, we examine a scenario involving a highly heterogeneous medium in which permeability varies significantly across the domain. The domain is a 2D square discretized with a cartesian grid of 150×150 elements. Two parameters determine the anisotropy of the permeability's spatial distribution and its primary direction. The method for generating this permeability field is outlined in [51]. In Fig. 20, two extreme cases are presented: the first, at the top, exhibits highly anisotropic permeability whose principal axis is slanted at 45°, while the second case, at the bottom, has only a slight anisotropy.

To induce a pressure gradient within the domain, we enforce pressure values at the boundaries: a null value at the top and a unitary value at the bottom. Null flux is imposed on the vertical edges. As a result of the heterogeneous permeability field, the pressure exhibits non-smooth variations with intricate patterns (see Fig. 20).

The variation in permeability from cell to cell means that the problem is non-affine. This requires re-discretization and the assembly of a full order model matrix for each new value of the parameters that control the permeability distribution, if the POD method is used. Therefore, our focus is here solely on the neural network approach, which effectively handles non-affinity.

The architecture of the neural networks in the reduced model is outlined in Table 4. To enhance the efficiency of the networks by reducing the number of trainable weights, a non-symmetric autoencoder is employed. This autoencoder consists of a single-layer encoder and a four-layer decoder. We justify this structure empirically because we believe that the compression task of the encoder is simpler than the reconstruction task of the decoder. The compression rate is $\frac{n}{N} = 0.009\%$. In terms of the datasets, we used 700 snapshots for training, 150 for testing, and 150 for validation. It took approximately 1.5 hours to generate all the data. We trained

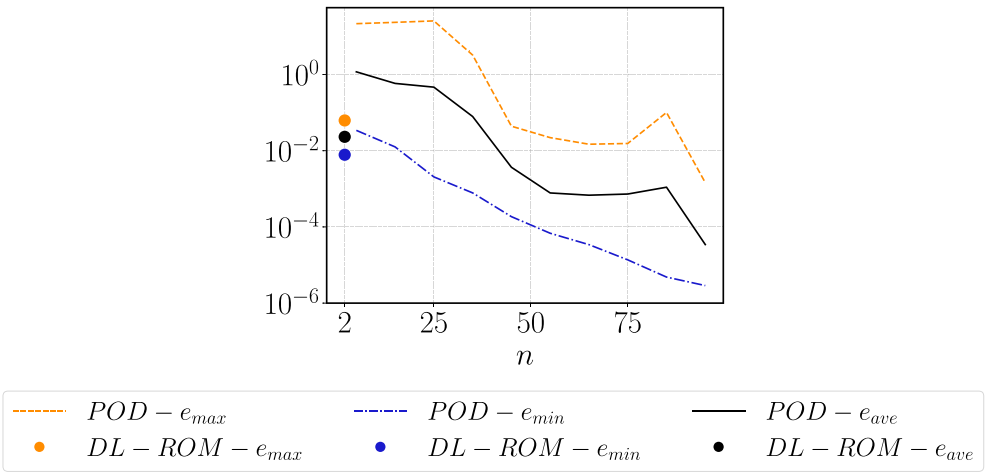


Fig. 16 Case study 3. e_{max}, e_{min}, e_{ave}, respectively, maximum, minimum and average error versus the size of the reduced space, n, of the reduced model obtained with the standard POD. Points represent the value errors for the DL-ROM, described in Section 6.2.2

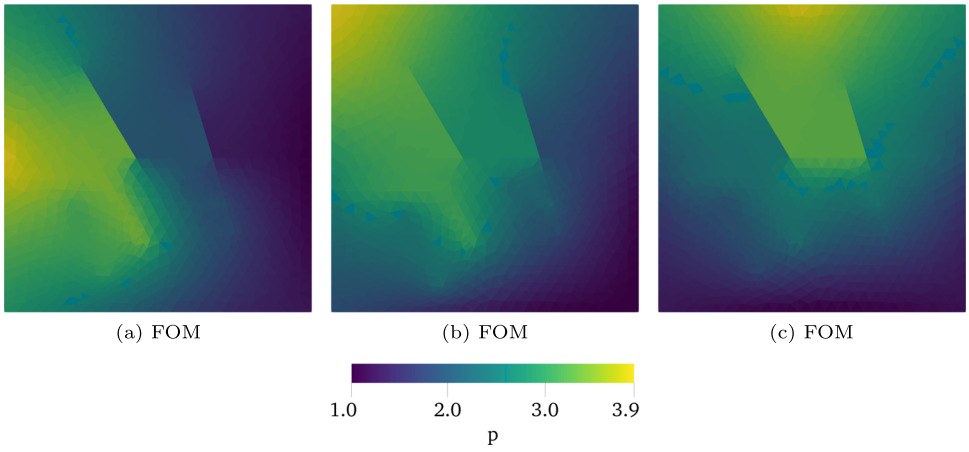


Fig. 17 Case study 3. Boundary conditions are applied such that the mean pressure gradient is (a) horizontal (b) slanted (c) vertical

the model for a total of 1031 epochs, which took around 1 hour, until the loss function reached a stable low value.

The error metrics underscore the robust reliability of the methodology, indeed we find: $e_{max} = 0.71\%$, $e_{min} = 0.15\%$, $e_{ave} = 0.29\%$, which constitutes a notable accomplishment given the substantial variability of the rock properties and the resulting pressure distributions. Indeed, as illustrated in Fig. 20, the congruence between the FOM and the reconstructed solution, even with two parameters not included in the training dataset, is evident. Furthermore, the relatively small error margin is attained with a significant reduction of the computational time w.r.t. the FOM, refer to Fig. 21 for a comparative analysis of the full-order and reduced-order model execution durations.

7 Multi-query application

In this section, we show possible practical applications of the reduced order model techniques. We first used a Monte Carlo strategy to sample the data and perform a sensitivity analysis, comparing the results of different reduced-order models and a full-order model. Afterwards, an inverse problem is solved exclusively with DL-ROM. For simplicity, we focus on a single quantity of interest for each study, such as the pressure jump between two relevant points. It is important to note that the reduced model computes the flow variables in the entire discrete domain, so no additional efforts would be required to investigate other quantities of interest within the same reduced model evaluation.

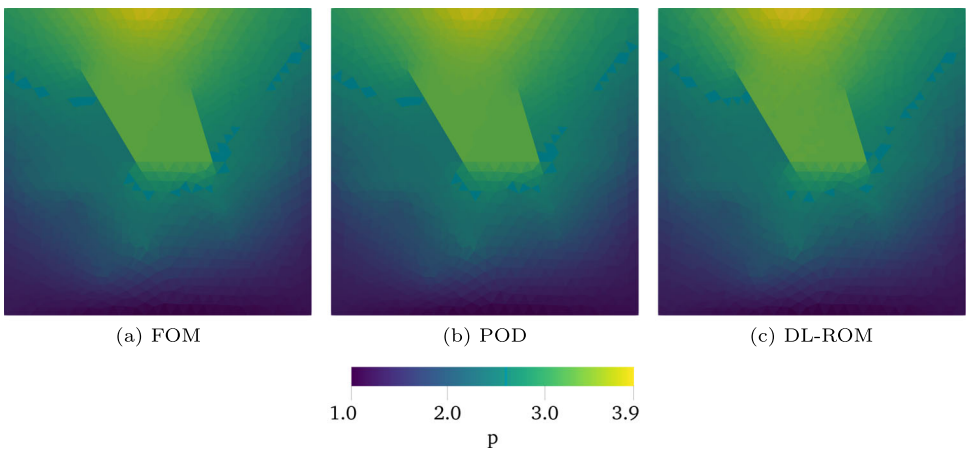


Fig. 18 Case study 3. Reconstructed solutions for a specific value of the parameter μ

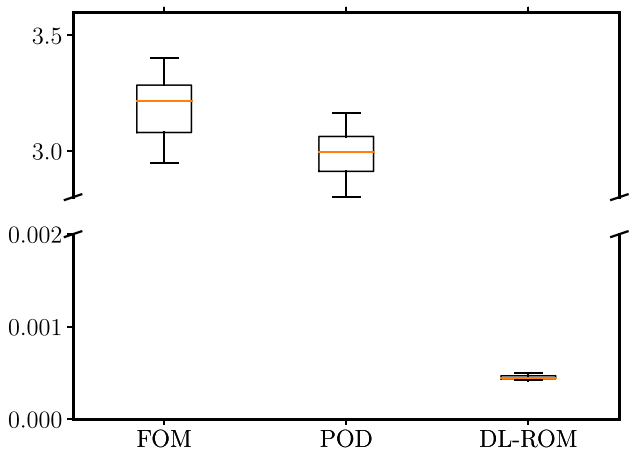


Fig. 19 Case study 3. Evaluation times in seconds of the full order model, reduced model obtained through POD, and reduced model obtained through the DL-ROM

7.1 Sensitivity analysis

We use a Monte Carlo technique to perform a sensitivity analysis on a problem based on test case 1 (see Section 6.1). We chose this test case because its small size allowed us to run the Monte Carlo analysis even with the full-order model, and thus compare the results with those obtained with the reduced models. We rely on the Chaospy library, a numerical toolbox to perform uncertainty quantification [52]. Our goal is to focus on a quantity of interest (q.o.i) represented by the pressure jump between injection and production, see Fig. 6, called Δp, and determine its mean value, $\overline{\Delta p}$, standard deviation $\tilde{\sigma}$, and the first-order sensitivity index (Sobol index) [53], $\tilde{s}_{1,i}$, $i = 1, \ldots, 5$, related to each entry, μ_i, of the parameter vector μ.

For the purpose of this example, we prepare a training data set of 200 snapshots and a validation data set of 20 samples. All data in the training data set are used to create the snapshot matrix to which the SVD decomposition is applied. Then, the left singular vector matrix, U, see Section 4.1, is truncated to $n = 13$ basis functions because it leads to a low enough reconstruction error, that is, $e_{ave} = 5\%$. The neural networks are trained until the loss function reaches a low enough value to have a satisfactory reconstruction error equal to $e_{ave} = 5.3\%$. Since the q.o.i. is known a priori, it is possible to add a term in the loss function related to the q.o.i., thus increasing the accuracy of the reduced model for only what is needed. For instance, in this case, we add the term ℓ_3 to the loss function defined in Eq. 12:

$$\begin{aligned}\mathscr{L} &= \alpha\ell_1 + \beta\ell_2 + \gamma\ell_3 = \\ &= \frac{\alpha}{N}\|u_N(\mu_i) - \Psi(\Psi'(u_N(\mu_i)))\|_2^2 + \frac{\beta}{n}\|\Psi'(u_N(\mu_i)) \\ &\quad - \varphi(\mu_i)\|_2^2 + \gamma(\Delta p_{rom} - \Delta p_{fom})^2\end{aligned}$$

where γ is a user-defined coefficient, Δp_{rom} and Δp_{fom} are, respectively, the jump of pressure between the injection and the production computed with the reduced order model and the full order model.

In order to reach stable values of the statistics, having a negligible error due to the finite sampling, we generate a number of 900 instances from a uniform distribution of the uncertain parameters to compute the desired statistics. This value is inferred from the convergence analysis, made computationally cheap by the neural network approach, shown in Fig. 22.

Figure 23 shows the probability density function (pdf) of the q.o.i. estimated using a Gaussian kernel estimator [54]. Its profile is well reproduced by both reduced models.

We list in Table 5 the summary statistics of our interests. We notice a good agreement between the values; most significant discrepancies occur for small quantities, such as $\tilde{s}_{1,1}$ and

Table 3 Case study 3. Architecture of the networks that make up the reduced model. "fc" stands for fully connected layer, tw is the number of trainable weights

	layer	tw	tw tot
Encoder	fc , 2378, 404, PReLU fc, 404, 2, PReLU	961 928	1 926 735
Decoder	fc, 2, 404, PReLU fc, 404, 2378, PReLU	964 303	
Reduced map network	fc, 2, 100, PReLU fc, 100, 2, PReLU	504	

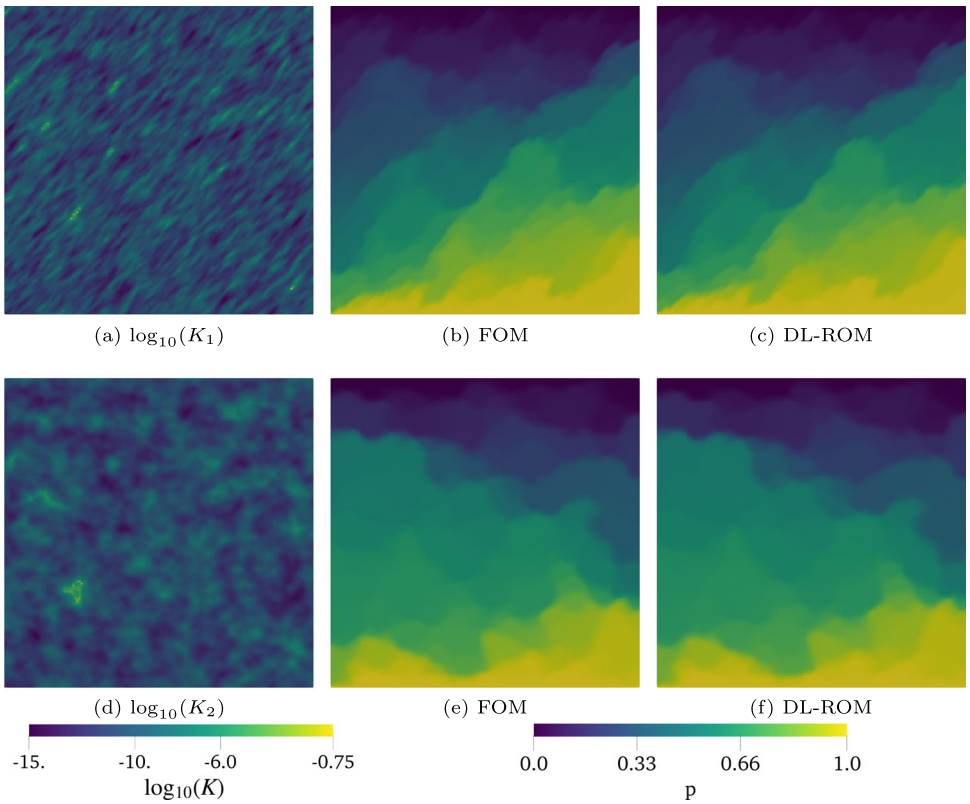


Fig. 20 Case study 4. The left panel represents the permeability os two extremal scenarios, specifically $\log_{10}(K_1)$ and $\log_{10}(K_2)$. The FOM and reconstructed pressures are shown in the middle and right panels, respectively

$\tilde{s}_{1,4}$, whose values are difficult to capture accurately. From the Sobol index, we can see that most of the variance in the pressure jump is due to the permeability of layer 1, the bottom layer, and layer 3, the caprock, since their permeabilities are low. The small displacement along the fault is less relevant, and the high-permeability upper layer and fault permeability play a minor role in this context.

We now want to estimate the total time required to assess this multi-query application. A single numerical solution of the full-order problem takes on average 0.58 s per core, see Fig. 10, using a 4 core CPU, we get a total time for the Monte Carlo evaluation of about 131 s. For the POD we need to consider the offline time to create the reduced model and the time to generate the required samples with the reduced model. Given the size of the training data set, we see that data generation takes $200 \times 0.58/4 = 29\ s$. The creation of the Φ matrix requires a negligible time compared to the other operations. The online time is 0.57 s, which is slightly less

Table 4 Case study 4. Architecture of the networks that make up the reduced model. "fc" stands for fully connected layer, tw is the number of trainable weights

	layer	# tw	# tw tot
Encoder	fc, 22500, 100, PReLU	45 003	2 338 531
Decoder	fc, 2, 100, PReLU	2 293 003	
	fc, 100, 100, PReLU		
	fc, 100, 100, PReLU		
	fc, 100, 22500, PReLU		
Reduced map network	fc, 2, 20, PReLU	525	
	fc, 20, 2, PReLU		

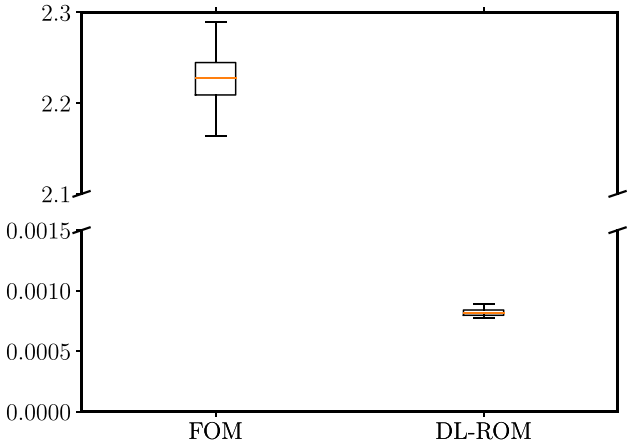


Fig. 21 Case study 4. Evaluation times in seconds of the full order model and reduced model obtained through the DL-ROM

than the full-order model time because the latter is already small and most of the time is spent deforming the mesh and reassembling the full-order matrix A. The POD sampling time results are equal to $900 \times 0.57/4 = 130\ s$, therefore, the total time to evaluate the sensitivity analysis is 159 s plus an additional time for additional routines that are common for all methods. Training in the neural network takes 400 epochs in 67 s. The overall sampling time is less than one second, which is negligible. The total time, which includes the training and validation data sets, is 96 s. This problem is very computationally cheap, so a reduced-order model technique does not show great computational time improvements, indeed the POD takes even longer than the full-order model. We recall that we chose this problem for the possibility of running the Monte Carlo analysis with the full-order model, hence to compare the accuracy of the results. Moreover, we repeat that these are just indicative numbers; the computational time depends on the relation between the algorithm, code, and hardware, since different parts of the algorithm run better on different hardware because of different code parallelization. Depending on the user's hardware availability, the outcomes may vary, but the neural network approach has an online time order of magnitude faster than the other methods, so for a problem large enough it will be convenient.

7.2 Inverse problem

The goal of this application is to determine the parameters such that a q.o.i. is equal to a desired value. We take Case 2 and set the pressure difference between injection and production equal to the desired value of $\Delta p_d = 0.188$ derived from the following set of parameters: $K_1 = 0.1$, $K_2 = 150$, $K_3 = 1 \times 10^{-4}$, $K_4 = 9.5 \times 10^{-4}$, $h = 0.09$. Assuming that some parameters, K_2 and K_4, are fairly known while some others are more uncertain, the ranges where we seek a solution are: $K_1 \in [10^{-4}, 1]$, $K_2 \in [10^2, 2 \times 10^2]$, $K_3 \in [10^{-6}, 10^{-4}]$, $K_4 \in [9 \times 10^{-4}, 10^{-3}]$, and $h \in [0.01, 0.1]$.

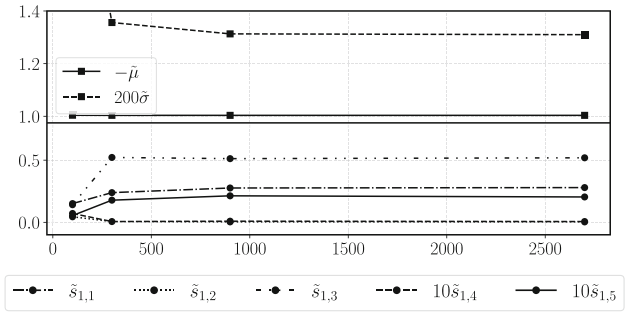


Fig. 22 Convergence of relevant statistics. The values reach a stable value with a set of 900 samples or more

In this example, we show only the application of DL-ROM. Similarly to the previous application, neural networks are trained on a small dataset made up of 200 snapshots because high precision is not required for the purpose of the current application. The training data set contains solutions sampled from the ranges defined in Section 6.1 that are stricter than those considered in this application, so we indirectly exploit the extrapolation capacity of the reduced model.

We cast the inverse problem as: $\min_\mu F(\mu)$, where $F(\mu) = (\Delta p(\mu) - \Delta p_d)^2$. We want to show that DL-ROM allows us to use heuristic optimization algorithms, which usually require a large number of evaluations of the objective function. We select the differential evolution algorithm [55] implemented in Scipy [56] with the default setting, except for the convergence tolerance: $tol = 0.001$ and $atol = 10^{-10}$. After 300 iterations, with a total number of objective function evaluations of 22581, the algorithm satisfies the convergence criterion. The optimal solution gives $K_1 = 0.251$, $K_2 = 153$, $K_3 = 9.1 \times 10^4$, $K_4 = 9.4 \times 10^{-4}$, and $h = 0.08$. We observe

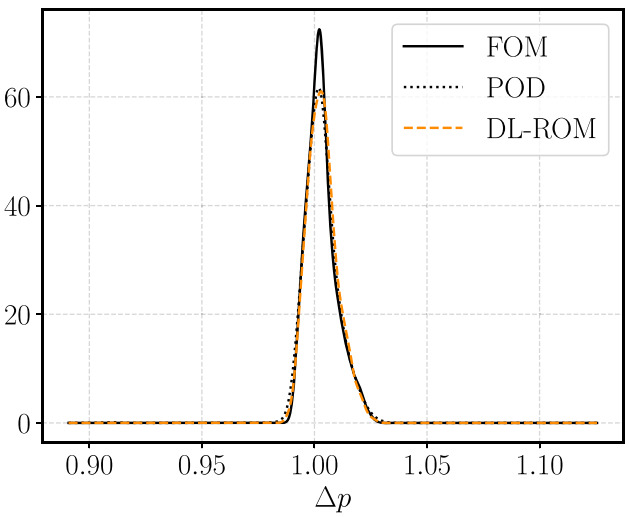


Fig. 23 Probability density function of the quantity of interest, Δp. Both POD and DL-ROM well reproduce the shape

Table 5 Mean value, $\overline{\Delta p}$, deviation, $\tilde{\sigma}$, and first order sensitivity index, $\tilde{s}_{1,i}$, of the quantity of interest obtained with data generated by the FOM, POD, and DL-ROM

	$\overline{\Delta p}$	$\tilde{\sigma} \times 10^3$	$\tilde{s}_{1,1} \times 10^1$	$\tilde{s}_{1,2}$	$\tilde{s}_{1,3} \times 10^1$	$\tilde{s}_{1,4} \times 10^3$	$\tilde{s}_{1,5} \times 10^2$
FOM	1.0033	6.61	3.11	7.04×10^{-8}	5.86	0.32	0.85
POD	1.0028	7.79	1.28	8.16×10^{-3}	3.62	9.37	2.45
DL-ROM	1.0036	6.57	2.76	3.73×10^{-3}	5.12	0.92	2.13

that the optimal values of the parameters are similar to the exact ones except for K_1 and a small error affects also the value of h.

The Δp obtained with the reduced model is equal to 0.188 as requested, while taking the optimal solution, recomputing the results of Δp with the results of the full-order model equal to 0.196, close enough to the desired value.

See Fig. 24 for the full-order model pressure field obtained with the optimal solution.

Due to the high number of objective function evaluations, we have considerable time savings: the offline phase takes about 75 min for snapshot generation and 4 min for neural network training, while 22581 runs would require 142 hours with the FOM but only 136 s with the use of the DL-ROM. Therefore, the total computational time is approximately 142 hours without the use of reduced models, and 1 h 18 min with the DL-ROM.

8 Conclusion

We focussed on reduced-order modeling techniques applied to the problem of a single-phase flow in rigid porous media with an arbitrary number of fractures and faults. The mixed-dimensional framework lets us efficiently deal with geometrical discontinuities (fractures or faults) in a geometry-deforming setup. It was possible to generate the data (solution of the full-order model) and create the reduced-order model without any further devices. We considered uncertainties in the parameters with respect to the physical properties of the rock and the geometry, whereas uncertainties in the fluid properties represent interesting possible future developments.

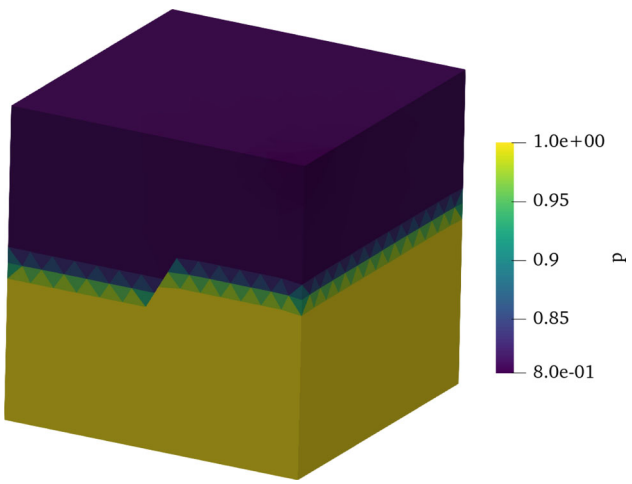


Fig. 24 Full order model pressure field obtained from the optimal solution

According to the results shown by our tests, the DL-ROM takes slightly longer compared to classic methods as the POD to generate the reduced model because of the training of the neural networks, but the online time is extremely low, which makes this approach promising, especially when the number of queries required is large.

Similarly to block-POD, the efficiency of DL-ROM could be improved by segregating variables and severing the links between disparate physical variables, i.e., p, p_γ, λ. This would decrease the total trainable weights, thereby expediting the training phase. Although we have conducted an initial review of this approach, it remains under development and is the subject of future research.

A disadvantage of the neural network approach is the large number of hyperparameters, such as the size of the training dataset, architecture of the neural networks, parameters of the optimization algorithm, etc., for which some user experience is needed since they affect the accuracy of the reduced model. We showed that the ROM strategies, and in particular the DL-ROM, lead to an advantage in terms of faster analysis with satisfying accuracy, so further investigations will be undertaken on the line of multifidelity ROM, and study of applications of ROM strategies to time-dependent problems simulating more realistic and complex physics. For instance, this could involve studying scenarios such as two-phase flows in fractured porous media, which presents, in addition to the geometric discontinuities, the challenge due to the sharp fronts resulting from the hyperbolic nature of the problem. This highly nonlinear scenario may pose strong difficulties for methods based on a linear map, $\mathcal{M}$, while promoting nonlinear methods such as the DL-ROM.

A Nomenclature

a	Trainable parameter of PReLU
A	Matrix describing the discrete equation
b	Discrete right-hand side
$C, C_d, C_{df}, C_s, C_{sf}$	Control points sets
$\overline{C}_s$	Set of surfaced where sliding conditions are applied
$d(x, y)$	Euclidean distance between x and y
D	Spatial dimension
e	Number of parameters
$e_{min}, e_{max}, e_{ave}$	Minimum, maximum, averaged relative errors between the FOM and ROM solution
f [1/s]	Scalar source or sink term
f_γ [1/s]	Scalar source or sink term in the fault
G	Matrix of displacement constraint
g	Radial basis function dependent on the distance, d, of two points, x_1, x_2
g^*	Radial basis function dependent on x_1, x_2
$g^\dagger$	Modified radial basis function
H_k	Matrix of no tangential contribution constraint applied to surface k
$\mathcal{I}$	influence function
l	Number of control points
N	Number of degrees of freedom of the full order problem
n	Number of degrees of freedom of the reduced problem
n_{br}	number of intersecting branches
$n_{\min}$	Minimal latent dimension
p [Pa]	Pressure
$\overline{p}$ [Pa]	Pressure on boundaries
p_γ [Pa]	Pressure in the fault
p_ι [Pa]	Pressure at the intersection
$\overline{\Delta p}$ [Pa]	Mean value of Δp
q [m s^{-1}]	Darcy velocity
$\overline{q}$ [m s^{-1}]	Darcy velocity on boundaries
K [m^{3}s/kg]	Intrinsic permeability scaled by the dynamic viscosity
K_n [m^{3}s/kg]	Normal fault permeability
K_τ [m^{3}s/kg]	In-plane fault permeability
K_ι [m^{3}s/kg]	Representative permeability at intersection
r	Map from γ to $\partial_{in}\Omega$
S	Snapshot matrix
s [m]	Displacement
$\overline{s}$ [m]	Known displacement
$\tilde{s}$	First order sensitivity index
t, b	Non-parallel tangent unit vectors of sliding surface
u_N	Full order model solution
$\tilde{u}_N$	Reconstructed solution
u_n	Reduced order model solution
U	Left singular vector matrix
U_{tr}	Left singular vector matrix truncated
V	Right singular vector matrix
z	Unknown of mesh deformation linear system
$\mathscr{L}$	Loss function

$\mathcal{M}$	Map from full order model space to reduced space
$\mathcal{S}$	Solution manifold
$\mathcal{V}_n$	Reduced problem solution space
$\mathcal{V}_N$	Full order model solution space
α, β	User-defined loss function weights
β	Side function
γ	Fault domain
$\partial_p\gamma$	Boundary of γ where Dirichlet boundary condition for the pressure is applied
$\partial_q\gamma$	Boundary of γ where Neumann boundary condition is applied
$\partial_{ex}\gamma$	Boundary of γ in contact with $\partial\Omega$
$\partial_{in}\gamma$	Boundary of γ not in contact with $\partial\Omega$
$\gamma^+(\gamma^-)$	Additional interfaces between the matrix domain, Ω and fault domain, γ
δ_n	Non-linear counterpart of Kolmogorov n-width.
ϵ [m]	Fault aperture
ζ	Unknown coefficients of the linear combination of radial functions
η	Exponent defining the permeability
$\lambda^+(\lambda^-)$ [m s^{-1}]	Volumetric fluid flux exchanged between subdomains
λ_γ [m s^{-1}]	Volumetric fluid flux exchanged between branches of a intersection
μ	Parameters
μ_{geom}	Geometrical parameters
μ_{phy}	Physical parameters
ν	Normal of a sliding surface
ρ	Activation function
σ	Right-hand side of mesh deformation system
$\tilde{\sigma}$	Standard deviation
θ	Scalar function μ-dependent
Θ	Parameter space
Σ	Singular values matrix
υ	Unit vector
υ_γ	Unit vector associated to γ
$\hat{\upsilon}$	Unit vector aligned with the fault
φ	Map $\varphi : \Theta \rightarrow \mathcal{V}_n$.In the DL-ROM approach, it is represented by the reduced map network
Φ	Transition matrix
Ψ	Map $\Psi : \mathcal{V}_n \rightarrow \mathcal{V}_N$.In the DL-ROM approach, it is represented by a decoder
Ψ'	Map $\Psi' : \mathcal{S} \rightarrow \mathcal{V}_n$.In the DL-ROM approach, it is represented by an encoder
Ω	Matrix domain
$\partial\Omega$	Boundary of Ω
$\partial_p\Omega$	Boundary of Ω where Dirichlet boundary condition for the pressure is applied
$\partial_q\Omega$	Boundary of Ω where Neumann boundary condition is applied
$\partial_{ex}\Omega$	External boundary of Ω
$\partial_{in}\Omega$	Internal boundary of Ω

Funding Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement. The research has been supported by the Italian Ministry of Universities and Research (MUR) under the project "PON Ricerca e Innovazione 2014-2020" and was carried out in collaboration with Eni S.p.A. The last three authors also gratefully acknowledge the support of the "Dipartimento di Eccellenza 2023-2027".
All authors warmly thank Andrea Manzoni, Stefano Micheletti, and Nicola Rares Franco for many insightful discussions.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Nordbotten, J.M., Celia, M.A.: Geological storage of CO_2 modeling approaches for large-scale simulation. Wiley (2012)
2. Lu, C., Sun, Y., Buscheck, T.A., Hao, Y., White, J.A., Chiaramonte, L.: Uncertainty quantification of CO_2 leakage through a fault with multiphase and nonisothermal effects. Greenhouse Gases: Science and Technology **2**(6), 445–459 (2012). https://doi.org/10.1002/ghg.1309
3. Quarteroni, A., Manzoni, A., Negri, F.: Reduced basis methods for partial differential equations. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-15431-2
4. Hesthaven, J.S., Rozza, G., Stamm, B.: Certified reduced basis methods for parametrized partial differential equations. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-22470-1
5. Brunton, S.L., Kutz, J.N.: Data-driven science and engineering: machine learning, dynamical systems, and control. Cambridge University Press (2019). https://doi.org/10.1017/9781108380690
6. Benner, P., Mehrmann, V.L., Sorensen, D.C.: Dimension reduction of large-scale systems. Springer (2005)
7. Maday, Y., Nguyen, N.C., Patera, A.T., Pau, S.H.: A general multipurpose interpolation procedure: the magic points. Commun. Pure Appl. Anal. **8**(1), 383–404 (2009). https://doi.org/10.3934/cpaa.2009.8.383
8. Schmid, P.J.: Dynamic mode decomposition of numerical and experimental data. J. Fluid Mech. **656**, 5–28 (2010). https://doi.org/10.1017/s0022112010001217
9. Kalur, A., Mortimer, P., Sirohi, J., Geelen, R., Willcox, K.E.: Data-driven closures for the dynamic mode decomposition using quadratic manifolds. In: AIAA AVIATION 2023 forum. American Institute of Aeronautics and Astronautics (2023). https://doi.org/10.2514/6.2023-4352
10. Peherstorfer, B., Willcox, K., Gunzburger, M.: Survey of multifidelity methods in uncertainty propagation, inference, and optimization. SIAM Rev. **60**(3), 550–591 (2018). https://doi.org/10.1137/16m1082469
11. Peherstorfer, B., Willcox, K.: Dynamic data-driven reduced-order models. Comput. Methods Appl. Mech. Eng. **291**, 21–41 (2015). https://doi.org/10.1016/j.cma.2015.03.018
12. Ahmed, H.F., Farooq, H., Akhtar, I., Bangash, Z.: Machine learning-based reduced-order modeling of hydrodynamic forces using pressure mode decomposition. Proc. Inst. Mech. Eng., Part G: J. Aerosp. Eng. **235**(16), 2517–2528 (2021). https://doi.org/10.1177/0954410021999864
13. Im, S., Lee, J., Cho, M.: Surrogate modeling of elasto-plastic problems via long short-term memory neural networks and proper orthogonal decomposition. Comput. Methods Appl. Mech. Eng. **385**, 114030 (2021). https://doi.org/10.1016/j.cma.2021.114030
14. Fu, J., Xiao, D., Fu, R., Li, C., Zhu, C., Arcucci, R., Navon, I.M.: Physics-data combined machine learning for parametric reduced-order modelling of nonlinear dynamical systems in small-data regimes. Comput. Methods Appl. Mech. Eng. **404**, 115771 (2023). https://doi.org/10.1016/j.cma.2022.115771
15. Gonzalez, F.J., Balajewicz, M.: Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. arXiv:1808.01346 (2018). https://doi.org/10.48550/ARXIV.1808.01346
16. Murata, T., Fukami, K., Fukagata, K.: Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. J. Fluid Mech. **882** (2019). https://doi.org/10.1017/jfm.2019.822
17. Hasegawa, K., Fukami, K., Murata, T., Fukagata, K.: Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. Theoret. Comput. Fluid Dyn. **34**(4), 367–383 (2020). https://doi.org/10.1007/s00162-020-00528-w
18. Fresca, S., Dede', L., Manzoni, A.: A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. J. Sci. Comput. **87**(2) (2021). https://doi.org/10.1007/s10915-021-01462-7
19. Fresca, S., Manzoni, A.: Real-time simulation of parameter-dependent fluid flows through deep learning-based reduced order models. Fluids **6**(7), 259 (2021). https://doi.org/10.3390/fluids6070259
20. Fresca, S., Manzoni, A.: POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. Comput. Methods Appl. Mech. Eng. **388**, 114181 (2022). https://doi.org/10.1016/j.cma.2021.114181
21. Fresca, S., Fatone, F., Manzoni, A.: Long-time prediction of nonlinear parametrized dynamical systems by deep learning-based reduced order models. Math. Eng. **5**(6), 1–36 (2023). https://doi.org/10.3934/mine.2023096
22. Franco, N., Manzoni, A., Zunino, P.: A deep learning approach to reduced order modelling of parameter dependent partial differential equations. Math. Comput. **92**(340), 483–524 (2022). https://doi.org/10.1090/mcom/3781
23. Franco, N.R., Fresca, S., Manzoni, A., Zunino, P.: Approximation bounds for convolutional neural networks in operator learning. Neural Netw. **161**, 129–141 (2023). https://doi.org/10.1016/j.neunet.2023.01.029
24. Fu, R., Xiao, D., Navon, I., Fang, F., Yang, L., Wang, C., Cheng, S.: A non-linear non-intrusive reduced order model of fluid flow by auto-encoder and self-attention deep learning methods. Int. J. Numer. Meth. Eng. **124**(13), 3087–3111 (2023). https://doi.org/10.1002/nme.7240
25. Regazzoni, F., Pagani, S., Salvador, M., Dede', L., Quarteroni, A.: Latent dynamics networks (LDNets): learning the intrinsic dynam-

ics of spatio-temporal processes (2023). https://doi.org/10.48550/ARXIV.2305.00094
26. Wangen, M.: Physical principles of sedimentary basin analysis. Cambridge University Press (2009)
27. Boffi, D., Brezzi, F., Fortin, M.: Mixed finite element methods and applications. Springer Series in Computational Mathematics. Springer, Berlin Heidelberg (2013)
28. Martin, V., Jaffré, J., Roberts, J.E.: Modeling fractures and barriers as interfaces for flow in porous media. SIAM J. Sci. Comput. **26**(5), 1667–1691 (2005). https://doi.org/10.1137/s1064827503429363
29. D'Angelo, C., Scotti, A.: A mixed finite element method for Darcy flow in fractured porous media with non-matching grids. Mathematical Modelling and Numerical Analysis **46**(02), 465–489 (2012). https://doi.org/10.1051/m2an/2011148
30. Flemisch, B., Berre, I., Boon, W., Fumagalli, A., Schwenck, N., Scotti, A., Stefansson, I., Tatomir, A.: Benchmarks for single-phase flow in fractured porous media. Adv. Water Resour. **111**, 239–258 (2018). https://doi.org/10.1016/j.advwatres.2017.10.036
31. Boon, W.M., Nordbotten, J.M., Yotov, I.: Robust discretization of flow in fractured porous media. SIAM J. Numer. Anal. **56**(4), 2203–2233 (2018). https://doi.org/10.1137/17m1139102
32. Berre, I., Boon, W.M., Flemisch, B., Fumagalli, A., Gläser, D., Keilegavlen, E., Scotti, A., Stefansson, I., Tatomir, A., Brenner, K., Burbulla, S., Devloo, P., Duran, O., Favino, M., Hennicker, J., Lee, I.H., Lipnikov, K., Masson, R., Mosthaf, K., Nestola, M.G.C., Ni, C.F., Nikitin, K., Schädle, P., Svyatskiy, D., Yanbarisov, R., Zulian, P.: Verification benchmarks for single-phase flow in three-dimensional fractured porous media. Adv. Water Resour. **147** (2020). https://doi.org/10.1016/j.advwatres.2020.103759
33. Formaggia, L., Fumagalli, A., Scotti, A., Ruffo, P.: A reduced model for darcy's problem in networks of fractures. ESAIM: Mathematical Modelling and Numerical Analysis **48**(4), 1089–1116 (2014). https://doi.org/10.1051/m2an/2013132
34. Nordbotten, J.M., Boon, W.M., Fumagalli, A., Keilegavlen, E.: Unified approach to discretization of flow in fractured porous media. Comput. Geosci. **23**(2), 225–237 (2018). https://doi.org/10.1007/s10596-018-9778-9
35. Blazek, J.: Computational Fluid Dynamics: Principles and Applications, 3rd edn. Butterworth-Heinemann, Oxford (2015). https://doi.org/10.1016/B978-0-08-099995-1.09986-3
36. Hirish, C.: Numerical Computation of Internal and External Flows. Elsevier (2007). https://doi.org/10.1016/b978-0-7506-6594-0.x5037-1
37. Aavatsmark, I.: An introduction to multipoint flux approximations for quadrilateral grids. Comput. Geosci. **6**(3), 405–432 (2002). https://doi.org/10.1023/a:1021291114475
38. Nordbotten, J.M., Keilegavlen, E.: In: Polyhedral Methods in Geosciences, pp. 119–158. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-69363-3_4
39. Starnoni, M., Berre, I., Keilegavlen, E., Nordbotten, J.M.: Consistent MPFA discretization for flow in the presence of gravity. Water Resour. Res. **55**(12), 10105–10118 (2019). https://doi.org/10.1029/2019wr025384
40. Stefansson, I., Berre, I., Keilegavlen, E.: Finite-volume discretisations for flow in fractured porous media. Transp. Porous Media **124**(2), 439–462 (2018). https://doi.org/10.1007/s11242-018-1077-3
41. Keilegavlen, E., Berge, R., Fumagalli, A., Starnoni, M., Stefansson, I., Varela, J., Berre, I.: Porepy: an open-source software for simulation of multiphysics processes in fractured porous media. arXiv:1908.09869 (2019). https://doi.org/10.48550/ARXIV.1908.09869
42. Benner, P., Gugercin, S., Willcox, K.: A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev. **57**(4), 483–531 (2015). https://doi.org/10.1137/130932715
43. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrika **1**(3), 211–218 (1936). https://doi.org/10.1007/bf02288367
44. Schmidt, E.: Zur theorie der linearen und nichtlinearen integralgleichungen. Math. Ann. **63**(4), 433–476 (1907). https://doi.org/10.1007/bf01449770
45. DeVore, R.A., Howard, R., Micchelli, C.: Optimal nonlinear approximation. Manuscripta Math. **63**(4), 469–478 (1989). https://doi.org/10.1007/bf01171759
46. de Boer, A., van der Schoot, M., Bijl, H.: Mesh deformation based on radial basis function interpolation. Computers & Structures **85**(11), 784–795 (2007). https://doi.org/10.1016/j.compstruc.2007.01.013
47. Forti, D., Rozza, G.: Efficient geometrical parametrisation techniques of interfaces for reduced-order modelling: application to fluid-structure interaction coupling problems. Int. J. Comput. Fluid Dyn. **28**(3–4), 158–169 (2014). https://doi.org/10.1080/10618562.2014.932352
48. Aubert, S., Mastrippolito, F., Rendu, Q., Buisson, M., Ducros, F.: Planar slip condition for mesh morphing using radial basis functions. Procedia Eng. **203**, 349–361 (2017). https://doi.org/10.1016/j.proeng.2017.09.819
49. Kingma, D.P., Ba, J.,: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014). https://doi.org/10.48550/ARXIV.1412.6980
50. Berre, I., Boon, W.M., Flemisch, B., Fumagalli, A., Gläser, D., Keilegavlen, E., Scotti, A., Stefansson, I., Tatomir, A., Brenner, K., Burbulla, S., Devloo, P., Duran, O., Favino, M., Hennicker, J., Lee, I.H., Lipnikov, K., Masson, R., Mosthaf, K., Nestola, M.G.C., Ni, C.F., Nikitin, K., Schädle, P., Svyatskiy, D., Yanbarisov, R., Zulian, P.: Verification benchmarks for single-phase flow in three-dimensional fractured porous media. Adv. Water Resour. **147**, 103759 (2021). https://doi.org/10.1016/j.advwatres.2020.103759
51. Winter, R., Valsamidou, A., Class, H., Flemisch, B.: A study on darcy versus forchheimer models for flow through heterogeneous landfills including macropores. Water **14**(4), 546 (2022). https://doi.org/10.3390/w14040546
52. Feinberg, J., Langtangen, H.P.: Chaospy: an open source tool for designing methods of uncertainty quantification. J. Comput. Sci. **11**, 46–57 (2015). https://doi.org/10.1016/j.jocs.2015.08.008
53. Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M.: Sensitivity Analysis in Practice, p. 232. Wiley (2004)
54. Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley (1992). https://doi.org/10.1002/9780470316849
55. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997). https://doi.org/10.1023/a:1008202821328
56. Virtanen, P., Gommers, R., Oliphant, T., et al.: SciPy 1.0: fundamental algorithms for scientific computing in python. Nature Methods **17**(3), 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2