International Journal for Numerical Methods in Engineering

WILEY

RESEARCH ARTICLE

Deep Learning-Based Reduced-Order Modeling of Darcy-Flow Systems With Local Mass Conservation

Wietse M. Boon[1] | Nicola R. Franco[2] | Alessio Fumagalli[2] | Paolo Zunino[2]

[1]Division of Energy and Technology, NORCE Norwegian Research Centre, Bergen, Norway | [2]MOX, Department of Mathematics, Politecnico di Milano, Milan, Italy

Correspondence: Wietse M. Boon (wibo@norceresearch.no)

Received: 4 July 2025 | **Revised:** 8 October 2025 | **Accepted:** 31 October 2025

Keywords: linear constraints | neural networks | reduced-order modeling

ABSTRACT

We propose a new reduced-order modeling (ROM) strategy for tackling parametrized Darcy-flow systems in which the constraint is given by mass conservation. Our approach employs classical neural-network architectures and supervised learning, but it is constructed in such a way that the resulting ROM is guaranteed to satisfy the linear constraints exactly. The procedure is based on a splitting of the solution into a particular solution satisfying the constraint and a homogenous solution. The homogeneous solution is approximated by mapping a suitable potential function, generated by a neural-network model, onto the kernel of the constraint operator. For the particular solution, instead, we propose an efficient spanning-tree algorithm. Starting from this paradigm, we present three approaches that follow this methodology, obtained by exploring different choices of the potential spaces: derived either via proper orthogonal decomposition (POD) or using the properties of a differential complex. To demonstrate the effectiveness of the proposed strategies and to emphasize their advantages over neural-network regression approaches, we present a series of numerical experiments, ranging from mixed-dimensional problems to nonlinear systems.

1 | Introduction

Numerical simulation of mathematical models based on partial differential equations (PDEs) is an essential component of many engineering applications. In areas such as geomechanics, hydrology, and management of subsurface resources, Darcy-type PDEs play a fundamental role in modeling the complex phenomenon of fluid flow in porous media [1, 2]. Typically, in the case of stationary problems, these are of the form

$$A_q \boldsymbol{q} + \nabla p = \boldsymbol{g}, \qquad \nabla \cdot \boldsymbol{q} = f, \tag{1.1}$$

where $\boldsymbol{q}$ and p denote the fluid velocity and its pressure field, respectively. Here, $\boldsymbol{g}$ and f are problem data, while A_q is some problem-specific invertible operator, possibly nonlinear. Due to the complexity of obtaining an accurate value

of these parameters, the model can be affected by uncertainty; an analysis of multiple scenarios is essential to obtain a reliable solution, especially for real applications. This challenge is not isolated to computational geosciences, but is also prevalent in various computational domains where numerical simulations are used to solve complex physical models [3–5].

To address these issues, a well-established approach is to replace the expensive calls to the numerical solver with a cheaper, but accurate, surrogate; a paradigm that is also known as reduced order modeling (ROM) [6, 7]. Here, we shall focus on data-driven nonintrusive ROMs based on deep learning techniques, which have recently proven extremely flexible and powerful (see, e.g., [8, 9]) The authors and coworkers have also recently contributed to this study [10, 11].

The driving idea behind these approaches is to construct the ROM by learning from a trusted high-fidelity solver, the full-order model (FOM), whose quality is taken as a ground truth reference. In practice, the FOM is exploited *offline* to sample a collection of PDE solutions, from which the ROM is left to learn the implicit mapping between the model parameters and the corresponding solutions. Then, after the training phase, the resulting ROM can be queried *online* at any time at a negligible computational cost. Compared to other, more traditional strategies, such as the Reduced Basis method, deep learning-based ROMs have the advantage of being completely nonintrusive, which makes them extremely fast online, and intrinsically nonlinear, which provides them with the ability to handle both complex and singular behaviors. The main contribution of this work lies in this context.

Specifically, we stress that our main goal is not to solve forward problems, meaning that we do not plan to replace classical numerical solvers. Instead, we aim at constructing computationally efficient surrogates that can facilitate other tasks that rely on a forward model, such as inverse modeling, parameter estimation, uncertainty quantification, and optimal control. In all such cases, a compromise between computational efficiency and overall accuracy is needed, and neural networks can provide a viable alternative. However, these methods have their limitations, which motivate this work.

One main issue is that, because of their intrinsically data-driven nature, deep learning-based ROMs are completely unaware of the underlying physics, which typically results in ROMs yielding unphysical results. For example, for problems such as (1.1), a naive data-driven ROM will likely violate the physical constraint regarding mass conservation. That is, given a suitable configuration of the model parameters, it will produce an output $\tilde{\boldsymbol{q}}$ that is a good proxy of the ground truth solution $\tilde{\boldsymbol{q}} \approx \boldsymbol{q}$, but for which $\nabla \cdot \tilde{\boldsymbol{q}} \neq f$. Clearly, there are many engineering applications for which this phenomenon is undesirable or even unacceptable. Although we may tolerate a small error in the constitutive equation, we might also demand that some of the physical constraints be satisfied *exactly*.

In Machine Learning terms, this corresponds to including the physical constraints within the *inductive bias* of the learning algorithm, thus making the model aware of the underlying mathematical structure. Inductive bias is the set of preexisting assumptions that a model uses to make predictions, effectively allowing it to extrapolate outside of the training set. In the context of data-driven ROMs, the introduction of inductive biases serves to mitigate the phenomenon of *overfitting* [12], and to accelerate the convergence of deep learning models toward the desired solution. When it comes to physical systems, this translates into the potential inclusion of biases based on first principles, such as conservation laws or geometric symmetries [13–15], among others. Inductive bias can be introduced at different levels of the surrogate model architecture. For example, Physics-Informed Neural Networks (PINNs) [9, 16] incorporate the governing PDEs directly into the loss function, thereby enforcing constraints in a *weak* sense. Other prominent operator-learning frameworks also embed strong inductive biases: DeepONets [17] exploit the underlying mathematical structure of continuous operators, introducing inductive biases for compositionality and locality inspired by the Kolmogorov–Arnold theorem; Fourier neural operators (FNOs) [18] operate in the frequency domain by learning only low-frequency modes through Fast Fourier Transforms, effectively acting as spectral filters that privilege smooth PDE solutions. More recently, physics-data combined approaches have been developed, in which physics is enforced through loss function regularization combined with data-driven terms, as in [19, 20].

More in general, because of their importance in flow and electromagnetic modeling, the aim of satisfying conservation laws has spawned a broad class of deep learning approaches. An overview is provided in [21], which emphasizes that enforcing constraints through the loss function, as in PINNs [9, 16] and in their conservative variation [22, 23], generally does not produce satisfactory results. Pioneering efforts in the development of exact constraint-preserving networks are reported in [24–26], where the proposed methods ensure that machine-learned models strictly enforce physics, even in challenging training scenarios, showing promising results in subsurface flows and electromagnetic modeling. Within the

same area of study, we also mention [27], in which analytical constraints are enforced by mapping into the kernel of the constraint matrix.

Here, we restrict our attention to linear constraints, with particular emphasis on equations in divergence form, as in (1.1). We propose a deep learning strategy that relies on separating the flux solution into two components: a homogeneous part and a nonhomogeneous part that satisfies the constraint of interest. These two contributions are then approximated separately. To approximate the homogeneous solution, we combine the action of a neural-network architecture, mapping the model parameters onto a suitable potential function, with that of a kernel projector, effectively binding the output within the null space of the constraint operator. At the same time, we exploit an efficient spanning-tree algorithm to retrieve the nonhomogeneous part, which is guaranteed to satisfy the linear constraint exactly. This construction ensures that the resulting reduced model is *conservative by design*, even in settings where standard neural-network surrogates would typically violate conservation. To the best of our knowledge, no comparable approach has been proposed in the context of deep learning ROMs, where enforcing nonhomogeneous constraints remains largely unexplored.

Our approach is reminiscent of [28] in the context of reduced basis methods, where the observation is made that, when employing proper orthogonal decomposition (POD) over solenoidal snapshots, each basis function of the reduced space is itself solenoidal. A *shift* of the solution is then introduced by means of local flux equilibration. We expand on this idea in several directions. First, we exploit a broader class of solenoidal spaces (not necessarily restricted to the POD basis), and second, we construct the shift function through a spanning-tree algorithm of linear complexity, which makes the computation scalable and efficient. Moreover, while previous works such as [27] have shown how to enforce analytical constraints by mapping into the kernel of the constraint operator, our contribution further develops these ideas by addressing nonhomogeneous constraints explicitly and by providing concrete realizations of kernel mappings within a deep learning ROM framework. In this sense, the novelty of our method lies in demonstrating how exact physical constraints can be embedded into neural surrogates at the architectural level, yielding models that are both computationally efficient and physically reliable.

The remainder of this article is structured as follows. The next two subsections introduce the model problem and the adopted notational conventions. Section 2 introduces the approach from an abstract perspective and presents the basic assumptions underlying our construction. These concern the three main constituents of our proposed ROM strategy: (i) a right-inverse operator, which we discuss in full detail in Section 3, (ii) a kernel-mapping, to which we devote Section 4, and a (iii) potential neural network, addressed in Section 5. Section 6 combines these three constituents to obtain constraint-preserving neural-network methods. Section 7 contains numerical experiments that illustrate the performance of the proposed methods compared to simple neural-network regression (NNR) models. Finally, concluding remarks are given in Section 8.

1.1 | Model Problem

On a Lipschitz domain Ω, we consider stationary flow problems of the form: find the velocity $\boldsymbol{q}$ and pressure p that satisfy

$$A_{\boldsymbol{q}}\boldsymbol{q} + \nabla p = \boldsymbol{g}, \qquad \nabla \cdot \boldsymbol{q} = f, \tag{1.2a}$$

in which $A_{\boldsymbol{q}}$ is a positive operator (possibly nonlinear), for given boundary conditions, $\boldsymbol{g}$ is a vector source term, and f is a mass source term. The boundary conditions are

$$p = p_0, \text{ on } \partial_p\Omega, \qquad \boldsymbol{q} \cdot \boldsymbol{\nu} = q_0, \text{ on } \partial_q\Omega, \tag{1.2b}$$

with $\boldsymbol{\nu}$ the outward unit vector normal to $\partial\Omega$ and the disjoint decomposition $\partial_p\Omega \cup \partial_q\Omega = \partial\Omega$ for which we assume that $|\partial_p\Omega| > 0$. We focus on two exemplary flow regimes that have this structure:

- Darcy flow: $A_{\boldsymbol{q}}\boldsymbol{w} := \kappa^{-1}\boldsymbol{w}$ for some symmetric positive definite tensor κ.
- Darcy-Forchheimer flow: $A_{\boldsymbol{q}}\boldsymbol{w} := (\kappa_0^{-1} + \kappa_1^{-1}|\boldsymbol{q}|)\boldsymbol{w}$ for symmetric positive definite tensors κ_0, κ_1.

We assume that this is a parameterized partial differential equation and we collect all relevant parameters of $A_{\boldsymbol{q}}$, $\boldsymbol{B}$, $\boldsymbol{g}$, and f into the vector $\mu \in \boldsymbol{M}$, where $\boldsymbol{M}$ is the parameter space. When necessary, we will emphasize the dependency on μ using a superscript, for example, f^{μ} instead of f.

We assume that the model problem has been discretized and that we seek the solution in finite-dimensional spaces $\boldsymbol{Q} \times P$. For our model problem, the chosen spaces are the lowest-order Raviart-Thomas [29] pair:

$$\boldsymbol{Q} := \mathbb{RT}_0(\Omega_h), \qquad P := \mathbb{P}_0(\Omega_h),$$

in which Ω_h is a simplicial tessellation of Ω.

1.2 | Preliminaries and Notation

Given a Hilbert space $\boldsymbol{Q}$, let $\boldsymbol{Q}'$ be its dual. For an operator $B : \boldsymbol{Q} \to P'$, let B^* denote its adjoint, so that $B^* : P \to \boldsymbol{Q}'$. We will use the Sans Serif font to indicate the equivalent matrix of linear operators, that is, B for B with its transpose denoted by $\mathsf{B}^\top$. Let $N_Q := \dim(\boldsymbol{Q})$ such that $\boldsymbol{q} \in \boldsymbol{Q}$ can be represented as a vector $\mathbf{q} \in \mathbb{R}^{N_Q}$ that contains the values of the degrees of freedom. Similarly, let N_P, N_R, and N_M denote the dimensions of the finite-dimensional spaces P, R, and M, respectively.

2 | A Potential-Based Approach to Satisfy Linear Constraints

We first cast the problem in a slightly more general framework. Let $\boldsymbol{Q}$ and P be Hilbert spaces. We consider a class of problems that have the following form: Given functionals $(\boldsymbol{g}, f) \in \boldsymbol{Q}' \times P'$, find $(\boldsymbol{q}, p) \in \boldsymbol{Q} \times P$ such that

$$A_{\boldsymbol{q}}\boldsymbol{q} - B^* p = \boldsymbol{g}, \qquad \text{in } \boldsymbol{Q}', \tag{2.1a}$$

$$B\boldsymbol{q} = f, \qquad \text{in } P'. \tag{2.1b}$$

Here, $A_{\boldsymbol{q}} : \boldsymbol{Q} \to \boldsymbol{Q}'$ is a (non)linear, positive operator, while $B : \boldsymbol{Q} \to P'$ is linear and surjective. The variable p is the Lagrange multiplier that enforces the constraint.

Our approach relies on a decomposition of $\boldsymbol{q} \in \boldsymbol{Q}$ into a *particular* solution $\boldsymbol{q}_f$ and a *homogeneous* solution $\boldsymbol{q}_0$,

$$\boldsymbol{q} = \boldsymbol{q}_f + \boldsymbol{q}_0, \qquad B\boldsymbol{q}_f = f, \qquad \boldsymbol{q}_0 \in \boldsymbol{Q}_0 := Ker(B, \boldsymbol{Q}). \tag{2.2}$$

For the computation of the particular and homogeneous parts of the solution, we assume that the following three constituents, which are essential for our construction, are available:

C1. A computationally efficient operator $S_I : P' \to \boldsymbol{Q}$ that is a right-inverse of B, that is, for which

$$BS_I = I. \tag{2.3a}$$

C2. An auxiliary Hilbert space $\boldsymbol{R}$, which we refer to as the *potential space*, and a linear map $S_0 : \boldsymbol{R} \to \boldsymbol{Q}_0$. Then, by definition, we have

$$BS_0 = 0. \tag{2.3b}$$

C3. A neural-network map $\mathcal{N} : M \to \boldsymbol{R}$ that is trained to make the following approximation,

$$S_0 \mathcal{N} \mu \approx (I - S_I B)\boldsymbol{q}^\mu, \tag{2.3c}$$

where $\boldsymbol{q}^\mu \in \boldsymbol{Q}$ is the true solution to (2.1) associated with the values of the corresponding parameters $\mu \in M$.

The following diagram illustrates the introduced above:

$$M \xrightarrow{\mathcal{N}} R \xrightarrow{S_0} Q \underset{S_I}{\overset{B}{\rightleftarrows}} P'. \tag{2.4}$$

We are now ready to describe the three-step solution technique, which is a generalization of the procedure presented in [30]:

1. Compute the particular solution $\boldsymbol{q}_f = S_I f \in \boldsymbol{Q}$.
2. Use the neural network and S_0 to compute the homogeneous solution $\tilde{\boldsymbol{q}}_0 = S_0 \mathcal{N} \mu$.
3. Set $\tilde{\boldsymbol{q}} = \boldsymbol{q}_f + \tilde{\boldsymbol{q}}_0$. The approximation $\tilde{p}$ can be post-processed as:

$$\tilde{p} = (S_I^* B^*)\tilde{p} = S_I^*(A_{\tilde{q}}\tilde{\boldsymbol{q}} - \boldsymbol{g}).$$

Lemma 1. *The approximate solution $\tilde{\boldsymbol{q}}$ satisfies the linear constraint (2.1b), regardless of the precision of the mapping $\mathcal{N}$.*

Proof. By C1-C2, we have $B\tilde{\boldsymbol{q}} = B(\boldsymbol{q}_f + \tilde{\boldsymbol{q}}_0) = B(S_I f + S_0 \mathcal{N} \mu) = f$. □

We dedicate the next three sections to an in-depth discussion about the possible choices for the three constituents of the proposed procedure. That is, we shall discuss how to choose: a right-inverse of B (C1) which allows us to rapidly construct a particular solution for the problem, a potential space supplied with a kernel-mapping (C2), and a suitable neural-network architecture (C3) to efficiently compute the homogeneous solution. In this way, we propose a computationally inexpensive framework for the problem that respects the linear constraint up to machine precision. This final step is presented in Section 6.

Remark. To ease the exposition, we adopt the nomenclature for Darcy-flow systems throughout the manuscript. However, the constituents presented in the next sections are generalizable to a larger class of linearly constrained problems. The only exceptions are Sections 3.2 and 4.3, which rely on specific properties of the divergence operator. For an extension to linearized elasticity, in which linear and angular momentum are conserved, we refer the interested reader to [31].

3 | Constituent 1: A Right-Inverse of the Operator B

The first constituent concerns an operator that can be used to generate a particular solution $\boldsymbol{q}_f$ such that $B\boldsymbol{q}_f = f$. In principle, any right-inverse of B can be chosen: here, we shall focus on those admitting an explicit construction. The main idea is to choose an appropriate operator $E : \boldsymbol{Q} \to \boldsymbol{Q}$ such that BE is invertible. Then, we may let

$$S_I := E(BE)^{-1}, \tag{3.1}$$

as it is straightforward to see that $BS_I = I$, as required by C1. In the next two subsections, two examples of admissible operators E are considered.

3.1 | The Moore-Penrose Inverse

As a first example, we may set $E = LB^*$ for some $L : \boldsymbol{Q}' \to \boldsymbol{Q}$. However, in this work, we require an exact, right-inverse operator to ensure a precise satisfaction of the constraint. The system BLB^* therefore needs to be solved to machine precision for each input, which may be too computationally demanding, unless specific tailored solvers are available or if $\mathrm{Ran}(B)$ is of low dimensionality.

Remark. This structure was explored in [30] by recognizing that the finite volume method using two-point flux approximation is of this form with L a diagonal matrix.

3.2 | A Spanning-Tree Solve for Conservation Equations

Next, we focus on the particular case in which B is a divergence operator acting on the Raviart-Thomas space of lowest order, on a simplicial grid. In this case, we leverage the structure of the operator B and the fact that $\mathbb{RT}_0$ has one degree of freedom per face to find an appropriate mapping E for (3.1). Our construction is similar in spirit to [32], Sec. 7.3, and a generalization to other differential operators can be found in [33]. We now detail the construction of a particular right-inverse $S_I : P' \to \boldsymbol{Q}$, which we refer to as a *spanning-tree solve*. The idea goes as follows:

1. Find a column of $\mathsf{B} \in \mathbb{R}^{n \times m}$ that has exactly one entry $b_{i_0 j_0}$. The index j_0 corresponds to the degree of freedom of the facet in $\boldsymbol{Q}$ that is on the boundary, and i_0 is its neighboring element.

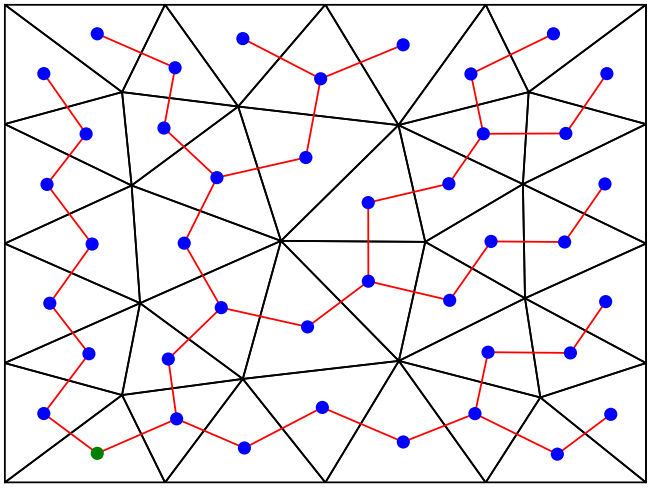

FIGURE 1 | A spanning-tree $\mathcal{T}$ superimposed on a triangular, unstructured grid. A particular solution $\boldsymbol{q}_f$ is generated that balances the mass source f by progressively solving the conservation equation from the leaves to the root of the tree at the domain boundary, illustrated in green in the bottom left of the figure. Only the degrees of freedom on faces cut by the tree are taken into account.

2. Construct a graph $\mathcal{G}$ with n nodes and m edges that has an incidence matrix with the same sparsity pattern as B. We emphasize that the graph nodes correspond to the mesh cells and the graph edges correspond to the mesh facets.
3. Generate a spanning-tree $\mathcal{T}$ of $\mathcal{G}$ with i_0 as its root. For example, we may use the tree generated by a breadth-first search of the graph, which is a computation of linear complexity. Note that $\mathcal{T}$ contains all nodes of $\mathcal{G}$ and a subset of its edges.
4. Let J be the set that contains the $(n-1)$ edges of $\mathcal{G}$ that are present in the tree $\mathcal{T}$, complemented by the edge with index j_0. Let $\Pi \in \mathbb{R}^{n\times m}$ be the restriction operator on the degrees of freedom of the n facet in the mesh that correspond to the edges in J.
5. Due to the tree structure, the matrix $\mathsf{B}\Pi^{\mathsf{T}} \in \mathbb{R}^{n\times n}$ is triangular, up to column/row permutations, and therefore leads to a system that is computationally easy to solve. We are now ready to define

$$\mathsf{S}_{\mathsf{I}} := \Pi^{\mathsf{T}}(\mathsf{B}\Pi^{\mathsf{T}})^{-1}. \tag{3.2}$$

We emphasize that this choice of S_I fits the format (3.1) by setting E as the adjoint of the restriction operator Π. In turn, C1 is directly satisfied. An example of a spanning tree is given in Figure 1.

Remark. Note that the set of right-inverses of B, namely

$$\{S : P' \to Q | BS = I\}, \tag{3.3}$$

is a convex set. This observation can be used to generate other candidates for the right-inverse of B, starting from the approaches described in Sections 3.1 and 3.2. For example, assume that $S_I^1, \dots, S_I^{N_t}$ are N_t right-inverses of B obtained by relying on N_t different spanning trees, as in Section 3.2. Then, the "average tree solver"

$$S_I^{\text{avg}} := \frac{1}{N_t}\sum_{i=1}^{N_t} S_I^i, \tag{3.4}$$

also operates as a right-inverse of B, thus fulfilling C1. In practice, as we shall discuss in Section 7, combining multiple trees as in (3.4) can substantially improve the quality of the model, especially in the approximation of the pressure field.

4 | Constituent 2: An Operator Mapping Onto the Kernel of B

The second constituent of our approach concerns an operator that can be used to compute the homogeneous solution $q_0 \in Q_0$. We propose three examples for this operator in the following subsections, and remark in each case on their surjectivity.

4.1 | A Choice Based on the Right-Inverse

We first consider a construction that follows directly from a given operator S_I that satisfies C1. Due to its simplicity, we present its definition together with its admissibility in the next proposition.

Proposition 1. *If S_I is chosen according to C1, then the choice*

$$\boldsymbol{R} := \boldsymbol{Q}, \qquad S_0 := I - S_I B, \tag{4.1a}$$

satisfies C2. Moreover, S_0 is a projection and

$$Ker(S_0) = \mathrm{Ran}(S_I B). \tag{4.1b}$$

Proof. A direct calculation gives us $BS_0 = B(I - S_I B) = B - B = 0$, which implies that $\mathrm{Ran}(S_0) \subseteq \boldsymbol{Q}_0$ verifying C2. Next, we note that

$$S_0 \boldsymbol{q}_0 = (I - S_I B)\boldsymbol{q}_0 = \boldsymbol{q}_0, \qquad \forall \boldsymbol{q}_0 \in \boldsymbol{Q}_0.$$

Hence S_0 is the identity operator on $\boldsymbol{Q}_0$ and is therefore a projection. To prove (4.1b), we first note that

$$S_0(S_I B) = S_I B - S_I(BS_I)B = 0$$

which shows that $Ker(S_0) \supseteq \mathrm{Ran}(S_I B)$. Secondly, for $q \in Ker(S_0)$, we have $(I - S_I B)q = 0 \Rightarrow q = S_I Bq$. In turn, $q \in \mathrm{Ran}(S_I B)$ from which we conclude $Ker(S_0) \subseteq \mathrm{Ran}(S_I B)$. □

4.2 | Proper Orthogonal Decomposition

We now consider an alternative mapping based on techniques common to reduced basis methods. We assume to be in the setting of model problem (1.2), so that $\boldsymbol{Q} = \mathbb{RT}_0$ and $\|\cdot\|_{\boldsymbol{Q}} = \|\cdot\|_{H(\mathrm{div})}$. With this setup, we assume that $\{\mu_i\}_{i=1}^{N_s}$ is a sampling of the parameter space, that is, $\mu_i \in \boldsymbol{M}$ for each index i with N_s the number of samples. Let $\{\boldsymbol{q}_0^{\mu_i}\}_{i=1}^{N_s}$ be the homogeneous parts of the corresponding flux fields. For given $n \leq N_Q$, we then construct the empirically optimal projector $V_n : \mathbb{R}^n \to \boldsymbol{Q}_0$ that minimizes the reconstruction error:

$$\begin{aligned} V_n &:= \operatorname*{argmin}_{W:\mathbb{R}^n \to \boldsymbol{Q}_0} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\boldsymbol{q}_0^{\mu_i} - WW^* \boldsymbol{q}_0^{\mu_i}\|_{\boldsymbol{Q}}^2 \\ &= \operatorname*{argmin}_{W:\mathbb{R}^n \to \boldsymbol{Q}_0} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\boldsymbol{q}_0^{\mu_i} - WW^* \boldsymbol{q}_0^{\mu_i}\|_{L^2}^2. \end{aligned} \tag{4.2}$$

The last equality follows from the fact that $\|\boldsymbol{q}\|_{\boldsymbol{Q}} = \|\boldsymbol{q}\|_{L^2}$ for all $\boldsymbol{q} \in \boldsymbol{Q}_0$. From a practical point of view, this is achieved by means of *POD, see, e.g.*, [7], Sec. 6.3. In particular, the solution to (4.2) is known in closed form and can be computed as follows. First, let the so-called *snapshots matrix* be defined as

$$\mathsf{U} := [\mathsf{q}_0^{\mu_1}, \dots, \mathsf{q}_0^{\mu_{N_s}}]^\top \in \mathbb{R}^{N_s \times N_Q}. \tag{4.3}$$

Second, let $\mathsf{M} \in \mathbb{R}^{N_Q \times N_Q}$ be the mass matrix representing the L^2-inner product over $\boldsymbol{Q}$, that is, the matrix that satisfies

$$\mathsf{u}^\top \mathsf{M} \mathsf{v} = \langle \boldsymbol{u}, \boldsymbol{v} \rangle_{L^2}$$

for all $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{Q}$ with vector representations $\mathsf{u}, \mathsf{v} \in \mathbb{R}^{N_Q}$. We then perform an eigenvalue decomposition to form matrices $\tilde{\mathsf{U}}, \Lambda \in \mathbb{R}^{N_s \times N_s}$ that satisfy

$$\mathsf{U}\mathsf{M}\mathsf{U}^\top = \tilde{\mathsf{U}} \Lambda \tilde{\mathsf{U}}^\top,$$

with $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_{N_s})$ containing the eigenvalues $\lambda_1 \geq \dots \geq \lambda_{N_s} \geq 0$. Let now $\tilde{\mathsf{U}}_n \in \mathbb{R}^{n \times N_s}$ be the matrix obtained by extracting the first n rows of $\tilde{\mathsf{U}}$. Similarly, let

$$\Lambda_n^{-1/2} := \mathrm{diag}\left(\frac{1}{\sqrt{\lambda_1}}, \dots, \frac{1}{\sqrt{\lambda_n}} \right).$$

Finally, the POD projector $V_n : \mathbb{R}^n \to \boldsymbol{Q}_0$ is defined according to its matrix representation as

$$V_n := \left(\Lambda_n^{-1/2}\tilde{\mathsf{U}}_n\mathsf{U}\right)^\top. \tag{4.4}$$

Proposition 2. *Let $n \leq N_Q$ and let V_n be as in (4.4). Then,*

$$\boldsymbol{R} := \mathbb{R}^n, \qquad S_0 := V_n,$$

satisfies C2.

Proof. By definitions (4.4) and (4.3), the range of V_n is a subspace of $\text{span}\{\boldsymbol{q}_0^{\mu_i}\}_{i=1}^{N_s}$. Consequently, $\text{Ran}(V_n) \subseteq \boldsymbol{Q}_0$. □

It is important to note that this choice of S_0 is generally not surjective on the kernel $\boldsymbol{Q}_0$. However, the range of V_n can be a good approximation of $\boldsymbol{Q}_0$ for sufficiently large n.

4.3 | Differential Complexes

Our final example concerns the case in which B corresponds to a differential operator. In particular, if a $d^k : \boldsymbol{Q} \to P$ exists such that

$$\langle B\boldsymbol{q}, \tilde{p}\rangle = \langle d^k\boldsymbol{q}, \tilde{p}\rangle_P \tag{4.5}$$

and d^k is an operator in a *differential complex*. Recall that a differential complex is a sequence of spaces Λ^k connected by operators $d^k : \Lambda^k \to \Lambda^{k+1}$ such that $d^k d^{k-1} = 0$ for all k. Identifying $P := \Lambda^{k+1}$, $\boldsymbol{Q} := \Lambda^k$, and $\boldsymbol{R} := \Lambda^{k-1}$, the differential complex is illustrated as

$$\cdots \xrightarrow{d^{k-2}} R \xrightarrow{d^{k-1}} Q \xrightarrow{d^k} P \xrightarrow{d^{k+1}} \cdots \tag{4.6}$$

The following result is directly obtained from the definitions.

Proposition 3. *Let d^k be such that (4.5) holds and let (4.5) be a differential complex with $\boldsymbol{Q} = \Lambda^k$. Then the choice*

$$\boldsymbol{R} := \Lambda^{k-1}, \qquad S_0 := d^{k-1},$$

satisfies C2.

In our model problem (1.2), we have the divergence operator $d^k\boldsymbol{q} = \nabla\cdot\boldsymbol{q}$ and $\boldsymbol{Q} = \mathbb{RT}_0$. In the case in which Ω is three-dimensional, we can use the theory of finite-element exterior calculus [34, 35] and consider the following differential complex:

$$\mathbb{N}_0(\Omega_h) \overset{\nabla\times}{\to} \mathbb{RT}_0(\Omega_h) \overset{\nabla\cdot}{\to} \mathbb{P}_0(\Omega_h).$$

in which $\mathbb{N}_0$ is the Nédélec edge element of the first kind [36]. We emphasize that this is a differential complex due to the vector calculus identity $\nabla\cdot\nabla\times\boldsymbol{r} = 0$ for all sufficiently regular $\boldsymbol{r}$. An analogous differential complex exists in the two-dimensional case, in which the nodal Lagrange space $\mathbb{L}_1(\Omega_h)$ replaces $\mathbb{N}_0(\Omega_h)$ and we have the rotated gradient $\nabla^\perp r := [\partial_2 r,\ -\partial_1 r]^\top$ as the curl.

We remark that S_0 as chosen in Proposition 3 is surjective onto $\boldsymbol{Q}_0$ if the domain is contractible. Otherwise, for example, if the domain has holes, the homogeneous solution may not be fully computable in this way. Extra steps then need to be taken to find the part of the solution that is in $Ker(d^k)/\text{Ran}(d^{k-1})$, also known as the cohomology space. In this work, we focus on the simplified case with zero cohomology.

5 | Constituent 3: A Neural-Network Model That Maps Onto the Potential Space

The third and final constituent of our approach consists of an efficient neural network $\mathcal{N} : M \to \boldsymbol{R}$ that can produce a suitable potential $\boldsymbol{r} = \boldsymbol{r}^\mu$ for any given parameter instance μ (cf. C3). We recall that the method proposed in Section 2 constructs an approximation $(\tilde{\boldsymbol{q}}, \tilde{p})$ for a given μ in the following manner:

$$\tilde{\boldsymbol{q}} = S_I f^\mu + S_0\mathcal{N}(\mu), \qquad \tilde{p} = S_I^*(A_{\tilde{q}}\tilde{\boldsymbol{q}} - g^\mu). \tag{5.1}$$

Since we have already defined the linear operators S_0 and S_I at this stage, we can directly train the neural network such that the map $\mu \mapsto (\tilde{\boldsymbol{q}}, \tilde{p})$ approximates the solution map $\mu \mapsto (\boldsymbol{q}^\mu, p^\mu)$. To this end, we note that the following holds true.

Lemma 2. *Assume that*:

i. The map $\mathcal{A} : \boldsymbol{q} \mapsto A_q$ is locally Lipschitz continuous.

ii. The solution map $\mathcal{Q} : \mu \mapsto \boldsymbol{q}^\mu$ is continuous.

iii. The source map $F : \mu \mapsto f^\mu$ is continuous.

Then, there exists a constant $C = C(\mathcal{A}, \mathcal{Q}, F, S_I, S_0, \mathcal{N}) > 0$ such that, for all $\mu \in M$ one has

$$\|p - \tilde{p}\|_P \leq C \|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}},$$

where $\boldsymbol{q} = \boldsymbol{q}^\mu$ and $p = p^\mu$, while $\tilde{\boldsymbol{q}}$ and $\tilde{p}$ are as in (5.1).

Proof. Let $\boldsymbol{Q}_M := \{\boldsymbol{q}^{\mu'} | \mu' \in M\} \subset \boldsymbol{Q}$ be the solution manifold associated with the flux variable. We note that, by continuity of the solution map, $\boldsymbol{Q}_M$ is compact. Since locally Lipschitz continuous maps are Lipschitz over compact sets, we may denote the Lipschitz constant of $\mathcal{A}$ over $\boldsymbol{Q}_M$ by L_A.

We have,

$$\begin{aligned}
\|p - \tilde{p}\|_P = \|S_I^*(A_q \boldsymbol{q} - A_{\tilde{q}} \tilde{\boldsymbol{q}})\|_P &\leq \|S_I\| \, \|A_q \boldsymbol{q} - A_{\tilde{q}} \tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}'} \\
&\leq \|S_I\| \big(\|A_q \boldsymbol{q} - A_q \tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}'} + \|A_q \tilde{\boldsymbol{q}} - A_{\tilde{q}} \tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}'} \big) \\
&\leq \|S_I\| \big(\|A_q\| \, \|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}} + \|A_q - A_{\tilde{q}}\| \, \|\tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}} \big) \\
&\leq \|S_I\| \big(\|A_q\| \, \|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}} + L_A \|\boldsymbol{q} - \tilde{\boldsymbol{q}}\| \, \|\tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}} \big),
\end{aligned}$$

in which $\|S_I\|$ and $\|A_q\|$ refer to the operator norms.

We now note that, since the maps $S_I, F, S_0, \mathcal{N}$ are continuous, so is the map $\mu \mapsto S_I f^\mu + S_0 \mathcal{N}(\mu)$. The compactness of M then implies that some $c' > 0$ exists such that $\|\tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}} \leq c'$, regardless of $\mu \in M$. Similarly, we can bound $\|A_q\|$ by some $c'' > 0$. It follows that

$$\|p - \tilde{p}\|_P \leq \|S_I\|(c'' + Lc') \|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{\boldsymbol{Q}},$$

as claimed. □

In other words, Lemma 2 shows that it suffices to train the potential network $\mathcal{N}$ to produce a reliable approximation of the flux, without directly taking care of the pressure. This allows us to avoid reassembling $A_{\tilde{q}}$ during the training phase. In light of these considerations, one possibility is to directly train the neural-network model $\mathcal{N}$ by minimizing the following loss function

$$\begin{aligned}
\mathcal{L}(\mathcal{N}) = \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \boldsymbol{q}^{\mu_i} - \left[S_I f_i^\mu + S_0 \mathcal{N}(\mu_i) \right] \right\|_{\boldsymbol{Q}}^2 &= \frac{1}{N_s} \sum_{i=1}^{N_s} \|(I - S_I B) \boldsymbol{q}^{\mu_i} - S_0 \mathcal{N}(\mu_i)\|_{\boldsymbol{Q}}^2 \\
&= \frac{1}{N_s} \sum_{i=1}^{N_s} \|\boldsymbol{q}_0^{\mu_i} - S_0 \mathcal{N}(\mu_i)\|_{\boldsymbol{Q}}^2,
\end{aligned} \tag{5.2}$$

where $\boldsymbol{q}_0^\mu := (I - S_I B) \boldsymbol{q}^\mu$, in line with our adopted notation. Here, $\{\mu_i, \boldsymbol{q}^{\mu_i}\}_{i=1}^{N_s} \subset M \times \boldsymbol{Q}$ is a suitable collection of training data, previously generated via the FOM.

In practice, optimizing (5.2) can become problematic if the potential space $\boldsymbol{R}$ is high-dimensional, as the network is required to produce a large number of outputs. In that case, one may enhance the training by introducing a *latent regularization*. This idea was first proposed by Fresca et al. in the so-called DL-ROM approach [10], where the network

architecture is decomposed in two components, $\phi : M \to \mathbb{R}^n$ and $\Psi : \mathbb{R}^n \to \boldsymbol{R}$, so that $\mathcal{N} = \Psi \circ \phi$. This splitting emphasizes the existence of a latent space, here denoted by $\mathbb{R}^n$, which lives in between the several hidden states of the overall architecture $\mathcal{N}$. Then, the DL-ROM approach proposes to replace (5.2) with

$$\mathscr{L}_{\text{reg}}(\phi, \Psi, \Psi') = \frac{1}{N_s}\sum_{i=1}^{N_s} \left\| \boldsymbol{q}_0^{\mu_i} - S_0(\Psi \circ \phi)(\mu_i) \right\|_{\boldsymbol{Q}}^2 + \lambda \frac{1}{N_s}\sum_{i=1}^{N_s} \left\| \Psi'(\boldsymbol{q}_0^{\mu_i}) - \phi(\mu_i) \right\|_{\mathbb{R}^n}^2, \tag{5.3}$$

where $\lambda > 0$ is a suitable regularization parameter, while $\Psi' : \boldsymbol{Q} \to \mathbb{R}^n$ is an auxiliary network that is used to enforce a connection between the final output and the latent representation of the model [37]. In summary, this approach requires the training of three networks: ϕ, Ψ, and Ψ'.

Remark. In the DL-ROM literature, for example, [10, 11], the maps Ψ' and Ψ are usually referred to as *encoder* and *decoder*. This is because the same idea can be explained using the concept of *autoencoders*. To see this, note that the regularization term makes the networks act so that $\Psi'(\boldsymbol{q}_0^{\mu}) \approx \phi(\mu)$ and $\boldsymbol{q}_0^{\mu} \approx S_0\Psi(\phi(\mu))$. It then follows that

$$\boldsymbol{q}_0^{\mu} \approx S_0(\Psi(\Psi'(\boldsymbol{q}_0^{\mu}))), \tag{5.4}$$

meaning that the map $S_0 \circ \Psi \circ \Psi'$ can operate as an autoencoder, effectively compressing and reconstructing the homogeneous part of flux solutions (this is why we chose to represent the latent dimension with the letter n, to highlight its parallelism with the reduced dimension in the POD approach, cf. Section 4.2). Here, however, we consider the DL-ROM approach as a regularization strategy for $\mathcal{N} := \Psi \circ \phi$, as talking about autoencoders may generate confusion between the flux space $\boldsymbol{Q}$ and the potential space $\boldsymbol{R}$. Furthermore, for our purposes, we are not interested in the quality of the reconstruction error (5.4).

6 | Combining the Constituents: Neural-Network Methods Respecting Linear Constraints

We are now ready to combine the three constituents and present the overall workflow of our approach. Based on the strategies described in the previous sections, we derive three implementations, each with its own benefits and limitations. For each, we choose the operator S_I to be the spanning-tree solve from Section 3.2, or an averaged variation (cf. Remark 3.2). Thus, the main differences lie in the choice of S_0, cf. Section 4, and in the corresponding neural-network architectures and training strategies. We highlight these in more detail in the following subsections.

6.1 | Conservative POD-NN

Our first proposal is to set $S_0 := V_n$, that is, as the POD projector from Section 4.2. The POD matrix is constructed starting from the homogeneous part of the flux, thus relying on the snapshots $\{\boldsymbol{q}_0^{\mu_i}\}_{i=1}^{N_s}$. This choice is particularly suited if the homogeneous part of the solution manifold, namely

$$\mathcal{S}_0 := \{(I - S_I B)\boldsymbol{q}^{\mu} | \mu \in M\} \subset \boldsymbol{Q}_0,$$

exhibits a fast decay of the Kolmogorov n-width (equivalently: it can be easily approximated by linear subspaces of reasonably small dimension). In practice, this can be deduced by looking at the decay of the singular values of the snapshot matrix.

In this case, the potential network, $\mathcal{N} : M \to \mathbb{R}^n$, can be implemented as any classical deep feedforward network, and no particular training strategy is required. Recall that n is the number of modes in the reduced basis obtained by POD. Additionally, thanks to the orthonormality of V_n, the minimization of the loss function in (5.2) can be equivalently replaced with that of

$$\mathscr{L}_{\text{POD}}(\mathcal{N}) = \frac{1}{N_s}\sum_{i=1}^{N_s} \| V_n^* \boldsymbol{q}_0^{\mu_i} - \mathcal{N}(\mu_i) \|_{\mathbb{R}^n}^2, \tag{6.1}$$

which is computationally more efficient to deal with. This fact is briefly summarized below.

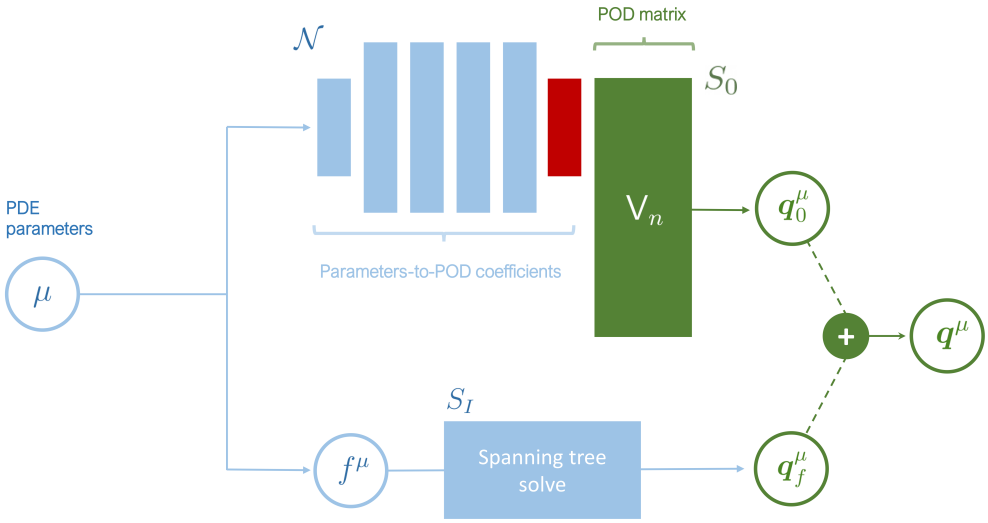

FIGURE 2 | Visual representation of the *Conservative POD-NN* approach, Section 6.1. In red, the potential space representation (here coinciding with the POD latent space).

Lemma 3. *Let $\mathcal{N}$ be any class of neural-network architectures from M to $\mathbb{R}^n$. Then,*

$$\operatorname*{argmin}_{\mathcal{N}\in\mathcal{N}} \mathscr{L}(\mathcal{N}) = \operatorname*{argmin}_{\mathcal{N}\in\mathcal{N}} \mathscr{L}_{\mathrm{POD}}(\mathcal{N}).$$

Proof. Due to the orthogonality of projection residuals, one has

$$\begin{aligned}\|\boldsymbol{q}_0^{\mu_i} - V_n\mathcal{N}(\mu_i)\|_{\boldsymbol{Q}}^2 &= \|\boldsymbol{q}_0^{\mu_i} - V_nV_n^*\boldsymbol{q}_0^{\mu_i}\|_{\boldsymbol{Q}}^2 + \|V_nV_n^*\boldsymbol{q}_0^{\mu_i} - V_n\mathcal{N}(\mu_i)\|_{\boldsymbol{Q}}^2\\ &= \|\boldsymbol{q}_0^{\mu_i} - V_nV_n^*\boldsymbol{q}_0^{\mu_i}\|_{\boldsymbol{Q}}^2 + \|V_n^*\boldsymbol{q}_0^{\mu_i} - \mathcal{N}(\mu_i)\|_{\mathbb{R}^n}^2.\end{aligned}$$

Since the first term in the last equality is independent of $\mathcal{N}$, the conclusion follows. □

We term this approach "Conservative POD-NN" because it forms a natural adaptation of the POD-NN approach introduced in [38]. The overall idea is illustrated in Figure 2 and summarized in Algorithm 1.

6.2 | Conservative DL-ROM: Curl-Variation

Our second proposal uses the construction in Section 4.3, thus assuming the presence of a suitable differential complex underlying our problem of interest. For simplicity, we directly focus on the case of the model problem (1.2) in three dimensions, in which $\boldsymbol{B}$ is the divergence on $\boldsymbol{Q} = \mathbb{R}\mathbb{T}_0(\Omega_h)$. We then let

$$S_0 := \nabla\times, \qquad \mathcal{N} : \boldsymbol{M} \to \mathbb{N}_0(\Omega_h).$$

In this case, the output of the neural-network model is the Nédélec finite-element space, which is high-dimensional. We therefore opt for the DL-ROM training strategy by splitting $\mathcal{N} = \Psi \circ \phi$ and introducing the auxiliary network Ψ'. A visual representation is given in Figure 3 and Algorithm 2 presents a synthetic summary. Compared to the construction from Section 6.1, this approach may be better suited when the solution manifold $\mathcal{S}_0$ exhibits a complicated structure that is not easily captured by linear subspaces. This situation is typically encountered if the PDE features a strong interaction between the space variable, x, and the model parameters, μ: see, e.g., the discussion in [11] and the examples reported therein.

With little abuse of notation, we shall also use the same terminology for the 2D-case, in which $\mathbb{N}_0(\Omega_h)$ is replaced by the nodal space $\mathbb{L}_1(\Omega_h)$, and $\nabla\times$ by $\nabla^\perp$, cf. the discussion in Section 4.3.

ALGORITHM 1 | Training and implementation of the Conservative POD-NN, Section 6.1. Here, `vstack` denotes the vertical stacking of multiple arrays.

Input : FOM solver $\texttt{FOM} = \texttt{FOM}(\mu)$, parameter space M, spanning-tree solver S_I, constraint operator B, source function `source`, basis dimension n, mass matrix M, sample size N_s, DNN architecture class $\mathcal{N}$.

Output: Trained Conservative POD-NN model.

$\mathsf{P},\ \mathsf{Q}^0 = [\,],[\,]$ **for** $i = 1, \dots, N_s$ **do**

  $\mu \leftarrow$ random sample from M

  $\boldsymbol{q} \leftarrow \texttt{FOM}(\mu)$

  $\boldsymbol{q}_0 \leftarrow (I - S_I B)\boldsymbol{q}$

  $\mathsf{P} \leftarrow \texttt{vstack}(\mathsf{P}, \mu)$

  $\mathsf{Q}^0 \leftarrow \texttt{vstack}(\mathsf{Q}^0, \boldsymbol{q}_0)$

end

$\mathsf{V} \leftarrow \texttt{POD}(\mathsf{Q}^0, \mathsf{M}, n)$

$\mathsf{C} \leftarrow \mathsf{Q}^0\mathsf{V}$

$\mathcal{N}_* \leftarrow \text{argmin}_{\mathcal{N}\in\mathcal{N}} \ \frac{1}{N_s}\sum_{i=1}^{N_s} \|\mathsf{C}_{i,\cdot} - \mathcal{N}(\mathsf{P}_{i,\cdot})\|^2$

def $\texttt{ROM} = \texttt{ROM}(\mu)$:

  $f \leftarrow \texttt{source}(\mu)$

  $\boldsymbol{q}_f \leftarrow S_I f$

  $\boldsymbol{q}_0 \leftarrow \mathsf{V}\mathcal{N}(\mu)$

  return $\boldsymbol{q}_0 + \boldsymbol{q}_f$

return *ROM*

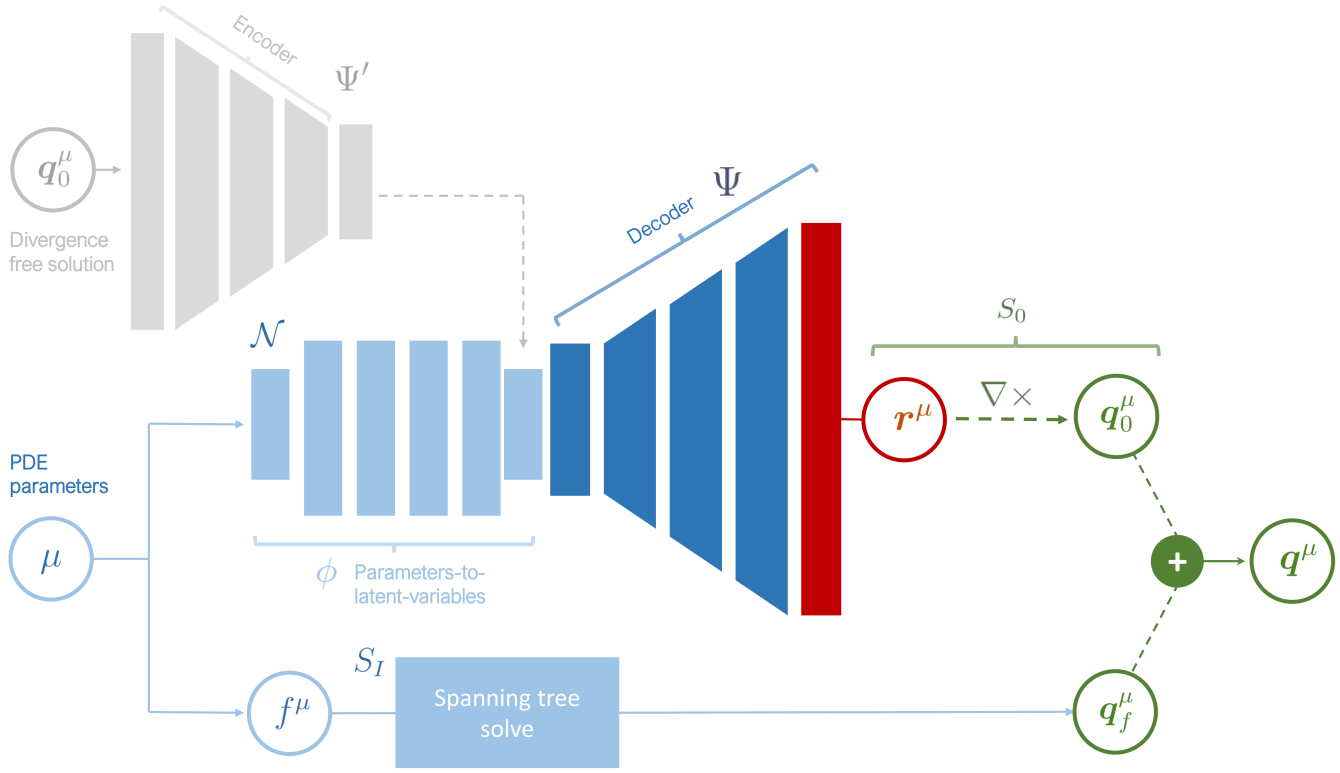

FIGURE 3 | Visual representation of the *Conservative DL-ROM* approach (curl-variation), Section 6.2. The potential network, $\mathcal{N}$ is comprised of two sequential blocks, so that $\mathcal{N} = \Psi \circ \phi$. An auxiliary encoder network (in gray) is used to introduce an internal regularization: the encoder is discarded after training. In red, the potential space representation.

ALGORITHM 2 | Training and implementation of the Conservative DL-ROMs, Sections 6.2–6.3.

Input : FOM solver `FOM` = `FOM`(μ), parameter space M, spanning-tree solver S_I, constraint operator B, source function `source`, mass matrix M, sample size N_s, type *type*, dictionary of decoder architectures $\mathscr{D} = \mathscr{D}[type]$, class of encoder architectures $\mathscr{E}$, class of latent networks $\mathscr{N}$, regularization hyperparameter λ, curl matrix Curl (optional)

Output: Trained Conservative DL-ROM model.

$\mathsf{P},\ \mathsf{Q}^0 = [\,],[\,]$

for $i = 1, \dots, N_s$ **do**

  $\mu \leftarrow$ random sample from M

  $\boldsymbol{q} \leftarrow$ `FOM`(μ)

  $\boldsymbol{q}_0 \leftarrow (I - S_I B)\boldsymbol{q}$

  $\mathsf{P} \leftarrow$ `vstack`(P, μ)

  $\mathsf{Q}^0 \leftarrow$ `vstack`($\mathsf{Q}^0, \boldsymbol{q}_0$)

end

if(*type* ==”curl”):

  $S_0 \leftarrow \mathsf{Curl}$

else:

  $S_0 \leftarrow I - S_I B$

$$\phi_*, \Psi_*, \Psi'_* \leftarrow \operatorname{argmin}_{\phi,\Psi,\Psi' \in \mathscr{N} \times \mathscr{D}[type] \times \mathscr{E}} \frac{1}{N_s} \sum_{i=1}^{N_s} [\mathsf{Q}^0_{i,\cdot} - S_0\Psi(\phi(\mathsf{P}_{i,\cdot}))]^T \mathsf{M} [\mathsf{Q}^0_{i,\cdot} - S_0\Psi(\phi(\mathsf{P}_{i,\cdot}))] + \lambda \cdot \frac{1}{N_s} \sum_{i=1}^{N_s} \|\Psi'(\mathsf{Q}^0_{i,\cdot}) - \phi(\mathsf{P}_{i,\cdot})\|^2$$

$\mathscr{N} \leftarrow \Psi_* \circ \phi_*$

def `DLROM` = `DLROM`(μ):

  $f \leftarrow$ `source`(μ)

  $\boldsymbol{q}_f \leftarrow S_I f$

  $\boldsymbol{q}_0 \leftarrow S_0 \mathscr{N}(\mu)$

  return $\boldsymbol{q}_0 + \boldsymbol{q}_f$

return *DLROM*

6.3 | Conservative DL-ROM: Spanning Tree (SpT) Variation

Our last proposal is to set $S_0 := I - S_I B$ and consequently $\mathscr{N} : M \to \boldsymbol{Q}$. As in the previous case, to handle the high-dimensionality at output, we implement this approach within the DL-ROM framework. Thus, the overall workflow can be represented exactly as in Figure 3, with the curl operator replaced by the redefined S_0 (see also Algorithm 2).

Compared to the Conservative POD-NN, this approach has the same advantages as its Curl counterpart. However, it also comes with increased flexibility, as it does not require the existence of an underlying differential complex. It can therefore be readily applied to a larger class of problems (as soon as a suitable right-inverse S_I is available). However, this additional flexibility comes at a cost, which we pay in terms of network complexity. In general, the output of the network, that is, the potential space R, is smaller in the Curl-variation than in the SpT-variation. This fact, whose nature is purely geometrical, is briefly summarized by the two lemmas below.

For 2-dimensional domains, the potential spaces for the curl and the spanning-tree variations are $\mathbb{L}_1(\Omega_h)$ and $\mathbb{RT}_0(\Omega_h)$, respectively. The degrees of freedom of the former correspond to mesh nodes, while those of the latter are represented by the mesh edges. The dimensions of these spaces are related as follows.

Lemma 4. *In a 2-dimensional setting, the simplicial tessellation Ω_h satisfies*

$$\text{nodes}(\Omega_h) = \text{edges}(\Omega_h) - \big[\text{elements}(\Omega_h) - 1\big]. \tag{6.2}$$

Proof. This is a reformulation of Euler's formula for planar graphs. □

The difference in output space dimension between the two approaches, therefore, becomes larger for finer meshes. A similar but weaker result holds in 3D. We recall that, in this case, the potential space of the curl-variation, $\mathbb{N}_0(\Omega_h)$, has the degrees of freedom defined over the mesh edges, while for the SpT-variation, the degrees of freedom are defined over the mesh faces.

Lemma 5. *In a* 3*-dimensional setting, the simplicial tessellation* Ω_h *satisfies*

$$\text{edges}(\Omega_h) \leq \text{faces}(\Omega_h) + 2. \tag{6.3}$$

Proof. If Ω_h consists of a single tetrahedron, then $\text{edges}(\Omega_h) = 6 = \text{faces}(\Omega_h) + 2$. Now, in order to address the general case, we note that any expansion of an existing simplicial grid by a new adjacent element (two elements are adjacent iff they share a common face) only increases the difference between the number of edges and faces. There are four possibilities to expand a simplicial grid, depending on how many faces the new element shares with the existing grid:

1. Adding a new element that shares one face with the grid introduces three new faces and three new edges, so the difference $\text{faces}(\Omega_h) - \text{edges}(\Omega_h)$ remains the same.
2. If the new element shares two faces with the grid, then two faces and one new edge need to be added. The difference increases by one.
3. If the new element shares three faces with the grid, then one only needs to add a single face. The difference increases by one.
4. The new element shares four faces with the grid, meaning that only the internal volume of the element is missing (no inclusion of extra faces or edges is required).

In short, the difference between number of faces and edges cannot decrease as the mesh is expanded, and the result follows. □

We emphasize that (6.3) only provides a weak bound, as one has $\text{edges}(\Omega_h) < \text{faces}(\Omega_h)$ in most practical cases.

7 | Numerical Results

We assess the performance of the proposed approaches for three different test cases, ranging from mixed-dimensional to nonlinear PDEs governing Darcy-flow systems. In the first test case, Section 7.1, we consider a simple 2D problem in which the model parameters present a strong interaction with the space variable. In Section 7.2, we explore a geometrically more involved scenario, concerning a Darcy flow in a fractured, three-dimensional medium. Finally, in Section 7.3, we consider a nonlinear model based on the Darcy-Forchheimer law. In order to present a systematic analysis for all the test cases, we proceed according to the following standardized protocol

1. We use the FOM to generate a collection of training data, $\{\mu_i, \boldsymbol{q}^{\mu_i}\}_{i=1}^{N_s}$, which we sample via Latin hypercube over the parameter space. In doing so, we implement the operator S_I using the multitree variation discussed in Remark 3.2, specifically using a total of 10 spanning trees. The parameter spaces of the three test cases are presented in Equations (7.4), (7.6) and (7.8), respectively.
2. We train the three ROMs as detailed in Sections 6.1-6.3.
3. We evaluate the quality of the flux approximations of the three ROMs by computing the relative error below

$$\mathscr{E}(\text{ROM}) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\|\boldsymbol{q}_i^{\text{test}} - \text{ROM}(\mu_i^{\text{test}})\|_X}{\|\boldsymbol{q}_i^{\text{test}}\|_X}, \tag{7.1}$$

where $\|\cdot\|_X$ is either the L^2-norm or the $H(\text{div})$-norm $\|\boldsymbol{q}\|_{H(\text{div})}^2 = \|\boldsymbol{q}\|_\Omega^2 + \|\nabla \cdot \boldsymbol{q}\|_\Omega^2$, to incorporate the satisfaction of mass conservation. Here, $\{\mu_i^{\text{test}}, \boldsymbol{q}_i^{\text{test}}\}_{i=1}^{N_{\text{test}}}$ are an additional collection of FOM data, the so-called test set, that we use for a posteriori evaluation.

4. We postprocess the flux approximations to evaluate the quality of the corresponding pressure fields. This is done on the same test set as in the previous step.

To better justify the effectiveness of the proposed strategies, we also compare the performances of our approaches with those of a naive NNR. That is, we build a benchmark ROM by directly training a neural-network model Φ to learn the map $\mu \to \boldsymbol{q}^{\mu}$ in a supervised way, which in practice corresponds to minimizing the loss function below

$$\mathscr{L}(\Phi) := \frac{1}{N_s}\sum_{i=1}^{N_s} \|\boldsymbol{q}^{\mu_i} - \Phi(\mu_i)\|_X^2, \tag{7.2}$$

where as before, $\|\cdot\|_X$ can be either the L^2 or the $H(\mathrm{div})$ norm: depending on this choice, we end up with two possible ROMs, which we refer to as "NNR L^2" and "NNR $H(\mathrm{div})$", respectively. Note that in both cases, the resulting ROM is not guaranteed to yield physical outputs in the sense that it may violate the linear constraint in the PDE model. In order to emphasize this fact, and to simultaneously verify the conservation properties of the proposed approaches, we introduce the following metrics,

$$\mathrm{ACV} := \frac{1}{N_{\mathrm{test}}}\sum_{i=1}^{N_{\mathrm{test}}} \|B\tilde{\boldsymbol{q}}_i^{\mathrm{test}} - f_{\mu_i^{\mathrm{test}}}\|_{(P,\|\cdot\|_\infty)'}, \qquad \mathrm{MCV} := \max_{i=1,\dots,N_{\mathrm{test}}} \|B\tilde{\boldsymbol{q}}_i^{\mathrm{test}} - f_{\mu_i^{\mathrm{test}}}\|_{(P,\|\cdot\|_\infty)'}, \tag{7.3}$$

where $\tilde{\boldsymbol{q}}_i^{\mathrm{test}} := \mathrm{ROM}(\mu_i^{\mathrm{test}})$. The above serve to quantify the extent to which the mass balance equation is violated across the test set. The acronyms ACV and MCV stand for *average constraint violation* and *maximum constraint violation*, respectively. As indicated in the formula, we measure such residuals using the dual norm

$$\|f\|_{(P,\|\cdot\|_\infty)'} := \sup_{\substack{p\in P \\ \|p\|_\infty=1}} |f(p)|.$$

Equivalently, if $\tilde{f} \in P$ is a Riesz representative of f, that is, $f(p) = \int_\Omega \tilde{f}p$ for all $p \in P$, then $\|f\|_{(P,\|\cdot\|_\infty)'} = \int_\Omega |\tilde{f}|$, which is the usual L^1 norm.

Technical details concerning the design of the neural-network architectures, both for the proposed approaches and the benchmark models, are reported in the Appendix A. There, the interested reader can also find additional results where we test other design choices (changes in the activation functions) and other training strategies (weighted loss for the $H(\mathrm{div})$ NNR surrogate): we anticipate that these changes can occasionally affect the performance of the models, but the conclusions remain the same. For this reason, when designing neural networks, we follow classical guidelines and refrain from using more sophisticated approaches such as adaptive activation functions [39, 40].

Finally, we mention that FOM simulations were computed in Python, using PorePy [41], PyGeoN [42] and dedicated modules that are publicly available. Conservative ROMs were implemented in Pytorch [43] using the dlroms package [44]. Demonstrative code and trained ROMs are available in our GitHub repository.[1]

7.1 | Dirichlet Problem With Sinusoidal Source

The first test case features problem (1.2) with $A_{\boldsymbol{q}} = I$. We let $\Omega = (0,1)^2$ be the domain of interest, which we discretized with a mesh size of $1/32$. This results in a computational grid that has 2400 cells, 3664 faces, and 1265 nodes. We set homogeneous boundary conditions for the pressure and a parametrized source function as

$$f(x_0, x_1) = \sin(\mu_0 2\pi x_0)\sin(\mu_1 2\pi x_1), \qquad \mu_0, \mu_1 \in [0,4]. \tag{7.4}$$

We moreover set a constant vector source term $\boldsymbol{g} \equiv [1.0, 0.0]^T$ in (1.1).

We use the FOM solver to sample a total of 2000 different PDE solutions, where $N_s = 1500$ are used for training and $N_{\mathrm{test}} = 500$ for testing. To keep the paper self-contained, we present all the technical details related to the structural hyperparameters of the neural-network architectures in Appendix A.

The results are in Figures 4–7 and Table 1. We note that, among the NNR approaches, the L^2-variant performs significantly better than its $H(\mathrm{div})$-counterpart. This is notable as it highlights that the inclusion of a divergence term in the loss

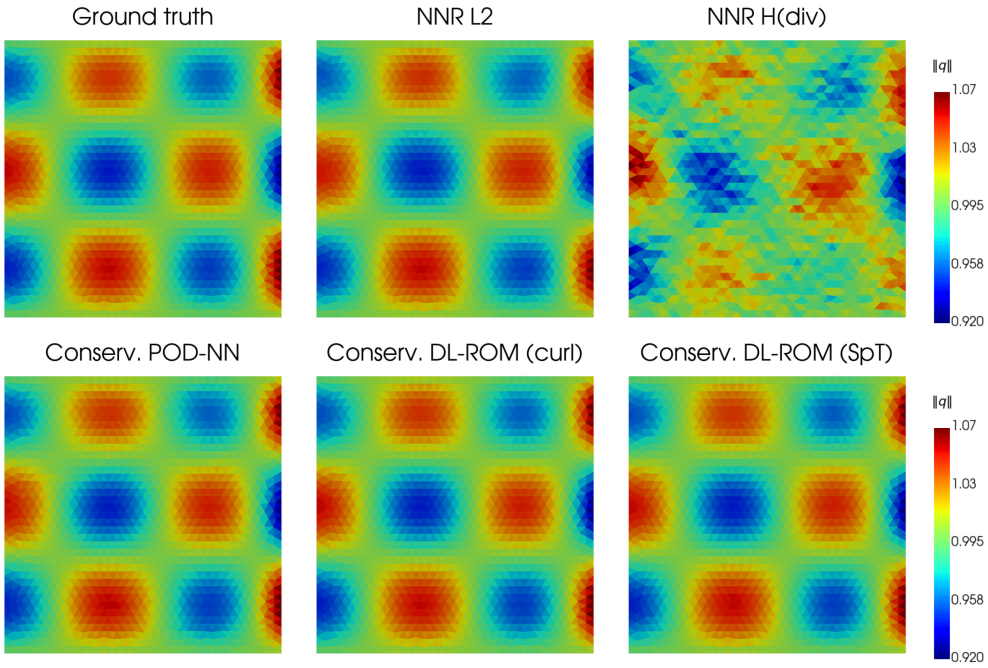

FIGURE 4 | Magnitude of the flow fields $|\boldsymbol{q}|$ for Case 1, Section 7.1. FOM solution (top left) vs ROM approximations for an unseen configuration of the problem parameters, $\mu = [1.332, 1.423]$. See Figure 6 for the corresponding pressure fields.

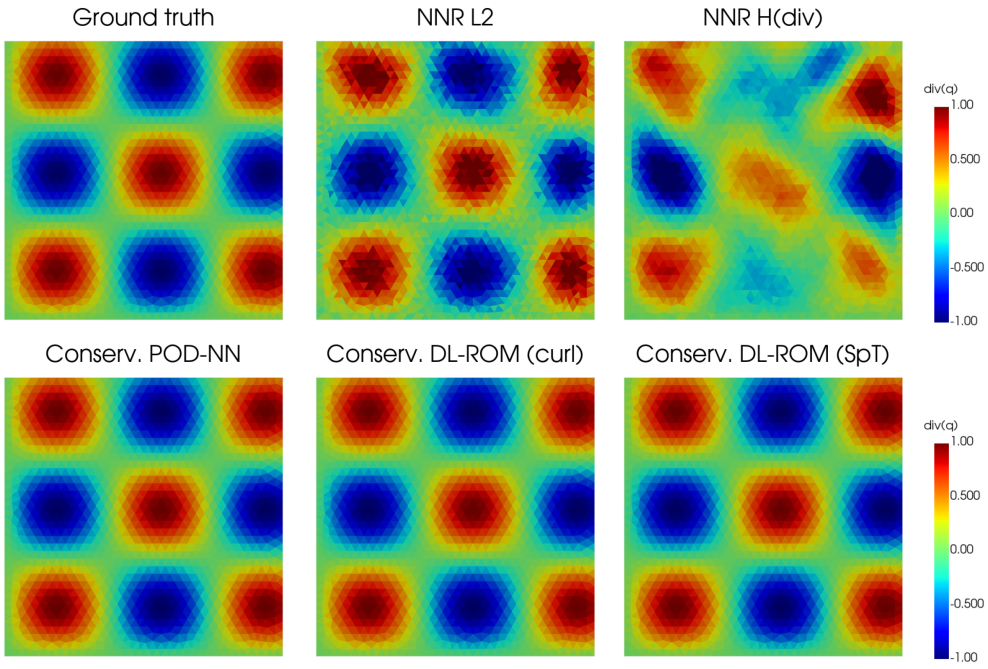

FIGURE 5 | Divergences of the flux fields $\nabla \cdot \boldsymbol{q}$ for Case 1, Section 7.1. FOM reference on the top left. ROM results are obtained by taking the divergence of the predicted flux fields. Results refer to an unseen configuration of the problem parameters (inference phase), as in Figures 4 and 6, $\mu = [1.332, 1.423]$.

function is not guaranteed to yield better results. Instead, the ROM performance may actually deteriorate. This behavior is likely caused by the intrinsic difficulty in minimizing the loss function, which becomes more and more challenging as we include extra terms in it.

Nevertheless, while the approximations proposed by the L^2-surrogate are graphically comparable to the reference one, both in terms of flux and pressure, they are not physical, as they violate the conservation of mass. This can be clearly seen

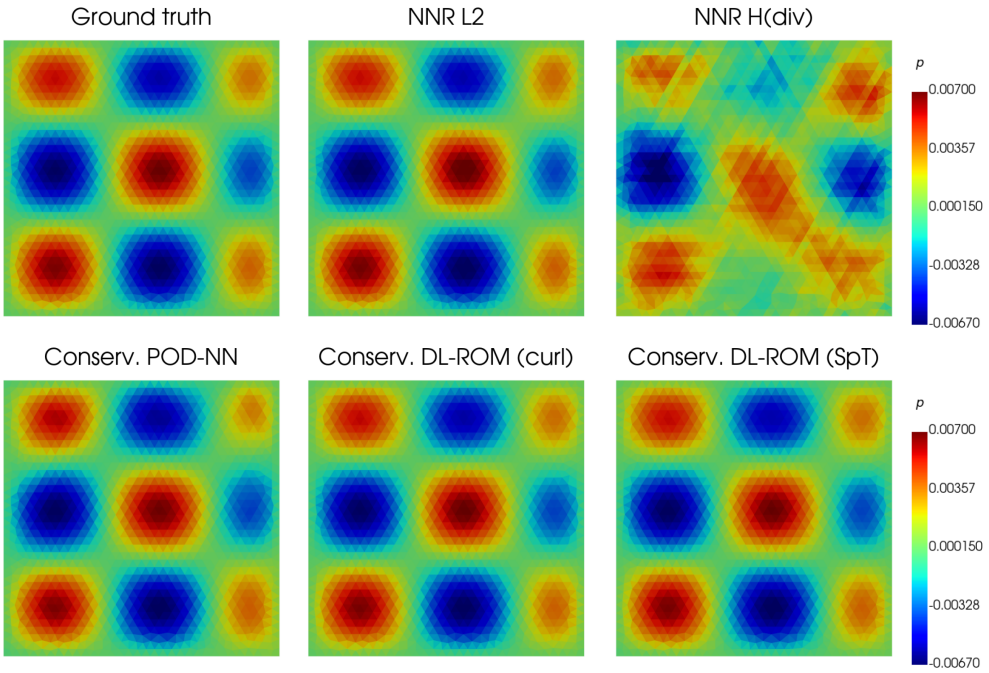

FIGURE 6 | Pressure fields p for Case 1, Section 7.1. FOM solution (top left) vs ROM approximations for an unseen configuration of the problem parameters, $\mu = [1.332, 1.423]$. See Figure 4 for the corresponding flow fields.

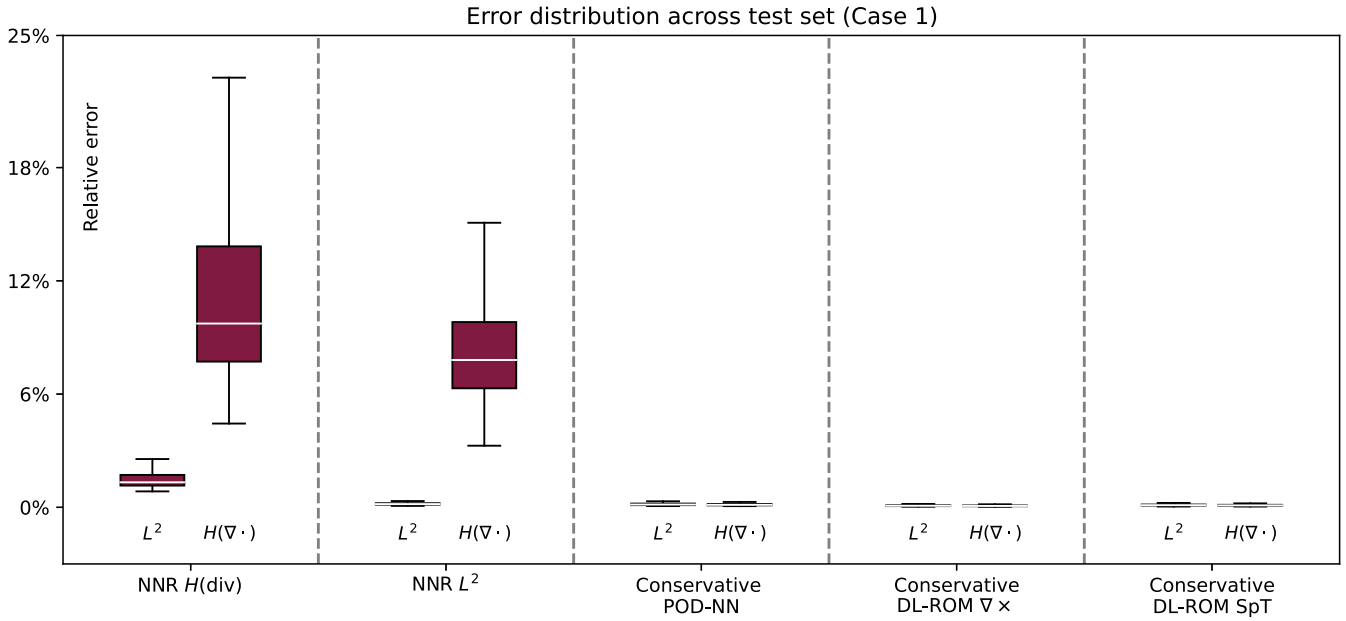

FIGURE 7 | Flux error distributions for the first case study, Section 7.1.

in Figure 5, where it becomes apparent that the conservative approaches are the only ones capturing the divergence of the flux field exactly, even when evaluated at inference, that is, when tested on unseen values of the problem parameters. Quantitatively speaking, this is made evident in Table 1: while naive regression surrogates tend to violate the mass balance equation, all conservative approaches satisfy the constraint close to machine precision, consistently across all test instances. This is especially true for Curl-DL-ROM (MCV $\sim 10^{-16}$) where the exactness of the differential complex guarantees minimal rounding errors. Conservative POD-NN and SpT DL-ROM, instead, present some fluctuations caused by small, but not entirely neglibigle, numerical errors (MCV $\sim 10^{-14}$). We recall that the implementation of Conservative POD-NN requires the solution to an eigenvalue problem; thus, the basis functions obtained via POD will only be homogeneous up to numerical errors. Similarly, SpT-DL-ROM requires the (offline) solution of a linear system and the storing of $(I - S_I B)$, which may generate small rounding errors.

TABLE 1 | Models comparison for the first case study, Section 7.1.

Model	Map onto kernel S_0	L^2 error Flow q	$H(\text{div})$ error Flow q	L^2 error Pressure p	Training time	ACV	MCV
NNR L^2	None	0.20%	9.00%	10.52%	4 m 50.0 s	3.9e-02	2.6e-01
NNR $H(\text{div})$	None	1.79%	13.35%	146.30%	6 m 13.4 s	4.3e-02	2.8e-01
Conserv. POD-NN	V_n	0.18%	0.16%	40.54%	2 m 50.1 s	3.9e-14	4.6e-14
Conserv. DL-ROM	$\nabla\times$	0.10%	0.09%	9.31%	4 m 52.7 s	1.2e-16	1.9e-16
Conserv. DL-ROM	$I - S_I B$	0.15%	0.14%	32.50%	8 m 5.02 s	4.0e-14	4.3e-14

Note: All the errors reported refer to the average performance over the test set. ACV and MCV quantify the average and maximum constraint violation, respectively: see Equation (7.3).

The inconsistency of naive nonlinear regression models can also be appreciated in Figure 7 and Table 1, where we see that the $H(\text{div})$ relative errors of such models are much higher than those in the L^2-norm. This indicates a violation of the linear constraint, as for any $\boldsymbol{q}, \tilde{\boldsymbol{q}}$ satisfying $\nabla \cdot \boldsymbol{q} = f = \nabla \cdot \tilde{\boldsymbol{q}}$ one would expect that

$$\frac{\|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{L^2}}{\|\boldsymbol{q}\|_{L^2}} \geq \frac{\|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{L^2}}{\|\boldsymbol{q}\|_{H(\text{div})}} = \frac{\|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{H(\text{div})}}{\|\boldsymbol{q}\|_{H(\text{div})}}, \tag{7.5}$$

That is, in relative terms, the errors in $H(\text{div})$ should be smaller than in L^2. Indeed, this is what we observe for all conservative approaches. Among these, the best one is undoubtedly the Curl-DL-ROM, which manages to: (i) capture the flux field, with a relative error of 0.1% and an exact conservation of mass, (ii) retrieve a good approximation of the pressure field, and (iii) keep the computational cost of the training phase under control. It is noteworthy that the spanning-tree variation of the DL-ROM approach achieves similar results, but at a higher computational cost. On the other hand, the Conservative POD-NN is undeniably the most cost-effective option for training, albeit with slightly inferior performance (see, e.g., the small artifacts and deformations in Figures 4–6). In this concern, we recall that, due to (7.4), the problem under study features a strong interaction between the space variable and the model parameters. It is therefore not surprising to see that, even with as many as 100 basis functions (cf. Appendix A), projection-based techniques do not rank as top performers. We also point out that, as clearly indicated in Figure 7, all the conservative approaches perform very well *overall*, that is, throughout the test set and not merely *on average*. In contrast, the performances of the NNR surrogates are much more volatile, especially in the $H(\text{div})$ norm.

Finally, it is interesting to note that, in line with Lemma 2, the ROMs reporting the smallest L^2 errors on the flux are also those providing a better approximation of the pressure fields. While the errors increase by two orders of magnitude, we consider these results satisfactory, as the pressure field was never explicitly involved during the training phase. We emphasize that the main purpose of this work is to provide a reliable approximation of the flux component. Further improvements are required if a higher quality of the pressure approximation is required, but that is beyond the scope of this work.

7.2 | Darcy Flow in a Fractured Porous Medium

As a second test case, we consider flow in a fractured porous medium in which five fractures are represented by planar two-dimensional inclusions in a three-dimensional medium. We provide a concise presentation of the model in Appendix B.

We set the conductivity of the bulk matrix to unity and set the aperture constant as $\epsilon_i = 10^{-4}$ for all subdomains Ω_i. Let us consider the following parametrization for the fracture permeability K_{frac}, the pressure boundary conditions, and source term f:

$$K_{frac} = 10^{\mu_0}, \qquad \mu_0 \in [3, 5], \tag{7.6a}$$

$$p|_{\partial\Omega}(\boldsymbol{x}) = [\mu_1, \mu_2, \mu_3]^\top \cdot \boldsymbol{x}, \qquad \mu_1, \mu_2, \mu_3 \in [0, 1], \tag{7.6b}$$

$$f(\boldsymbol{x}) = \begin{cases} \mu_4, & \boldsymbol{x} \in \Omega_{frac} \\ 0, & \boldsymbol{x} \in \Omega_{bulk} \end{cases} \qquad \mu_4 \in [1, 2]. \tag{7.6c}$$

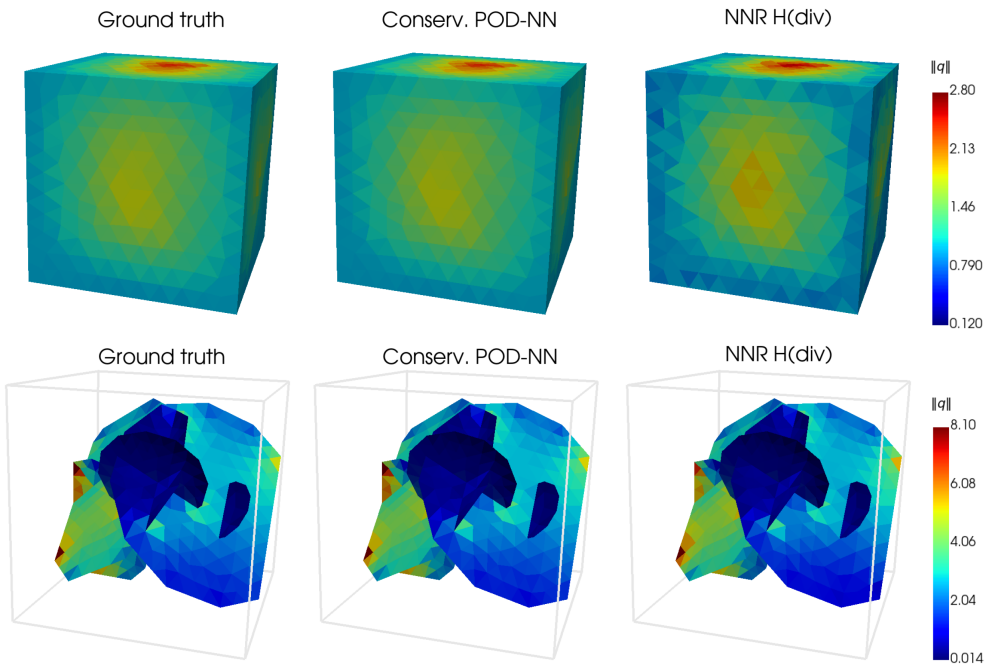

FIGURE 8 | Magnitude of the flow fields $|\boldsymbol{q}|$ for Case 2, Section 7.2. FOM solution (left) vs ROM approximations for an unseen configuration of the problem parameters, $\mu = [4.947, 0.406, 0.713, 0.552, 1.651]$. The top row depicts the bulk flux, whereas the bottom row concerns the fracture flux. The bulk color bar has been rescaled to match surface values. See Figure 9 for the corresponding pressure fields.

For the FOM solver, we rely on a mixed-dimensional mesh with mesh size $h = 0.1$, which results in a total of 17,249 degrees for the flux space. The FOM is used to generate a total of 1000 snapshots, $N_s = 800$ for training and $N_{\text{test}} = 200$ for testing.

We report in Figures 8 and 9 the solutions obtained with the FOM along with some of the reduced approaches. Since a proper visualization requires plotting both the bulk and the fractures, we have opted to show only the best and worst performing methods of the five ROMs.

As for the previous test case, we see that the approximations provided by the NNR surrogates are visibly unphysical. Furthermore, the results confirm that a forceful inclusion of the divergence term in the loss function is not beneficial, making the $H(\text{div})$ approach the worst performer. Instead, all conservative ROMs are capable of providing a good approximation of the flux $\boldsymbol{q}$, with L^2 and $H(\text{div})$ errors below 3% (Figure 10). Additionally, as testified by the physical consistency metrics, ACV and MCV, all conservative methods satisfy the mass balance equation up to a tolerance of 10^{-13}, with small deviations caused by the different scales characterizing the mixed-dimensional problem under study: see Table 2.

This time, among the three techniques, POD-NN excels, with average relative errors of 1.16% and 0.50% for the flux and pressure, respectively. Furthermore, it is the fastest ROM to train, as it only requires the optimization of a small DNN architecture (here, the potential space was $\dim(\boldsymbol{R}) = \dim(V_n) = n = 10$). Once again, this is not particularly surprising as the underlying PDE is mostly diffusive and the parameters do not directly interact with the space variable. Thus, the solution manifold associated with (7.6a) is expected to present a fast decay of its Kolmogorov n-width, meaning that it can be easily approximated by linear subspaces (such as the one given by the POD basis). This means that, while all conservative ROMs are satisfactory, prior knowledge of the problem can guide in the selection of the appropriate ROM.

7.3 | Darcy-Forchheimer Flow

As a final test case, we consider a Darcy-Forchheimer flow system on $\Omega := (0, 1)^2$, given by the following nonlinear system of equations:

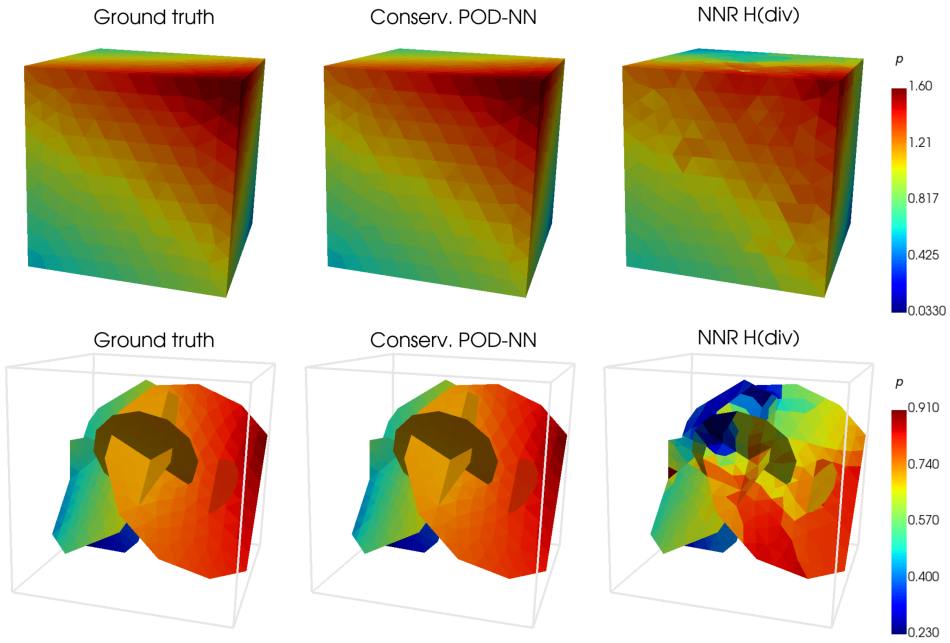

FIGURE 9 | Pressure fields p for Case 2, Section 7.2. FOM solution (left) vs ROM approximations for an unseen configuration of the problem parameters, $\mu = [4.947, 0.406, 0.713, 0.552, 1.651]$. Top: bulk pressure; bottom: fracture pressure. See Figure 8 for the corresponding flow fields.

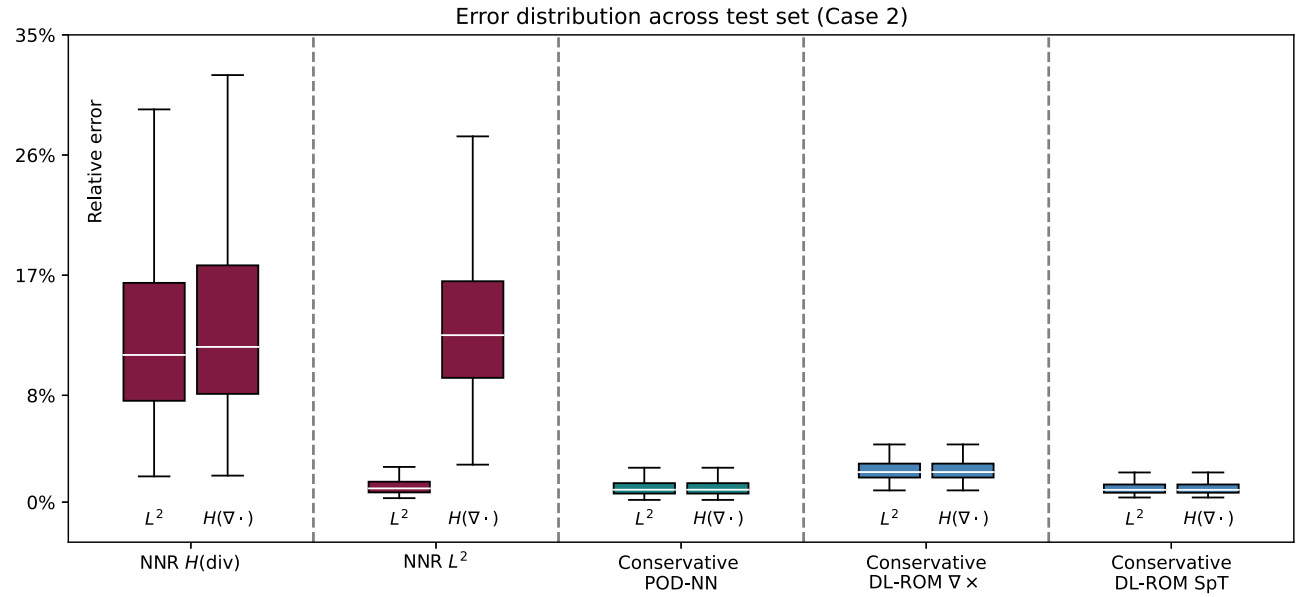

FIGURE 10 | Flux error distributions for the second case study, Section 7.2.

TABLE 2 | Models comparison for the second case study, Section 7.2.

Model	Map onto kernel S_0	L^2 error Flow q	H(div) error Flow q	L^2 error Pressure p	Training	ACV	MCV
NNR L^2	None	1.25%	14.50%	10.15%	24 m 24.5 s	1.5e-01	5.1e-01
NNR H(div)	None	13.10%	14.05%	317.81%	33 m 14.7 s	9.6e-02	3.0e-01
Conserv. POD-NN	V_n	1.16%	1.16%	0.50%	3 m 08.4 s	2.9e-14	1.2e-13
Conserv. DL-ROM	$\mathfrak{D}\times$	2.59%	2.59%	69.52%	31 m 25.2 s	2.1e-14	1.1e-13
Conserv. DL-ROM	$I - S_I B$	1.20%	1.20%	44.47%	48 m 16.9 s	5.4e-14	2.7e-13

Note: Table entries read as in Table 1. $\mathfrak{D}\times$ = generalized mixed-dimensional Curl, see Appendix B.

$$\kappa_0^{-1}(1+\kappa_1^{-1}|\boldsymbol{q}|)\boldsymbol{q}+\nabla p=0, \qquad \text{in } \Omega, \tag{7.7a}$$

$$\nabla\cdot\boldsymbol{q}=f, \qquad \text{in } \Omega, \tag{7.7b}$$

$$p=g, \qquad \text{on } \partial\Omega. \tag{7.7c}$$

We assume that the source terms, material parameters, and boundary conditions are parametrized as follows:

$$f(x_0,x_1)=\mu_0\sin(2\pi x_0)+(1-\mu_0)\sin(2\pi x_1), \qquad \mu_0\in[0,1], \tag{7.8a}$$

$$g(x_0,x_1)=\mu_1 x_0 x_1, \qquad \mu_1\in[0,1], \tag{7.8b}$$

$$\kappa_0=10^{\mu_2},\ \kappa_1=10^{\mu_3}, \qquad \mu_2\in[-2,1],\ \mu_3\in[0,2]. \tag{7.8c}$$

For the FOM solver, we rely on a structured triangular grid of size $h=1/32$. We generate a total of $N_s=800$ training snapshots and $N_{\text{test}}=200$ testing instances.

We remark that this problem constitutes a prototypical scenario in which our reduced-order modeling (ROM) approach would be of particular interest because (i) it features a fairly expensive FOM (nearly $\sim 20s$ to set up and run a single simulation, too costly for applications in which several different parameter values need to be investigated) and (ii) it concerns a nonlinear PDE, which poses significant challenges for intrusive ROMs based on linear projections, such as POD-Galerkin methods. As before, all the technical details about the design of ROMs and DNN architectures can be found in Appendix A.

The results are presented in Figures 11, 12 and Table 3. As for the previous test cases, the $H(\text{div})$-NNR model ranks as the worst performer, highlighting once again how "physics-informed" loss functions can be a double-edged sword. On the other hand, while the L^2-surrogate appears to be reasonably accurate, it is still physically inconsistent, as it returns flux fields that violate conservation of mass, cf. (7.5) and Table 3. This is also evident in Figure 12, where we clearly see that the $H(\text{div})$ errors are much higher and much more spread compared to the L^2 ones.

In contrast, all conservative ROMs show satisfactory results, both in terms of flux and pressure approximation, with Curl-DL-ROM reporting the best performance. This applies not only to the average expressivity (Table 1), but also to the overall quality of the approximation, as depicted in Figure 12. There, in fact, we see that the relative errors of conservative

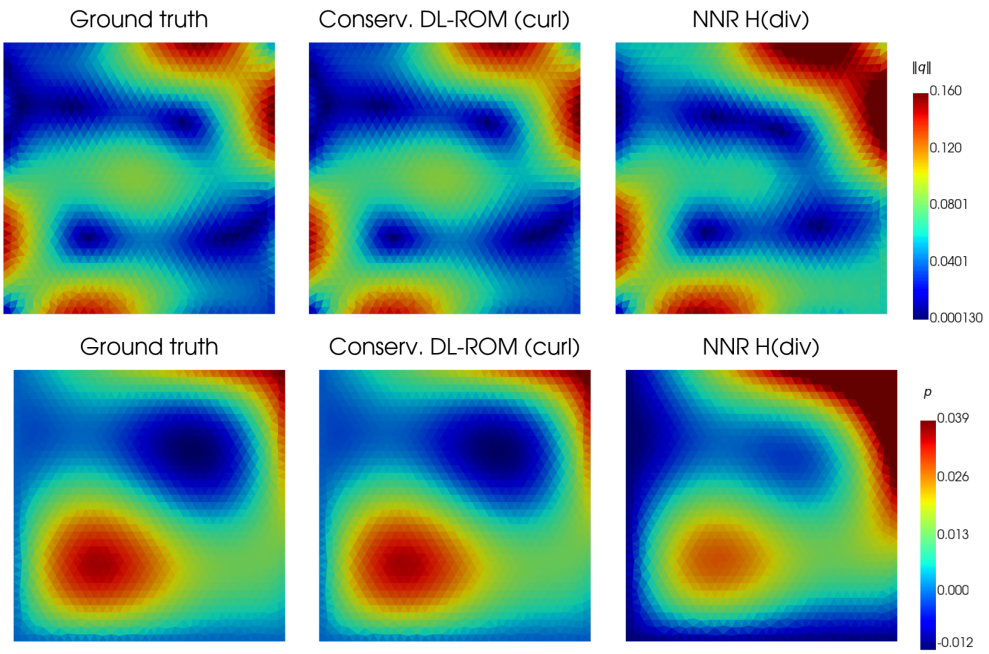

FIGURE 11 | Magnitude of the velocity fields $|\boldsymbol{q}|$ (top) and pressure fields p (bottom) for Case 3, Section 7.1. FOM solution vs ROM approximations for an unseen configuration of the problem parameters, $\mu=[0.388, 0.040, -0.193, 0.744]$.

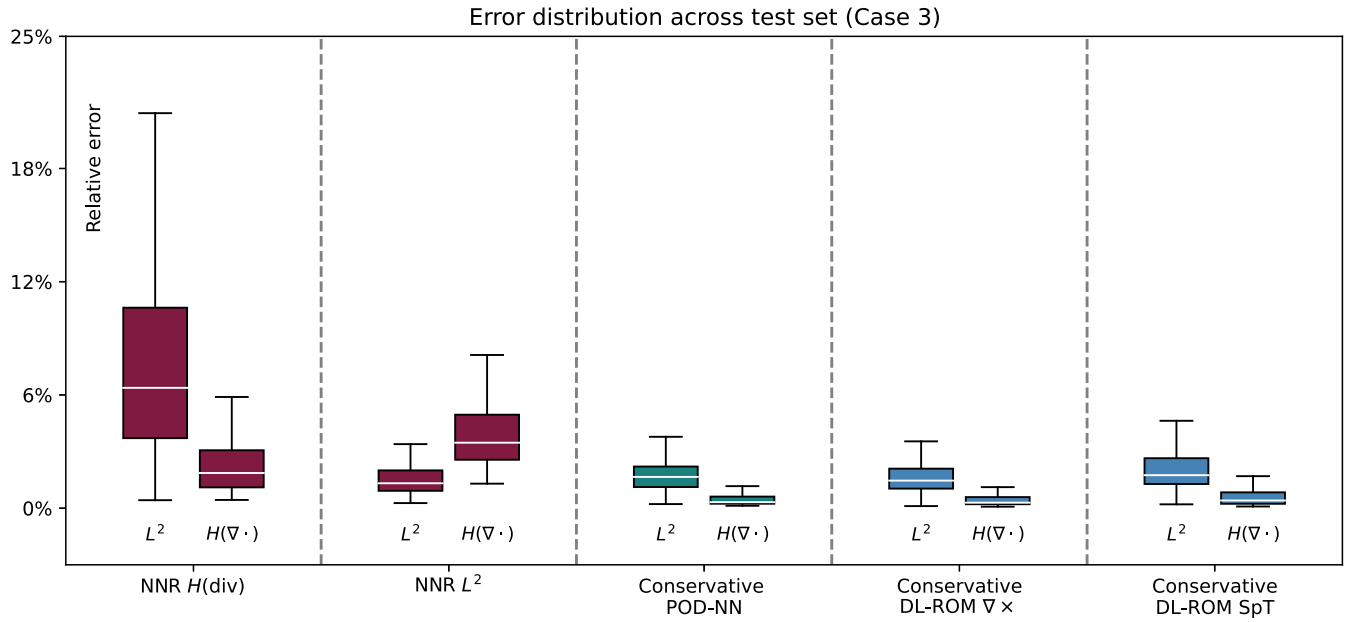

FIGURE 12 | Flux error distributions for the third case study, Section 7.3.

TABLE 3 | Models comparison for the third case study, Section 7.3.

Model	Map onto kernel S_0	L^2 error Flow q	H(div) error Flow q	L^2 error Pressure p	Training	ACV	MCV
NNR L^2	None	2.71%	4.04%	3.47%	3 m 52.5 s	2.6e-02	2.4e-01
NNR H(div)	None	13.43%	3.11%	20.18%	4 m 42.4 s	3.9e-03	4.1e-02
Conserv. POD-NN	V_n	2.76%	0.66%	4.03%	2 m 32.1 s	1.4e-14	1.2e-13
Conserv. DL-ROM	∇×	2.31%	0.63%	2.80%	4 m 19.7 s	1.5e-16	1.5e-15
Conserv. DL-ROM	$I - S_I B$	2.82%	0.87%	3.29%	7 m 14.1 s	3.0e-14	2.1e-13

Note: Table entries read as in Table 1.

ROMs are subject to fluctuations, but always fall below 6%. This is especially true for Conservative POD-NN and Conservative Curl-DL-ROM, whose performance is proven to be rather robust, as indicated by the width of the box plots. For instance, we see that more than 150 out of 200 test instances were approximated with L^2 errors below 3%.

As previously noted, despite its simplicity, this case study presents a situation in which the FOM can quickly become computationally prohibitive if a large number of simulations, obtained for varying values of $\mu_0, \dots, \mu_3$, is required (e.g., in applications involving uncertainty quantification and optimal control). Motivated by this fact, Table 4 shows the computational speed-up achieved by our best-performing ROM (Curl DL-ROM) when compared to the FOM. Wall-times refer to a single forward pass $\boldsymbol{\mu} \mapsto \boldsymbol{q}$. Notably, our proposed approach is $\sim$ 570 times faster than the reference model. Furthermore, this speed-up remains unmatched if we reduce the number of iterations in the nonlinear solver inside the FOM. In fact, while the finite-element model becomes faster, the drop in accuracy is far worse compared to the neural-network model. Although not reported, we mention that similar observations hold for Conservative POD-NN and SpT-DL-ROM.

8 | Concluding Remarks

We proposed an innovative deep learning strategy for the model order reduction of Darcy-flow systems. The methodology hinges on the idea of splitting the flux into homogeneous and particular solutions, employing a neural-network architecture and a kernel projector for the approximation of the former, and an efficient spanning-tree algorithm for the latter.

TABLE 4 | Computational speed-up of ROM vs FOM.

Model	Specifics	Computational cost	L^2 error
Conserv. DL-ROM	$S_0 = \nabla\times$	0.009 s	0.47%
FOM	25 iterations	5.195 s	0.00%
FOM	10 iterations	2.221 s	0.52%
FOM	5 iterations	1.092 s	11.64%

Note: Results refer to the third case study (nonlinear Darcy-Forchheimer flow), Section 7.3, and are limited to the best performing ROM, cf. Table 3. All models were evaluated at inference time—*online phase*—for an unseen value of the problem parameters, $\boldsymbol{\mu} = [0.32, 0.37, 0.95, 0.10]$. For benchmarking, FOM performances are reported for a varying number of fixed-point iterations. Errors are computed using the converged FOM (25 iterations) as a reference.

We proposed three model order reduction strategies, all adhering to linear constraints and based upon neural networks:

a. *Conservative POD-NN*: an approach based on a data-driven potential subspace;
b. *Curl DL-ROM*: a technique based on the theory of differential complexes;
c. *SpT DL-ROM*: a strategy that relies entirely on an efficient spanning-tree solve.

To substantiate the efficacy of the proposed strategies, we compared their performances with those attained by naive NNR. This comparative analysis was carried out across three different test cases: a 2D problem with a nontrivial dependency on the model parameters, a mixed-dimensional model of flow in fractured porous medium, and a nonlinear Darcy-Forchheimer flow system.

Our numerical experiments show that all conservative approaches can successfully incorporate the linear constraint, thus overcoming the mass conservation issues arising in NNR approaches. This is made possible by our construction with a potential space, and, as the experiments show, the same result cannot be easily achieved by simply including the linear constraint within the loss function. In fact, as testified by the poor performance of the $H(\mathrm{div})$ NNR surrogate, the inclusion of a divergence term does not invariably yield superior results. Instead, it may actually impair ROM performance. Notably, the same conclusions can be drawn even if one refines the training strategy of the $H(\mathrm{div})$ surrogate by balancing the divergence term, or if one changes the design of the neural-network models by modifying their activation function (cf. Appendix A).

We moreover observe that none of the three approaches, conservative POD-NN, Curl DL-ROM, and SpT DL-ROM, is unquestionably better than the others for all problems. The two DL-ROM techniques are better suited if the equation features either strong nonlinearities or space-varying parameters. Otherwise, conservative POD-NN can be a good fit, with high accuracy and reduced training times. It should be noted that although our construction was focused on providing a reliable approximation of the flux, the three approaches were still capable of retrieving a good approximation of the pressure fields as well. In general, the quality of this approximation was better for ROMs that were sufficiently L^2-accurate on the flux. In this context, although our findings are promising, there may still be room for improvement.

Our approach relies on the linearity of the constraint operator B and is therefore not directly applicable to more general problems with nonlinear or inequality constraints. However, those cases may be handled by setting up an iterative scheme in which our approach is applied to linearizations of the constraint operator. We reserve such extensions as a topic for future research.

Our proposal stands as a data-driven, but physical, nonintrusive alternative to other well-established model order reduction techniques, such as the Reduced Basis method. In particular, it can be of high interest whenever intrusive strategies are unavailable, possibly due to high computational costs or code inaccessibility, or when projection-based methods encounter substantial difficulties, for example, in the case of slow decay in the Kolmogorov n-width.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 101031434—MiDiROM. The present research is part of the activities of the project Dipartimento di Eccellenza 2023–2027, funded by MUR. The authors are members of the Gruppo Nazionale per il Calcolo Scientifico (GNCS)

of the Istituto Nazionale di Alta Matematica (INdAM). NF was supported by project DREAM (Reduced Order Modeling and Deep Learning for the real-time approximation of PDEs), grant no. FIS00003154, funded by MUR and by the Italian Science Fund (FIS). PZ acknowledges support from the MUR PRIN 2022 project 2022WKWZA8 *Immersed methods for multiscale and multiphysics problems* (IMMEDIATE), part of the Next Generation EU program Mission 4, Component 2, CUP D53D23006010006.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in GitHub at https://github.com/compgeo-mox/conservative_ml.

Endnotes

[1] https://github.com/compgeo-mox/conservative_ml.

References

1. S. Barbeiro and M. F. Wheeler, "A Priori Error Estimates for the Numerical Solution of a Coupled Geomechanics and Reservoir Flow Model With Stress-Dependent Permeability," *Computational Geosciences* 14 (2010): 755–768.

2. J. P. Eberhard, "Simulation of Lognormal Random Fields With Varying Resolution Scale and Local Average for Darcy Flow," *Computing and Visualization in Science* 9 (2006): 1–10.

3. F. Chinesta, A. Leygue, F. Bordeu, et al., "PGD-Based Computational Vademecum for Efficient Design, Optimization and Control," *Archives of Computational Methods in Engineering* 20, no. 1 (2013): 31–59.

4. P. Benner, S. Gugercin, and K. Willcox, "A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems," *SIAM Review* 57, no. 4 (2015): 483–531.

5. B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization," *SIAM Review* 60, no. 3 (2018): 550–591.

6. J. S. Hesthaven, C. Pagliantini, and G. Rozza, "Reduced Basis Methods for Time-Dependent Problems," *Acta Numerica* 31 (2022): 265–345.

7. A. Quarteroni, A. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations: An Introduction*, vol. 92 (Springer, 2015).

8. G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider, "A Theoretical Analysis of Deep Neural Networks and Parametric PDEs," *Constructive Approximation* 55, no. 1 (2022): 73–125.

9. S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific Machine Learning Through Physics–Informed Neural Networks: Where We Are and What's Next," *Journal of Scientific Computing* 92, no. 3 (2022): 88.

10. S. Fresca, L. Dede', and A. Manzoni, "A Comprehensive Deep Learning-Based Approach to Reduced Order Modeling of Nonlinear Time-Dependent Parametrized PDEs," *Journal of Scientific Computing* 87 (2021): 1–36.

11. N. Franco, A. Manzoni, and P. Zunino, "A Deep Learning Approach to Reduced Order Modelling of Parameter Dependent Partial Differential Equations," *Mathematics of Computation* 92, no. 340 (2023): 483–524.

12. X. Ying, "An Overview of Overfitting and Its Solutions," in *Journal of Physics: Conference Series*, vol. 1168 (IOP Publishing, 2019), 022022.

13. M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson, "Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data," in *Proceedings of the 37th International Conference on Machine Learning ICML'20* (JMLR.org, 2020).

14. K. Lee and K. T. Carlberg, "Model Reduction of Dynamical Systems on Nonlinear Manifolds Using Deep Convolutional Autoencoders," *Journal of Computational Physics* 404 (2020): 108973.

15. B. M. N. Smets, J. Portegies, E. J. Bekkers, and R. Duits, "PDE-Based Group Equivariant Convolutional Neural Networks," *Journal of Mathematical Imaging and Vision* 65, no. 1 (2023): 209–239.

16. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational Physics* 378 (2019): 686–707.

17. L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators," *Nature Machine Intelligence* 3, no. 3 (2021): 218–229.

18. N. Kovachki, Z. Li, B. Liu, et al., "Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs," *Journal of Machine Learning Research* 24 (2023): 1–97.

19. J. Fu, D. Xiao, R. Fu, et al., "Physics-Data Combined Machine Learning for Parametric Reduced-Order Modelling of Nonlinear Dynamical Systems in Small-Data Regimes," *Computer Methods in Applied Mechanics and Engineering* 404 (2023): 115771.

20. X. Pan and D. Xiao, "Domain Decomposition for Physics-Data Combined Neural Network Based Parametric Reduced Order Modelling," *Journal of Computational Physics* 519 (2024): 113452.

21. D. Hansen, D. M. Robinson, S. Alizadeh, G. Gupta, and M. Mahoney, "Learning Physical Models That Can Respect Conservation Laws," in *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)* (2023), 12469–12510, https://doi.org/10.1016/j.physd.2023.133952.

22. A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative Physics-Informed Neural Networks on Discrete Domains for Conservation Laws: Applications to Forward and Inverse Problems," *Computer Methods in Applied Mechanics and Engineering* 365 (2020): 113028.

23. Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-Informed Neural Networks for High-Speed Flows," *Computer Methods in Applied Mechanics and Engineering* 360 (2020): 112789.

24. T. Boesen, E. Haber, and U. M. Ascher, "Neural DAEs: Constrained Neural Networks," *SIAM Journal on Scientific Computing* 47, no. 2 (2025): C291–C312.

25. L. Ruthotto and E. Haber, "Deep Neural Networks Motivated by Partial Differential Equations," *Journal of Mathematical Imaging and Vision* 62 (2020): 352–364.

26. N. Trask, A. Huang, and X. Hu, "Enforcing Exact Physics in Scientific Machine Learning: A Data-Driven Exterior Calculus on Graphs," *Journal of Computational Physics* 456 (2022): 110969.

27. T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine, "Enforcing Analytic Constraints in Neural Networks Emulating Physical Systems," *Physical Review Letters* 126, no. 9 (2021): 098302.

28. S. Rave and F. Schindler, "A Locally Conservative Reduced Flux Reconstruction for Elliptic Problems," *PAMM* 19, no. 1 (2019): e201900026.

29. P. A. Raviart and J. M. Thomas, "A Mixed Finite Element Method for 2nd Order Elliptic Problems," in *Mathematical Aspects of Finite Element Methods*, vol. 606, ed. I. Galligani and E. Magenes (Springer, 1977), 292–315.

30. W. M. Boon and A. Fumagalli, "A Reduced Basis Method for Darcy Flow Systems That Ensures Local Mass Conservation by Using Exact Discrete Complexes," *Journal of Scientific Computing* 94, no. 3 (2023): 64.

31. W. M. Boon, N. R. Franco, and A. Fumagalli, "Neural Network Solvers for Parametrized Elasticity Problems That Conserve Linear and Angular Momentum," *Computer Methods in Applied Mechanics and Engineering* 437 (2025): 117759.

32. P. Jiránek, Z. Strakoš, and M. Vohralík, "A Posteriori Error Estimates Including Algebraic Error and Stopping Criteria for Iterative Solvers," *SIAM Journal on Scientific Computing* 32, no. 3 (2010): 1567–1590.

33. W. M. Boon, "Solvers for Mixed Finite Element Problems Using Poincaré Operators Based on Spanning Trees," 2024 arXiv:2410.08830.

34. D. N. Arnold, R. S. Falk, and R. Winther, "Finite Element Exterior Calculus, Homological Techniques, and Applications," *Acta Numerica* 15 (2006): 1–155.

35. D. N. Arnold, *Finite Element Exterior Calculus* (SIAM, 2018).

36. J. C. Nédélec, "Mixed Finite Elements in R3," *Numerische Mathematik* 35 (1980): 315–341.

37. N. R. Franco, D. Fraulin, A. Manzoni, and P. Zunino, "On the Latent Dimension of Deep Autoencoders for Reduced Order Modeling of PDEs Parametrized by Random Fields," *Advances in Computational Mathematics* 50, no. 5 (2024): 96.

38. J. S. Hesthaven and S. Ubbiali, "Non-Intrusive Reduced Order Modeling of Nonlinear Problems Using Neural Networks," *Journal of Computational Physics* 363 (2018): 55–78.

39. A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive Activation Functions Accelerate Convergence in Deep and Physics-Informed Neural Networks," *Journal of Computational Physics* 404 (2020): 109136.

40. A. D. Jagtap, Y. Shin, K. Kawaguchi, and G. E. Karniadakis, "Deep Kronecker Neural Networks: A General Framework for Neural Networks With Adaptive Activation Functions," *Neurocomputing* 468 (2022): 165–180.

41. E. Keilegavlen, R. Berge, A. Fumagalli, et al., "PorePy: An Open-Source Software for Simulation of Multiphysics Processes in Fractured Porous Media," *Computational Geosciences* 25 (2020): 243–265.

42. W. M. Boon and A. Fumagalli, "PyGeoN: A Python Package for Geo-Numerics; v. 0.3,"

43. A. Paszke, S. Gross, F. Massa, et al., "Pytorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32 (Curran Associates, Inc., 2019).

44. N. R. Franco and d. NicolaRFranco, "First Release (v1.0.0)," (2024) Zenodo, https://doi.org/10.5281/zenodo.13254758.

45. H. N. Mhaskar, "Neural Networks for Optimal Approximation of Smooth and Analytic Functions," *Neural Computation* 8, no. 1 (1996): 164–177.

46. S. Mishra and T. K. Rusch, "Enhancing Accuracy of Deep Learning Algorithms by Training With Low-Discrepancy Sequences," *SIAM Journal on Numerical Analysis* 59, no. 3 (2021): 1811–1834.

47. I. G. Farcaş, B. Peherstorfer, T. Neckel, F. Jenko, and H. J. Bungartz, "Context-Aware Learning of Hierarchies of Low-Fidelity Models for Multi-Fidelity Uncertainty Quantification," *Computer Methods in Applied Mechanics and Engineering* 406 (2023): 115908.

48. P. Vitullo, N. R. Franco, and P. Zunino, "Deep Learning Enhanced Cost-Aware Multi-Fidelity Uncertainty Quantification of a Computational Model for Radiotherapy," *Foundations of Data Science* 7, no. 1 (2025): 386–417.

49. I. Berre, W. M. Boon, B. Flemisch, et al., "Verification Benchmarks for Single-Phase Flow in Three-Dimensional Fractured Porous Media," *Advances in Water Resources* 147 (2021): 103759.

50. W. M. Boon, J. M. Nordbotten, and I. Yotov, "Robust Discretization of Flow in Fractured Porous Media," *SIAM Journal on Numerical Analysis* 56, no. 4 (2018): 2203–2233.

51. W. M. Boon, J. M. Nordbotten, and J. E. Vatne, "Functional Analysis and Exterior Calculus on Mixed-Dimensional Geometries," *Annali di Matematica Pura ed Applicata (1923 -)* 200, no. 2 (2021): 757–789.

52. A. Budisa, W. M. Boon, and X. Hu, "Mixed-Dimensional Auxiliary Space Preconditioners," *SIAM Journal on Scientific Computing* 42, no. 5 (2020): A3367–A3396.

Appendix A

Neural Network Architectures

In this Section, we report all the technical details about the neural network architectures employed across the three experiments. We mention that all DNN architectures were constructed using dense layers, primarily relying on the 0.1-leakyReLU activation. All networks were trained using the L-BFGS optimizer with standard parameter values (learning rate = 1) and for a total of 500 epochs.

We recall that a dense layer with input dimension m_1, output dimension m_2 and activation $\rho : \mathbb{R} \to \mathbb{R}$ is a nonlinear map from $\mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ of the form

$$\mathbf{v} \mapsto \rho(\mathbf{W}\mathbf{v} + \mathbf{b})$$

where $\mathbf{W} \in \mathbb{R}^{m_2 \times m_1}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^{m_2}$ the bias vector and, with little abuse of notation,

$$\rho([w_1, \dots, w_m]^T) := [\rho(w_1), \dots, \rho(w_m)]^T.$$

Then, classical deep feed-forward neural networks (such as those employed here) are obtained via sequential composition of multiple layers.

N.B.: in what follows, ρ is always assumed to be the 0.1-leakyReLU activation, namely,

$$\rho : x \mapsto 0.1x\hat{1}_{(-\infty,0)}(x) + x\hat{1}_{[0,+\infty)}(x).$$

Architectures for Case 1

For this test case, see Tables A1–A3, all architectures employ a *feature layer*, whose purpose is to provide a preliminary preprocessing of the input parameters. The latter consists of a non-learnable map $F : \mathbb{R}^2 \to \mathbb{R}^{225}$ acting as

$$[\mu_0, \mu_1] \mapsto \left[\sin\left(2\pi\mu_0 \frac{\mathrm{mod}(j, 15)}{14}\right) \sin\left(2\pi\mu_1 \frac{j - \mathrm{mod}(j, 15)}{14 \cdot 15}\right)\right]_{j=1}^{225}.$$

That is, F maps the two parameters $[\mu_0, \mu_1]$ onto a discrete representation of the source term $f^{\mu_0,\mu_1}(x_0, x_1) = \sin(\mu_0 2\pi x_0)\sin(\mu_1 2\pi x_1)$, obtained by evaluating the latter over a uniform 15×15 grid. As the PDE solution only depends on $[\mu_0, \mu_1]$ through f^{μ_0,μ_1}, this transformation can be seen as a natural preprocessing of the input data (see Tables A1–A12).

Architectures for Case 2

We report the architectures for the 2nd test case in Tables A4–A6. We recall that the problem features 5 scalar parameters, while the FOM dimension (for the flux field) is 17249.

Architectures for Case 3

We report the architectures for the last test case (nonlinear flow) in Tables A7–A9. We recall that the problem features 4 parameters, while the FOM dimension (for the flux) is 3664.

TABLE A1 | Potential network for the Conservative POD-NN approach, §7.1.

Layer	Input	Output	Activation
Feature extractor	2	225	—
Dense	225	100	ρ
Dense	100	100	ρ
Dense	100	100	—

Note: The output dimension coincides with the POD space dimension, here $n = 100$. The first layer is not learnable and coincides with the feature map F.

TABLE A2 | Architectures of the Conservative DL-ROM approach §7.1 (both versions).

Potential network $\mathcal{N} = \Psi \circ \phi$				**Encoder Ψ'**			
Layer	**Input**	**Output**	**Activ.**	**Layer**	**Input**	**Output**	**Activ.**
Feature extractor	2	225	—	Dense	3664	100	ρ
Dense	225	100	ρ	Dense	100	100	ρ
Dense	100	200	ρ				
Dense	200	dim($\boldsymbol{R}$)	—				

Note: Potential network on the left (above dashed line = ϕ, below dashed line = Ψ), auxiliary encoder module on the right. Here, dim($\boldsymbol{R}$) equals either 1265 or 3664, depending on the ROM variation, that is, Curl and SpT, respectively.

TABLE A3 | Architecture for the neural network regression surrogates, §7.1.

Layer	Input	Output	Activation
Feature extractor	2	225	—
Dense	225	100	ρ
Dense	100	200	ρ
Dense	200	500	ρ
Dense	500	3664	—

Supplementary Experiments and Results

We report the results of some additional experiments that serve as Supporting Information for Section 7.

Smoothness of the Activation Function

We rerun all the test cases by replacing the 0.1-leakyReLU activation with the GeLU activation in all neural network architectures, (both for the proposed approaches as well as the benchmark models),

$$\rho_{\text{GeLU}}(x) := x\Phi(x),$$

where Φ is the cumulative distribution function of a standard normal distribution in 1d. This was done to assess whether the different methodologies could benefit from the use of smoother activations, as the deep learning theory suggests; see, for example, [45, 46]. Results are in Table A10.

In general, we see that using a smoother activation function appears to be beneficial for all of the approaches considered, as essentially all of the errors are slightly smaller than the ones reported in Section 7. However, the conclusion is identical: the NNR benchmarks perform nicely in the L^2 sense but, differently from the conservative approaches, they struggle in the $H(\text{div})$-norm, indicating a violation of the conservation constraint.

Hyperparameter Tuning for Nonlinear Regression Models Using the $H(\text{div})$ Loss

We considered alternative implementations of the $H(\text{div})$-NNR benchmark, obtained by training the model with a weighted loss function based on the norm

$$\|\boldsymbol{q}\|_\lambda := \|\boldsymbol{q}\|_{L^2(\Omega)} + \lambda\|\nabla \cdot \boldsymbol{q}\|_{L^2(\Omega)}.$$

Specifically, we repeated this process for different λ and selected the best model a posteriori. This is to verify that, even in this way, the $H(\text{div})$ NNR surrogate is not competitive when compared to our proposed approaches. In light of the results discussed in Appendix Section A, this analysis was conducted directly using the GeLU activation. The results can be found in Table A11.

TABLE A4 | Conservative POD-NN, potential network for §7.2.

Layer	Input	Output	Activation
Dense	5	200	ρ
Dense	200	100	ρ
Dense	100	10	—

Note: Output dimension = POD space dimension, here $n = 10$.

TABLE A5 | Architectures of the Conservative DL-ROM approach for §7.2 (both versions).

Potential network $\mathcal{N} = \Psi \circ \phi$				**Encoder Ψ'**			
Layer	**Input**	**Output**	**Activ.**	**Layer**	**Input**	**Output**	**Activ.**
Dense	5	200	ρ	Dense	17249	10	ρ
Dense	200	10	ρ				
Dense	10	500	ρ				
Dense	500	dim($\boldsymbol{R}$)	—				

Note: Potential network on the left (above dashed line = ϕ, below dashed line = Ψ), auxiliary encoder module on the right. Here, dim($\boldsymbol{R}$) equals either 11,182 or 17,249, depending on the ROM variation, that is, Curl and SpT, respectively.

TABLE A6 | Architecture for the neural network regression surrogates used in §7.2.

Layer	Input	Output	Activation
Dense	5	200	ρ
Dense	200	10	ρ
Dense	100	500	ρ
Dense	500	17249	—

In general, we note that the $H(\text{div})$-NNR surrogate typically performs worse than its L^2 counterpart (obtained for $\lambda = 0$), except for some special cases. Furthermore, the optimal choice for the λ coefficient is clearly problem dependent, which further hinders the usefulness of this approach. On top of this, even with this additional optimization, the $H(\text{div})$-NNR surrogate is not able to match the accuracy of the proposed conservative models.

Training Time Vs Mesh Resolution

In order to assess the scalability of the proposed approaches, we repeated the analysis presented in Section 3 for varying resolutions of the FOM. Precisely, we considered two alternative discretizations of the model problem (but the same number of training data), using a coarser $h_{\text{coarse}} = 1/16$ and a finer $h_{\text{fine}} = 1/64$ mesh size, respectively. Coherent with the other analyses reported in this section, the experiment was carried out using the GeLU activation for all neural network architectures. The results, which are limited to the three proposed approaches, are presented in Table A12. Note that the results for $h = 1/32$ are precisely the ones in Table A10.

Unsurprisingly, the performance and the efficiency of Conservative POD-NN are completely unaffected by the resolution of the FOM. In the latter approach, the size of the neural network architecture depends only on the number of POD coefficients retained during truncation, and is therefore independent of h. This independence becomes especially evident in the limit $h \downarrow 0^+$, where the numerical solutions converge to the true PDE solutions and the POD expansion approaches a Karhunen–Loève decomposition. Specifically, in the case of Conservative POD-NN, the potential space dimension is fixed at $n = 4$, which allows the architecture to be trained efficiently regardless of h. Interestingly, training even converged faster for $h = 1/64$ than for $h = 1/16$.

Curl-DL-ROM also scaled well in terms of training times and overall accuracy, despite a small but consistent increase in the errors. This is likely caused by the fact that, in this case, the dimension of the potential space $\boldsymbol{R}$ is increasing as $h \downarrow 0^+$. This can result in larger neural network architectures that require more training data in order to generalize properly. This effect becomes even more evident when considering SpT-DL-ROM. There, the potential space becomes too large and the model is not able to scale properly. For this reason, while Conservative POD-NN is guaranteed to scale even beyond our experiments, the same conclusion cannot be drawn for the other two approaches.

As a final note, we point out that Table A12 does not include the offline cost required to generate the training data, which clearly increases with the resolution of the FOM. This is because the latter was the same for all approaches, and is thus not of interest for the present analysis. Furthermore, in this work, we have made the assumption that a ROM is needed, focusing on the development of

TABLE A7 | Potential network for the Conservative POD-NN approach, §7.3.

Layer	Input	Output	Activation
Dense	4	50	ρ
Dense	50	50	ρ
Dense	50	100	ρ
Dense	100	4	—

TABLE A8 | Architectures of the Conservative DL-ROM approach, §7.3, (both versions).

Potential network $\mathcal{N} = \Psi \circ \phi$				**Encoder Ψ'**			
Layer	**Input**	**Output**	**Activ.**	**Layer**	**Input**	**Output**	**Activ.**
Dense	4	50	ρ	Dense	3664	10	ρ
Dense	50	50	ρ				
Dense	50	4	ρ				
Dense	4	50	ρ				
Dense	50	dim($\boldsymbol{R}$)	—				

Note: Potential network on the left (above dashed line = ϕ, below dashed line = Ψ), encoder module on the right. Here, dim($\boldsymbol{R}$) = 11, 182, 17, 249 for the Curl and SpT variations, respectively.

TABLE A9 | Architecture for the neural network regression surrogates, §7.3.

Layer	Input	Output	Activation
Dense	4	50	ρ
Dense	50	50	ρ
Dense	50	4	ρ
Dense	4	50	ρ
Dense	50	3644	—

physically consistent surrogates rather than discussing optimal ways for managing a given computational budget. For practical insights on the latter topic, we refer the interested reader to [47, 48], which provides precise guidelines on how to balance data generation costs, training costs, and FOM evaluations in many-query applications concerning uncertainty quantification.

Appendix B

Mixed-Dimensional Model for Flow in Fractured Porous Media

We give a concise exposition of the fracture flow model used in Section 7.2. The interested reader is referred to [49] for more details concerning the model, [50] for the mixed finite element method, and [51] for the admissible geometries and mixed-dimensional function spaces.

Let us decompose the domain into subdomains Ω_i with d_i its dimension and i from the index set $\mathcal{I}$. Fractures will thus be given by subdomains with $d_i = 2$, fracture intersections are one-dimensional with $d_i = 1$, and the subdomains with $d_i = 0$ represent the intersection points. For given Ω_i, let $\partial_j\Omega$ with $d_j = d_i - 1$ be the part of its boundary that coincides geometrically with a lower-dimensional neighbor Ω_j. Let $\boldsymbol{\nu}_i$ on $\partial_j\Omega$ be the unit normal vector, oriented outward with respect to Ω_i.

We continue with the material parameters. Let ε_i be a given length constant that represents the fracture aperture if $d_i = 2$. Moreover, ε_i^2 represents the cross-sectional area of intersection lines with $d_i = 1$, and ε_i^3 represents the volume of intersection points for $d_i = 0$. Next, for given fracture conductivity K_{frac}, we define the effective conductivities:

$$K_{\parallel,i} := \varepsilon_i^{3-d_i} K_{frac} \text{ on } \Omega_i, \qquad K_{\perp,ij} := \varepsilon_i^{3-d_i} \frac{2}{\varepsilon_j} K_{frac} \text{ on } \partial_j\Omega_i.$$

For ease of notation, we collect the flux and pressure variables into mixed-dimensional variables, which we denote using a Gothic font:

$$\mathfrak{p} := \bigoplus_{\substack{i\in\mathcal{I}\\ 0\le d_i\le 3}} p_i, \qquad \mathfrak{q} := \bigoplus_{\substack{i\in\mathcal{I}\\ 1\le d_i\le 3}} q_i.$$

TABLE A10 | Experimental results for the three case studies when using the GeLU activation.

	$\|\|\cdot\|\|$	NNR L^2	NNR $H(\text{div})$	Conserv. POD-NN	Curl DL-ROM	SpT DL-ROM
Case 1	L^2	0.07%	0.12%	0.10%	0.10%	0.21%
	$H(\text{div})$	4.78%	5.22%	0.09%	0.09%	0.19%
Case 2	L^2	0.27%	0.32%	0.38%	1.51%	0.57%
	$H(\text{div})$	6.33%	3.75%	0.38%	1.51%	0.57%
Case 3	L^2	0.72%	1.23%	0.93%	1.52%	0.78%
	$H(\text{div})$	3.60%	0.66%	0.19%	0.32%	0.16%

Note: Errors refer to the approximation of the flux field $\boldsymbol{q}$ and are computed as norm-relative averages over the test set (cf. Section 7). Here, Conserv. POD-NN is Conservative POD-NN.

TABLE A11 | Numerical results for the $H(\text{div})$-NNR benchmark model when changing the weight of the divergence term, λ, in the loss function.

	$\|\|\cdot\|\|$	$\lambda = 0$	λ =1e-4	λ =1e-3	λ =1e-2	λ =1e-1	$\lambda = 1$
Case 1	L^2	0.07%	0.12%	0.45%	2.73%	2.72%	2.67%
	$H(\text{div})$	4.78%	5.22%	12.65%	42.61%	42.15%	41.12%
Case 2	L^2	0.27%	0.24%	0.32%	1.01%	1.64%	18.67%
	$H(\text{div})$	6.33%	4.74%	3.75%	5.26%	3.11%	19.59%
Case 3	L^2	0.72%	0.76%	0.68%	1.23%	4.46%	356.22%
	$H(\text{div})$	3.60%	1.68%	0.99%	0.66%	2.05%	77.52%

Note: As in Table A10, error metrics are computed over the test set by taking the average of the norm-relative error for the flux field in the L^2 and $H(\text{div})$ norms, respectively. NB: the case $\lambda = 0$ refers to the L^2 NNR surrogate, and is only reported for comparison.

TABLE A12 | Training times vs FOM dimension for the third test case, Section 7.3.

Model	Map to kernel S_0	h	dim(Q)	dim(R)	L^2 error on q	Training time
Conserv. POD-NN	V_n	1/16	956	4	0.92%	1 m 09.5 s
		1/32	3664	4	0.93%	2 m 26.1 s
		1/64	14408	4	1.04%	58.7 s
Conserv. DL-ROM	$\nabla\times$	1/16	956	341	1.30%	4 m 20.1 s
		1/32	3664	1265	1.52%	4 m 14.2 s
		1/64	14408	4889	1.86%	4 m 39.3 s
Conserv. DL-ROM	$I - S_I B$	1/16	956	956	1.35%	4 m 27.5 s
		1/32	3664	3664	0.78%	7 m 09.5 s
		1/64	14408	14408	9.18%	1 h 06 m 22.9 s

The fracture flow model, cf. [49, 50], then takes the form (1.1) with

$$\langle A\mathfrak{q}, \tilde{\mathfrak{q}}\rangle := \sum_{\substack{i\in I\\ 1\le d_i\le 3}} (K_{\parallel,i}^{-1}\boldsymbol{q}_i, \tilde{\boldsymbol{q}}_i)_{\Omega_i} + \sum_{\substack{j\in I\\ d_j=d_i-1}} (K_{\perp,ij}^{-1}\boldsymbol{\nu}_i\cdot\boldsymbol{q}_i, \boldsymbol{\nu}_i\cdot\tilde{\boldsymbol{q}}_i)_{\partial_j\Omega_i}$$

$$\langle B\mathfrak{q}, \tilde{\mathfrak{p}}\rangle := \sum_{\substack{i\in I\\ 1\le d_i\le 3}} (\nabla\cdot\boldsymbol{q}_i, \tilde{p}_i)_{\Omega_i} - \sum_{\substack{j\in I\\ d_j=d_i-1}} (\boldsymbol{\nu}_i\cdot\boldsymbol{q}_i, \tilde{p}_j)_{\partial_j\Omega_i}$$

The operator B corresponds to the mixed-dimensional divergence operator, which we denote by $\mathfrak{D}\cdot\mathfrak{q}$. This operator is part of a differential complex [51] and the mixed-dimensional curl $\mathfrak{D}\times$ has the property $\mathfrak{D}\cdot(\mathfrak{D}\times\mathfrak{r}) = 0$ for all $\mathfrak{r}$ in its domain. Precise definitions of the mixed-dimensional curl can be found in [30, 51, 52].

To conclude this section, we define the norms on the mixed-dimensional variables as follows:

$$\|\mathfrak{p}\|_P^2 := \sum_{i \in I} \|p_i\|_{\Omega_i}^2$$

$$\|\mathfrak{q}\|_Q^2 := \left(\sum_{\substack{i \in I \\ 1 \leq d_i \leq 3}} \|\boldsymbol{q}_i\|_{\Omega_i}^2 + \sum_{\substack{j \in I \\ d_j = d_i - 1}} \|\boldsymbol{\nu}_i \cdot \boldsymbol{q}_i\|_{\partial_j \Omega_i}^2 \right) + \|\mathfrak{D} \cdot \mathfrak{q}\|_P^2.$$

With a slight abuse of notation, we refer to these norms as L^2 and $H(\mathrm{div})$, respectively, in Section 7.2.