LOD Surrogate Modelling for multiscale Darcy-Flow Problems

Marc Haltmayer [*1] **Jaemin Seo** [*1] **Yu Seung Lee** [1] **Sungyeop Lee** [2] **Jaehoon Jeong** [2] **Jae Yong Lee** [1]

Abstract

Multiscale problems are notoriously difficult to tackle using traditional numerical methods, as accurately resolving fine-scale features often requires prohibitively fine discretizations. This challenge is particularly pronounced in applications such as materials science, fluid dynamics, climate systems, chemical processes, and complex networks. Recent neural operator models provide a promising data-driven alternative, but frequently struggle to achieve sufficient accuracy in the presence of strongly heterogeneous or oscillatory coefficients. In this work, we focus on the solution of elliptic PDEs with rough and high-contrast inputs. The Localized Orthogonal Decomposition (LOD) method is a well-established numerical approach for such problems, but it comes, however, at a substantial computational cost. In this work, we investigate the performance of popular neural operator architectures on these challenging multiscale problems and identify key limitations in their ability to resolve fine-scale structure. We introduce **LOD-MSNO** (LOD-Multiscale Neural Operator), a hybrid approach that leverages the LOD method as a strong multiscale prior by building on its representation of the solution as a linear combination of problem-adapted basis functions, while addressing its main computational bottlenecks through data-driven operator learning. We demonstrate that our proposed method outperforms current neural operator baselines in terms of accuracy for challenging multiscale input, while mainly retaining the computational efficiency of neural operator models.

[*]Equal contribution [1]Department of Artificial Intelligence, Chung-Ang University, 84 Heukseok-ro, Dongjak-gu, Seoul, Republic of Korea [2]Computational Science and Engineering Team, Samsung Electronics, 1 Samsung-ro, Giheung-gu, Yongin-si, Gyeonggi-do, 17113, Republic of Korea. Correspondence to: Jaeyong Lee <jaeyong@cau.ac.kr>.

Accepted by AI4Science workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

1. Introduction

Multiscale systems, characterized by the coexistence of important features across multiple spatial and temporal scales, arise in a wide range of scientific and engineering domains. Modeling such systems typically requires the integration of multiple models operating at different resolutions, where microscale phenomena must often be resolved to accurately capture system behavior. A representative example is given by Darcy Flow,

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f \qquad \text{in } D, \\ u &= 0 \qquad \text{on } \partial D, \end{aligned} \tag{1}$$

where $D \subset \mathbb{R}^d$, $d = 2, 3$ is a bounded physical domain, $u : D \to \mathbb{R}$ denotes the pressure, $f \in L^2(D)$ is a given source term, and $\kappa : D \to \mathbb{R}_+$ is a heterogeneous permeability coefficient. The Darcy Flow equation can be viewed as a heterogeneous extension of Poisson's equation, which arises naturally across a range of physical settings involving transport through complex media. For example, it describes fluid flow in porous subsurface (Bear, 1988) formations, while analogous formulations govern heat conduction in materials (Yang & Huang, 2024) with spatially varying thermal properties and potential fields in electrically inhomogeneous media (Kirkham, 2023). Across these problems, the spatial variability of the coefficient field plays a central role, as even fine-scale heterogeneities can significantly influence the global solution behavior, leading to pronounced multiscale effects. This interplay between fine-scale structure and macroscopic response poses significant challenges for both modeling and computation, often requiring high-resolution simulations to accurately capture the underlying physics. Such demands are particularly critical in practical engineering workflows, where repeated evaluations under varying material configurations or design parameters are often required. As a result, developing accurate and efficient solution strategies for this class of problems remains a key objective in both scientific computing and industrial applications.

In this context, neural operator methods have emerged as promising alternatives, as they aim to learn mappings between infinite-dimensional function spaces and thereby approximate solution operators of parametric PDEs directly. However, despite their favorable computational properties

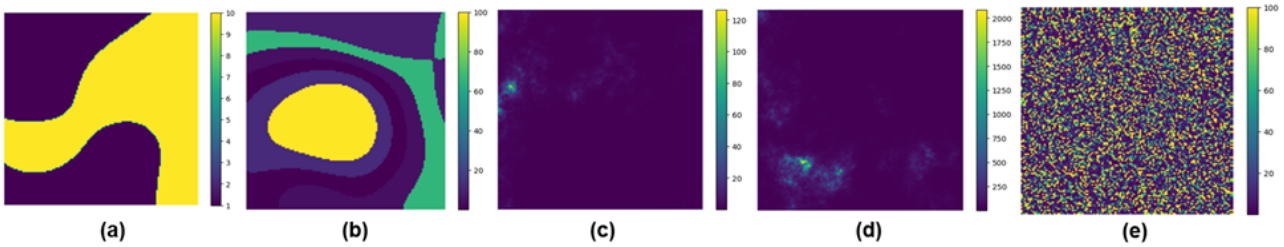

Figure 1. Example permeability fields $\kappa(x)$ (color bars show ranges). (a) Two-faces piecewise-constant as considered in many related works. (b) Six-faces piecewise-constant. (c) lognormal random field. (d) High-contrast lognormal random field spanning several orders of magnitude, yielding a particularly difficult regime due to extreme coefficient variation. (e) Cell-wise i.i.d. discrete random field sampled from a finite level set.

and remarkable success in learning solution operators of parametric PDEs, neural operator models exhibit important limitations when applied to multiscale problems with rough and strongly heterogeneous inputs. Much of the existing neural operator literature focuses on relatively simplified flow configurations; for instance, benchmark studies involving the Fourier Neural Operator (FNO) (Li et al., 2021), Deep Operator Network (DeepONet) (Lu et al., 2022), and Transformer-based architectures such as Transolver (Wu et al., 2024b), and related approaches frequently consider two-phase Darcy flow problems with moderately varying permeability fields (Figure 1(a)). While these settings provide useful testbeds for model development and evaluation, they do not fully reflect the complexity encountered in realistic subsurface flow applications. In particular, multi-phase Darcy flow in highly heterogeneous and highly oscillatory media (Figure 1(b)-(e)) remains largely unexplored in the neural operator community, despite its relevance in groundwater hydrology, petroleum engineering, and carbon sequestration.

Contributions. In this work, we address these gaps and make the following contributions.

- We identify a key shortcoming of existing neural operator models when applied to multiscale problems with rough, high-contrast coefficients. Using Darcy flow as a prototypical example, we demonstrate the challenges posed by highly heterogeneous and multiscale permeability fields through a series of carefully designed datasets based on applications in industry that exhibit complex fine-scale features.

- We propose **LOD-MSNO**, an operator learning model specifically designed for multiscale PDE problems that follows a coefficient-learning paradigm. The approach adopts a hybrid strategy by using the FEM-based Localized Orthogonal Decomposition (LOD) approximation as a strong numerical prior and employing neural networks to learn the associated multiscale basis functions and expansion coefficients.

- We demonstrate that our method can outperform established neural operator benchmarks on the proposed multiscale datasets. These results validate the feasibility of our idea and highlight its potential for reliable and efficient surrogate modeling of multiscale PDE problems.

2. Related work

Multiscale Numerical Methods. The numerical approximation of PDEs with highly heterogeneous and multiscale coefficients has been extensively studied in computational mathematics. Classical *multiscale methods* (Pavliotis & Stuart, 2008) and *numerical homogenization* (Blanc & Le Bris, 2023) techniques aim to derive effective coarse-scale models that capture fine-scale effects implicitly. While successful in settings with clear scale separation, their performance degrades when coefficients exhibit strong local variations or lack periodicity. Localized approaches, most notably the *LOD* (Målqvist & Peterseim, 2020) method, address these limitations albeit with a high computational cost.

Neural Operators. Neural operator methods aim to learn mappings between function spaces directly. *DeepONet* (Lu et al., 2021b) represents operators using coupled branch and trunk networks, with extensions such as *Physics-informed DeepONet (PI-DeepONet)* (Goswami et al., 2022) or *HyperDeepONet* (Lee et al., 2023). A complementary class of methods is based on spectral representations. The *FNO* (Li et al., 2021) and its variants, including *Convolutional Neural Operators (CNO)* (Raonić et al., 2023), *U-shaped Neural Operators (UNO)* (Rahman et al., 2023) and the physics-informed *PINO* (Li et al., 2024), have demonstrated strong performance on parametric PDE benchmarks. However, their reliance on global spectral representations can limit their ability to resolve localized fine-scale features in highly heterogeneous and multiscale settings. Recent work has explored attention-based and graph-based operator learning, for example, is *Transolver* (Wu et al., 2024b) is a representative example that leverages Transformer architectures to model long-range dependencies.

Hybrid approaches/Coefficient Learning. Recent work has explored hybrid approaches that combine classical numerical methods, such as finite element or spectral discretizations, with neural networks in order to leverage the strengths of both paradigms (Fanaskov & Oseledets, 2023). In these methods, the solution is represented in a fixed basis, and the learning task is to infer the corresponding expansion coefficients. For example, Finite element operator network (FEONet) (Lee et al., 2025) integrates neural networks with the continuous Galerkin FEM by directly learning solution coefficients, enabling physics-aware training without requiring labeled data. Related spectral approaches, such as SCLON (Choi et al., 2024) and ULGNet (Choi et al., 2023), learn coefficients in spectral or Galerkin bases, further demonstrating the effectiveness of coefficient learning as a hybrid modeling strategy for PDEs. Other hybrid AI-based approaches have also demonstrated substantial computational speedups and improved flexibility, with successful applications reported in areas such as solid mechanics (Kalina et al., 2023; Yizheng et al., 2026), fluid dynamics (Vinuesa et al., 2022), climate science (Gentine et al., 2018), and biomedical systems (C.Y.Peng et al., 2021).

3. Preliminaries

3.1. Problem statement and mathematical setup

We consider the Darcy flow problem with homogeneous Dirichlet boundary conditions (1) where $f \in L^2(D)$ and $\kappa\colon D \to \mathbb{R}$ denotes the heterogeneous diffusion coefficient. Throughout this work, we assume that κ is uniformly elliptic and almost everywhere bounded. However, no further smoothness or scale separation assumptions on κ are imposed; in particular, κ may exhibit highly oscillatory or high-contrast behavior. A fine-scale Finite-Element approximation for (1) is obtained by seeking $u_h \in V_h$ such that

$$a(u_h, v_h) := \int \kappa \nabla u_h \cdot \nabla v_h = \int_D f v_h \, dx \quad \forall v_h \in V_h, \tag{2}$$

where V_h denotes a Finite-Element space like $V_h := P_1(\mathcal{T}_h) \cap H_0^1(D)$ with

$$P_1(\mathcal{T}_h) := \left\{ v \in C^0(D) \;\middle|\; v|_K \text{ is affine for all } K \in \mathcal{T}_h \right\}$$

and $\mathcal{T}_h$ being a triangulation of the domain D. Writing $u_h = \sum_{i=0}^{N_h-1} \boldsymbol{u}_i \phi_h^i$, as a linear combination of the nodal basis functions ϕ_h^i associated with $\mathcal{T}_h$, this problem is equivalent to solving the linear system $\mathbf{A}\boldsymbol{u} = \mathbf{f}$ with $A_{ij} = a(\phi_h^i, \phi_h^j)$ and $f_i = \int_D f \phi_h^i, dx$. In order to accurately resolve the fine-scale features of a potentially highly varying and oscillatory coefficient κ, the mesh size h must typically be chosen very small, which may result in prohibitively high computational cost for solving the linear system. On the other hand, a coarse-scale finite element approximation based on a coarser approximation space V_H with $h < H$ generally fails to capture the influence of the unresolved fine-scale variations in κ and therefore leads to poor accuracy.

3.2. Localized orthogonal decomposition (LOD)

The LOD method, originally proposed in (Målqvist & Peterseim, 2011), is a Finite-Element method designed to overcome this difficulty by constructing problem-adapted multiscale basis functions that combine the computational efficiency of coarse discretizations with the accuracy of fine-scale resolution.

3.2.1. Orthogonal decomposition of scales

The central idea of the LOD method is to construct an accurate approximation of the solution in a *low-dimensional* space of the same dimension as a coarse finite element space V_H, while enriching this space with fine-scale information from a high-dimensional space V_h. To this end, first define a projection (or quasi-interpolation) operator $\mathcal{I}_H\colon V \to V_H$, which projects functions from the energy space $V = H_0^1(D)$ onto its associated counterpart in the coarse finite element space V_H. We define the space of microscopic fine-scale functions as

$$W := \ker(\mathcal{I}_H) = \{ v \in V \mid \mathcal{I}_H(v) = 0 \}. \tag{3}$$

The space W contains all functions with fine-scale features that cannot be represented in the coarse space V_H and yields an orthogonal decomposition $V = V_H \oplus W$ w.r.t. the standard inner product on V. In order to construct an improved approximation space compared to V_H, we exploit the *problem-specific inner product* on V induced by the bilinear form

$$a(u, v) := \int_D \kappa \nabla u \cdot \nabla v \, dx. \tag{4}$$

and define the improved multiscale approximation space as

$$V_H^{\text{ms}} := \{ v_H^{\text{ms}} \in V \mid a(v_H^{\text{ms}}, w) = 0 \;\text{ for all } w \in W \}, \tag{5}$$

which is nothing else but the orthogonal complement of the subspace W in V with respect to the inner product a, leading to the $\mathcal{A}$-orthogonal decomposition $V = V_H^{\text{ms}} \oplus W$. By construction, V_H^{ms} has the same dimension as V_H and consists of functions that are $\mathcal{A}$-orthogonal to all unresolved fine-scale components. Based on this decomposition, we define the *ideal numerical homogenization method* as follows: given $f \in L^2(D)$, find $u_H^{\text{ms}} \in V_H^{\text{ms}}$ such that

$$a(u_H^{\text{ms}}, v) = \int_D f v \, dx \quad \text{for all } v \in V_H^{\text{ms}}. \tag{6}$$

It is "ideal" in the sense that V_H^{ms} contains all coarse-resolvable fine-scale features of the multiscale coefficient κ.

3.2.2. Practical realization and localization of the orthogonal decomposition

The ideal numerical homogenization method introduced above is formulated in the abstract multiscale space $V_H^{\mathrm{ms}} \subset V$. In order to make this formulation computationally feasible, we next clarify how the corresponding variational problem can be solved in practice. To this end, consider the *correction operator* $Q\colon V \to W$, defined as the $a(\cdot,\cdot)$-orthogonal projection onto the detail space $W = \ker(\mathcal{I}_H)$. More precisely, for a given $v \in V$, the correction $Qv \in W$ is defined as the unique solution of

$$a(Qv, w) = a(v, w) \quad \text{for all } w \in W. \tag{7}$$

By construction, $(1-Q)v$ is a-orthogonal to W. One can show that the operator $(1-Q)\colon V_H \to V_H^{\mathrm{ms}}$ is bijective, with inverse given by the projection $\mathcal{I}_H$. Consequently, every function $v_H^{\mathrm{ms}} \in V_H^{\mathrm{ms}}$ can be written uniquely as $v_H^{\mathrm{ms}} = (1-Q)v_H$ for some $v_H \in V_H$. Using this observation, the ideal multiscale problem can be reformulated as follows: find $u_H \in V_H$ such that

$$a\big((1-Q)u_H, (1-Q)v_H\big) = \int_D f(1-Q)v_H \, dx \tag{8}$$

for all $v_H \in V_H$. This formulation is more favorable from a computational perspective, as it depends on the discrete, low-dimensional unknown $u_H \in V_H$ instead of the rather abstract object $u_H^{\mathrm{ms}} \in V_H^{\mathrm{ms}}$. In practice, the computation of the correction operator Q itself is localized to small patches around each coarse element in order to make the method computationally feasible. This localization is motivated by the exponential decay of the corrector functions away from their associated coarse elements (Målqvist & Peterseim, 2020). For a given coarse element $T \in \mathcal{T}_H$, we define a patch $\mathcal{N}^\ell(T)$ consisting of all coarse elements within ℓ layers of T, and compute the localized correction operator Q_h^ℓ by solving fine-scale elliptic problems of the form (7) on these patches. The global localized correction operator is obtained by aggregating the contributions over all coarse elements: $Q_h^\ell := \sum_{T \in \mathcal{T}_H} Q_{T,h}^\ell$. Using this operator, the fully discrete multiscale finite element space is defined as $V_{H,\ell}^{\mathrm{ms},h} = \operatorname{span}\left\{\Phi_H^j - Q_h^\ell(\Phi_H^j)\right\}$ where $\{\Phi_H^j\}$ denotes the standard coarse finite element basis. This space has the same dimension as the coarse space but is enriched with fine-scale information induced by the heterogeneous coefficient. The final LOD solution is obtained by performing a Galerkin projection of the original problem onto this problem-adapted multiscale space.

3.3. Limitations of the LOD method

Despite its strong theoretical guarantees, the LOD method suffers from notable computational limitations. In particular, the LOD-based discretization requires the solution of fine-scale auxiliary problems on overlapping subdomains $\mathcal{N}^\ell(T)$ for each realization of the random coefficient. As the localization parameter ℓ increases, these subdomains grow in size and overlap more strongly, leading to a rapidly increasing computational cost for the construction of the correction operator Q_h^ℓ. However, theoretical accuracy results typically require choosing $\ell \approx C|\log H|$ for some constant $C > 0$, which may result in relatively large localization patches in practice. This renders the computation of the correctors a major computational bottleneck.

Moreover, the corrected basis functions lose the local support of standard finite element nodal basis functions: Instead of being supported on a single coarse element T, the LOD basis functions have support on the entire patch $\mathcal{N}^\ell(S)$, thereby significantly polluting the sparsity structure of the resulting system matrix. This effect is further exacerbated in settings where the resulting sparsity-polluted linear systems must be solved repeatedly. These considerations motivate the development of fully surrogate-based models that bypass both the explicit computation of the basis corrections Q_h^ℓ and the solution of the associated dense or weakly sparse linear systems.

4. Proposed method: LOD-MSNO

Central to the LOD method are the *corrected basis functions*

$$\Psi_H^j := \Phi_H^j - Q_h^\ell(\Phi_H^j), \qquad j = 1, \ldots, N_H, \tag{9}$$

where $\{\Phi_H^j\}$ denotes the standard coarse nodal basis functions and Q_h^ℓ is the localized correction operator computed from fine-scale problems posed on overlapping patches of diameter proportional to the localization parameter ℓ. Using the corrected basis $\{\Psi_H^j\}$, the LOD-solution is represented as

$$u_{\mathrm{LOD}} = \sum_{j=1}^{N_H} \boldsymbol{u}_{\mathrm{LOD}}^{(j)} \Psi_H^j, \tag{10}$$

where the coefficient vector $\boldsymbol{u}_{\mathrm{LOD}} \in \mathbb{R}^{N_H}$ is obtained by solving the corresponding system of equations that can be derived by inserting (10) into the weak form in the space $V_{H,\ell}^{\mathrm{ms},h}$. The goal of LOD-MSNO is to learn the coefficients $\boldsymbol{u}_{\mathrm{LOD}}$ of the LOD solution representation directly, while avoiding the explicit and computationally expensive assembly of the linear system and its solution.

4.1. Learning the coefficients

The coefficient vector $\boldsymbol{u}_{\mathrm{LOD}}$ is defined as the solution of the linear system

$$\boldsymbol{A}_{\mathrm{LOD}} \boldsymbol{u}_{\mathrm{LOD}} = \boldsymbol{f}_{\mathrm{LOD}}, \tag{11}$$

where $\boldsymbol{A}_{\mathrm{LOD}}$ denotes the stiffness matrix and $\boldsymbol{f}_{\mathrm{LOD}}$ denotes the load vector expressed in the corrected LOD basis.

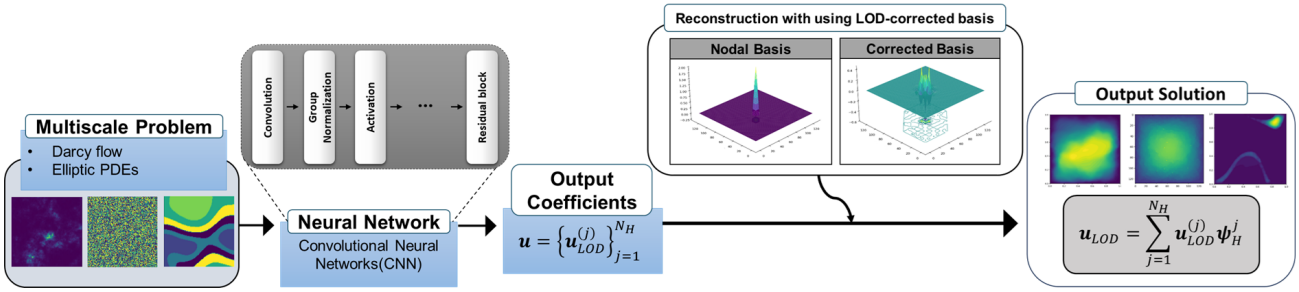

Figure 2. Schematic of the proposed LOD-MSNO. Neural network predicts the coarse LOD coefficient vector.

The focus of LOD-MSNO lies on learning $\boldsymbol{u}_{\mathrm{LOD}}$ directly without assembling and solving (11) explicitly. To train a neural network surrogate for the coefficient vector $\boldsymbol{u}_{\mathrm{LOD}}$, we proceed as follows. Given permeability coefficients $\kappa_i := \kappa(\omega_i)$ parameterized by a random set of parameters $\omega_i \in \Omega$, where $(\Omega, \mathcal{F}, \mathbb{P})$ is a possibly high-dimensional probability space, we construct a dataset consisting of tuples

$$\{(\boldsymbol{A}_{\mathrm{LOD}}(\omega_i), \boldsymbol{u}_{\mathrm{LOD}}(\omega_i))\}_{i=1}^{M},$$

where $\boldsymbol{A}_{\mathrm{LOD}}(\omega_i)$ denotes the LOD stiffness matrix and $\boldsymbol{u}_{\mathrm{LOD}}(\omega_i)$ denotes the ground truth coefficient vector corresponding to ω_i (to $\kappa(\omega_i)$). Since the space $V_{H,\ell}^{\mathrm{ms},h} = \operatorname{span}\left\{\Psi_H^j\right\}$ is typically low-dimensional, saving the matrices $\boldsymbol{A}_{\mathrm{LOD}}$ remains feasible in terms of memory consumption and computing them is necessary anyway to obtain the ground truths $\boldsymbol{u}_{\mathrm{LOD}}(\omega_i)$. Rather than employing a standard L^2- or mean-square-based regression loss, we adopt an energy-based loss which is defined as

$$\mathcal{L}_{E,M}(\boldsymbol{u}) := \frac{1}{M}\sum_{i=1}^{M}\left|\boldsymbol{A}_{\mathrm{LOD}}^{1/2}(\omega_i)\big(\boldsymbol{u}_{\mathrm{LOD}}(\omega_i) - \boldsymbol{u}(\omega_i)\big)\right|. \tag{12}$$

The choice of the energy-based loss (12) is directly motivated by the variational formulation of the underlying elliptic PDE. At the discrete level, this variational principle is associated with the (discrete) energy norm of a coefficient vector $\boldsymbol{v}$, which is given by

$$\|\boldsymbol{v}\|_{\boldsymbol{A}_{\mathrm{LOD}}}^2 = \boldsymbol{v}^\top \boldsymbol{A}_{\mathrm{LOD}}\boldsymbol{v}.$$

Consequently, the proposed loss (12) measures the error between the predicted and true coefficients in the natural energy norm associated with the elliptic operator. We hypothesize that employing this energy-based loss can be beneficial for generalization, as it is more closely aligned with the variational principle underlying the PDE. We empirically observe that training with the energy-based loss (12) leads to improved accuracy and better generalization, particularly for coefficients exhibiting strong heterogeneity and high contrast. An ablation study comparing different loss functions is provided in Figure 4. In the next section, we prove an error estimate for our LOD-MSNO model and analyze how using the energy-based loss affects its convergence properties.

4.2. Error Analysis

Let $\mathcal{N}$ be any class of $\mathbb{R}^{N_H}$-valued functions that are parameterized as a neural network, and let

$$\boldsymbol{u}_{\mathrm{LOD}}^{E,M} := \operatorname*{argmin}_{\boldsymbol{u}\in\mathcal{N}} \mathcal{L}_{E,M}(\boldsymbol{u})$$

be the neural network that minimizes the loss function (12). In this section, we derive a quantitative error estimate between the exact solution of problem (1) and the proposed LOD-MSNO approximation given by the linear combination of the predicted coefficients $\widehat{\boldsymbol{u}}_{\mathrm{LOD}}(\omega)$, given an input $\omega \in \Omega$, and the LOD-basis functions:

$$\widehat{u}_{\mathrm{LOD}}(x;\omega) := \sum_{j=1}^{N_H}\big(\boldsymbol{u}_{\mathrm{LOD}}^{E,M(j)}(\omega)\big)\Psi_H^j(x;\omega).$$

Theorem 4.1 (Error estimate for LOD-MSNO)**.** *Let $\lambda_{\min}(\omega)$ and $\lambda_{\max}(\omega)$ denote the smallest and largest eigenvalues of the LOD stiffness matrix $\boldsymbol{A}_{\mathrm{LOD}}(\omega)$ for a given input representation ω, and assume that there exist constants $c_0, c_1 > 0$ such that*

$$\operatorname*{ess\,inf}_{\omega\in\Omega} \lambda_{\min}(\omega) \geq c_0 > 0,$$
$$\|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \leq c_1 < \infty,$$

and patch size chosen as $\ell \approx |\log H|$. Let $R_M(\mathcal{G}^E)$ denote the empirical Rademacher complexity of the function class $\mathcal{G}^E$ defined by

$$\mathcal{G}^E := \left\{\omega \mapsto \left|\boldsymbol{A}_{\mathrm{LOD}}^{1/2}(\omega)\big(\boldsymbol{u}_{\mathrm{LOD}}(\omega) - \boldsymbol{u}(\omega)\big)\right| : \boldsymbol{u} \in \mathcal{N}\right\}.$$

Then

$$\mathbb{E}\big[\|u - \widehat{u}_{\mathrm{LOD}}\|_{L^1(\Omega;L^2(D))}\big] \lesssim H^2 + \frac{1}{\sqrt{c_0}} R_M(\mathcal{G}^E) + \frac{2c_1}{\sqrt{c_0}} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}. \tag{13}$$

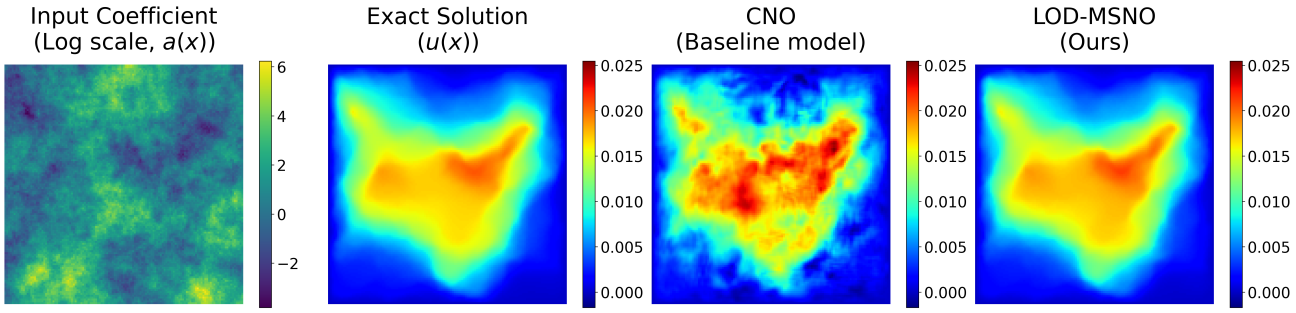

Figure 3. Prediction errors for lognormal2 coefficients using the best-performing baseline model and LOD-MSNO(ours).

Table 1. Test relative L^2 errors on the fine grid with $h = 2^{-7}$.

Dataset	FNO	CNO	UNO	PI-DeepONet	Transolver	PINO	FEONet	**LOD-MSNO (ours)**
Lognormal1	0.049±0.001	0.018±0.001	0.044±0.000	0.138±0.001	0.082±0.022	0.025±0.000	0.099±0.000	**0.012**±0.002
Lognormal2	0.163±0.002	0.093±0.003	0.160±0.003	2.028±0.084	0.341±0.016	0.145±0.002	1.805±0.073	**0.042**±0.006
Quantile	0.254±0.010	0.120±0.007	0.239±0.008	1.282±0.015	0.610±0.050	0.203±0.012	0.129±0.020	**0.118**±0.017
Checkerboard	0.064±0.000	**0.010**±0.001	0.064±0.001	0.062±0.000	0.053±0.003	0.035±0.000	0.668±0.109	0.018±0.003

Theorem 4.1 establishes that the total error between the true solution of (1) is primarily governed by three contributions. The first term reflects the approximation properties of the LOD method and recovers the optimal bound when $\ell \approx |\log(H)|$. The second term captures the generalization capability of the neural network class, which can be quantified via its Rademacher complexity. The third term depends on both the expressive power of the neural network class $\mathcal{N}$ and the spectral properties of the LOD stiffness matrix corresponding to a given input $\omega \in \Omega$. By picking a neural network class $\mathcal{N}$ with strong enough approximation (third term) and generalization properties (second term), and by choosing H small enough (first term), we can achieve that the expected error between the true solution u and the LOD-MSNO prediction $\widehat{u}_{\mathrm{LOD}}$ converges to zero. A complete proof of the theorem is provided in the appendix.

5. Experiments

We evaluate the proposed LOD-MSNO on the steady Darcy flow problem, to learn the mapping from the permeability coefficient κ to the corresponding coefficients $\boldsymbol{u}_{\mathrm{LOD}}$ in (10). Our model follows a hybrid *coefficient-learning* paradigm that combines the multiscale structure of the LOD method with neural networks. Using the corrected LOD basis as a fixed representation, we train a convolutional neural network, termed `LODMimeticNet`, to directly predict the LOD coefficient vector from the input permeability field. The architecture consists of a convolutional encoder that learns a restriction from the fine grid to the coarse LOD grid, followed by a residual mixing trunk operating at coarse resolution. The network is built from Conv–GroupNorm–GELU blocks and ResNet-style residual layers, enabling stable training and effective multiscale feature extraction. Coarse-scale mixing is performed entirely at the LOD resolution, and the final output corresponds to the interior LOD coefficients, consistent with homogeneous Dirichlet boundary conditions. This design explicitly mirrors the structure of the LOD discretization while replacing the expensive assembly and solution of the dense LOD system with a learned surrogate. Further details on the architecture of `LODMimeticNet` can be found in the appendix.

Baseline models. We compare our method against a range of established baselines to ensure a balanced and meaningful evaluation. First, we include FNO, CNO, and UNO, which are widely used and representative neural operator architectures based on global and multiresolution convolutional designs. Similarly, Transolver is incorporated as a transformer-based operator model capable of capturing long-range dependencies. These models serve as standard data-driven operator learning benchmarks. To ensure fairness with respect to physics-informed approaches, we additionally compare against PINO, as our method also incorporates physical structure—specifically through the use of problem-adapted basis functions and the inclusion of the LOD stiffness matrix $\boldsymbol{A}_{\mathrm{LOD}}$ in the loss function. We further include FEONet as a representative hybrid neural network–numerical method, since it integrates classical finite element method (FEM) principles into the learning process, providing a relevant point of comparison for methods that combine learned and discretization-based components. Finally, we consider PI-DeepONet, which is both physics-informed and structurally similar to our approach in that it represents the solution as a linear expansion over a set of basis functions. This makes it a particularly meaningful baseline for evaluating the advantages of our LOD-based formulation. Our LOD-MSNO model is trained to predict

the LOD coefficient vector $\boldsymbol{u}_{\mathrm{LOD}}$, from which the full fine-scale solution is reconstructed using the corrected LOD basis.

5.1. Benchmark datasets

As depicted in Figure 1, training and test datasets are generated using four distinct classes of permeability fields designed to probe different multiscale and high-contrast regimes. In the following, we refer to them as *lognormal1*, *lognormal2*, *quantile*, and *checkerboard*. These datasets are specifically constructed to highlight a key shortcoming of neural operator models when applied to multiscale problems, and are inspired by industry-relevant real-world applications. A more detailed description of the dataset characteristics and its generation process can be found in the appendix.

Lognormal coefficients. For the lognormal datasets, we consider permeability fields of the form $\kappa(x) = \exp(Z(x))$ where Z is a centered Gaussian random field with Whittle–Matérn covariance with smoothness parameter $\nu > 0$, correlation length $\ell > 0$, and variance $\sigma^2 > 0$. We distinguish two regimes: (1) *Lognormal1* with $\sigma = 1.0$, $\ell = 0.1$, (2) *Lognormal2* with $\sigma = 2.0$, $\ell = 0.3$. The second setting exhibits significantly stronger oscillations and substantially higher contrast, making it particularly challenging for operator learning methods.

Quantile coefficients. For the quantile dataset, we first sample a Gaussian random field Z with squared-exponential covariance and define a lognormal field $\exp(Z)$. The resulting values are then discretized into a finite set of prescribed permeability levels via empirical quantiles and randomly permuted across the domain, yielding piecewise-constant coefficients with sharp jumps and high contrast. This construction produces highly heterogeneous coefficients with nontrivial global structure and limited smoothness.

Checkerboard coefficients. The checkerboard coefficients are defined as piecewise-constant random fields on a fine Cartesian grid, where each cell independently takes values from a fixed set of permeability levels. This yields coefficients with discontinuities at the grid scale and no spatial correlation.

5.2. Evaluation of baseline models and LOD-MSNO

For LOD-MSNO, we fix the coarse and fine mesh sizes to $H = 2^{-4}$ and $h = 2^{-7}$, respectively. The model is trained to predict the LOD coefficient vector $\boldsymbol{u}_{\mathrm{LOD}}$ associated with the corrected multiscale basis $\{\Psi_H^i\}$. At inference time, the predicted coefficients are combined with the corrected basis functions to reconstruct the solution, which is then evaluated on a uniform 128×128 grid. Training of LOD-MSNO is performed using a loss function based on the loss function (12). In contrast, all baseline models are trained directly on pairs $\{(\kappa_i, u_h^{(i)})\}$ of permeability fields and corresponding fine-scale reference solutions, using the relative L^2 error as the training objective. For the models LOD-MSNO, FNO, CNO, and UNO, we apply the same input preprocessing and normalization pipeline, consisting of a logarithmic transformation of the permeability field, gradient-magnitude and coordinate channels, and channel-wise input as well as per-component output normalization (see Appendix C for details). For PI-DeepONet and Transolver, which do not support additional input channels, the gradient-magnitude and coordinate augmentations (`add_grad` and `add_coords`) are therefore omitted. For evaluation, we report the mean and standard deviation of the relative L^2 error (Rel-L^2) computed on the fine grid, where u denotes the reference solution and $\widehat{u}$ the model prediction. The aggregated results are summarized in Table 1. The results indicate that LOD-MSNO attains competitive and, in three out of four cases, improved relative L^2-error accuracy compared to the considered baselines, with particularly strong performance on the multiscale lognormal datasets, while remaining comparable to the best-performing methods on the other coefficient configurations. In comparison to FEONet and PI-DeepONet, the results highlight the advantage of employing problem-adapted multiscale basis functions. While FEONet relies on a fixed coarse $P1$-FEM basis and PI-DeepONet learns a global set of basis functions via the trunk network, LOD-MSNO leverages LOD basis functions that are specifically tailored to the underlying coefficient field. This leads to consistently improved accuracy, particularly in strongly heterogeneous regimes. Moreover, when comparing to physics-informed approaches such as PINO and PI-DeepONet, the results suggest that incorporating physical structure through a strong numerical prior is more effective than enforcing the PDE via autodifferentiation-based loss terms. In particular, constructing the solution using the LOD framework allows the model to directly encode relevant multiscale features of the problem, which appears to be a more robust way of injecting physical information than relying solely on PDE residual minimization.

5.3. Ablation study

We conduct an ablation study to assess the impact of the training loss function on the performance and generalization ability of LOD-MSNO. In particular, we train the model using three alternative losses: the standard L^2 loss, the relative L^2 loss, and the L^1 loss. All models are trained on 500 samples with lognormal2 and quantile-type permeability fields, and we report the evolution of the test relative L^2 error during training. As shown in figure 4, training with the energy-based loss yields the lowest final error for both coefficient classes, while the L^1 and (relative) L^2 saturate at higher error levels. Especially in the lognormal2 case, the

Table 2. Ablation on preprocessing and input-channel augmentation for LOD-MSNO. Fine-grid test relative L^2 errors.

Dataset	Training setting	Variants			
		LOD-MSNO	w/o log + normalization	w/o add_grad + add_coords	w/o all
	log + normalization	✓	✗	✓	✗
	add_grad + add_coords	✓	✓	✗	✗
Lognormal1		**0.025**	0.080	0.034	0.236
Lognormal2		**0.042**	0.135	0.096	1.215
Quantile		**0.089**	0.247	0.319	0.313
Checkerboard		**0.020**	0.020	0.056	0.188

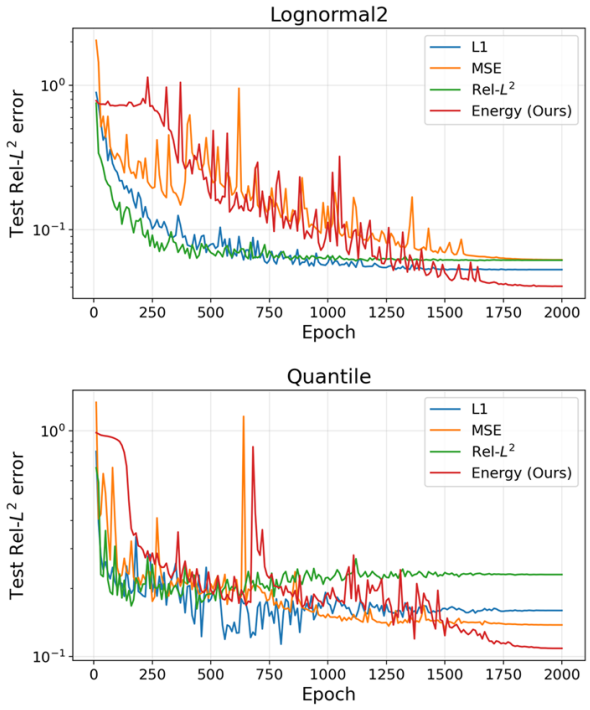

Figure 4. Loss ablation for predicting $\boldsymbol{u}_{\mathrm{LOD}}$. Test Rel-$L^2$ error vs. epoch for lognormal2 (top) and quantile (bottom).

energy-loss exhibits the most stable convergence behavior, whereas the L^2- and L^1-based trainings display larger fluctuations and higher asymptotic errors. Table 2 reports the impact of the proposed preprocessing and channel augmentation strategies on the performance of LOD-MSNO. All models are trained using the energy-norm loss on 500 samples from the Quantile dataset. Across all datasets, the full configuration—combining logarithmic scaling, normalization, and the additional gradient-magnitude and coordinate channels—consistently yields the lowest relative L^2 errors. Removing either the log-scaling and normalization or the auxiliary channels leads to a noticeable degradation in accuracy, while omitting all preprocessing steps results in a substantial loss of performance. These results demonstrate that both the normalization strategy and the inclusion of gradient and coordinate information play an important role in enabling accurate and stable learning for LOD-MSNO.

6. Conclusion and Outlook

In this work, we focused on learning the LOD solution coefficients, while the construction of the corrected multiscale basis functions, as outlined in Section 4.1, was not surrogated. An interesting extension of the proposed framework is therefore to move beyond a fixed, precomputed multiscale basis and instead learn the basis functions directly from data. In the LOD setting, this corresponds to learning the correction operator Q_h^ℓ as a function of the input coefficient κ.

Such an approach would further increase the computational advantages of our model, removing the need to numerically compute the LOD basis to obtain the full solution. A natural strategy is to couple the learning of the correction operator with the coefficient prediction, instead of learning both components separately. In this setting, one model would predict the coefficient vector, while another (or a shared architecture) would learn the correction operator defining the multiscale basis. The training objective could combine multiple terms: for the coefficient prediction, an energy-based loss together with a fine-scale reconstruction loss; and for the learned basis, a supervised loss (when reference correction operators are available) augmented by a fine-scale reconstruction loss to ensure consistency of the resulting solution. However, a key limitation of this approach lies in the cost of data generation. Constructing a sufficiently large and diverse dataset of correction operators Q_h^ℓ may require solving a large number of localized fine-scale problems, which can be computationally expensive. In practice, generating thousands of such operators could become a bottleneck, potentially limiting the applicability of this approach and the performance gain in terms of computational time.

Furthermore, since LOD methods are available for a wide range of problem classes, including elliptic eigenvalue problems and parabolic equations, extending LOD-MSNO to these settings constitutes another promising avenue for future work.

Impact Statement

This paper presents work to make advancements in the field of Scientific Machine Learning, particularly for the numerical solution of multiscale partial differential equation problems. By combining established numerical analysis techniques with data-driven models, the proposed methods aim to improve computational efficiency and accessibility of high-fidelity simulations in scientific and engineering applications. The authors do not anticipate any immediate or unique negative societal impacts arising from this work.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd (IO250307-12279-01), by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [RS-2021-II211341, Artificial Intelligence Graduate School Program (Chung-Ang University)] and by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2025-02303239).

References

Bear, J. *Dynamics of Fluids in Porous Media*. Dover Civil and Mechanical Engineering Series. Dover, 1988. ISBN 9780486656755. URL https://books.google.co.kr/books?id=-XEaxd3hGzoC.

Blanc, X. and Le Bris, C. *Homogenization Theory for Multiscale Problems: An introduction*. MS&A, 21. Springer Nature Switzerland, 2023. ISBN 978-3-031-21832-3. doi: 10.1007/978-3-031-21833-0.

Bonizzoni, F. and Nobile, F. Perturbation analysis for the Darcy problem with log-normal permeability. *SIAM/ASA J. Uncertain. Quantif.*, 2(1):223–244, 2014. ISSN 2166-2525. doi: 10.1137/130949415. URL https://doi.org/10.1137/130949415.

Choi, J., Kim, N., and Hong, Y. Unsupervised legendre-galerkin neural network for solving partial differential equations. *IEEE Access*, PP:1–1, 01 2023. doi: 10.1109/ACCESS.2023.3244681.

Choi, J., Yun, T., Kim, N., and Hong, Y. Spectral operator learning for parametric pdes without data reliance. *Computer Methods in Applied Mechanics and Engineering*, 420:116678, 02 2024. doi: 10.1016/j.cma.2023.116678.

C.Y.Peng, G., Alber, M., Tepole, A. B., Cannon, W. R., De, S., Dura-Bernal, S., Garikipati, K., Karniadakis, G., Lytton, W. W., Perdikaris, P., Petzold, L., and Kuhl, E. Multiscale modeling meets machine learning: what can we learn? *Archives of computational methods in engineering*, 28.3:1017–1037, 2021. URL https://link.springer.com/article/10.1007/s11831-020-09405-5.

Engwer, C., Henning, P., Målqvist, A., and Peterseim, D. Efficient implementation of the localized orthogonal decomposition method. *Computer Methods in Applied Mechanics and Engineering*, 350:123–153, 2019. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2019.02.040. URL https://www.sciencedirect.com/science/article/pii/S0045782519301112.

Fanaskov, V. S. and Oseledets, I. V. Spectral neural operators. In *Doklady Mathematics*, pp. S226–S232. Springer, 2023.

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., and Yacalis, G. Could machine learning break the convection parameterization deadlock? *Geophysical Rresearch Letters*, 45.11:5742–5751, 2018. URL https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2018GL078202.

Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E. Physics-informed deep neural operator networks. *ArXiv*, abs/2207.05748, 2022. URL https://api.semanticscholar.org/CorpusID:250626955.

Kalina, K. A., Linden, L., Brummund, J., and Kastner, M. Feann: an efficient data-driven multiscale approach based on physics-constrained neural networks and automated data mining. *Computational Mechanics*, 71.5:827–851, 2023. URL https://link.springer.com/article/10.1007/s00466-022-02260-0.

Kirkham, M. Chapter 22 - electrical analogs for water movement through the soil-plant-atmosphere continuum. In *Principles of Soil and Plant Water Relations (Third Edition)*, pp. 431–449. Academic Press, third edition edition, 2023. ISBN 978-0-323-95641-3. doi: https://doi.org/10.1016/B978-0-323-95641-3.00030-1. URL https://www.sciencedirect.com/science/article/pii/B9780323956413000301.

Lee, J. Y., Cho, S. W., and Hwang, H. J. Hyperdeeponet: learning operator with complex target function space using the limited resources via hypernetwork, 2023. URL https://arxiv.org/abs/2312.15949.

Lee, J. Y., Ko, S., and Hong, Y. Finite element operator network for solving elliptic-type parametric pdes. *SIAM Journal on Scientific Computing*, 47(2):C501–C528, 2025. doi: 10.1137/23M1623707. URL https://doi.org/10.1137/23M1623707.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier

neural operator for parametric partial differential equations, 2021. URL https://arxiv.org/abs/2010.08895.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM / IMS J. Data Sci.*, 1(3), May 2024. doi: 10.1145/3648506. URL https://doi.org/10.1145/3648506.

Liu, R., Lehman, J., Molino, P., Such, F. P., Frank, E., Sergeev, A., and Yosinski, J. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 9628–9639, Red Hook, NY, USA, 2018. Curran Associates Inc.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 03 2021a. doi: 10.1038/s42256-021-00302-5.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021b. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5. URL http://dx.doi.org/10.1038/s42256-021-00302-5.

Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.114778. URL https://www.sciencedirect.com/science/article/pii/S0045782522001207.

Målqvist, A. and Peterseim, D. Localization of elliptic multiscale problems. *Math. Comput.*, 83:2583–2603, 2011. URL https://api.semanticscholar.org/CorpusID:267820598.

Målqvist, A. and Peterseim, D. *Numerical Homogenization by Localized Orthogonal Decomposition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020. doi: 10.1137/1.9781611976458. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611976458.

Pavliotis, G. A. and Stuart, A. M. *Multiscale Methods: Averaging and Homogenization*, volume 53 of *Texts in Applied Mathematics*. Springer Science & Business Media, New York, 2008. ISBN 978-0-387-73828-4.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-no: U-shaped neural operators, 2023. URL https://arxiv.org/abs/2204.11127.

Raonić, B., Molinaro, R., De Ryck, T., Rohner, T., Bartolucci, F., Alaifari, R., Mishra, S., and de Bézenac, E. Convolutional neural operators for robust and accurate learning of pdes. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Vinuesa, Ricardo, , and Brunton, S. L. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2.6:358–366, 2022. URL https://www.nature.com/articles/s43588-022-00264-7.

Wu, H., Luo, H., Wang, H., Wang, J., and Long, M. Transolver: a fast transformer solver for pdes on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Wu, H., Luo, H., Wang, H., Wang, J., and Long, M. Transolver: a fast transformer solver for pdes on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024b.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Yang, F.-B. and Huang, J.-P. *Convective Heat Transfer in Porous Materials*, pp. 129–143. Springer Nature Singapore, Singapore, 2024. ISBN 978-981-97-0487-3. doi: 10.1007/978-981-97-0487-3_7. URL https://doi.org/10.1007/978-981-97-0487-3_7.

Yizheng, W., Li, X., Yan, Z., Ma, S., Bai, J., Liu, B., Zhuang, X., Rabczuk, T., and Liu, Y. A pretraining-finetuning computational framework for material homogenization. *International Journal of Mechanical Sciences*, 314:111388, 2026. URL https://www.sciencedirect.com/science/article/abs/pii/S0020740326002444.

A. Notations

The list of notations used throughout the paper is provided in Table 3.

Table 3. Notations

Notation	Meaning
D	physical domain of the Darcy problem
Ω	parameter/probability domain for coefficient realizations and training samples
x	physical variable in D
ω	parameter / random input variable in Ω
κ	heterogeneous permeability coefficient in the Darcy problem
u	exact solution of (1)
u_h	fine-scale finite element approximation of u
u_{LOD}	LOD approximation / reference multiscale solution
$u_{\mathrm{LOD}}^{E,M}$	learned LOD solution by minizing the empirical loss funtion
$\mathcal{N}^{\ell}(T)$	ℓ-layer coarse patch around the coarse element T
$\boldsymbol{u}_{\mathrm{LOD}}$	coefficient vector of u_{LOD} in the basis $\{\Psi_H^i\}$
$\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}$	population minimizer for the energy-based loss
$\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}$	empirical minimizer for the energy-based loss
$\boldsymbol{A}_h,\ \boldsymbol{f}_h$	fine-scale stiffness matrix and load vector
$\boldsymbol{A}_{\mathrm{LOD}},\ \boldsymbol{f}_{\mathrm{LOD}}$	LOD stiffness matrix and LOD load vector
h	fine mesh size
H	coarse mesh size
$N_h,\ N_H$	numbers of fine and coarse degrees of freedom (or basis functions/nodes, depending on context)
M	number of training samples used in the empirical loss
$\mathcal{N}$	neural network class used to approximate the LOD coefficients
$\mathcal{G}^E$	function class associated with the energy-based loss
$R_M(\mathcal{G}^E)$	empirical Rademacher complexity of $\mathcal{G}^E$
$\mathbb{N},\ \mathbb{R}$	sets of natural and real numbers

B. Details on the LOD method

Interpolation operator. In order to introduce an A-orthogonal decomposition of V, we require a projection operator

$$\mathcal{I}_H : V \to V_H,$$

which maps fine-scale functions to the coarse finite element space. The operator $\mathcal{I}_H$ is assumed to be a projection in the algebraic sense, i.e.,

$$\mathcal{I}_H \circ \mathcal{I}_H = \mathcal{I}_H,$$

and to satisfy appropriate stability properties, see (Målqvist & Peterseim, 2020) for details. The purpose of this projection is to characterize the fine-scale "detail space" $\ker(\mathcal{I}_H)$ and thereby enable a systematic separation of coarse and fine scales. Several realizations of such a projection are possible. Typical examples include the global L^2-orthogonal projection onto the coarse finite element space V_H, defined by

$$(\mathcal{I}_H v, \Phi_H)_{L^2(\Omega)} = (v, \Phi_H)_{L^2(\Omega)} \quad \text{for all } \Phi_H \in V_H,$$

as well as locally defined L^2-projections based on nodal patches. These local constructions exploit projections onto polynomial spaces restricted to coarse neighborhoods and lead to operators that satisfy the required stability properties on quasi-uniform meshes. In contrast, nodal interpolation operators generally fail to provide sufficient stability in the H^1-norm and are therefore not suitable for the Localized Orthogonal Decomposition framework; see (Målqvist & Peterseim, 2020) for a detailed discussion of admissible choices.

Computation of the correction operator. Based on the projection $\mathcal{I}_H$, the Localized Orthogonal Decomposition method introduces a correction operator Q that accounts for the unresolved fine-scale features. For each coarse element $T \in \mathcal{T}_H$, we associate an element correction operator

$$Q_T : V \to W,$$

where $W := \ker(\mathcal{I}_H)$. The operator Q_T is defined by requiring that $Q_T v \in W$ solves the local variational problem

$$a(Q_T v, w) = \int_T \kappa \nabla v \cdot \nabla w \, dx \quad \text{for all } w \in W.$$

Using these local contributions, the global correction operator can be written as

$$Qv = \sum_{T \in \mathcal{T}_H} Q_T v.$$

This decomposition enables an element-wise and thus parallel computation of the correction operator. Moreover, it can be shown that the element correctors decay exponentially away from the corresponding coarse element, which motivates the introduction of localized approximations.

To this end, for a given subset $S \subset \Omega$, we define a sequence of patches recursively by

$$\begin{aligned} &\mathcal{N}^0(S) := S, \\ &\mathcal{N}^\ell(S) := \bigcup \left\{ T \in \mathcal{T}_H \;\middle|\; T \cap \mathcal{N}^{\ell-1}(S) \neq \emptyset \right\}, \quad \ell \in \mathbb{N}. \end{aligned}$$

Based on these patches, we define localized detail spaces $W^\ell(T) \subset W$ consisting of functions supported in $\mathcal{N}^\ell(T)$ and extended by zero outside this region. The localized element corrector

$$Q_T^\ell : V \to W^\ell(T)$$

is then defined as the solution of

$$a(Q_T^\ell v, w) = \int_T \kappa \nabla v \cdot \nabla w \, dx \quad \text{for all } w \in W^\ell(T).$$

For practical computations, all spaces are discretized using the fine-scale finite element space V_h. For a given $v_h \in V_h$, the fully discrete localized element corrector

$$Q_{T,h}^\ell(v_h) \in W^\ell(T) \cap V_h$$

is defined as the solution of

$$a\big(Q_{T,h}^\ell(v_h), w_h\big) = \int_T \kappa \nabla v_h \cdot \nabla w_h \, dx \quad \text{for all } w_h \in W^\ell(T) \cap V_h.$$

The corresponding discrete localized correction operator is then given by

$$Q_h^\ell := \sum_{T \in \mathcal{T}_H} Q_{T,h}^\ell.$$

Finally, the fully discrete Localized Orthogonal Decomposition method reads as follows: given $f \in L^2(D)$, find $u_{H,\ell,h}^{\mathrm{ms}} \in V_{H,\ell}^{\mathrm{ms},h}$ such that

$$a(u_{H,\ell,h}^{\mathrm{ms}}, v) = \int_D f v \, dx \quad \text{for all } v \in V_{H,\ell}^{\mathrm{ms},h},$$

where the multiscale space is defined by

$$V_{H,\ell}^{\mathrm{ms},h} := \operatorname{span} \left\{ \Phi_H^i - Q_h^\ell(\Phi_H^i) \;\middle|\; Z_i \in \mathcal{N}_H \right\}.$$

This space has the same dimension as the coarse finite element space V_H, and the method can be interpreted as a Galerkin approximation of the original problem in a problem-adapted multiscale space. Both the correction operator Q_h^ℓ and the final solution $u_{H,\ell,h}^{\mathrm{ms}}$ can be obtained by assembling and solving the corresponding linear systems; see (Engwer et al., 2019) for implementation details.

C. Theoretical Analysis

C.1. SPD-property of the LOD-stiffness matrix

Lemma C.1. *Let $D \subset \mathbb{R}^d$ be the physical domain and let Ω be the input domain of the neural network. Assume that $A(x,\omega) \in \mathbb{R}^{d\times d}$ is symmetric and uniformly elliptic, i.e.,*

$$\alpha|\xi|^2 \leq \xi^T A(x,\omega)\xi \leq \beta|\xi|^2, \quad \forall \xi \in \mathbb{R}^d,$$

for a.e. $x \in D$ and a.e. $\omega \in \Omega$. For each fixed $\omega \in \Omega$, define the bilinear form

$$a_\omega(u,v) := \int_D (A(x,\omega)\nabla u)\cdot \nabla v \, dx,$$

and let $A_H(\omega)$ be the coarse-scale stiffness matrix associated with a_ω. Let $V_{\mathrm{ms}}(\omega)$ be the matrix that represents the change of basis from the nodal basis functions of V_H to the LOD-corrected basis and define

$$A_{\mathrm{LOD}}(\omega) := V_{\mathrm{ms}}(\omega)\, A_h(\omega)\, V_{\mathrm{ms}}(\omega)^T.$$

Then, for a.e. $\omega \in \Omega$, the matrix $A_{\mathrm{LOD}}(\omega)$ is symmetric positive definite.

Proof. We first establish that the bilinear form a_ω is coercive on $V = H^1(D)$. Fix $\omega \in \Omega$ and let $u \in H_0^1(D)$. The Poincaré inequality gives:

$$||u||_{L^2(D)} \leq C_p ||\nabla u||_{L^2(D)}$$

Therefore:

$$\begin{aligned}\|u\|^2_{H^1(D)} &= \|u\|^2_{L^2(D)} + \|\nabla u\|^2_{L^2(D)} \\ &\leq (C_p^2+1)\|\nabla u\|^2_{L^2(D)}.\end{aligned}$$

Using this and the uniform ellipticity of A we obtain

$$\begin{aligned}a_\omega(u,u) &= \int_D (A(x,\omega)\nabla u)\cdot\nabla u\, dx \\ &\geq \alpha\|\nabla u\|^2_{L^2(D)} \geq \frac{\alpha}{C_p^2+1}\|u\|^2_{H^1(D)}.\end{aligned}$$

Next, the coarse-scale stiffness matrix $A_h(\omega)$ associated with a_ω, defined by

$$(A_H(\omega))_{ij} = a_\omega(\Phi_i, \Phi_j)$$

is positive-definite: Let $c \neq 0$ and write $v_H \in V_H$ as $v_h = \sum_i c_i\varphi_i$ in the nodal basis. Then $v_H \neq 0$, and therefore

$$c^T A_h(\omega) c = a_\omega(v_h, v_h) > 0$$

The LOD construction gives the multiscale space

$$V_{\mathrm{ms}}(\omega) = (1 - \mathcal{C}^\ell(\omega))V_H,$$

and the map $(1-\mathcal{C}^\ell(\omega)) : V_H \to V_{\mathrm{ms}}(\omega)$ is bijective. Therefore, the associated change-of-basis matrix $B_{\mathrm{ms}}(\omega)$ has full rank. Let $x \neq 0$ and set

$$y := B_{\mathrm{ms}}(\omega)^T x.$$

Since $B_{\mathrm{ms}}(\omega)$ has full rank, we have $y \neq 0$. Thus

$$x^T A_{\mathrm{LOD}}(\omega) x = x^T B_{\mathrm{ms}}(\omega) A_H(\omega) B_{\mathrm{ms}}(\omega)^T x = y^T A_H(\omega) y > 0.$$

Symmetry follows from the symmetry of $A_h(\omega)$. □

C.2. Approximation error for the energy-based loss function

Lemma C.2 (Uniform norm equivalence)**.** *Denote with $A_{\mathrm{LOD}}(\omega)$ the LOD-stiffness matrix and let $\lambda_{\min}(\omega)$ and $\lambda_{\max}(\omega)$ denote its smallest and largest eigenvalues. Then for every measurable $e : \Omega \to \mathbb{R}^{N_H}$ we have,*

$$\sqrt{\lambda_{\min}(\omega)}\,|e(\omega)| \le |A_{\mathrm{LOD}}(\omega)^{1/2} e(\omega)| \le \sqrt{\lambda_{\max}(\omega)}\,|e(\omega)|$$

for a.e. $\omega \in \Omega$. Consequently,

$$\|A_{\mathrm{LOD}}^{1/2} e\|_{L^\infty(\Omega)} \le \|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \|e\|_{L^\infty(\Omega)},$$

and

$$\|e\|_{L^\infty(\Omega)} \le \left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)} \|A_{\mathrm{LOD}}^{1/2} e\|_{L^\infty(\Omega)}.$$

Proof. Since $A_{\mathrm{LOD}}(\omega)$ is symmetric positive definite, it admits an orthonormal eigenbasis. Hence, for any $z \in \mathbb{R}^{N_H}$,

$$\lambda_{\min}(\omega)|z|^2 \le z^T A_{\mathrm{LOD}}(\omega) z = |A_{\mathrm{LOD}}(\omega)^{1/2} z|^2 \le \lambda_{\max}(\omega)|z|^2.$$

Taking $z = e(\omega)$ and then square roots yields

$$\sqrt{\lambda_{\min}(\omega)}\,|e(\omega)| \le |A_{\mathrm{LOD}}(\omega)^{1/2} e(\omega)| \le \sqrt{\lambda_{\max}(\omega)}\,|e(\omega)|.$$

Taking essential suprema over $\omega \in \Omega$ gives the two L^∞ estimates. □

Lemma C.3 (Approximation error for the energy-based loss)**.** *Denote by $\lambda_{\min}(\omega)$ and $\lambda_{\max}(\omega)$ the smallest and biggest eigenvalues of $\boldsymbol{A}_{\mathrm{LOD}}$ respectively. Assume that there exists a positive constant $c_0 > 0$ such that*

$$\operatorname*{ess\,inf}_{\omega \in \Omega} \lambda_{\min}(\omega) \ge c_0 > 0.$$

Denote by $\boldsymbol{u}_{\mathrm{LOD}} \colon \Omega \longrightarrow \mathbb{R}^{N_H}$ the true coefficient map of the LOD approximation, i.e. the solution of the linear system (11) for a given $\omega \in \Omega$ Let $\mathcal{N}$ be any neural network class, and define

$$\boldsymbol{u}_{\mathrm{LOD}}^E \in \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{N}} \mathcal{L}_E(\boldsymbol{u}).$$

Then

$$\|\boldsymbol{u}_{\mathrm{LOD}} - \boldsymbol{u}_{\mathrm{LOD}}^E\|_{L^1(\Omega)} \le c_0^{-1/2} \inf_{\boldsymbol{u} \in \mathcal{N}} \|A_{\mathrm{LOD}}^{1/2}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.$$

If, in addition,

$$\|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} := c_1 < \infty,$$

for some $c_1 > 0$, then

$$\|\boldsymbol{u}_{\mathrm{LOD}} - \boldsymbol{u}_{\mathrm{LOD}}^E\|_{L^1(\Omega)} \le \frac{c_1}{\sqrt{c_0}} \inf_{\boldsymbol{u} \in \mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}$$

Proof. Set

$$e(\omega) := \boldsymbol{u}_{\mathrm{LOD}}(\omega) - \boldsymbol{u}_{\mathrm{LOD}}^E(\omega).$$

By the uniform norm equivalence lemma,

$$|e(\omega)| \le \frac{|A_{\mathrm{LOD}}^{1/2}(\omega) e(\omega)|}{\sqrt{\lambda_{\min}(\omega)}}$$

for a.e. $\omega \in \Omega$. Integrating over Ω gives

$$\|\boldsymbol{u}_{\mathrm{LOD}} - \boldsymbol{u}_{\mathrm{LOD}}^E\|_{L^1(\Omega)} \le \left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)} \|A_{\mathrm{LOD}}^{1/2}(\boldsymbol{u}_{\mathrm{LOD}} - \boldsymbol{u}_{\mathrm{LOD}}^E)\|_{L^1(\Omega)}.$$

Since $\boldsymbol{u}_{\mathrm{LOD}}^E$ minimizes $\mathcal{L}_E$ over $\mathcal{N}$,

$$\|A_{\mathrm{LOD}}^{1/2}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)} = \inf_{\boldsymbol{u} \in \mathcal{N}} \mathcal{L}_E(\boldsymbol{u}).$$

Combining the two estimates yields

$$\|\boldsymbol{u}_{\mathrm{LOD}} - \boldsymbol{u}_{\mathrm{LOD}}^{E}\|_{L^1(\Omega)} \leq \left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)} \inf_{\boldsymbol{u}\in\mathcal{N}} \|A_{\mathrm{LOD}}^{1/2}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.$$

The bound with $c_0^{-1/2}$ follows from

$$\left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)} \leq c_0^{-1/2}.$$

For the final estimate, use the upper spectral bound

$$|A_{\mathrm{LOD}}(\omega)^{1/2} v| \leq \sqrt{\lambda_{\max}(\omega)}\, |v|$$

with $v = \boldsymbol{u}_{\mathrm{LOD}}(\omega) - \boldsymbol{u}_{\mathrm{LOD}}^{E}(\omega)$ to get

$$\|A_{\mathrm{LOD}}^{1/2}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)} \leq \|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.$$

Substituting this into the previous bound gives

$$\|\boldsymbol{u}_{\mathrm{LOD}} - \boldsymbol{u}_{\mathrm{LOD}}^{E}\|_{L^1(\Omega)} \leq \left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)} \|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.$$

□

C.3. Generalization error for the energy-based loss function

Lemma C.4 (Generalization error for the energy-based loss)**.** *Assume that*

$$\operatorname*{ess\,inf}_{\omega\in\Omega} \lambda_{\min}(\omega) \geq c_0 > 0, \qquad \operatorname*{ess\,sup}_{\omega\in\Omega} \lambda_{\max}(\omega) \leq c_1 < \infty.$$

Let $\omega_1, \ldots, \omega_M$ be i.i.d. samples in Ω. Define the empirical energy loss as the Monte-Carlo approximation of the energy loss

$$\mathcal{L}_{E,M}(\boldsymbol{u}) := \frac{|\Omega|}{M} \sum_{i=1}^{M} \left| \boldsymbol{A}_{\mathrm{LOD}}(\omega_i)^{1/2} \big(\boldsymbol{u}_{\mathrm{LOD}}(\omega_i) - \boldsymbol{u}(\omega_i)\big) \right|.$$

and let

$$\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M} \in \operatorname*{argmin}_{\boldsymbol{u}\in\mathcal{N}} \mathcal{L}_{E,M}(\boldsymbol{u})$$

be a minimizer of the empirical energy loss function. Define the function class associated with the Neural Network class $\mathcal{N}$ and the energy loss

$$\mathcal{G}^E := \Big\{ \omega \mapsto \left| A_{\mathrm{LOD}}^{1/2}(\omega) \big(\boldsymbol{u}_{\mathrm{LOD}}(\omega) - \boldsymbol{u}(\omega)\big) \right| : \ \boldsymbol{u} \in \mathcal{N} \Big\}.$$

Then

$$\begin{aligned}
&\mathbb{E}\left[\left\| \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M} \right\|_{L^1(\Omega)} \right] \\
&\lesssim \left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)} R_M(\mathcal{G}^E) \\
&\quad + \left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)} \|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.
\end{aligned}$$

Here, $R_M(\mathcal{G}^E)$ denotes the empirical Rademacher complexity of the function class $\mathcal{G}^E$ and the expectation is taken with respect to the random training samples $\omega_1, \ldots, \omega_M$.

Proof. Set

$$C_{\min} := \left\| \frac{1}{\sqrt{\lambda_{\min}}} \right\|_{L^\infty(\Omega)}.$$

By the lower spectral bound,

$$|v(\omega)| \leq \frac{|A_{\mathrm{LOD}}^{1/2}(\omega) v(\omega)|}{\sqrt{\lambda_{\min}(\omega)}}$$

for a.e. $\omega \in \Omega$. Hence

$$\begin{aligned}
&\mathbb{E}\left[\left\| \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M} \right\|_{L^1(\Omega)} \right] \\
&\leq C_{\min}\, \mathbb{E}\left[\left\| A_{\mathrm{LOD}}^{1/2} \big(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M} \big) \right\|_{L^1(\Omega)} \right] \\
&\leq C_{\min}\, \mathbb{E}\left[\left\| A_{\mathrm{LOD}}^{1/2} (\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \boldsymbol{u}_{\mathrm{LOD}}) \right\|_{L^1(\Omega)} \right] \\
&\quad + C_{\min}\, \mathbb{E}\left[\left\| A_{\mathrm{LOD}}^{1/2} (\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M} - \boldsymbol{u}_{\mathrm{LOD}}) \right\|_{L^1(\Omega)} \right] \\
&= C_{\min}\, \mathbb{E}\Big[\mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}) + \mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) \Big].
\end{aligned}$$

Since $\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}$ minimizes the population loss,

$$\mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}) \leq \mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}),$$

and therefore

$$\mathbb{E}\left[\left\| \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M} \right\|_{L^1(\Omega)} \right] \leq 2 C_{\min}\, \mathbb{E}\Big[\mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) \Big].$$

Now,

$$\begin{aligned}
\mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) &= \Big(\mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) - \mathcal{L}_{E,M}(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) \Big) + \mathcal{L}_{E,M}(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) \\
&\leq \sup_{\boldsymbol{u} \in \mathcal{N}} \big| \mathcal{L}_E(\boldsymbol{u}) - \mathcal{L}_{E,M}(\boldsymbol{u}) \big| + \mathcal{L}_{E,M}(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) \\
&\leq \sup_{\boldsymbol{u} \in \mathcal{N}} \big| \mathcal{L}_E(\boldsymbol{u}) - \mathcal{L}_{E,M}(\boldsymbol{u}) \big| + \mathcal{L}_{E,M}(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}) \\
&= \sup_{\boldsymbol{u} \in \mathcal{N}} \big| \mathcal{L}_E(\boldsymbol{u}) - \mathcal{L}_{E,M}(\boldsymbol{u}) \big| + \Big(\mathcal{L}_{E,M}(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}) - \mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}) \Big) + \mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}) \\
&\leq 2 \sup_{\boldsymbol{u} \in \mathcal{N}} \big| \mathcal{L}_E(\boldsymbol{u}) - \mathcal{L}_{E,M}(\boldsymbol{u}) \big| \\
&\quad + \inf_{\boldsymbol{u} \in \mathcal{N}} \mathcal{L}_E(\boldsymbol{u}).
\end{aligned}$$

Taking expectation and using the fact that the Rademacher complexity serves as a measure between the error of population risk minimizer and empirical rsik minimizer, we obtain:

$$\mathbb{E}\left[\sup_{\boldsymbol{u} \in \mathcal{N}} \big| \mathcal{L}_E(\boldsymbol{u}) - \mathcal{L}_{E,M}(\boldsymbol{u}) \big| \right] \lesssim R_M(\mathcal{G}^E),$$

we obtain

$$\mathbb{E}\Big[\mathcal{L}_E(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}) \Big] \lesssim R_M(\mathcal{G}^E) + \inf_{\boldsymbol{u} \in \mathcal{N}} \mathcal{L}_E(\boldsymbol{u}).$$

Substituting this into the previous estimate yields

$$\mathbb{E}\left[\left\| \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M} \right\|_{L^1(\Omega)} \right] \lesssim C_{\min} \left(R_M(\mathcal{G}^E) + \inf_{\boldsymbol{u} \in \mathcal{N}} \mathcal{L}_E(\boldsymbol{u}) \right).$$

Finally, by the upper spectral bound,

$$\mathcal{L}_E(\boldsymbol{u}) = \|A_{\mathrm{LOD}}^{1/2}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)} \leq \|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}$$

Therefore,

$$\inf_{\boldsymbol{u}\in\mathcal{N}} \mathcal{L}_E(\boldsymbol{u}) \leq \|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.$$

Combining the above bounds gives the claimed conclusion,

$$\begin{aligned}
&\mathbb{E}\left[\left\|\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}\right\|_{L^1(\Omega)}\right] \\
&\lesssim \left\|\frac{1}{\sqrt{\lambda_{\min}}}\right\|_{L^\infty(\Omega)} R_M(\mathcal{G}^E) \\
&\quad + \left\|\frac{1}{\sqrt{\lambda_{\min}}}\right\|_{L^\infty(\Omega)} \|\sqrt{\lambda_{\max}}\|_{L^\infty(\Omega)} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.
\end{aligned}$$

□

C.4. Convergence of the learned localized LOD solution

Theorem C.5. *Assume that $R_M(\mathcal{G}^E)$ converges to 0 as $M \to \infty$ and that there exits $\boldsymbol{u} \in \mathcal{N}$ such that $\|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}$ gets arbitrarily small.*

Define the reference LOD solution and the learned LOD approximation by

$$u_{\mathrm{LOD}} = \sum_{j=1}^{N_H} \boldsymbol{u}_{\mathrm{LOD}}^{(j)} \Psi_H^j$$

$$u_{\mathrm{LOD}}^{E,M}(x,\omega) := \sum_{j=1}^{N_H} \big(\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M(j)}(\omega)\big) \Psi_H^j(x,\omega).$$

Then we have

$$\lim_{M\to\infty} \mathbb{E}\Big[\|u_{\mathrm{LOD}} - u_{\mathrm{LOD}}^{E,M}\|_{L^1(\Omega;L^2(D))}\Big] = 0. \tag{3.12}$$

where the expectation is taken over the random sampling $\omega_j \sim \mathbb{P}_\Omega$.

Proof. From the definition, we have that

$$\begin{aligned}
\|u_{\mathrm{LOD}} - u_{\mathrm{LOD}}^{E,M}\|_{L^1(\Omega;L^2(D))} &= \int_\Omega \left\|\sum_{j=1}^{N_H} \big(\boldsymbol{u}_{\mathrm{LOD}}^{(j)} - (\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M(j)})\big) \Psi_H^j\right\|_{L^2(D)} d\omega \\
&\lesssim \int_\Omega \left(\sum_{j=1}^{N_H} \big|\boldsymbol{u}_{\mathrm{LOD}}^{(j)} - (\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M(j)})\big| \|\Psi_H^j\|_{L^2(D)}\right) d\omega \\
&\lesssim \max_{1\leq j\leq N_H} \|\Psi_H^j\|_{L^2(D)} \int_\Omega \left(\sum_{j=1}^{N_H} \big|\boldsymbol{u}_{\mathrm{LOD}}^{(j)} - (\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M(j)})\big|\right) d\omega \\
&\lesssim \max_{1\leq j\leq N_H} \|\Psi_H^j\|_{L^2(D)} \|\boldsymbol{u}_{\mathrm{LOD}}^{(j)} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M(j)}\|_{L^1(\Omega)}.
\end{aligned} \tag{3.13}$$

Therefore, for fixed $H > 0$, by $\boldsymbol{C.3}$ and $\boldsymbol{C.4}$, we can conclude that for $M \to \infty$,

$$\mathbb{E}\Big[\|u_{\mathrm{LOD}} - u_{\mathrm{LOD}}^{E,M}\|_{L^1(\Omega;L^2(D))}\Big] \lesssim \mathbb{E}\Big[\|\boldsymbol{u}_{\mathrm{LOD}} - \boldsymbol{u}_{\mathrm{LOD}}^{E,M}\|_{L^1(\Omega;L^2(D))}\Big] \lesssim \mathbb{E}\Big[\|\boldsymbol{u}_{\mathrm{LOD}} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}\|_{L^1(\Omega)}\Big] + \mathbb{E}\Big[\|\widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E,M}\|_{L^1(\Omega)}\Big] \to 0.$$

due to the assumptions that we made. □

C.5. Proof of Theorem 4.1

Proof. We decompose the total error as

$$u - \widehat{u}_{\mathrm{LOD}}^{E,M} = (u - u_{\mathrm{LOD}}) + (u_{\mathrm{LOD}} - \widehat{u}_{\mathrm{LOD}}^{E}) + (\widehat{u}_{\mathrm{LOD}}^{E} - \widehat{u}_{\mathrm{LOD}}^{E,M}).$$

Lemma C.3 yields

$$\|u - u_{\mathrm{LOD}}\|_{L^1(\Omega;L^2(D))} \lesssim H^2.$$

Lemma C.4 implies

$$\|\boldsymbol{u}_{\mathrm{LOD}} - \widehat{\boldsymbol{u}}_{\mathrm{LOD}}^{E}\|_{L^1(\Omega)} \leq \frac{c_1}{\sqrt{c_0}} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)},$$

while Lemma C.5 gives

$$\mathbb{E}[\|\widehat{u}_{\mathrm{LOD}}^{E} - \widehat{u}_{\mathrm{LOD}}^{E,M}\|_{L^1(\Omega)}] \lesssim \frac{1}{\sqrt{c_0}} R_M(\mathcal{G}^E) + \frac{c_1}{\sqrt{c_0}} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.$$

Combining the three terms proves

$$\mathbb{E}\Big[\|u - \widehat{u}_{\mathrm{LOD}}^{E,M}\|_{L^1(\Omega;L^2(D))}\Big] \lesssim H^2 + \frac{1}{\sqrt{c_0}} R_M(\mathcal{G}^E) + \frac{2c_1}{\sqrt{c_0}} \inf_{\boldsymbol{u}\in\mathcal{N}} \|(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{LOD}})\|_{L^1(\Omega)}.$$

□

D. Baseline models

D.1. Common input–output format and training losses

All baselines learn the Darcy solution operator on a regular $s \times s$ grid. The input is a coefficient field (and optional engineered channels), and the output is the scalar solution field. Concretely, the network input is a tensor in $\mathbb{R}^{B\times s\times s\times C_{\mathrm{in}}}$ (or equivalently $B \times C_{\mathrm{in}} \times s \times s$), and the output prediction is $u_\theta \in \mathbb{R}^{B\times s\times s}$. Here C_{in} is the number of input channels used in the experiment (denoted by `Cin` in our code).

For supervised training, we use the standard data loss

$$\mathcal{L}_{\mathrm{data}}(\theta) = \|u_\theta - u\|_2^2.$$

For physics-informed training of PI-DeepONet and PINO, we additionally enforce the Darcy equation

$$-\nabla \cdot \big(\kappa(x)\nabla u(x)\big) = f(x)$$

through coordinate-based automatic differentiation rather than a finite-volume/finite-difference stencil. Let

$$\mathcal{G} = \{x_{ij}\}_{i,j=1}^{s}$$

denote the regular grid points, and let $u_\theta(x;\kappa)$ be the network prediction evaluated at coordinate $x \in \Omega$. We define the pointwise residual by

$$r_\theta(x;\kappa) = -\partial_{x_1}(\kappa(x)\,\partial_{x_1} u_\theta(x;\kappa)) - \partial_{x_2}(\kappa(x)\,\partial_{x_2} u_\theta(x;\kappa)) - f(x).$$

In implementation, the spatial coordinates are treated as differentiable inputs, ∇u_θ is obtained by automatic differentiation with respect to those coordinates, and $\kappa(x)$ is evaluated from the raw coefficient field at the same coordinates by differentiable interpolation. The physics loss is then

$$\mathcal{L}_{\mathrm{pde}}(\theta) = \frac{1}{|\mathcal{G}|}\sum_{x\in\mathcal{G}} |r_\theta(x;\kappa)|^2, \qquad \mathcal{L}_{\mathrm{PI}}(\theta) = \mathcal{L}_{\mathrm{data}}(\theta) + \lambda\mathcal{L}_{\mathrm{pde}}(\theta).$$

D.2. Fourier neural operator (FNO)

The Fourier Neural Operator (FNO), introduced by Li et al. (Li et al., 2021), is a neural operator architecture for learning mappings between infinite-dimensional function spaces, with a particular focus on approximating solution operators of parametric partial differential equations. The key idea of the FNO is to parameterize an integral operator in Fourier space, enabling efficient global convolution and resolution-invariant generalization.

Given a function v_t defined on a spatial domain, the FNO constructs a learnable integral operator $\mathcal{K}(\phi)$ of the form

$$(\mathcal{K}(\phi)v_t)(x) = \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_\phi) \cdot (\mathcal{F}v_t)\right)(x),$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively, and κ_ϕ is a learnable kernel. This formulation corresponds to a global convolution in physical space and allows the operator to capture long-range dependencies efficiently.

In practice, the continuous Fourier transforms are replaced by discrete fast Fourier transforms. Specifically:

- The operators $\mathcal{F}$ and $\mathcal{F}^{-1}$ are replaced by the discrete fast Fourier transforms $\hat{\mathcal{F}}$ and $\hat{\mathcal{F}}^{-1}$.
- The input $\hat{v} \in \mathbb{R}^{s_1 \times \cdots \times s_d \times n}$ represents the evaluation of the function v on a uniform grid with $s_j \in \mathbb{N}$ points in each spatial dimension.
- The Fourier transform of the kernel $\mathcal{F}(\kappa)$ is replaced by a complex-valued weight tensor $\boldsymbol{T} \in \mathbb{C}^{s_1 \times \cdots \times s_d \times n \times n}$, whose entries correspond to the learnable Fourier modes of the kernel.

To reduce computational complexity and improve generalization, the FNO typically employs a low-frequency truncation in Fourier space. Concretely, only a prescribed number of low-frequency modes are retained in each spatial dimension, while the remaining high-frequency modes are set to zero. With this truncation, one Fourier layer of the FNO is given by

$$\hat{v} \;\mapsto\; \hat{\mathcal{F}}^{-1}\left(\boldsymbol{T}_{\text{low}} \cdot \hat{\mathcal{F}}(\hat{v})\right),$$

where $\boldsymbol{T}_{\text{low}}$ denotes the restriction of $\boldsymbol{T}$ to the retained low-frequency Fourier modes. By stacking multiple such Fourier layers and interleaving them with pointwise nonlinearities, the Fourier Neural Operator is able to approximate nonlinear operators with global receptive fields while remaining computationally efficient.

Experimental configuration. In our experiments, the FNO uses 4 Fourier layers with channel width 64, retaining 12 low-frequency modes per spatial dimension in each spectral convolution layer.

D.3. Convolutional neural operator (CNO)

The Convolutional Neural Operator (CNO) (Raonić et al., 2023) is designed to approximate operators $\mathbf{G} : \mathcal{B}_w(D, \mathbb{R}^{d_X}) \to \mathcal{B}_w(D, \mathbb{R}^{d_Y})$ between band-limited function spaces. By restricting both the input and output to band-limited spaces, the CNO mitigates aliasing errors and enables stable learning across different spatial resolutions.

The CNO represents the target operator as a composition of lifting, convolutional operator layers, and projection, expressed as

$$\mathcal{G} : u \mapsto P(u) = v_0 \mapsto v_1 \mapsto \cdots \mapsto v_L \mapsto Q(v_L) = \bar{u},$$

where P is a lifting operator that maps the input function u to a higher-dimensional feature representation, Q is a projection operator to the target space, and the intermediate states $\{v_\ell\}_{\ell=0}^{L}$ are defined recursively by

$$v_{\ell+1} = \mathcal{P}_\ell \circ \Sigma_\ell \circ \mathcal{K}_\ell(v_\ell), \qquad 1 \le \ell \le L-1.$$

Here, $\mathcal{K}_\ell$ denotes a convolutional operator, Σ_ℓ a pointwise nonlinearity, and $\mathcal{P}_\ell$ a resolution-changing operator.

Architecturally, the CNO adopts a modified operator U-Net structure composed of band-preserving building blocks. In the encoder, the lifted representation v_0 is processed by a sequence of downsampling blocks (D-blocks), which progressively decrease the spatial resolution while increasing the number of channels. This operation expands the feature width and

contracts the spatial height, enabling multiscale feature extraction. The downsampled representations are then processed by resolution-invariant residual blocks (R-blocks), which take the form

$$\mathbf{R}_w(v) = v + K_w^{(2)}\Big(\Sigma_{w\to w}\Big(K_w^{(1)}(v)\Big)\Big),$$

where $K_w^{(1)}$ and $K_w^{(2)}$ are convolutional operators and $\Sigma_{w\to w}$ denotes a band-preserving activation.

The decoder reconstructs high-resolution outputs by successively upsampling the feature maps while reducing the channel dimension. To compensate for the potential loss of high-frequency information caused by upsampling, feature maps from earlier encoder stages are concatenated with decoder features, ensuring the preservation of fine-scale details. Finally, the projection operator Q maps the decoded representation to the desired output dimension, yielding the approximation $\bar{u}$ of the target function.

Experimental configuration. We use a 4-level encoder–decoder CNO with base channel width 64 and 2 residual (R-)blocks at each resolution.

D.4. U-shaped neural operator (UNO)

The U-shaped Neural Operator (UNO) (Rahman et al., 2023) is a neural operator architecture that adopts a U-shaped structure to learn mappings between function spaces, sharing similarities with the Convolutional Neural Operator while differing in the functional roles of the encoder and decoder. In particular, the UNO explicitly models operator layers that act between function spaces defined on domains of varying resolution and channel width.

Each UNO layer is formulated as a map between function spaces

$$G_i : \left\{v_i : D_i \to \mathbb{R}^{d_{v_i}}\right\} \longrightarrow \left\{v_{i+1} : D_{i+1} \to \mathbb{R}^{d_{v_{i+1}}}\right\},$$

where both the spatial domain D_i and the channel dimension d_{v_i} may vary across layers. The architecture is organized into an encoder–decoder structure that progressively contracts and expands the domain while adjusting the channel width.

The encoder is responsible for shrinking the spatial domain and increasing the channel dimension, satisfying

$$\mu(D_i) \geq \mu(D_{i+1}), \qquad d_{v_{i+1}} \geq d_{v_i},$$

where $\mu(D_i)$ denotes the measure of the domain D_i. As a result, the encoder produces a sequence of feature functions defined on progressively smaller domains with richer channel representations. Conversely, the decoder performs the inverse operation by expanding the spatial domain and reducing the channel width in order to match the target output space, expressed as

$$\mu(D_i) \leq \mu(D_{i+1}), \qquad d_{v_{i+1}} \leq d_{v_i}.$$

By symmetrically combining these encoder and decoder mappings, the U-shaped Neural Operator enables multiscale operator learning, allowing information to be processed at different spatial resolutions while ultimately reconstructing outputs on the original domain.

Experimental configuration. We use an 4-stage UNO with base channel width 64 in the lifting layer and scale the width across encoder/decoder stages accordingly.

D.5. Deep operator network (DeepONet / PI-DeepONet)

Deep Operator Networks (DeepONets) (Lu et al., 2021a) learn nonlinear operators by combining two subnetworks: a branch net that encodes the input function and a trunk net that encodes the query location. For an input coefficient field κ and a spatial point $y \in \Omega$, DeepONet represents the solution operator in the separable form

$$u_\theta(y;\kappa) = \sum_{j=1}^{p} b_{\theta,j}(\kappa)\, t_{\theta,j}(y), \tag{14}$$

where $b_\theta(\kappa) \in \mathbb{R}^p$ is produced by the branch net and $t_\theta(y) \in \mathbb{R}^p$ is produced by the trunk net. Evaluating (14) over all grid points yields the full predicted field.

Physics-informed variants of DeepONet (often referred to as PI-DeepONet) incorporate the governing PDE as an additional training signal. In our experiments, we adopt a PI-DeepONet-style training of DeepONet by using the common PI objective in Section D.1, where the physics loss is computed by automatic differentiation with respect to the trunk coordinates.

In our implementation, the branch net takes as input the discretized coefficient field (stacked channels as described in Section D.1) and maps it to a latent code, while the trunk net takes the coordinate $y = (x_1, x_2)$. Their inner product yields the scalar solution value at y, and batching over all grid locations produces the full field prediction. For the physics loss, we differentiate the prediction with respect to y, form $\kappa(y)\nabla u_\theta(y;\kappa)$ using the coefficient sampled at the same coordinates, and penalize the residual

$$-\partial_{x_1}(\kappa(y)\partial_{x_1}u_\theta(y;\kappa)) - \partial_{x_2}(\kappa(y)\partial_{x_2}u_\theta(y;\kappa)) - f(y).$$

Experimental configuration. For the plain DeepONet baseline, we set the latent dimension to $p = 128$. Both the branch and trunk nets are MLPs of depth 4 and 4, respectively, with hidden width 128.

For the PI-DeepONet baseline, we use branch layers $(d_b, 100, 100, 100, 100)$ and trunk layers $(2, 100, 100, 100, 100)$, where $d_b = C_{\text{in}}s^2$ is the flattened branch input dimension.

D.6. Transolver (Physics-Attention)

Transolver (Wu et al., 2024a) is an operator learning architecture built on physics-attention, which compresses dense attention over grid tokens into a two-stage slicing–token mechanism. Let $x \in \mathbb{R}^{N\times d}$ denote the per-point features on a grid ($N = s^2$). Transolver first assigns each grid point to one of S latent slices by

$$A = \text{softmax}\left(\frac{W_s x}{\tau}\right) \in \mathbb{R}^{N\times S},$$

then forms slice tokens by normalized pooling,

$$T = \frac{A^\top x}{A^\top \mathbf{1}} \in \mathbb{R}^{S\times d}.$$

Multi-head self-attention is applied among the S tokens (cost $\mathcal{O}(S^2)$), and the updated tokens are projected back to the grid by deslicing, $\tilde{x} = AT'$. This reduces the quadratic cost in N to $\mathcal{O}(NS + S^2)$ while retaining global information flow, which is beneficial for elliptic problems where long-range dependencies are intrinsic.

Experimental configuration. We use feature dimension $d = 128$, number of slices $S = 64$, number of attention heads 8, and temperature $\tau = 0.5$ for the slicing softmax.

D.7. Physics-informed Neural Operator (PINO)

The Physics-Informed Neural Operator (PINO) combines an operator-learning backbone with a PDE residual during training. In our implementation, the PINO baseline uses an FNO-style 2D architecture on the regular $s \times s$ grid. Given the preprocessed coefficient tensor, we append the spatial coordinates to the input channels and form an input tensor in

$$\mathbb{R}^{B\times s\times s\times (C_{\text{in}}+2)}.$$

A lifting linear layer first maps these per-grid features to a hidden representation, after which a sequence of spectral convolution layers and pointwise linear layers is applied in parallel and summed at each block. Finally, a pointwise projection head maps the hidden representation back to a scalar solution field.

Concretely, if v_ℓ denotes the hidden feature map at layer ℓ, one PINO block takes the form

$$v_{\ell+1}(x) = \sigma(\mathcal{K}_\ell(v_\ell)(x) + W_\ell v_\ell(x)),$$

where $\mathcal{K}_\ell$ is a Fourier spectral convolution retaining only low-frequency modes and W_ℓ is a learnable pointwise linear map. Stacking such blocks yields a neural operator with a global receptive field and efficient spectral mixing.

For physics-informed training, PINO uses the same PI objective from Section D.1. In particular, the network prediction is differentiated with respect to the appended coordinates by automatic differentiation, and we penalize the Darcy residual

$$-\partial_{x_1}(\kappa(x)\partial_{x_1}u_\theta(x;\kappa)) - \partial_{x_2}(\kappa(x)\partial_{x_2}u_\theta(x;\kappa)) - f(x).$$

As in the PI-DeepONet baseline, the coefficient multiplying the gradient is obtained from the raw coefficient field at the same coordinates.

Experimental configuration. We use four spectral blocks, hidden width 64, retain 12 Fourier modes in each spatial dimension at every block, use a pointwise projection dimension of 128, GELU activations, and no domain padding.

D.8. Finite-Element Operator Network (FEONet)

FEONet is a neural operator model grounded in the finite element method (FEM), using nodal basis functions $\{\phi_i\}$ to approximate solutions as a linear combination

$$u = \sum_{i=1}^{N_H} \alpha_i \phi_i$$

where the index i denotes the nodes of a triangulation of the physical domain and N_H the number of (coarse) nodes. The basic idea of FEONet is to predict the FEM-coefficients $\{\alpha_i\}$. For the Darcy equation 1, the training procedure of FEONet is based on the weak formulation, which reads: Find $u \in V$ such that

$$\int_\Omega \kappa \nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx \quad \forall v \in V.$$

Using the FEM ansatz $u = \sum_j \alpha_j \phi_j$ and testing with ϕ_i, one obtains the linear system $A\alpha = b$ with the stiffness matrix $A = (A_{ij})$ and the load vector $f = (f_i)$ defined as

$$A_{ij} = \int_\Omega \kappa \nabla \phi_j \cdot \nabla \phi_i \, dx, \quad b_i = \int_\Omega f \phi_i \, dx.$$

for $i, j = 1, ..., N_H$. FEONet is then trained by sampling permeability fields κ_i and computing corresponding stiffness and load vector samples $\{(A_i(\kappa_i), b_i(\kappa_i))\}_{i=1}^N$ and then using the weak-form residual loss over samples with index i:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|A_i \alpha_i - b_i\|.$$

Note that, by using this loss function, we do not need the ground-truth FEM-coefficients α_i to train our model. This way, FEONet can be interpreted as a **hybrid** Operator Learning model that combines neural networks with the conventional FEM-method, which is **data-free** and **fully physics-informed**.

Experimental configuration. In our implementation of FEONet, we use a CNN-based feature extractor for the input κ, followed by a Multi-Layer Perceptron to predict the coefficients α_i based on these features. For comparability with LOD-MSNO, the input of the model is the permeability field κ on a grid with the fine resolution h and the output are the coefficients α_i for the coarse-scale-basis functions ϕ_i, $i = 1, ..., N_H$.

E. Neural architecture for predicting LOD coefficients

Task and I/O. Let $a(x) > 0$ denote a heterogeneous permeability field on a bounded domain with homogeneous Dirichlet boundary conditions. Our network takes a fine-grid discretization $a_h \in \mathbb{R}^{n_h \times n_h}$ and outputs the interior coefficient vector of the LOD basis on a coarse grid with n_H nodes per spatial direction. i.e., $\widehat{u}_{\text{int}} \in \mathbb{R}^D$ where $D = (n_H - 2)^2$.

We package preprocessing and (de)normalization inside a single model so that the same transforms are applied in both training and inference.

E.1. Input preprocessing

We form a multi-channel tensor $X \in \mathbb{R}^{B \times C_{\text{in}} \times n_h \times n_h}$ from the raw field a_h. Denote by b the *base channel*:

$$b(i, j) = \log(\max\{a_h(i, j), \varepsilon\}) \tag{15}$$

where $\varepsilon > 0$ is a small constant to avoid numerical issues. The log transform compresses the dynamic range of high-contrast permeability fields. This is consistent with the common modeling practice in Darcy-flow settings (Bonizzoni & Nobile, 2014).

Gradient-magnitude channel (`add_grad`). To expose sharp interfaces/discontinuities in a_h, we append a gradient-magnitude channel computed by forward differences (with zero padding at the boundary):

$$g_x(i,j) = \begin{cases} b(i,j+1) - b(i,j), & j < n_h - 1, \\ 0, & j = n_h - 1, \end{cases} \tag{16}$$

$$g_y(i,j) = \begin{cases} b(i+1,j) - b(i,j), & i < n_h - 1, \\ 0, & i = n_h - 1, \end{cases} \tag{17}$$

$$g(i,j) = \sqrt{g_x(i,j)^2 + g_y(i,j)^2 + \delta}, \tag{18}$$

with $\delta > 0$ small.

Coordinate channels (`add_coords`). Convolutional networks are translation-equivariant, while boundary-value problems on bounded domains are position-dependent. We therefore append explicit coordinates (CoordConv-style positional encoding (Liu et al., 2018)):

$$x(j) = -1 + \frac{2j}{n_h - 1}, \qquad y(i) = -1 + \frac{2i}{n_h - 1}, \tag{19}$$

Channel count. The number of input channels is

$$C_{\text{in}} \;=\; 1 \;+\; \mathbb{I}[\texttt{add_grad}] \;+\; 2\,\mathbb{I}[\texttt{add_coords}], \tag{20}$$

and the final preprocessed tensor is formed by concatenation along the channel dimension.

E.2. Normalization (input/output)

Channel-wise input normalization. Let $X \in \mathbb{R}^{B \times C_{\text{in}} \times n_h \times n_h}$ be the preprocessed input. We compute per-channel statistics (μ_c, σ_c) on the training set after preprocessing:

$$\mu_c \;=\; \mathbb{E}[X_c], \qquad \sigma_c^2 \;=\; \mathbb{E}[X_c^2] - \mu_c^2, \tag{21}$$

where expectations are taken over all training samples and spatial locations. We then normalize as

$$\widetilde{X}_{b,c,i,j} \;=\; \frac{X_{b,c,i,j} - \mu_c}{\sigma_c + \epsilon}. \tag{22}$$

The values μ, σ are stored as non-trainable buffers and reused at test time.

Per-component output normalization. Let $u_{\text{int}} \in \mathbb{R}^D$ denote the target interior coefficient vector. We compute per-component statistics $(\mu_k^{(u)}, \sigma_k^{(u)})$ on the training set and normalize:

$$\widetilde{u}_k \;=\; \frac{u_{\text{int},k} - \mu_k^{(u)}}{\sigma_k^{(u)} + \epsilon}. \tag{23}$$

The core network is trained to predict $\widehat{\widetilde{u}}$; inference returns the decoded coefficients $\widehat{u}_{\text{int}}$ via the inverse transform of (23).

E.3. Core CNN: `LODMimeticNet`

Building block: Conv–GroupNorm–GELU (GNAct). We use the block

$$\text{GNAct}(z) \;=\; \text{GELU}(\text{GN}(\text{Conv}_{k\times k,s}(z)))\,, \tag{24}$$

where GN denotes Group Normalization (Wu & He, 2018). GELU is a smooth activation commonly defined as $\text{GELU}(x) = x\Phi(x)$ with Φ the standard normal CDF. For GroupNorm, we use $G = \min(8, C)$ groups for a C-channel tensor.

Residual block (ResBlock2D). At a fixed resolution and channel width C, we use a ResNet-style block:

$$\text{ResBlock}(x) = \text{GELU}\left(x + \text{GN}\left(\text{Conv}_{3\times 3,1}\left(\text{GNAct}(x)\right)\right)\right) \tag{25}$$

Encoder: learned restriction from $n_h \times n_h$ to $n_H \times n_H$. Starting from $\widetilde{X} \in \mathbb{R}^{B\times C_{\text{in}}\times n_h\times n_h}$, we apply:

- **Stem (stride 1):** two GNAct blocks (kernel 3, padding 1): $C_{\text{in}} \to \texttt{base} \to \texttt{base}$.
- **Downsampling stages (stride 2):** $\log_2((n_h-1)/(n_H-1))$ stages, each consisting of one GNAct with stride 2 followed by one ResBlock:

$$n_h \times n_h \to \frac{n_h+1}{2} \times \frac{n_h+1}{2} \to \cdots \to n_H \times n_H$$
$$\texttt{base} \to 2\texttt{base} \to \cdots \to 2\frac{n_h-1}{n_H-1}\texttt{base}$$

Thus, the spatial resolution matches the coarse-grid node resolution (n_H, n_H).

Coarse-scale mixing and readout. At resolution $n_H \times n_H$, we apply:

- **FirstMix:** GNAct (stride 1) increasing channels followed by `first_mix_blocks` residual blocks.
- **Mixing trunk:** `mix_blocks` residual blocks at fixed width.
- **Head:** a 1×1 convolution to produce a scalar coarse-grid map $S \in \mathbb{R}^{B\times 1\times n_H\times n_H}$. We remove the channel dimension and extract the interior coefficients:

$$\widehat{\widetilde{u}}_{\text{int}} \;=\; \text{vec}\Big(S_{(:,\,1:n_H-1,\,1:n_H-1)}\Big) \in \mathbb{R}^{B\times D}, \tag{26}$$

  where the slicing $(1 : n_H - 1)$ removes the boundary nodes in each direction, consistent with homogeneous Dirichlet conditions.

Hyperparameters used. We use $\texttt{base} = 32$, GroupNorm with $G = \min(8, C)$ groups, $\texttt{add_grad} = \texttt{True}$, and $\texttt{add_coords} = \texttt{True}$. For the coarse-scale mixing, we set $\texttt{first_mix_blocks} = 1$ and $\texttt{mix_blocks} = 6$.

E.4. Data generation and dataset construction

Goal and supervised learning target. For each realization of a heterogeneous coefficient field κ, we generate a *LOD-reduced* coarse problem and its solution. Concretely, for every sample we compute

$$\big(\kappa_h,\ \mathbf{A}_{\text{LOD}},\ \mathbf{u}_{\text{LOD}}\big),$$

where κ_h denotes the coefficient sampled on the fine mesh $\mathcal{T}_h$ (stored as nodal values), $\mathbf{A}_{\text{LOD}}$ is the (interior) coarse LOD stiffness matrix and load vector, and $\mathbf{u}_{\text{LOD}}$ is the corresponding (interior) coarse LOD solution. This provides paired input–output data for learning an approximation of the parameter-to-solution map $\kappa \mapsto u$ at coarse-grid resolution.

Meshes and coefficient resolution. We generate two nested conforming triangulations $\mathcal{T}_H$ and $\mathcal{T}_h$ of $\Omega = (0,1)^2$ with $H = 2^{-H_{\text{exp}}}$ and $h = 2^{-h_{\text{exp}}}$, assuming that $\mathcal{T}_h$ is a refinement of $\mathcal{T}_H$. The fine mesh $\mathcal{T}_h$ is chosen sufficiently small to resolve the spatial variations of κ. All discrete corrector problems are solved on $V_h = P_1(\mathcal{T}_h) \cap H_0^1(\Omega)$.

Coefficient families. We consider several coefficient models κ :

- **Quantile (discontinuous/high contrast).** We first sample a smooth Gaussian random field g on the fine nodes with squared-exponential covariance,

$$\mathbb{E}\big[g(x)\,g(y)\big] = \sigma^2 \exp\Big(-\tfrac{\|x-y\|^2}{2\ell^2}\Big),$$

  and set $\kappa = \exp(g)$ (lognormal transform). We then compute empirical quantile edges of κ and map nodal values into a shuffled discrete set $\{\kappa_1, \ldots, \kappa_m\}$ by quantile binning; finally, we define an elementwise-constant coefficient on fine triangles by averaging nodal values per element.

- **Lognormal Matérn (continuous).** We sample a Matérn Gaussian field Z at fine nodes and set $\kappa = \exp(Z)$.

$$c(x, x') = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} \|x - x'\|_2 \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} \|x - x'\|_2 \right),$$

  To evaluate $\kappa(x)$ at quadrature points during assembly, we use a scattered interpolant based on the fine-node samples.

- **Piecewise-constant patterns (checkerboard).** We generate κ as a discrete random field on a uniform grid (either coarse- or fine-scale), including checkerboard, horizontal stripes, and vertical stripes, and define $\kappa(x, y)$ by O(1) indexing into the corresponding cell.

Per-sample LOD assembly. Fix a localization depth $\ell \in \mathbb{N}$. For each coefficient realization κ, we perform:

1. **Fine stiffness assembly.** Assemble the fine stiffness matrix from the fine nodal basis functions

$$\mathbf{A}_h = \left[a(\phi_h^i, \phi_h^j) \right]_{i,j}, \qquad a(u, v) = \int_\Omega \kappa \, \nabla u \cdot \nabla v \, dx.$$

2. **Localized correctors.** For every coarse element $T \in \mathcal{T}_H$, form the ℓ-layer patch $\mathcal{N}^\ell(T)$ and solve the discrete localized corrector problems

$$Q_{T,h}^\ell(\Phi_H^i) \in W^\ell(T) \cap V_h, \qquad a\big(Q_{T,h}^\ell(\Phi_H^i), w_h\big) = \int_T \kappa \, \nabla \Phi_H^i \cdot \nabla w_h \, dx,$$

   for all $w_h \in W^\ell(T) \cap V_h$, where $W = \ker(\mathcal{I}_H)$ and $W^\ell(T)$ denotes functions supported in $\mathcal{N}^\ell(T)$. Summing all element contributions yields the global discrete correction operator

$$Q_h^\ell := \sum_{T \in \mathcal{T}_H} Q_{T,h}^\ell.$$

3. **Multiscale basis and coarse operator.** We assemble the multiscale basis matrix

$$\mathbf{G} := \mathbf{P}_1 - \mathbf{Q}_h^\ell,$$

   and obtain the (Galerkin-projected) LOD coarse operator and load

$$\mathbf{A}_{\mathrm{LOD}} = \mathbf{G} \, \mathbf{A}_h \, \mathbf{G}^\top, \qquad \mathbf{f}_{\mathrm{LOD}} = \mathbf{G} \, \mathbf{f}_h.$$

4. **Dirichlet restriction and solve.** We remove boundary coarse nodes using a diagonal mask $\mathbf{B}_H$ and solve the interior system

$$\mathbf{A}_{\mathrm{LOD,int}} \, \mathbf{u}_{\mathrm{LOD,int}} = \mathbf{f}_{\mathrm{LOD,int}}.$$

   The vector $\mathbf{u}_{\mathrm{LOD,int}}$ is the supervision target used for training. Optionally, we also compute a fine reference solution $\mathbf{u}_h$ by solving $\mathbf{A}_h \mathbf{u}_h = \mathbf{f}_h$ (with Dirichlet constraints) for diagnostic comparisons.

F. Training and Inference Time Comparisons

Table 4 summarizes the data-generation costs for different mesh resolutions; all times are measured in seconds per sample. For the LOD basis construction, we report the local patch-corrector cost, the global basis assembly cost, and their sum parallelized over 16 CPU cores, the sequential global basis assembly cost, and their sum. The localized corrector problems $Q_{T,h}^\ell$ are embarrassingly parallel—each depends only on the fine-grid stiffness matrix restricted to its patch—so the patch-corrector step distributes exactly across cores; scaling to more than 16 cores would reduce this component proportionally. By contrast, assembling the LOD basis (given by the rows of the matrix $G = P_1 - Q_h^\ell$, which is the difference of the prolongation matrix P1 that encodes the coarse nodal basis functions and the correction operator Q_h^ℓ) via global sparse-matrix stacking and multiplication across all patches such that is an aggregation step that does not decompose into independent

Table 4. Comparison of data-generation times, measured in seconds per sample. Local patch-corrector times assume 16 CPU cores. Global basis assembly involves global sparse matrix operations and is measured sequentially on the same CPU.

Resolution	Fine FEM	LOD Basis			LOD Coeff		
		Local patch correctors	Global basis assembly	Total	Coeff. assembly	Coeff. solve	Total
$h = 7,\ H = 4$	0.1135	0.7842	0.4890	1.2732	0.2145	0.0020	0.2165
$h = 8,\ H = 4$	0.5590	4.2527	1.8327	6.0854	0.8500	0.0021	0.8521
$h = 8,\ H = 5$	0.5606	4.5842	4.7611	9.3453	1.2688	0.0752	1.3440
$h = 9,\ H = 6$	4.3828	20.0033	72.9008	92.9041	6.9178	3.1047	10.0225

Table 5. Training and inference times (seconds).

Resolution	Training		Inference	
	LOD-MSNO	CNO	LOD-MSNO	CNO
$h = 7,\ H = 4$	1186	10366	0.0036	0.0061
$h = 8,\ H = 4$	3922	36076	0.0036	0.0064
$h = 8,\ H = 5$	4508	36040	0.0036	0.0064
$h = 9,\ H = 6$	27334	137714	0.0086	0.0207

patch-level tasks and is therefore measured without inter-task parallelization. Classical LOD timings were obtained on a 16-core CPU; all neural-network timings (training and inference) were measured on an NVIDIA A100 80GB GPU.

The results show that the localized basis-corrector computations dominate the classical LOD data-generation cost, especially as the fine mesh is refined. With 16-core parallelization of the local corrector step, the per-sample patch-corrector to abtian the LOD basis functions time is substantially reduced compared to the sequential cost when computing all patch corrections sequentially; using more than 16 cores would decrease this cost further proportionally. The global basis assembly and the coefficient steps remain sequential, and for fine coarse spaces the global assembly itself becomes a dominant cost—at $h = 9$, $H = 6$, it accounts for the majority of the basis construction time. In particular, as H is refined, the coarse LOD solve becomes comparable in cost to the fine FEM solve despite involving fewer degrees of freedom, which highlights the effect of the non-sparsity of the LOD system as a major computational bottleneck. This observation motivates the coefficient surrogate, whose direct inference of the expansion coefficients avoids repeatedly assembling and solving this dense coarse system and thereby contributes directly to the speedup of LOD-MSNO.

Table 5 reports the corresponding neural-network training and single-sample inference times for LOD-MSNO and CNO. All neural-network models were trained using 1000 training samples, and the coefficient model and CNO were trained for the same number of epochs. We compare against CNO because it is the strongest benchmark model in terms of accuracy among the standard neural-operator baselines considered in Table 1.

The results show that LOD-MSNO is faster to train than CNO across all tested resolutions. At inference time, the neural surrogates are much faster than the corresponding traditional LOD computations, as can be inferred from the per-sample LOD basis and coefficient generation times in Table 4.