Transolver-GMsFEM: A Hybrid Framework for High-Contrast Multiscale PDEs on Irregular Grids

Alexander Rudikov[1,2] **Sergei Stepanov**[3] **Vladimir Fanaskov**[1,4]

Eric Chung[5] **Ekaterina Muravleva**[6] **Ivan Oseledets**[1,2]

[1]AXXX, [2]Institute of Numerical Mathematics, [3]North-Eastern Federal University,
[4]Applied AI Institute, [5]The Chinese University of Hong Kong, [6]AI4Science

Abstract

High-contrast multiscale PDE problems are common in real-world applications, yet current neural PDE solvers struggle to achieve sufficient accuracy on such tasks. The Generalized Multiscale Finite Element Method (GMsFEM) addresses this by compressing fine-scale heterogeneity into localized basis functions which are used to obtain more accurate solutions than neural PDE solvers. However, constructing these basis functions requires solving many local eigenvalue problems – the major computational bottleneck. We address this issue by proposing **Transolver-GMsFEM**, a new hybrid framework that utilizes the efficiency of neural PDE solver for predicting multiscale basis functions while preserving the solution quality of GMsFEM. Experiments on 2D/3D steady-state and time-dependent high-contrast multiscale PDEs on irregular grids show proposed method achieves over $100\times$ speedup for basis construction. Crucially, our experiments demonstrate that Transolver-GMsFEM significantly outperforms state-of-the-art neural PDE solvers in both accuracy and robustness, especially in out-of-distribution tasks where pure neural PDE solvers dramatically fail.

1 Introduction

Many real-world applications require the solution of partial differential equations (PDEs) with high-contrast coefficients Wazwaz [2026], Evans [2022], Spiridonov et al. [2020], He and Ren [2009], Vasilyeva et al. [2023], Stepanov et al. [2018]. Neural PDE solvers have recently emerged as fast surrogate models that learn the mapping from problem inputs, such as geometry, coefficients, and boundary conditions, to solution fields directly from data. One widely used class of neural PDE solvers is neural operators (NOs), which learn mappings between function spaces Li et al. [2020a], Tran et al. [2021], Li et al. [2023a], Wen et al. [2022], Rahman et al. [2022], Ronneberger et al. [2015], Li et al. [2020b, 2023b]. However, some of these operators are restricted to Cartesian grids and cannot be directly applied to complex engineering geometries Li et al. [2020a], Tran et al. [2021], Wen et al. [2022], Rahman et al. [2022], Ronneberger et al. [2015]. Also, NOs often struggle to achieve good accuracy on million-scale meshes.

More recently, transformer-based neural PDE solvers have shown strong performance on arbitrary geometries and large-scale problems. In particular, GNOT Hao et al. [2023] and GAOT Wen et al. [2025] employ specialized geometric encoding and multiscale attention to handle long-range physical correlations. Transolver Wu et al. [2024] further advanced this field by introducing a physics-attention mechanism to model long-range physical dependencies. Transolver++ Luo et al. [2025] substantially strengthens this framework with an optimized architecture and an Adaptive Temperature mechanism that prevents performance degradation on large, high-resolution meshes. Transolver++

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

has demonstrated superiority over more than 20 advanced baselines, achieves accurate simulations on million-scale geometries, and supports arbitrary geometries.

Despite this progress, purely neural PDE solvers have limited accuracy and robustness on high-contrast multiscale problems. For these, the Generalized Multiscale Finite Element Method (GMsFEM) remains a most effective classical approach. GMsFEM Efendiev et al. [2013, 2014] compresses fine-scale heterogeneity into multiscale basis functions Efendiev et al. [2011]. These functions form a projection matrix mapping the fine-grid problem into coarse space, enabling efficient computation while preserving high accuracy.

GMsFEM consists of **offline** and **online** stages. In the offline stage, multiscale basis functions are generated for each input coefficient field by solving local spectral problems; these functions encode important information about heterogeneities. For time-dependent problems, offline basis functions are often insufficient. To improve the accuracy, we add online basis functions during the online stage. The construction of the online basis functions is based on the local residuals Chung et al. [2014]. Because both stages require repeated local solvers, offline and online basis construction together form the main computational bottleneck of GMsFEM.

A previous acceleration strategy based on F-FNO Rudikov et al. [2025] reduced the cost of offline basis construction, but it is largely limited to regular grids and steady-state PDEs. To overcome these limitations and retain efficiency on irregular grids, we propose Transolver-GMsFEM, a hybrid framework that uses Transolver++ to predict multiscale basis functions while preserving the accuracy of classical GMsFEM. Crucially, whereas Rudikov et al. [2025] requires separate models for different local domain types, Transolver++ leverages its ability to handle arbitrary geometries to generalize across all domain variations. Our main contributions are:

- **Hybrid multiscale framework:** We develop Transolver-GMsFEM, which integrates a state-of-the-art Transformer-based neural PDE solver with GMsFEM to solve high-contrast PDEs on irregular meshes, achieving higher accuracy than state-of-the-art neural PDE solvers on both 2D and 3D datasets.
- **Robust OOD generalization:** We show stable behavior on out-of-distribution inputs, including coefficients shifted within the same generative family, coefficients from structurally distinct families, and OOD right-hand sides. In contrast, neural PDE solvers show substantial degradation.
- **Selective sampling strategy:** We introduce a selective local-domain sampling scheme that prioritizes high-contrast regions, improving basis function prediction accuracy by 15–20% over random sampling while simultaneously reducing model training time.
- **Dual-stage acceleration:** We achieve substantial dual-stage acceleration: for 2D problems, approximately **100**$\times$ speedup in offline basis construction and **5**$\times$ speedup in online basis functions construction; for 3D problems, up to **265**$\times$ and **7**$\times$ speedups in the offline and online stages, respectively.

2 Problem Statement and Motivation

Let $\mathcal{D} \subset \mathbb{R}^d$ be the computational domain. As an example, we consider high-contrast problem

$$-\nabla \cdot \big(\kappa(x)\nabla u\big) = f,\ x \in \mathcal{D}, \quad u = 0,\ x \in \partial\mathcal{D}, \tag{1}$$

and the corresponding time-dependent problem:

$$\frac{\partial u}{\partial t} - \nabla \cdot \big(\kappa(x)\nabla u\big) = f(x),\ x \in \mathcal{D}, \quad u = 0,\ x \in \partial\mathcal{D}, \quad u|_{t=0} = 0,\ x \in \mathcal{D}, \tag{2}$$

where $f \in L^2(\mathcal{D})$ represents the source term and $\kappa(x)$ is a heterogeneous field with high contrast. The difficulty in the numerical approximation of (1) and (2) arises from the complexity of the coefficient $\kappa(x)$, which can have multiple scales. Discretizing (1) and (2) using traditional numerical methods based on the finite element method Zienkiewicz and Taylor [2005], Ŝolín [2005] requires a fine grid to resolve all underlying scales. This leads to extremely large and ill-conditioned linear systems, demanding excessive computational time and memory. Therefore, it is desirable to develop efficient numerical schemes using fewer degrees of freedom while maintaining high accuracy.

3 Preliminaries

To discretize (1), we introduce the notion of fine and coarse grids. Let $\mathcal{T}^H$ be a usual partition of the computational domain $\mathcal{D}$ into finite elements. We refer to this partition as the coarse grid. The fine grid partition is denoted by $\mathcal{T}^h$ with $h \ll H$, and is, by definition, a refinement of the coarse grid. Let N_c be the number of nodes $\{x_i\}_{i=1}^{N_c}$ in the coarse grid $\mathcal{T}^H$. The coarse mesh $\mathcal{T}^H$ is structured and composed of $N_v \times N_v$ coarse elements K_j; this implies $N_c = (N_v + 1)^2$. For each coarse node x_i, we define a coarse neighborhood ω_i by

$$\omega_i = \bigcup \{K_j \in \mathcal{T}^H : x_i \in \overline{K}_j\}.$$

In other words, ω_i is the union of all coarse elements $K_j \in \mathcal{T}^H$ sharing the coarse node x_i. We classify these local domains for the 2D case into three geometric types – **interior**, **border**, and **corner** – based on the number of coarse cells K_j within the local domain ω_i (see Appendix A).

Let $V_0^h(\mathcal{D})$ be the space of piecewise linear continuous functions on a fine partition $\mathcal{T}^h$ that vanish on $\partial\mathcal{D}$. The standard Galerkin finite element discretization of (1) (or the spatial discretization of (2)) yields the linear system

$$Au = b, \tag{3}$$

where the stiffness matrix A and the load vector b are defined for all $u, v \in V_0^h(\mathcal{D})$ by

$$v^\top A u = \int_{\mathcal{D}} \kappa \nabla u \cdot \nabla v \, dx, \qquad v^\top b = \int_{\mathcal{D}} f v \, dx.$$

4 Offline GMsFEM

Construction of multiscale basis functions consists of two major ingredients: (i) **local spectral problem** and (ii) **global coupling**. We seek multiscale basis functions localized in each coarse neighborhood $\omega_i \subset \mathcal{D}$, denoted by $\{\psi_k^{\omega_i}\}_{k=1}^L$, where L is a fixed number of basis functions per neighborhood. The multiscale solution u_{ms} is then represented as a linear combination:

$$u_{\text{ms}}(x) = \sum_{i=1}^{N_c} \sum_{k=1}^{L} c_k^i \psi_k^{\omega_i}(x), \tag{4}$$

and belongs to the approximation space $V_{\text{off}} = \text{span}\{\psi_k^{\omega_i} : 1 \leqslant i \leqslant N_c, \, 1 \leqslant k \leqslant L\}$.

4.1 Local Spectral Problem

For each local domain $\omega_i \subset \mathcal{D}$, we consider the local spectral problem (see Appendix B for details)

$$-\nabla \cdot (\kappa(x) \nabla \phi) = \lambda \kappa(x) \phi \ \text{ in } \omega_i, \quad n \cdot (\kappa(x) \nabla \phi) = 0 \ \text{ on } \partial\omega_i, \tag{5}$$

where n is the outward unit normal to $\partial\omega_i$. We denote its eigenvalues and eigenvectors by $\{\lambda_k^{\omega_i}\}_{k=1}^{l_i}$ and $\{\phi_k^{\omega_i}\}_{k=1}^{l_i}$, respectively. The first eigenvalue satisfies $\lambda_1^{\omega_i} = 0$, and we can arrange the non-negative real eigenvalues in the non-decreasing order $\lambda_1^{\omega_i} \leqslant \lambda_2^{\omega_i} \leqslant \ldots \leqslant \lambda_{l_i}^{\omega_i}$.

We choose the first L eigenvectors ($L \leqslant l_i$) corresponding to the smallest eigenvalues, $\{\phi_k^{\omega_i}\}_{k=1}^L$, to construct the local basis in each local domain ω_i. These eigenvectors represent the dominant components of the solution that capture the effects of $\kappa(x)$ at the coarse scale.

4.2 Global Coupling

The global coupling combines these eigenvectors to provide an accurate approximation of the solution. The global coupling in GMsFEM relies on a set of partition of unity functions $\{\chi_i\}_{i=1}^{N_c}$ that satisfy $\sum_{i=1}^{N_c} \chi_i = 1$. We use multiscale partition of unity functions that are designed to incorporate the local variations of $\kappa(x)$. Specifically, for each local domain ω_i, the function χ_i is obtained by solving the following problem

$$-\nabla \cdot (\kappa \nabla \chi_i) = 0 \ \text{ in each } K \subset \omega_i, \quad \chi_i = \chi_i^0 \ \text{ on } \partial K, \tag{6}$$

where χ_i^0 is the standard piecewise linear partition of unity function defined on local domain ω_i. By substituting $\chi_i = \varepsilon_i + \chi_i^0$, we can rewrite (6) as

$$-\nabla \cdot (\kappa \nabla \varepsilon_i) = g \text{ in each } K \subset \omega_i, \quad \varepsilon_i = 0 \text{ on } \partial K, \tag{7}$$

where $g = \nabla \cdot (\kappa \nabla \chi_i^0)$ is the source term.

Offline basis functions are defined by $\psi_k^{\omega_i} = \chi_i \phi_k^{\omega_i}$ for $k = 1, \ldots, L$. The **projection matrix** is $R_0^\top = [\psi_1^{\omega_1}, \ldots, \psi_L^{\omega_1}, \ldots, \psi_1^{\omega_{N_c}}, \ldots, \psi_L^{\omega_{N_c}}]$. Instead of solving the fine-scale system (3), we solve the reduced coarse-scale system $A_0 u_0 = f_0$, where $A_0 = R_0 A R_0^\top$ is the projected matrix and $f_0 = R_0 b$ is the projected forcing vector. The fine-scale solution (4) is reconstructed as $u_{\text{ms}} = R_0^\top u_0$.

5 Online GMsFEM

In the **online stage**, we enrich the offline space V_{off} with **online basis functions** ϑ^{ω_i}. By online basis functions, we mean basis functions that are computed during the iterative process, contrary to offline basis functions $\{\psi_k^{\omega_i}\}_{k=1}^L$ that are computed before the iterative process. The adaptive enrichment algorithm Chung et al. [2015] is stated as follows. We use the index $m \geqslant 1$ to represent the enrichment level. At the enrichment level $m = L$, we use $V_{\text{ms}}^m = V_{\text{off}}$ to denote the corresponding GMsFEM space and $u_{\text{ms}}^m = u_{\text{ms}}$ the corresponding solution obtained in (4). The online basis functions are constructed as $\vartheta^{\omega_i} = \chi_i \phi^{\text{on},\omega_i}$ within the local domains ω_i (see Appendix C and D for details).

For each ω_i, the online basis function $\phi_n^{\text{on},\omega_i}$ for time-step n is obtained by solving a local problem

$$\int_{\omega_i} \frac{\phi_n^{\text{on},\omega_i}}{\tau} v \, dx + \int_{\omega_i} \kappa \nabla \phi_n^{\text{on},\omega_i} \cdot \nabla v \, dx = R_i^n(v), \quad \forall v \in H_0^1(\omega_i) \cap V, \tag{8}$$

where the local residual functional $R_i^n : V_i \to \mathbb{R}$ is defined by

$$R_i^n(v) = \int_{\omega_i} \left(\frac{u_{\text{ms}}^{m+1,n-1}|_{\omega_i}}{\tau} + f \right) v \, dx - \int_{\omega_i} \frac{u_{\text{ms}}^{m,n}|_{\omega_i}}{\tau} v \, dx - \int_{\omega_i} \kappa \nabla u_{\text{ms}}^{m,n}|_{\omega_i} \cdot \nabla v \, dx.$$

Here, $u_{\text{ms}}^{m,n}|_{\omega_i}$ denotes the offline GMsFEM solution at time step n restricted to ω_i, and $u_{\text{ms}}^{m+1,n-1}|_{\omega_i}$ denotes the online GMsFEM solution at time step $n-1$ restricted to ω_i. To maintain efficiency, the online basis functions are updated only every fifth time step, referred to as **enrichment steps**.

Let $V_{\text{ms}}^{m+1,n} = V_{\text{off}} \oplus \text{span}\{\vartheta_n^{\omega_i} : 1 \leqslant i \leqslant N_c\}$ be the new approximation space, and let $u_{\text{ms}}^{m+1,n}$ be the corresponding solution. As a result, the following multiscale space and projection matrix are:

$$V_{\text{ms}}^{m+1,n} = \text{span}\{\psi_1^{\omega_1}, \ldots, \psi_L^{\omega_{N_c}}, \vartheta_n^{\omega_1}, \ldots, \vartheta_n^{\omega_{N_c}}\}, \; R_0^{m+1,n} = \left[\psi_1^{\omega_1}, \ldots, \psi_L^{\omega_{N_c}}, \vartheta_n^{\omega_1}, \ldots, \vartheta_n^{\omega_{N_c}}\right]^\top.$$

6 Proposed Method

Offline GMsFEM requires solving computationally intensive local problems for each ω_i: the local spectral problem (5) and local problem (7) for ε_i. Additionally, time-dependent problems require solving (8) at enrichment steps for $\phi_n^{\text{on},\omega_i}$. We propose to accelerate these computations by using trained neural PDE solvers (Transolver++ Luo et al. [2025]). These models provide rapid approximations, significantly reducing computational costs while preserving accuracy. Specifically, we employ three specialized models for GMsFEM tasks: **(1)** spectral eigenvector prediction, **(2)** partition of unity correction, and **(3)** online basis function prediction.

6.1 Model Formulation and Loss Functions

Model 1: Spectral Eigenvector Prediction. The first model approximates the solution of the local spectral problem (5). Given the heterogeneous coefficient field $\kappa|_{\omega_i}$ restricted to ω_i, the model predicts the first L eigenvectors $\{\phi_k^{\omega_i}\}_{k=1}^L$. Formally, the mapping is $\mathcal{G}_1 : \kappa|_{\omega_i} \mapsto \{\widetilde{\phi}_k^{\omega_i}\}_{k=1}^L$, where $\widetilde{\phi}_k^{\omega_i}$ denotes the predicted eigenvector.

Loss Function. Since eigenvectors are defined only up to orthogonal transformations, we evaluate the subspace alignment error:

$$\mathcal{L}_1 = \mathbb{E}_i \left[L - \left\| Q_{W^i}^\top Q_{\widetilde{W}^i} \right\|_F^2 \right],$$

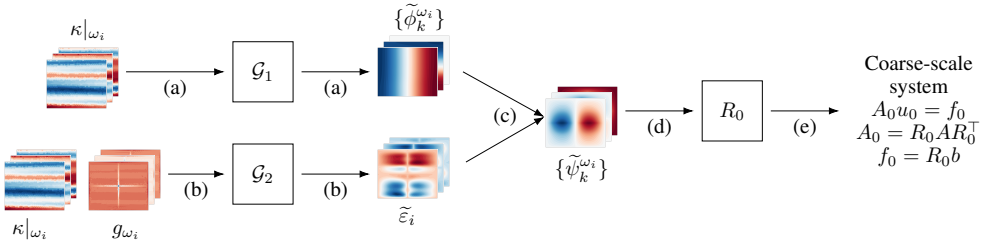


Figure 1: Inference for **offline** Transolver-GMsFEM: **(a)** Trained model $\mathcal{G}_1$ takes as input the local coefficient field $\kappa|_{\omega_i}$ on ω_i and outputs the eigenvectors $\{\widetilde{\phi}_k^{\omega_i}\}$. **(b)** Trained model $\mathcal{G}_2$ takes as input the coefficient field $\kappa|_{\omega_i}$ and the local source term g_{ω_i}, where $g = \nabla \cdot (\kappa \nabla \chi_i^0)$ and outputs the correction term $\widetilde{\varepsilon}_i$. **(c)** These outputs are combined to calculate the offline basis functions $\{\widetilde{\psi}_k^{\omega_i}\}$ through $\widetilde{\psi}_k^{\omega_i} = (\widetilde{\varepsilon}_i + \chi_i^0)\widetilde{\phi}_k^{\omega_i}$. **(d)** The local basis functions are then assembled to form the global projection matrix R_0, which is used to construct the coarse-scale system. **(e)** Finally, the coarse-scale system is solved to obtain the multiscale solution. *Notably, the inference from $\mathcal{G}_1$ and $\mathcal{G}_2$ replaces the traditional, computationally expensive basis function calculations of GMsFEM.*

where $W^i = \text{span}\{\phi_1^{\omega_i}, \ldots, \phi_L^{\omega_i}\}$ denotes the target eigenvector subspace and $\widetilde{W}^i = \text{span}\{\widetilde{\phi}_1^{\omega_i}, \ldots, \widetilde{\phi}_L^{\omega_i}\}$ denotes the predicted eigenvector subspace. The matrices Q_{W^i} and $Q_{\widetilde{W}^i}$ contain orthonormal bases for these subspaces. The term $\|Q_{W^i}^\top Q_{\widetilde{W}^i}\|_F^2$ quantifies the subspace overlap, achieving its maximum value L when the two spaces are perfectly aligned.

This loss formulation was previously suggested for training F-FNO to predict multiscale basis functions $\{\psi_k^{\omega_i}\}$ Rudikov et al. [2025].

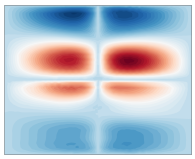


Figure 2: ε_i example.

Model 2: Partition of Unity Correction. This model computes the correction ε_i appearing in (7). The inputs are the local coefficient field $\kappa|_{\omega_i}$ restricted to the local domain ω_i and the source term $g_{\omega_i} = \nabla \cdot (\kappa|_{\omega_i} \nabla \chi_i^0)$. The output is the correction ε_i, from which the partition of unity function is recovered as $\chi_i = \varepsilon_i + \chi_i^0$. The mapping is $\mathcal{G}_2 : (\kappa|_{\omega_i}, g_{\omega_i}) \mapsto \widetilde{\varepsilon}_i$, where $\widetilde{\varepsilon}_i$ denotes the predicted correction.

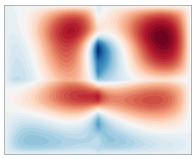


Figure 3: $\phi_n^{\text{on},\omega_i}$ example.

Loss Function. The correction is measured using a weighted Sobolev-type norm that incorporates gradient information:

$$\mathcal{L}_2 = \mathbb{E}_i \left[\left\| \varepsilon_i - \widetilde{\varepsilon}_i \right\|_2^2 + \left\| \sqrt{\kappa|\omega_i} (\nabla \varepsilon_i - \nabla \widetilde{\varepsilon}_i) \right\|_2^2 \right].$$

This specific formulation is chosen because ε_i exhibits sharp gradients, as illustrated in Figure 2. By weighting the gradient difference with the coefficient $\sqrt{\kappa}$, the loss ensures the model accurately captures sharp local features and maintains stability in high-contrast regions.

Model 3: Online Basis Function Prediction. For time-dependent problems, the third model approximates the solution to (8) at each online enrichment step. The inputs consist of the local right-hand side $f|_{\omega_i}$, the local coefficient field $\kappa|_{\omega_i}$, the offline solution $u_{\text{ms}}^{m,n}|_{\omega_i}$, and the online solution $u_{\text{ms}}^{m+1,n-1}|_{\omega_i}$ from the previous time step, all restricted to the ω_i. The output is the online basis function $\phi_n^{\text{on},\omega_i}$ at time step n. The mapping is $\mathcal{G}_3 : (u_{\text{ms}}^{m,n}, u_{\text{ms}}^{m+1,n-1}, f, \kappa)|_{\omega_i} \mapsto \widetilde{\phi}_n^{\text{on},\omega_i}$, where $\widetilde{\phi}_n^{\text{on},\omega_i}$ is the predicted online basis function.

Loss Function. We employ a relative error formulation with gradients:

$$\mathcal{L}_3 = \mathbb{E}_{i,n} \left[\frac{\left\| \phi_n^{\text{on},\omega_i} - \widetilde{\phi}_n^{\text{on},\omega_i} \right\|_2^2}{\left\| \phi_n^{\text{on},\omega_i} \right\|_2^2} + \frac{\left\| \sqrt{\kappa|_{\omega_i}} (\nabla \phi_n^{\text{on},\omega_i} - \nabla \widetilde{\phi}_n^{\text{on},\omega_i}) \right\|_2^2}{\left\| \sqrt{\kappa|_{\omega_i}} \nabla \phi_n^{\text{on},\omega_i} \right\|_2^2} \right].$$

This relative formulation is essential because $\phi_n^{\text{on},\omega_i}$ often varies greatly in magnitude across ω_i and time steps; as shown in Figure 3, these functions have complex profiles with sharp transitions.

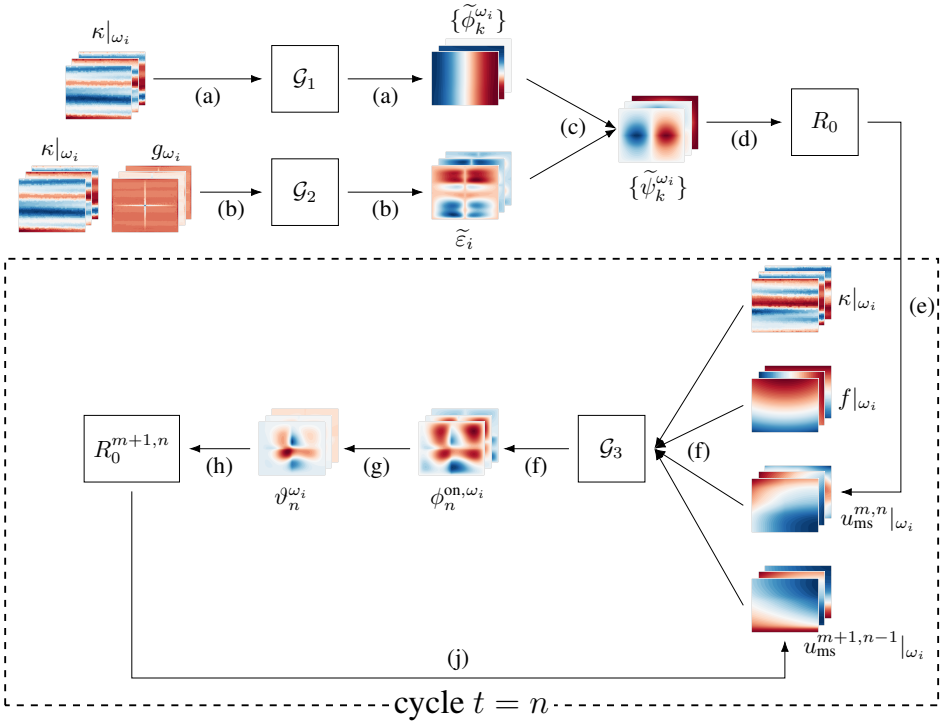


Figure 4: Inference for online Transolver-GMsFEM: **(a)–(d)** The initial **offline stage** follows the offline Transolver-GMsFEM procedure to construct the initial projection matrix R_0. To maintain computational efficiency, the online enrichment steps **(e)–(h)** are performed only periodically (e.g., every fifth timestep). **(e)** At these enrichment steps, the coarse scale system $A_0 u_0^{m,n} = f_0^n$ is solved, where $A_0 = R_0 A R_0^\top$ and the time-dependent source term is $f_0^n = R_0 b^n$, to obtain the multiscale solution $u_{\mathrm{ms}}^{m,n} = R_0^\top u_0^{m,n}$. **(f)** During these enrichment steps, the local right-hand side $f^n|_{\omega_i}$, the current coarse solution $u_{\mathrm{ms}}^{m,n}|_{\omega_i}$, the online solution from the previous step $u_{\mathrm{ms}}^{m+1,n-1}|_{\omega_i}$ (initialized as zero for $n = 0$), and the coefficient field $\kappa|_{\omega_i}$ are provided as inputs to the trained model $\mathcal{G}_3$. Model $\mathcal{G}_3$ generates $\widetilde{\phi}_n^{\mathrm{on},\omega_i}$, effectively replacing the direct GMsFEM online basis calculation with high-speed inference. **(g)** These vectors are used to construct online basis functions $\widetilde{\vartheta}_n^{\omega_i} = (\widetilde{\varepsilon}_i + \chi_i^0)\widetilde{\phi}_n^{\mathrm{on},\omega_i}$, which are then assembled into **(h)** the updated online projection matrix $R_0^{m+1,n}$. If the timestep n is not an enrichment step, previous online projection matrix is reused $R_0^{m+1,n} = R_0^{m+1,n-1}$. **(j)** Finally, on each timestep n, the updated coarse system $A_0^{m+1,n} u_0^{m+1,n} = f_0^{m+1,n}$ is solved to yield the final online multiscale solution $u_{\mathrm{ms}}^{m+1,n} = (R_0^{m+1,n})^\top u_0^{m+1,n}$ for the current timestep.

6.2 Theoretical guarantees

Let W_i be the span of the first L basis functions on ω_i, $\widetilde{W}_i$ the subspace predicted by $\mathcal{G}_1$. The $\mathcal{G}_1$ is trained with $\mathcal{L}_1$ loss, then $\mathcal{L}_{\max} = \max_i \left[L - \left\| Q_{W^i}^\top Q_{\widetilde{W}^i} \right\|_F^2 \right]$ vanishes iff $\widetilde{W}_i = W_i$ for all i.

Theorem 6.1. *Let $\widetilde{u}_{\mathrm{ms}}$ be the Transolver-GMsFEM solution computed with the eigenvectors predicted by $\mathcal{G}_1$ and the exact partition of unity $\{\chi_i\}_{i=1}^{N_c}$. Under Assumptions E.1–E.3 (see Appendix E),*

$$\|u - \widetilde{u}_{\mathrm{ms}}\|_V \leq C_0 \frac{H}{\sqrt{\Lambda_0^*}} \|f\|_{L^2(\mathcal{D})} + \kappa_{\mathrm{ov}}\, C \sqrt{\mathcal{L}_{\max}}\, \|u\|_V,$$

where $\Lambda_0^* = H^2 \min_i \lambda_{L+1}^{\omega_i}$, $\kappa_{\mathrm{ov}} = 2^d$, and C_0, C are independent of H and N_c. The first term is the GMsFEM bound; the second does not grow with N_c (proof in Appendix E).

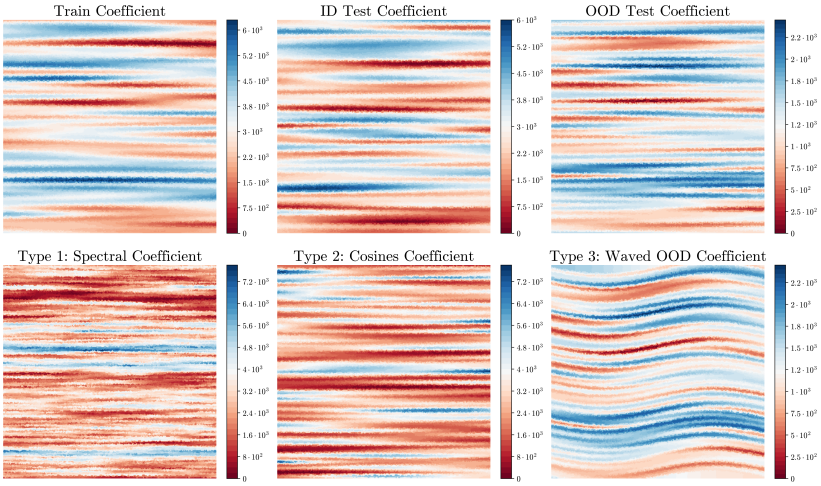


Figure 5: Examples of train and in-distribution (ID) test coefficient fields, together with four out-of-distribution (OOD) coefficient types.

6.3 Models Training

To optimize training efficiency and model performance, we do not use all N_c local domains. Instead, we follow the **Selective Sampling Strategy** (detailed in Appendix F), specifically identifying "extreme-value" domains where κ approaches its global extreme values. The models $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ are trained on these selected local domains. Consequently, the models develop a worst-case robustness. Post-training, the pipeline branches into **two distinct tracks**: one for steady-state problems (see Figure 1) and another for time-dependent problems (see Figure 4).

7 Experiments

We utilized a dataset consisting of 2D high-contrast coefficient fields (detailed in Appendix G), which was partitioned into $N_{\text{train}} = 700$ training, $N_{\text{val}} = 100$ validation, and $N_{\text{test}} = 200$ test samples. Representative examples of the training and in-distribution (ID) test coefficient fields are illustrated in Figure 5. To evaluate the robustness of our approach, we also generated four types of out-of-distribution (OOD) coefficient fields (200 samples for each type) as detailed in Appendix G.2. A sample OOD field from the same generative family (with shifted statistics) is shown in the top-right panel of Figure 5. The bottom row of the same figure visualizes structurally distinct OOD coefficient families, generated by entirely different processes, which are used for further robustness testing. The structured coarse mesh is 10×10, resulting in $N_c = 121$ nodes, while the irregular fine grid consists of $151{,}940$ cells and $76{,}471$ vertices.

We extensively evaluate Transolver-GMsFEM on five benchmark problems: (i) Diffusion with Dirichlet BC (see Appendix H.1), (ii) Diffusion with mixed BC (see Appendix H.2), (iii) Steady-State Richards Equation (see Appendix H.4), (iv) Heat Equation (see Appendix H.5), and (v) Time-Dependent Richards Equation (see Appendix H.6). For steady-state problems, we consider two different right-hand sides, given by (19) and (20) (see Appendix G.3). Conversely, for time-dependent problems, we restrict our evaluation to (20). Furthermore, since online basis functions in time-dependent equations are sensitive to the source term, we also tested an OOD right-hand side (21).

To further evaluate the scalability, we conducted additional experiments on a 3D mesh consisting of $1{,}382{,}363$ cells and $238{,}152$ vertices. Due to space constraints, the experimental setup and detailed results are provided in Appendices I and P, respectively.

To test whether the method depends on rectangular local domains, we use perforated benchmarks with circular (2D)/spherical (3D) voids of varying number and position. The boundaries of the local domains are therefore curved and non-polygonal; setup and results are in Appendices H.3 and Q.

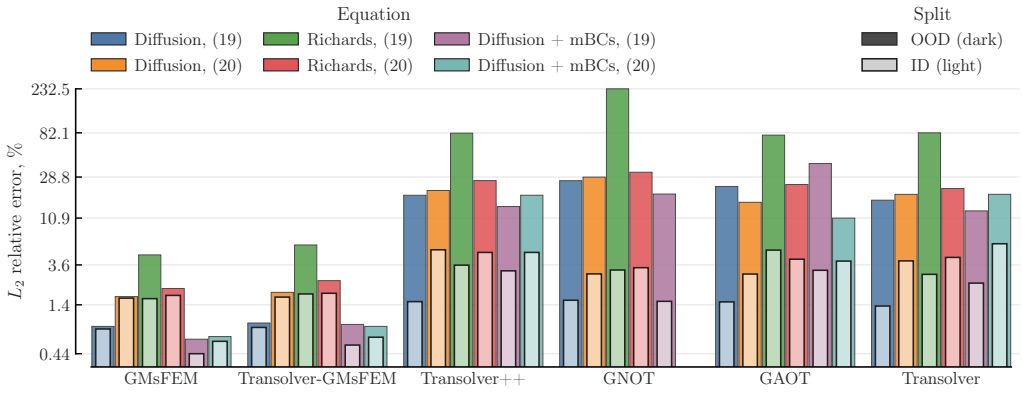


Figure 6: Relative L_2 errors for steady-state problems. Results evaluate the proposed method against classical GMsFEM and four neural baselines under ID and OOD conditions (see Tables 7 and 9).

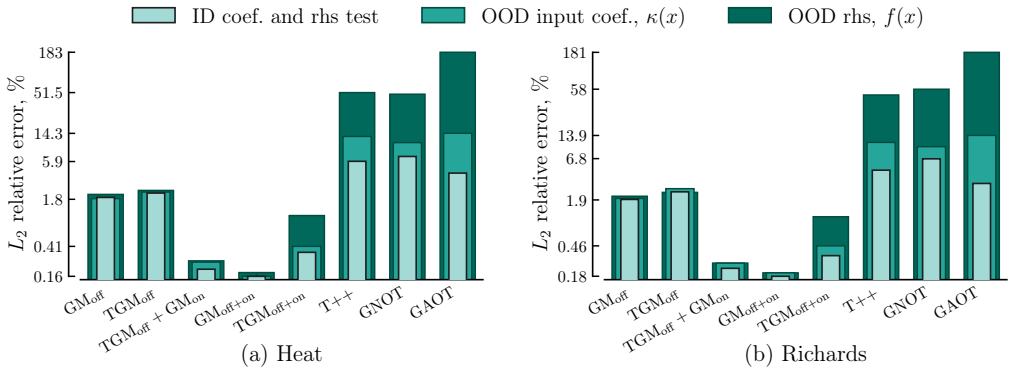


Figure 7: Relative L_2 errors for time-dependent problems. Method abbreviations: $\mathrm{GM}_{\mathrm{off}}$ = GMsFEM (offline); $\mathrm{TGM}_{\mathrm{off}}$ = Transolver-GMsFEM (offline); $\mathrm{TGM}_{\mathrm{off}} + \mathrm{GM}_{\mathrm{on}}$ = Transolver-GMsFEM (offline) + GMsFEM (online); $\mathrm{GM}_{\mathrm{off+on}}$ = GMsFEM (offline+online); $\mathrm{TGM}_{\mathrm{off+on}}$ = Transolver-GMsFEM (offline+online); T++ = Transolver++. Detailed results are in Tables 8, 10, and 11.

To measure quality of the obtained solutions on fine grid, we use the following metrics:

$$L_2 = \mathbb{E}_i \left[\sqrt{\frac{\int_{\mathcal{D}} \left| u_h^i - \tilde{u}_h^i \right|^2 dx}{\int_{\mathcal{D}} \left| u_h^i \right|^2 dx}} \right], \quad H^1 = \mathbb{E}_i \left[\sqrt{\frac{\int_{\mathcal{D}} \left| \nabla u_h^i - \nabla \tilde{u}_h^i \right|^2 dx}{\int_{\mathcal{D}} \left| \nabla u_h^i \right|^2 dx}} \right].$$

We evaluate our results against Transolver++, Transolver, GNOT, GAOT baselines, which processes the global coefficient field and right-hand side to produce the solution on fine grid as output. We optimized baselines for all benchmarks and right-hand sides through a comprehensive grid search of architecture and training hyperparameters, as detailed in Appendix J.5-J.8. We additionally ablate the relative versus absolute loss for $\mathcal{G}_2$ and $\mathcal{G}_3$ (see Appendix K) and replace the Transolver++ predictor with GAOT (GAOT-GMsFEM; see Appendix L).

All numerical experiments were conducted on a single Nvidia Tesla A100 GPU with 80GB memory.

7.1 Results for steady-state problems

In this section, we present the results for the first three steady-state benchmarks. To determine the optimal training configuration for the models $\mathcal{G}_1$ ($L = 8$) and $\mathcal{G}_2$, we performed a comprehensive grid search over architectural and training hyperparameters. Full specifications are available in Appendix J. All resulting L_2 and H^1 metrics are calculated as the mean across 200 test samples.

Domain Sampling Strategy. Training on every local domain for each field would yield $121 \times 700 = 84,700$ samples and a significant imbalance of local domain types (81 interior, 36 border, and 4 corner). To mitigate this, we selected a balanced subset of 44 domains (12 interior, 28 border, and 4 corner), ensuring the border samples were distributed equally across all four boundaries.

We compared a **Random Sampling Strategy**, which selects domains randomly from the available set, and a **Selective Sampling Strategy**, a targeted approach where half of the interior and border domains are chosen based on local coefficient fields containing values close (± 100) to the global extrema to ensure exposure to high-contrast features (see Appendix F).

As shown in Table 1, the **Selective** approach consistently outperforms **Random** sampling, reducing L_2 relative errors by 15–20% and significantly lowering H^1 errors. This confirms that prioritizing high-contrast regions better captures solution gradients and improves overall model robustness.

Table 1: Performance comparison of Transolver-GMsFEM trained on samples selected by Selective Sampling Strategy or by Random Sampling.

	Selective		Random	
Dataset	L_2	H^1	L_2	H^1
Diffusion, (19)	**1.43**	**21.09**	1.71	23.76
Diffusion, (20)	**2.50**	**23.70**	3.15	27.66
Richards, (19)	**3.29**	**24.04**	4.07	28.50
Richards, (20)	**2.93**	**24.38**	3.71	28.79
Diffusion + mBCs, (19)	**1.09**	**18.48**	1.38	21.78
Diffusion + mBCs, (20)	**1.28**	**18.04**	1.56	20.57

Transolver-GMsFEM vs. Classical GMsFEM vs. baselines. To evaluate the efficacy of our proposed pipeline, we compare Transolver-GMsFEM against the classical GMsFEM and four baselines. For GMsFEM, we utilize the true eigenvectors $\{\phi_k^{\omega_i}\}_{k=1}^L$ (see Figures 20, 21, Appendix M) and partition of unity function χ_i. In contrast, Transolver-GMsFEM employs predicted basis functions $\{\widetilde{\phi}_k^{\omega_i}\}_{k=1}^L$ and corrections $\widetilde{\varepsilon}_i$ generated by trained models $\mathcal{G}_1$ and $\mathcal{G}_2$.

Detailed performance metrics for $L = 4$ and $L = 8$ basis functions, including L_2 and H^1 errors, are provided in Appendix N.1 (Tables 6 and 7). Figure 6 illustrates the L_2 error for $L = 8$ across both ID and OOD coefficient fields (see Appendix O for detailed OOD results). Furthermore, we test our method on a structurally different class of OOD coefficient fields (visualized in Figure 5). The L_2 errors for these unseen generative families are displayed in Figure 8, while the complete L_2 and H^1 metrics are provided in Appendix Table 13. The results demonstrate that Transolver-GMsFEM closely approximates the accuracy of classical GMsFEM, with nearly identical performance in the $L = 8$ configuration. Notably, our method significantly outperforms all baselines across all benchmarks. This gap is most evident across all OOD scenarios: while baseline models exhibit severe performance degradation – with errors frequently exceeding 20%.

The largest performance gap between Transolver-GMsFEM and the baselines occurs in the Diffusion with mixed BC benchmark with the spatially variable source term (20). To highlight this, Figure 22 visualizes the absolute error for a representative sample (No. 801), contrasting our approach with the Transolver++ baseline (see Appendix N.2), with corresponding input field shown in Figure 11.

Table 2 summarizes the generation times for $\{\phi_k^{\omega_i}\}_{k=1}^L$, where Transolver-GMsFEM achieves a $90\times$ to $100\times$ acceleration over classical GMsFEM. In 3D scenarios, the speedup reaches approximately $230\times$ for $L = 8$ (see Table 16, Appendix P). A comprehensive analysis of computational costs and wall-clock times is provided in Appendix R.

Table 2: Basis generation time: GMsFEM vs. Transolver-GMsFEM (Tr-GMsFEM).

L	Method	Time, s
4	GMsFEM (offline)	476.74
	Tr-GMsFEM (offline)	**5.32**
8	GMsFEM (offline)	650.98
	Tr-GMsFEM (offline)	**6.12**
4	GMsFEM (online)	6.66
	Tr-GMsFEM (online)	**1.28**

7.2 Results for time-dependent problems

In this section, we present the results for the two time-dependent benchmarks. For the offline stage, we utilized models $\mathcal{G}_1$ ($L = 4$) and $\mathcal{G}_2$, which were trained previously. To optimize the configuration for model $\mathcal{G}_3$ (which predicts $\phi_n^{\text{on},\omega_i}$; see Figure 19, Appendix M), we conducted a grid search across architectures and hyperparameters. Full details are in Appendix J.

Figure 7 shows mean L_2 metrics averaged across all test samples and all timesteps for ID and OOD cases. The mean L_2 and H^1 errors, averaged across all test samples and all timesteps, are summarized in Table 8. The hybrid approach (Transolver-GMsFEM (offline) + GMsFEM (online)) achieves an L_2 error close to the fully numerical GMsFEM (offline + online) baseline. Both the offline-only and fully integrated versions of Transolver-GMsFEM significantly outperform baselines (Transolver++, GNOT,

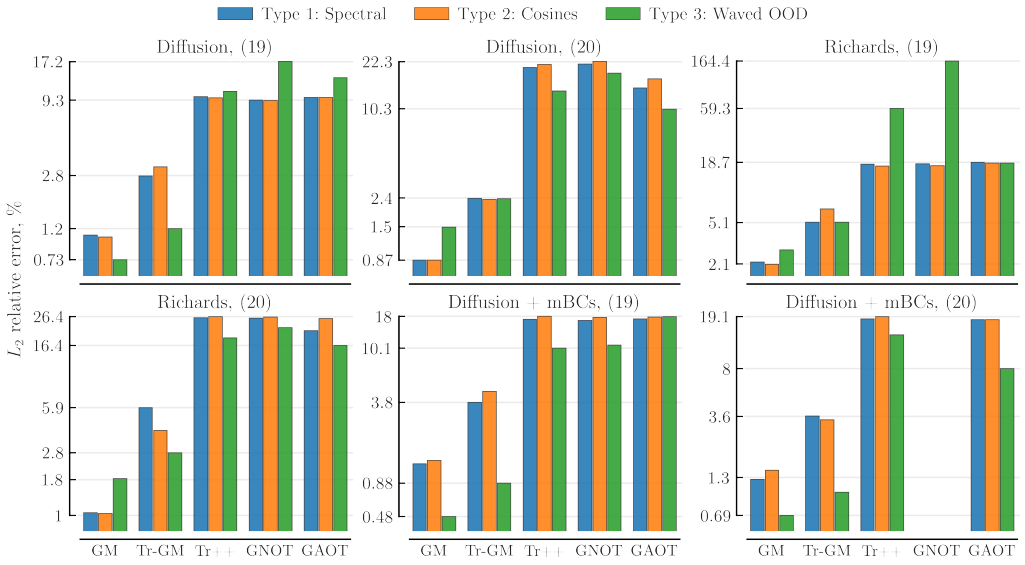


Figure 8: Relative L_2 errors for steady-state problems using OOD coefficients. Results evaluate the proposed method against classical GMsFEM and four baselines (see Table 13).

GAOT). Error evolution is illustrated in Figs. 23–26 (see Appendices N.3, N.4). Comprehensive results are provided in Appendix N (see Table 8) and Appendix O (see Tables 10, 11 for OOD coefficients and RHS).

Table 2 shows Transolver-GMsFEM (online) requires 1.28s per enrichment step vs. 6.66s for classical GMsFEM. For a 20-timestep simulation (4 updates), cumulative time drops from 26.64s to 5.12s. In 3D cases, the computation of online basis functions achieves an approximate 7× speedup (see Table 16, Appendix P).

8 Conclusion

In this work, we propose Transolver-GMsFEM, a novel hybrid framework that integrates state-of-the-art Transolver++ architectures with the GMsFEM. By leveraging neural PDE solvers, we mitigate the primary computational bottlenecks of the traditional GMsFEM. We also introduced a Selective Sampling Strategy – a targeted domain selection method that prioritizes extreme high-contrast features – enhancing model accuracy by 15-20% over standard random sampling.

We evaluated Transolver-GMsFEM on 2D and 3D, steady-state and time-dependent, linear and nonlinear high-contrast problems on irregular grids, including perforated domains with curved local boundaries and a 3D mesh with 1.38M cells. The method accelerates basis construction by up to 106× in the offline and 5× in the online stage in 2D, and up to 265× and 7× in 3D, and it remains 3.4× faster than a 48-core parallel GMsFEM. Its accuracy stays close to classical GMsFEM: for steady-state problems with $L = 8$, the L_2 error differs by at most 0.2%, and in time-dependent problems the predicted offline basis combined with GMsFEM online enrichment yields L_2 errors of 0.20–0.23% versus 0.16–0.18% for full GMsFEM, while the fully learned pipeline keeps them at 0.34%. Transolver-GMsFEM outperforms Transolver++, Transolver, GNOT, and GAOT on all benchmarks, and the gap widens under distribution shift: on shifted and structurally different coefficient families its L_2 error stays below 7%, and on an unseen source term the fully learned pipeline stays near 1%, whereas the baselines reach L_2 errors of up to 232%.

Although our results on perforated media demonstrate strong performance, all reported cases employ structured coarse grids. Nevertheless, the underlying principles of our hybrid approach are general and can be extended to unstructured coarse meshes and more complex geometries, as well as adapted to other predictor models.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgments and Disclosure of Funding

We are deeply grateful to Prof. Yalchin Efendiev (Texas A&M University) for his generous support, insightful guidance, and the many fruitful discussions that shaped this work. His deep expertise in multiscale methods and his thoughtful feedback were invaluable at every stage of this research.

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The authors declare no competing interests.

References

Abdul-Majid Wazwaz. *Partial differential equations*. CRC Press, 2026.

Lawrence C Evans. *Partial differential equations*, volume 19. American mathematical society, 2022.

Denis Spiridonov, Maria Vasilyeva, Eric T Chung, Yalchin Efendiev, and Raghavendra Jana. Multiscale model reduction of the unsaturated flow problem in heterogeneous porous media with rough surface topography. *Mathematics*, 8(6):904, 2020.

Xinguang He and Li Ren. An adaptive multiscale finite element method for unsaturated flow problems in heterogeneous porous media. *Journal of Hydrology*, 374(1-2):56–70, 2009.

Maria Vasilyeva, SM Mallikarjunaiah, and D Palaniappan. Multiscale model reduction technique for fluid flows with heterogeneous porous inclusions. *Journal of Computational and Applied Mathematics*, 424:114976, 2023.

Sergei Stepanov, Maria Vasilyeva, and Vasiliy I Vasil'ev. Generalized multiscale discontinuous Galerkin method for solving the heat problem with phase change. *Journal of Computational and Applied Mathematics*, 340: 645–652, 2018.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023a.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-FNO—An enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023b.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pages 12556–12569. PMLR, 2023.

Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for pdes on arbitrary domains. *arXiv preprint arXiv:2505.18781*, 2025.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An Accurate Neural Solver for Pdes on Million-Scale Geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Yalchin Efendiev, Juan Galvis, and Thomas Y Hou. Generalized multiscale finite element methods (GMsFEM). *Journal of computational physics*, 251:116–135, 2013.

Yalchin Efendiev, Guanglian Li, Michael Presho, et al. Generalized multiscale finite element methods: Oversampling strategies. *International Journal for Multiscale Computational Engineering*, 12(6), 2014.

Yalchin Efendiev, Juan Galvis, and Xiao-Hui Wu. Multiscale finite element methods for high-contrast problems using local spectral basis functions. *Journal of Computational Physics*, 230(4):937–955, 2011.

Eric T Chung, Yalchin Efendiev, and Guanglian Li. An adaptive GMsFEM for high-contrast flow problems. *Journal of Computational Physics*, 273:54–76, 2014.

Alexander Rudikov, Vladimir Fanaskov, Sergei Stepanov, Buzheng Shan, Ekaterina Muravleva, Yalchin Efendiev, and Ivan Oseledets. Locally Subspace-Informed Neural Operators for Efficient Multiscale PDE Solving. *arXiv preprint arXiv:2505.16030*, 2025.

Olgierd C Zienkiewicz and Robert Leroy Taylor. *The finite element method set*. Elsevier, 2005.

Pavel Ŝolín. *Partial differential equations and the finite element method*. John Wiley & Sons, 2005.

Eric T Chung, Yalchin Efendiev, and Wing Tat Leung. Residual-driven online generalized multiscale finite element methods. *Journal of Computational Physics*, 302:176–190, 2015.

Eugene Wong. Stochastic processes in information and dynamical systems. *(No Title)*, 1971.

Benjamin Ganis, Hector Klie, Mary F Wheeler, Tim Wildey, Ivan Yotov, and Dongxiao Zhang. Stochastic collocation and mixed finite elements for flow in porous media. *Computer methods in applied mechanics and engineering*, 197(43-44):3547–3559, 2008.

Roland Haverkamp, Michel Vauclin, Jaoudat Touma, PJ Wierenga, and Georges Vachaud. A comparison of numerical simulation models for one-dimensional infiltration. *Soil Science Society of America Journal*, 41(2): 285–294, 1977.

A Local Domain Classification

Local domains ω_i vary in shape depending on the position of the coarse node x_i within the computational domain $\mathcal{D}$. In 2D, we classify local domains into **three geometric types** based on the number of coarse elements they contain:

- **Interior domains**: consisting of 4 coarse elements K_j, where x_i is an interior node of $\mathcal{T}^H$;
- **Border domains**: consisting of 2 coarse elements K_j, where x_i lies on $\partial\mathcal{D}$ but not at a corner;
- **Corner domains**: consisting of 1 coarse element K_j, where x_i is a corner node of $\mathcal{D}$.

In 3D, there are **four types** of local domains:

- **Interior domains**: consisting of 8 coarse elements K_j, where x_i is an interior node of $\mathcal{T}^H$;
- **Face domains**: consisting of 4 coarse elements K_j, where x_i lies on the interior of a face of $\partial\mathcal{D}$;
- **Edge domains**: consisting of 2 coarse elements K_j, where x_i lies on an edge of $\partial\mathcal{D}$ but is not a corner;
- **Corner domains**: consisting of 1 coarse element K_j, where x_i is a corner node of $\mathcal{D}$.

This classification accounts for the geometric variability of local domains and the relative placement of the coarse node x_i shared by all cells in each domain. Each local domain $\omega_i \subset \mathcal{D}$ is discretized using the fine grid $\mathcal{T}^h$. An example of different geometric types in 2D for $N_v = 5$ is shown in Figure 9.

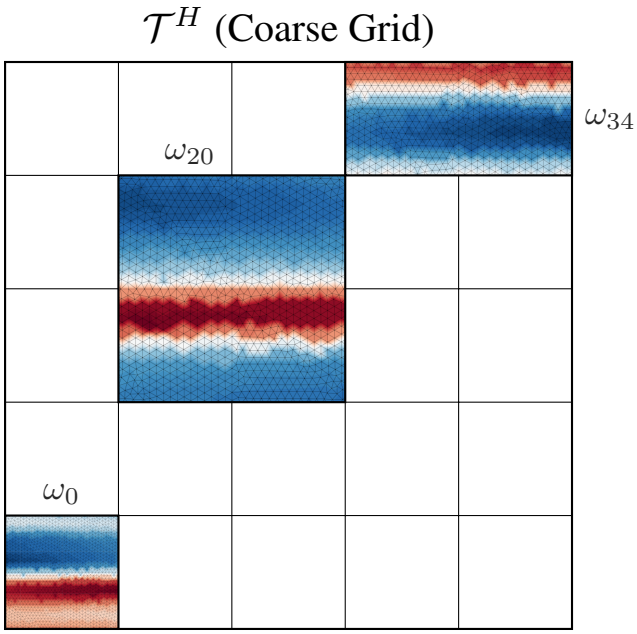


Figure 9: Illustration of a 5×5 ($N_c = 36$) coarse structured grid $\mathcal{T}^H$ showing local domains of three geometric types: **corner** domain ω_0, **border** domain ω_{34}, and **interior** domain ω_{20}.

B Local Spectral Problem

To define the multiscale offline basis functions $\{\psi_k^{\omega_i}\}_{k=1}^{L}$, we introduce a local spectral problem. For each domain $\omega_i \subset \mathcal{D}$, let $V^h(\omega_i)$ denote the restriction of the fine-scale finite element space to ω_i, with $l_i = \dim V^h(\omega_i)$. We define the local stiffness matrix A^{ω_i} by

$$v^\top A^{\omega_i} w = \int_{\omega_i} \kappa \nabla v \cdot \nabla w \, dx \quad \forall v, w \in V^h(\omega_i),$$

and the local mass matrix M^{ω_i} by

$$v^\top M^{\omega_i} w = \int_{\omega_i} \kappa \, v \, w \, dx \quad \forall v, w \in V^h(\omega_i).$$

We consider the generalized eigenvalue problem

$$A^{\omega_i} \phi = \lambda M^{\omega_i} \phi,$$

and denote its eigenvalues and eigenvectors by $\{\lambda_k^{\omega_i}\}_{k=1}^{l_i}$ and $\{\phi_k^{\omega_i}\}_{k=1}^{l_i}$, respectively. Note that the eigenvectors $\{\phi_k^{\omega_i}\}_{k=1}^{l_i}$ form an orthonormal basis of $V^h(\omega_i)$ with respect to the M^{ω_i} inner product.

C Online Enrichment Procedure

We provide a detailed derivation of the online basis function enrichment strategy described in Section 5.

Consider a given coarse neighborhood ω_i, and suppose that we need to add a basis function $\phi^{\text{on}} \in V_i = H_0^1(\omega_i) \cap V$ on the i-th coarse neighborhood ω_i. Let $V_{\text{off}}^{m+1} = V_{\text{off}} \oplus \text{span}\{\phi^{\text{on}}\}$ be the new approximation space, and let $u_{\text{ms}}^{m+1} \in V_{\text{off}}^{m+1}$ be the corresponding GMsFEM solution. It is easy to see that u_{ms}^{m+1} satisfies

$$\|u - u_{\text{ms}}^{m+1}\|_V^2 = \inf_{v \in V_{\text{off}}^{m+1}} \|u - v\|_V^2,$$

where $\|v\|_V^2 = a(v, v)$. Taking $v = u_{\text{ms}}^m + \alpha \phi^{\text{on}} \in V_{\text{off}}^{m+1}$, where α is a scalar to be determined, we have

$$\|u - u_{\text{ms}}^{m+1}\|_V^2 \leqslant \|u - u_{\text{ms}}^m - \alpha \phi^{\text{on}}\|_V^2 = \|u - u_{\text{ms}}^m\|_V^2 - 2\alpha a(u - u_{\text{ms}}^m, \phi^{\text{on}}) + \alpha^2 a(\phi^{\text{on}}, \phi^{\text{on}}).$$

The last two terms in the above inequality measure the amount of reduction in error when the new basis function ϕ^{on} is added to the space V_{off}^m. To determine ϕ^{on}, we first assume that the basis function ϕ^{on} is normalized so that

$a(\phi^{\text{on}}, \phi^{\text{on}}) = 1$. In order to maximize the reduction in error, one can show that $\alpha = a(u - u_{\text{ms}}^m, \phi^{\text{on}})$. Using this choice of α, we have

$$\|u - u_{\text{ms}}^{m+1}\|_V^2 \leqslant \|u - u_{\text{ms}}^m\|_V^2 - |a(u - u_{\text{ms}}^m, \phi^{\text{on}})|^2.$$

Since $\phi^{\text{on}} \in V_i \subset V$ and using the Galerkin orthogonality $a(u - u_{\text{ms}}^m, v) = R_i(v)$ for all $v \in V_i$, where $R_i(v) = (f, v) - a(u_{\text{ms}}^m, v)$ is the local residual functional, we have

$$\|u - u_{\text{ms}}^{m+1}\|_V^2 \leqslant \|u - u_{\text{ms}}^m\|_V^2 - |R_i(\phi^{\text{on}})|^2.$$

We then find $\phi^{\text{on}} \in V_i$ to maximize the local residual $|R_i(\phi^{\text{on}})|^2$. The functional norm of the residual R_i is defined as

$$\|R_i\|_{V_i^*} = \sup_{\substack{v \in V_i \\ v \neq 0}} \frac{|R_i(v)|}{\|v\|_V} = \sup_{\substack{v \in V_i \\ \|v\|_V = 1}} |R_i(v)|.$$

Clearly, the maximum of $|R_i(\phi^{\text{on}})|$ over all $\phi^{\text{on}} \in V_i$ with $\|\phi^{\text{on}}\|_V = 1$ equals the functional norm $\|R_i\|_{V_i^*}$. Hence, the new online basis function $\phi^{\text{on}} \in V_i$ can be obtained by solving the local problem

$$a(\phi^{\text{on}}, v) = R_i(v), \quad \forall v \in V_i, \tag{9}$$

and then normalizing it in the energy norm. In addition, the residual norm $\|R_i\|_{V_i^*}$ provides a measure of the amount of reduction in energy error.

C.1 Convergence Analysis

The convergence of the online adaptive procedure has been studied in Chung et al. [2014, 2015]. To simplify notation, we write $r_i = \|R_i\|_{V_i^*}$. It has been shown that

$$\|u - u_{\text{ms}}^{m+1}\|_V^2 \leqslant \left(1 - \frac{\lambda_{n_i+1}^{\omega_i}}{C_{\text{err}}} \cdot \frac{r_i^2 \big(\lambda_{n_i+1}^{\omega_i}\big)^{-1}}{\sum_{k=1}^{N_c} r_k^2 \big(\lambda_{n_k+1}^{\omega_k}\big)^{-1}}\right) \|u - u_{\text{ms}}^m\|_V^2, \tag{10}$$

where n_i is the number of offline basis functions in ω_i, $\lambda_{n_i+1}^{\omega_i}$ is the (n_i+1)-th eigenvalue from the local spectral problem in ω_i, and C_{err} is a constant independent of the contrast.

The above inequality gives the convergence of the online adaptive GMsFEM with a precise convergence rate for the case when one online basis function is added per iteration. The estimate (10) shows that the eigenvectors corresponding to very small eigenvalues need to be included in the offline space in order to achieve a significant error reduction in one iteration. We call this algorithm the online adaptive GMsFEM since online basis functions are used for enrichment.

D Online GMsFEM for Time-Dependent Problems

We provide details of the online enrichment procedure for time-dependent problems. Let $u_{\text{ms}}^{m,n}$ and $u_{\text{ms}}^{m+1,n}$ denote the multiscale solutions at enrichment level m and $m+1$, respectively, at time step n.

D.1 Coarse Problem with Offline Basis

At enrichment steps, we compute the solution $u_{\text{ms}}^{m,n}$ using only the offline basis. The projected stiffness matrix A_0^n and the projected forcing vector b_0^n are defined as

$$A_0^n = R_0 A^n R_0^\top, \quad b_0^n = R_0 b^n,$$

The coarse solution $u_0^{m,n}$ satisfies the time-stepping scheme

$$\frac{u_0^{m,n} - u_0^{m+1,n-1}}{\tau} + A_0^n u_0^{m,n} = b_0^n, \tag{11}$$

where τ is a time-step size, $u_0^{m+1,n-1} = R_0 u_{\text{ms}}^{m+1,n-1}$ is the projection of the online GMsFEM solution at time step $n-1$ onto the offline space. For the initial condition ($n = 0$), we set $u_{\text{ms}}^{m+1,n-1} = 0$. The fine-scale solution is then reconstructed as $u_{\text{ms}}^{m,n} = R_0^\top u_0^{m,n}$.

D.2 Local Problem for Online Basis Functions

For each coarse neighborhood ω_i, the online basis function $\phi_n^{\text{on},\omega_i}$ is obtained by solving a local problem (9). The weak formulation reads: find $\phi_n^{\text{on},\omega_i} \in V_i$ such that

$$\int_{\omega_i} \frac{\phi_n^{\text{on},\omega_i}}{\tau} v \, dx + \int_{\omega_i} \kappa \nabla \phi_n^{\text{on},\omega_i} \cdot \nabla v \, dx = R_i^n(v), \quad \forall v \in V_i, \tag{12}$$

where the local residual functional $R_i^n : V_i \to \mathbb{R}$ is defined by

$$R_i^n(v) = \int_{\omega_i} \left(\frac{u_{\text{ms}}^{m+1,n-1}|_{\omega_i}}{\tau} + f \right) v \, dx - \int_{\omega_i} \frac{u_{\text{ms}}^{m,n}|_{\omega_i}}{\tau} v \, dx - \int_{\omega_i} \kappa \nabla u_{\text{ms}}^{m,n}|_{\omega_i} \cdot \nabla v \, dx.$$

Here, $u_{\text{ms}}^{m,n}|_{\omega_i}$ denotes the offline GMsFEM solution at time step n restricted to ω_i, and $u_{\text{ms}}^{m+1,n-1}|_{\omega_i}$ denotes the online GMsFEM solution at time step $n-1$ restricted to ω_i.

D.3 Enriched Projection Matrix

After computing the online basis functions $\{\phi_n^{\text{on},\omega_i}\}$ for all ω_i, the projection matrix $R_0^{m+1,n}$ is updated as

$$R_0^{m+1,n} = \left[\psi_1^{\omega_1}, \ldots, \psi_L^{\omega_1}, \ldots, \psi_1^{\omega_{N_c}}, \ldots, \psi_L^{\omega_{N_c}}, \vartheta_n^{\omega_1}, \ldots, \vartheta_n^{\omega_{N_c}} \right]^\top,$$

where $\psi_k^{\omega_i} = \chi_i \, \phi_k^{\omega_i}$ is the offline multiscale basis function, $\vartheta_n^{\omega_i} = \chi_i \phi_n^{\text{on},\omega_i}$ is the online multiscale basis function, and $\chi_i = \varepsilon_i + \chi_i^0$ is the partition of unity function for ω_i.

D.4 Coarse Problem with Enriched Basis

Using the $R_0^{m+1,n}$, the coarse-scale system becomes

$$A_0^{m+1,n} = R_0^{m+1,n} A^n (R_0^{m+1,n})^\top, \quad b_0^{m+1,n} = R_0^{m+1,n} b^n.$$

The projection of the online GMsFEM solution on coarse space $u_0^{m+1,n}$ satisfies

$$\frac{u_0^{m+1,n} - u_0^{m+1,n-1}}{\tau} + A_0^{m+1,n} u_0^{m+1,n} = b_0^{m+1,n}, \tag{13}$$

and the fine-scale solution is reconstructed as $u_{\text{ms}}^{m+1,n} = \left(R_0^{m+1,n}\right)^\top u_0^{m+1,n}$.

D.5 Summary of the Algorithm

At enrichment steps ($n \mod N_{\text{enrich}} = 0$):

1. Solve the coarse problem (11) with R_0 to obtain $u_{\text{ms}}^{m,n}$.
2. For each ω_i, solve (8) to obtain $\phi_n^{\text{on},\omega_i}$.
3. Construct $R_0^{m+1,n}$ by appending $\{\chi_i \phi_n^{\text{on},\omega_i}\}_{i=1}^{N_c}$ to R_0.
4. Solve the coarse problem (13) with $R_0^{m+1,n}$ to obtain $u_{\text{ms}}^{m+1,n}$.

At non-enrichment steps:

1. Retain the current projection matrix $R_0^{m+1,n}$.
2. Solve only (13) to advance $u_{\text{ms}}^{m+1,n}$.

E Theoretical Guarantees and Proofs

E.1 Setup

Let $\mathcal{D} \subset \mathbb{R}^d$, $d \in \{2, 3\}$, be the computational domain and $\mathcal{T}^H$ a coarse grid with nodes $\{x_i\}_{i=1}^{N_c}$ and coarse neighborhoods $\omega_i = \bigcup\{K \in \mathcal{T}^H : x_i \in \overline{K}\}$. Let $V = V_0^h(\mathcal{D})$ denote the conforming fine-scale finite element space with homogeneous Dirichlet boundary conditions, equipped with the global energy norm $\|u\|_V^2 = a(u,u)$, where

$$a(u,v) = \int_{\mathcal{D}} \kappa \, \nabla u \cdot \nabla v \, dx.$$

On each ω_i we use the local energy form

$$a_{\omega_i}(u,v) = \int_{\omega_i} \kappa \, \nabla u \cdot \nabla v \, dx$$

with $\|u\|_{V(\omega_i)}^2 = a_{\omega_i}(u,u)$, and the κ-weighted inner product

$$s_{\omega_i}(u,v) = \int_{\omega_i} \kappa \, u \, v \, dx$$

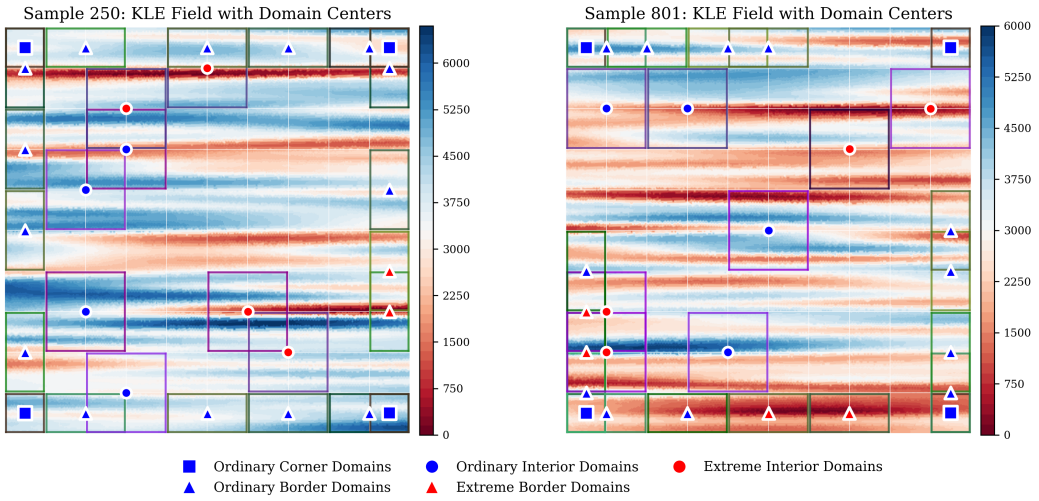


Figure 10: Example of subdomain selection for two sample coefficient fields (No. 250 and No. 801). The centers of the chosen domains ω_i are displayed alongside their corresponding boundaries $\partial\omega_i$.

with $\|u\|^2_{s(\omega_i)} = s_{\omega_i}(u,u)$; the same coefficient κ serves as the spectral weight throughout.

Let $W_i = \operatorname{span}\{\phi_k^{\omega_i}\}_{k=1}^{L}$ be the exact local GMsFEM subspace spanned by the first L eigenvectors of

$$a_{\omega_i}(\phi_k^{\omega_i}, v) = \lambda_k^{\omega_i}\, s_{\omega_i}(\phi_k^{\omega_i}, v) \quad \forall v, \tag{14}$$

ordered by $\lambda_1^{\omega_i} \leqslant \cdots \leqslant \lambda_L^{\omega_i} \leqslant \ldots$. Here $\lambda_1^{\omega_i} = 0$ with constant eigenvector, so $\mathbb{1} \in W_i$. Let $\widetilde{W}_i = \operatorname{span}\{\widetilde{\phi}_k^{\omega_i}\}_{k=1}^{L}$ be the network-predicted subspace.

Let $P_{W_i}, P_{\widetilde{W}_i}$ be the s_{ω_i}-orthogonal projectors onto $W_i, \widetilde{W}_i$. Fix s_{ω_i}-orthonormal bases $\{\phi_k^{\omega_i}\}$ of W_i and $\{\widetilde{\phi}_k^{\omega_i}\}$ of $\widetilde{W}_i$ (in matrix form, M^{ω_i}-orthonormal), encoded as isometric embeddings

$$Q_{W_i} : \mathbb{R}^L \to W_i, \quad Q_{W_i} c = \sum_k c_k\, \phi_k^{\omega_i}, \qquad Q_{\widetilde{W}_i} : \mathbb{R}^L \to \widetilde{W}_i, \quad Q_{\widetilde{W}_i} c = \sum_k c_k\, \widetilde{\phi}_k^{\omega_i}.$$

Orthonormality of the bases gives $Q_{W_i}^\top Q_{W_i} = Q_{\widetilde{W}_i}^\top Q_{\widetilde{W}_i} = I_L$, $P_{W_i} = Q_{W_i} Q_{W_i}^\top$, $P_{\widetilde{W}_i} = Q_{\widetilde{W}_i} Q_{\widetilde{W}_i}^\top$, and $Q_{W_i}^\top Q_{\widetilde{W}_i}$ is the cross-Gram matrix $\big(s_{\omega_i}(\phi_k^{\omega_i}, \widetilde{\phi}_m^{\omega_i})\big)_{km}$. (Here $Q_{W_i}^\top$ denotes the s_{ω_i}-adjoint of Q_{W_i}; in matrix form $Q_{W_i}^\top M^{\omega_i}$.)

The partition of unity $\{\chi_i\}_{i=1}^{N_c}$ is assumed to satisfy

$$\sum_i \chi_i = 1, \quad \operatorname{supp}\chi_i \subset \overline{\omega_i}, \quad 0 \leqslant \chi_i \leqslant 1, \quad |\nabla\chi_i| \leqslant C_\chi H^{-1}, \tag{15}$$

with C_χ independent of H, of i, and of the contrast. These properties hold for the piecewise (bi/tri)linear partition of unity $\{\chi_i^0\}$; for the multiscale partition of unity used in the numerical method the gradient bound may degrade with the contrast, in which case s_{ω_i} is replaced by the $\widetilde{\kappa}$-weighted form with $\widetilde{\kappa} = \kappa \sum_j |\nabla\chi_j|^2$ throughout. In our experiments the spectral problem is solved with the weight κ (see Appendix B); the analysis therefore applies verbatim only when the gradient bound in (15) holds for the partition of unity in use. We assume the standard boundary treatment of GMsFEM so that the trial spaces below are conforming, $V_{\mathrm{off}}, \widetilde{V}_{\mathrm{off}} \subset V$. The theorem below concerns model $\mathcal{G}_1$: both interpolants use the same, exact partition of unity, and the analysis is for the symmetric linear steady problem, for which Céa's lemma holds with constant one. In particular, the prediction errors of $\mathcal{G}_2$ (partition of unity correction) and $\mathcal{G}_3$ (online basis functions), as well as nonlinear and time-dependent problems, are not covered by the analysis below.

The multiscale interpolants and trial spaces are

$$I_{\mathrm{ms}} v = \sum_i \chi_i\, P_{W_i}(v|_{\omega_i}) \in V_{\mathrm{off}}, \quad \widetilde{I}_{\mathrm{ms}} v = \sum_i \chi_i\, P_{\widetilde{W}_i}(v|_{\omega_i}) \in \widetilde{V}_{\mathrm{off}},$$

where $V_{\mathrm{off}} = \operatorname{span}\{\chi_i \phi_k^{\omega_i}\}$ and $\widetilde{V}_{\mathrm{off}} = \operatorname{span}\{\chi_i \widetilde{\phi}_k^{\omega_i}\}$. The GMsFEM solutions $u_{\mathrm{ms}} \in V_{\mathrm{off}}$, $\widetilde{u}_{\mathrm{ms}} \in \widetilde{V}_{\mathrm{off}}$ are the Galerkin projections of the exact solution u; by Céa's lemma $\widetilde{u}_{\mathrm{ms}}$ minimizes the energy error over $\widetilde{V}_{\mathrm{off}}$.

We make three standing assumptions.

Assumption E.1 (constant reproduction)**.** For every coarse neighborhood, $\mathbb{1} \in \widetilde{W}_i$. Together with $\mathbb{1} \in W_i$ this gives $P_{W_i}\mathbb{1} = P_{\widetilde{W}_i}\mathbb{1} = \mathbb{1}$, hence $(P_{W_i} - P_{\widetilde{W}_i})c = 0$ for any constant c. This assumption can be enforced exactly at negligible cost, e.g. by predicting $L-1$ functions, appending the constant vector, and re-orthonormalizing, so that $\dim \widetilde{W}_i = L$.

Assumption E.2 (local spectral consistency)**.** There is a constant $\Lambda^{(i)} > 0$ with

$$a_{\omega_i}(g,\, g) \leqslant \Lambda^{(i)}\, s_{\omega_i}(g,\, g) \quad \forall g \in W_i + \widetilde{W}_i.$$

This Rayleigh bound is stated on the *joint* subspace $W_i + \widetilde{W}_i$ because $w_i = (P_{W_i} - P_{\widetilde{W}_i})u$ generally lies in $W_i + \widetilde{W}_i$ but not in W_i, so the bound $a_{\omega_i}(\cdot,\cdot) \leqslant \lambda_L^{\omega_i} s_{\omega_i}(\cdot,\cdot)$ valid on W_i does not apply; when $\widetilde{W}_i = W_i$ one has $\Lambda^{(i)} = \lambda_L^{\omega_i}$. For the κ-weighted spectral problem used here, a_{ω_i} scales as H^{d-2} and s_{ω_i} as H^d, so the Rayleigh quotient grows as H^{-2} and the bound scales as $\Lambda^{(i)} = \Lambda_0^{(i)}/H^2$ with $\Lambda_0^{(i)} = O(1)$ mesh-independent. The constant $\Lambda^{(i)}$ is finite for any fixed pair of finite-dimensional subspaces; the content of the assumption is its *uniformity* in H and N_c. For κ piecewise constant on the fine mesh, an elementwise inverse inequality gives $\Lambda^{(i)} \leqslant C_{\mathrm{inv}} h^{-2}$ with C_{inv} independent of the contrast (κ cancels on each fine element). In general this only yields $\Lambda_0^{(i)} = O\big((H/h)^2\big)$; the uniform bound $\Lambda_0^{(i)} = O(1)$ is therefore an assumption on the smoothness of the network output.

Assumption E.3 (uniform local spectral gap)**.** There is a constant $C_{P,0} > 0$, independent of H and of i, such that

$$\lambda_2^{\omega_i} \;\geqslant\; \frac{1}{C_{P,0}\, H^2} \quad \text{for all } i, \quad \text{i.e.,} \quad C_{P,0} := \max_i \big(H^2 \lambda_2^{\omega_i}\big)^{-1}.$$

Since functions with vanishing s_{ω_i}-weighted mean are s_{ω_i}-orthogonal to the constant eigenvector $\phi_1^{\omega_i}$, the min–max characterization of (14) yields the weighted Poincaré inequality $\|g\|^2_{s(\omega_i)} \leqslant C_P\, \|g\|^2_{V(\omega_i)}$ with $C_P = C_{P,0}\, H^2$ for every such g. Note that $C_{P,0}$ is in general contrast-dependent: for high-contrast channelized coefficients, $\lambda_2^{\omega_i}$ can be small, which is precisely why several eigenvectors are retained in GMsFEM.

Finally, recall the classical multiscale approximation estimate of Efendiev et al. [2011]: with $\Lambda^* = \min_i \lambda_{L+1}^{\omega_i}$ and the mesh-independent spectral gap $\Lambda_0^* := H^2\, \Lambda^*$,

$$\|u - I_{\mathrm{ms}} u\|_V \;\leqslant\; \frac{C_0}{\sqrt{\Lambda^*}}\, \|f\|_{L^2(\mathcal{D})} \;=\; C_0\, \frac{H}{\sqrt{\Lambda_0^*}}\, \|f\|_{L^2(\mathcal{D})}, \tag{16}$$

where C_0 is independent of the contrast $\kappa(x)$.[1]

Definition E.4 (overlap constant and subspace loss)**.** Let $\kappa_{\mathrm{ov}} = \max_K \#\{i : K \subset \omega_i\}$. Since a d-cube has 2^d vertices, $\kappa_{\mathrm{ov}} = 2^d$ ($\kappa_{\mathrm{ov}} = 4$ in 2D, 8 in 3D), independently of N_c. The local and worst-case subspace losses are

$$\mathcal{L}_1^{(i)} = L - \left\| Q_{W_i}^\top Q_{\widetilde{W}_i} \right\|_F^2, \quad \mathcal{L}_{\max} = \max_{1 \leqslant i \leqslant N_c} \mathcal{L}_1^{(i)}.$$

In terms of the principal angles $\{\theta_k\}_{k=1}^L$ between W_i and $\widetilde{W}_i$ (in the s_{ω_i}-inner product), $\left\| Q_{W_i}^\top Q_{\widetilde{W}_i} \right\|_F^2 = \sum_k \cos^2\theta_k$, so $0 \leqslant \mathcal{L}_1^{(i)} \leqslant L$, with $\mathcal{L}_1^{(i)} = 0$ iff $\widetilde{W}_i = W_i$.

Lemma E.5 (Hilbert–Schmidt distance of the projectors)**.** *For the rank-L s_{ω_i}-orthogonal projectors $P_{W_i}, P_{\widetilde{W}_i}$, $\left\| P_{W_i} - P_{\widetilde{W}_i} \right\|_{\mathrm{HS}}^2 = 2\, \mathcal{L}_1^{(i)}$, where $\|\cdot\|_{\mathrm{HS}}$ is the Hilbert–Schmidt norm in the s_{ω_i}-inner product. Moreover, $\|P_{W_i} - P_{\widetilde{W}_i}\|_{\mathrm{op}}^2 \leqslant \mathcal{L}_1^{(i)}$. The subspace loss $\mathcal{L}_1$ was introduced by Rudikov et al. [2025]. We restate it for the s_{ω_i}-weighted projectors used here.*

Proof. Idempotence and rank give $\mathrm{tr}(P_{W_i}) = \mathrm{tr}(P_{\widetilde{W}_i}) = L$. Substituting the factorizations and using cyclicity of the trace across $Q_{W_i}\,(Q_{W_i}^\top Q_{\widetilde{W}_i})\, Q_{\widetilde{W}_i}^\top$,

$$\mathrm{tr}(P_{W_i} P_{\widetilde{W}_i}) = \mathrm{tr}\big((Q_{W_i}^\top Q_{\widetilde{W}_i})(Q_{\widetilde{W}_i}^\top Q_{W_i})\big) = \left\| Q_{W_i}^\top Q_{\widetilde{W}_i} \right\|_F^2,$$

using $Q_{\widetilde{W}_i}^\top Q_{W_i} = (Q_{W_i}^\top Q_{\widetilde{W}_i})^\top$. Since $P_{W_i} - P_{\widetilde{W}_i}$ is s_{ω_i}-self-adjoint,

$$\begin{aligned} \left\| P_{W_i} - P_{\widetilde{W}_i} \right\|_{\mathrm{HS}}^2 &= \mathrm{tr}(P_{W_i}) + \mathrm{tr}(P_{\widetilde{W}_i}) - 2\ \mathrm{tr}(P_{W_i} P_{\widetilde{W}_i}) \\ &= 2L - 2\, \left\| Q_{W_i}^\top Q_{\widetilde{W}_i} \right\|_F^2 = 2\, \mathcal{L}_1^{(i)}. \end{aligned}$$

[1] The estimate is stated here in the form needed downstream; C_0 absorbs domain- and partition-of-unity-dependent factors and is contrast-independent owing to the spectral construction of the basis Efendiev et al. [2011]. Sharp variants in the cited literature employ a weighted data norm; only the H-scaling and the dependence on the spectral gap matter for what follows.

For the operator norm, recall that for two subspaces of equal dimension L the nonzero eigenvalues of the self-adjoint operator $P_{W_i} - P_{\widetilde{W}_i}$ are $\pm \sin\theta_k$, $k = 1, \dots, L$. Hence $\|P_{W_i} - P_{\widetilde{W}_i}\|_{\mathrm{op}}^2 = \max_k \sin^2\theta_k \leqslant \sum_k \sin^2\theta_k = L - \sum_k \cos^2\theta_k = \mathcal{L}_1^{(i)}$. □

Lemma E.6 (finite overlap)**.** *For any v_i with $\mathrm{supp}(v_i) \subset \overline{\omega_i}$, $\big\|\sum_i v_i\big\|_V^2 \leq \kappa_{\mathrm{ov}} \sum_i \|v_i\|_V^2$.*

Proof. On each coarse cell K only the v_i with $i \in \mathcal{I}(K) = \{i : K \subset \omega_i\}$ are nonzero, and $|\mathcal{I}(K)| \leqslant \kappa_{\mathrm{ov}}$. Cauchy–Schwarz gives $\big|\nabla \sum_{i\in\mathcal{I}(K)} v_i\big|^2 \leqslant \kappa_{\mathrm{ov}} \sum_{i\in\mathcal{I}(K)} |\nabla v_i|^2$. Multiplying by κ, integrating over K, summing over K, and using $\bigcup_{K\subset\omega_i} K = \omega_i$ yields the claim. □

Lemma E.7 (constant annihilation and spectral/Poincaré bounds)**.** *Let $\bar{u}_{\omega_i} = s_{\omega_i}(u, \mathbb{1})/s_{\omega_i}(\mathbb{1}, \mathbb{1})$ and $w_i = (P_{W_i} - P_{\widetilde{W}_i})\, u|_{\omega_i}$. Then:*

(i) under Assumption E.1, $w_i = (P_{W_i} - P_{\widetilde{W}_i})\,(u - \bar{u}_{\omega_i})|_{\omega_i}$;

(ii) under Assumption E.2, $a_{\omega_i}(w_i, w_i) \leqslant \Lambda^{(i)}\ \|w_i\|_{s(\omega_i)}^2$;

(iii) under Assumption E.3, $\big\|u - \bar{u}_{\omega_i}\big\|_{s(\omega_i)}^2 \leqslant C_P\ \|u\|_{V(\omega_i)}^2$ with $C_P = C_{P,0}\, H^2$, uniformly in i.

Proof. (i) By Assumption E.1, $(P_{W_i} - P_{\widetilde{W}_i})\bar{u}_{\omega_i} = 0$, so $w_i = (P_{W_i} - P_{\widetilde{W}_i})(u - \bar{u}_{\omega_i})$. (ii) Since $w_i \in W_i + \widetilde{W}_i$, this is Assumption E.2 applied to $g = w_i$. (iii) The function $u - \bar{u}_{\omega_i}$ has vanishing s_{ω_i}-weighted mean over ω_i, so Assumption E.3 applies; the conclusion follows from $\nabla \bar{u}_{\omega_i} = 0$. □

E.2 Local and global estimates

Proposition E.8 (local energy estimate)**.** *With $v_i = \chi_i w_i$ and $C^2 = \max_i 2\, C_{P,0}\big(\Lambda_0^{(i)} + C_\chi^2\big)$, which is independent of H and N_c,*

$$\|v_i\|_V^2 \leqslant C^2\, \mathcal{L}_1^{(i)}\ \|u\|_{V(\omega_i)}^2 .$$

Proof. By the Leibniz rule and $(a+b)^2 \leqslant 2a^2 + 2b^2$,

$$\|v_i\|_V^2 \leqslant 2\int_{\omega_i} \kappa\, \chi_i^2 |\nabla w_i|^2\ +\ 2\int_{\omega_i} \kappa\, |\nabla\chi_i|^2 |w_i|^2 .$$

Since $0 \leqslant \chi_i \leqslant 1$, the first term is $\leqslant a_{\omega_i}(w_i, w_i)$. For the second, (15) gives $|\nabla\chi_i| \leqslant C_\chi H^{-1}$, hence $\int_{\omega_i} \kappa |\nabla\chi_i|^2 |w_i|^2 \leqslant C_\chi^2 H^{-2}\, \|w_i\|_{s(\omega_i)}^2$. By Lemma E.7(ii),

$$\|v_i\|_V^2 \leqslant 2\, a_{\omega_i}(w_i, w_i) + 2\, C_\chi^2 H^{-2}\, \|w_i\|_{s(\omega_i)}^2 \leqslant 2\big(\Lambda^{(i)} + C_\chi^2 H^{-2}\big)\ \|w_i\|_{s(\omega_i)}^2 .$$

By Lemma E.7(i), the operator-norm bound of Lemma E.5, and Lemma E.7(iii),

$$\|w_i\|_{s(\omega_i)}^2 = \Big\|(P_{W_i} - P_{\widetilde{W}_i})(u - \bar{u}_{\omega_i})\Big\|_{s(\omega_i)}^2 \leqslant \|P_{W_i} - P_{\widetilde{W}_i}\|_{\mathrm{op}}^2\, \big\|u - \bar{u}_{\omega_i}\big\|_{s(\omega_i)}^2 \leqslant \mathcal{L}_1^{(i)}\, C_P\ \|u\|_{V(\omega_i)}^2 .$$

Substituting $\Lambda^{(i)} = \Lambda_0^{(i)}/H^2$ and $C_P = C_{P,0} H^2$, the mesh factors cancel:

$$\begin{aligned}\|v_i\|_V^2 &\leqslant 2 \cdot \big(\Lambda_0^{(i)} + C_\chi^2\big) H^{-2} \cdot C_{P,0} H^2\, \mathcal{L}_1^{(i)}\, \|u\|_{V(\omega_i)}^2 = 2 \cdot C_{P,0}\big(\Lambda_0^{(i)} + C_\chi^2\big)\mathcal{L}_1^{(i)}\, \|u\|_{V(\omega_i)}^2 \\ &\leqslant C^2 \mathcal{L}_1^{(i)}\, \|u\|_{V(\omega_i)}^2 .\end{aligned}$$

□

Proposition E.9 (control of the prediction error)**.** *With C as above, $\Big\|I_{\mathrm{ms}} u - \widetilde{I}_{\mathrm{ms}} u\Big\|_V \leqslant \kappa_{\mathrm{ov}}\, C\, \sqrt{\mathcal{L}_{\max}}\ \|u\|_V$.*

Proof. Write $I_{\mathrm{ms}} u - \widetilde{I}_{\mathrm{ms}} u = \sum_i v_i$ with $v_i = \chi_i (P_{W_i} - P_{\widetilde{W}_i}) u|_{\omega_i}$ and $\mathrm{supp}(v_i) \subset \overline{\omega_i}$. By Lemma E.6, Proposition E.8, and $\mathcal{L}_1^{(i)} \leqslant \mathcal{L}_{\max}$,

$$\Big\|I_{\mathrm{ms}} u - \widetilde{I}_{\mathrm{ms}} u\Big\|_V^2 \leqslant \kappa_{\mathrm{ov}} \sum_i \|v_i\|_V^2 \leqslant \kappa_{\mathrm{ov}} C^2\, \mathcal{L}_{\max} \sum_i \|u\|_{V(\omega_i)}^2 .$$

Finite overlap of the ω_i gives $\sum_i \|u\|_{V(\omega_i)}^2 \leqslant \kappa_{\mathrm{ov}}\, \|u\|_V^2$, so $\Big\|I_{\mathrm{ms}} u - \widetilde{I}_{\mathrm{ms}} u\Big\|_V^2 \leqslant \kappa_{\mathrm{ov}}^2 C^2\, \mathcal{L}_{\max}\ \|u\|_V^2$. Taking square roots finishes the proof. □

Remark E.10 (local weighting). The second inequality in the proof above can be kept in the sharper form

$$\left\| I_{\mathrm{ms}} u - \widetilde{I}_{\mathrm{ms}} u \right\|_V^2 \leqslant \kappa_{\mathrm{ov}}\, C^2 \sum_i \mathcal{L}_1^{(i)}\ \|u\|_{V(\omega_i)}^2 ,$$

which weights the local prediction errors by the local solution energy and is better aligned with the averaged training objective.

Theorem E.11 (uniform error control)**.** *Let $u \in V$ be the exact solution and $\widetilde{u}_{\mathrm{ms}}$ the Transolver-GMsFEM solution computed with the eigenvectors predicted by $\mathcal{G}_1$ and the exact partition of unity $\{\chi_i\}_{i=1}^{N_c}$. Under Assumptions E.1–E.3, with C_0 from* (16) *and $C^2 = \max_i 2 \cdot C_{P,0}\big(\Lambda_0^{(i)} + C_\chi^2\big)$, both independent of H and N_c,*

$$\boxed{\|u - \widetilde{u}_{\mathrm{ms}}\|_V \ \leqslant\ C_0\, \frac{H}{\sqrt{\Lambda_0^*}}\, \|f\|_{L^2(\mathcal{D})} \ +\ \kappa_{\mathrm{ov}}\, C\, \sqrt{\mathcal{L}_{\max}}\ \|u\|_V}\ .$$

Proof. Since $\widetilde{u}_{\mathrm{ms}}$ is the Galerkin projection onto $\widetilde{V}_{\mathrm{off}}$, Céa's lemma and the triangle inequality give

$$\|u - \widetilde{u}_{\mathrm{ms}}\|_V = \min_{v \in \widetilde{V}_{\mathrm{off}}} \|u - v\|_V \leqslant \left\| u - \widetilde{I}_{\mathrm{ms}} u \right\|_V \leqslant \|u - I_{\mathrm{ms}} u\|_V + \left\| I_{\mathrm{ms}} u - \widetilde{I}_{\mathrm{ms}} u \right\|_V .$$

Bounding the first term by (16) and the second by Proposition E.9 yields the result. □

Remark E.12 (scope and constants). (i) The constant $C^2 = \max_i 2\, C_{P,0}\big(\Lambda_0^{(i)} + C_\chi^2\big)$ inherits the contrast dependence of $C_{P,0} = \max_i (H^2 \lambda_2^{\omega_i})^{-1}$ (Assumption E.3); the bound is uniform in H and N_c, but not in the contrast. (ii) $\mathcal{L}_{\max}$ is an *a posteriori* quantity evaluated over all local domains of a given test sample, whereas $\mathcal{G}_1$ is trained on the average of $\mathcal{L}_1$ over the selected local domains. The selective sampling strategy (Appendix F), which emphasises extreme-value domains, is designed precisely to reduce the worst-case local loss that enters the bound. (iii) The analysis covers the offline stage for linear problems only; the errors of $\mathcal{G}_2$ and $\mathcal{G}_3$, as well as nonlinear and time-dependent problems, are not included.

F Selective Sampling Strategy

Using all $\{\omega_i\}_{i=1}^{N_c}$ local domains per sample for training would be computationally redundant and inefficient. Instead, we employ a selective sampling strategy that ensures diverse and informative training data.

For a coarse 2D grid with N_v cells per dimension, the total number of local domains is $N_c = (N_v + 1)^2$, distributed among the **three geometric types**: $(N_v - 1)^2$ **interior** domains, $4 \cdot (N_v - 1)$ **border** domains, and 4 **corner** domains (see more information about domain types in Section A). For example, with $N_v = 10$, the total number of domains is 121, consisting of 81 interior domains, 36 border domains, and 4 corner domains.

Our selection procedure identifies coarse neighborhoods ω_i containing extreme values of the coefficient field (within a tolerance δ of the global maximum or minimum) and those containing only intermediate values. For each geometric type, we select a fixed number of coarse neighborhoods: half are drawn randomly from the extreme-value coarse neighborhoods (supplemented by intermediate neighborhoods if necessary), and the remaining half are drawn from intermediate coarse neighborhoods. This balanced selection ensures that the training data captures both challenging high-contrast regions and typical coefficient variations.

Figure 10 illustrates the domain selection procedure for samples No. 250 and No. 801. For visualization purposes, 8 interior and 16 border domains are shown. In contrast, the model training utilized a larger subset consisting of 12 interior and 28 border domains to ensure comprehensive spatial coverage and exposure to high-contrast features.

For a coarse 3D grid with N_v cells per dimension, the total number of local domains is $N_c = (N_v + 1)^3$, distributed among the **four geometric types**: $(N_v - 1)^3$ **interior** domains, $6 \cdot (N_v - 1)^2$ **face** domains, $12 \cdot (N_v - 1)$ **edge** domains, and 8 **corner** domains (see more information about domain types in Section A). For example, with $N_v = 5$, the total number of domains is 216, consisting of 64 interior domains, 96 face domains, 48 edge domains, and 8 corner domains. We employ a selective sampling strategy to construct the training set, choosing 12 interior, 60 face, and 24 edge domains, along with all 8 corner domains. This distribution is designed to provide the neural operator with sufficient exposure to the varying levels of geometric constraints imposed by the boundary $\partial\mathcal{D}$, which dictate the local behavior of the solutions to (7)

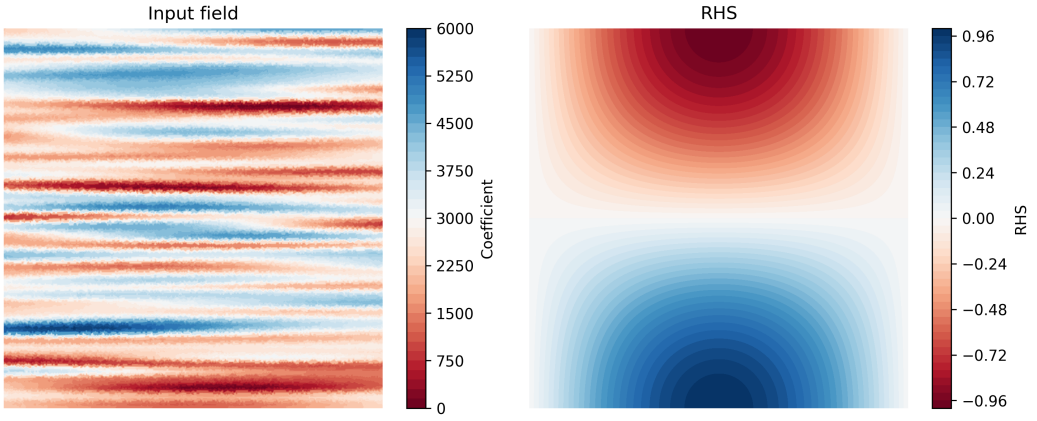
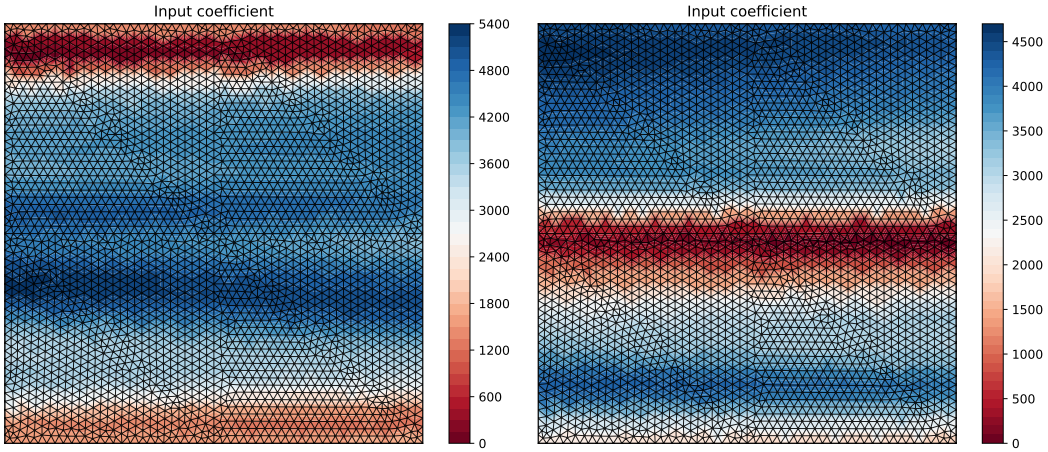


Figure 11: Example of input coefficient $\kappa(x)$ and right-hand side $f(x)$ defined by (20).



(a) Input coefficient, $\kappa(x)$ for No. 250 sample

(b) Input coefficient, $\kappa(x)$ for No. 801 sample

Figure 12: Examples of input coefficients $\kappa(x)$ for two samples: No. 250 and No. 801, on one local domain ω_{94}.

G Input Data

G.1 Input coefficients

We use the **Karhunen-Loève expansion** (KLE) [Wong, 1971, Ganis et al., 2008] to generate stochastic permeability fields. This method decomposes a random field into deterministic spatial functions and random coefficients.

1. Covariance Function. We assume the covariance function has an exponential form:

$$R(x, y) = \sigma_R^2 \exp(-\sqrt{\Delta}),$$

where

$$\Delta = \frac{|x_1 - x_2|^2}{l_x^2} + \frac{|y_1 - y_2|^2}{l_y^2}. \tag{17}$$

For 3D, the covariance function has an exponential form:

$$R(x, y, z) = \sigma_R^2 \exp(-\sqrt{\Delta}),$$

where

$$\Delta = \frac{|x_1 - x_2|^2}{l_x^2} + \frac{|y_1 - y_2|^2}{l_y^2} + \frac{|z_1 - z_2|^2}{l_z^2}. \tag{18}$$

2. Eigenvalue Problem. The eigenfunctions ϕ_k and eigenvalues λ_k are obtained by solving the homogeneous Fredholm integral equation:

$$\int_{\mathcal{D}} R(x,y)\phi_k(y)\,dy = \lambda_k \phi_k(x), \quad k = 1, 2, \ldots$$

3. Random Field Construction. The random field is represented as:

$$Y_L(x,\omega) = \sum_{k=1}^{L} \sqrt{\lambda_k}\,\theta_k(\omega)\,\phi_k(x),$$

where $\theta_k(\omega)$ are independent scalar random variables, and L is chosen to capture most of the field's energy by retaining the largest eigenvalues.

4. Permeability Field Generation. Each stochastic permeability field is defined as:

$$\kappa(x,\omega) = \exp\big(a_k \cdot \phi(x,\omega)\big),$$

where $\phi(x,\omega)$ represents the heterogeneous porosity field derived from $Y_L(x,\omega)$, and $a_k > 0$ is a scaling parameter that controls the contrast.

This KLE framework provides a systematic approach for generating realistic permeability fields with prescribed spatial correlation structures.

For the 2D case, we generate stochastic permeability-field datasets for training ($N_{\text{train}} = 700$ samples), testing ($N_{\text{test}} = 200$ samples), and validation ($N_{\text{val}} = 100$ samples), using correlation lengths $l_x = 0.02$, $l_y = 0.6$, and variance $\sigma_R^2 = 1.0$ (see (17)). Examples of the generated training (**Train Coefficient**) and test (**ID Test Coefficient**) datasets are shown in Figure 5.

For the 3D case, we generate stochastic permeability-field datasets for training ($N_{\text{train}} = 150$ samples), testing ($N_{\text{test}} = 50$ samples), and validation ($N_{\text{val}} = 50$ samples), using correlation lengths $l_x = 0.02$, $l_y = 0.6$, $l_z = 0.2$ and variance $\sigma_R^2 = 4.0$ (see (18)).

In this work, we use $\kappa(x)$ with values in the range $[1, 9600]$.

G.2 OOD input coefficients

Same-family OOD (Shifted covariance parameters) For the 2D case, we generate an out-of-distribution stochastic permeability-field dataset ($N_{\text{test}} = 200$ samples) using correlation lengths $\ell_x = 0.015$, $\ell_y = 0.5$, and $\sigma_R^2 = 2.0$ (see (17)). An example of the generated out-of-distribution test sample (**OOD Test Coefficient**) is shown in Figure 5.

For the 3D case, we generate an out-of-distribution stochastic permeability-field dataset ($N_{\text{test}} = 50$ samples) using correlation lengths $\ell_x = 0.015$, $\ell_y = 0.4$, $\ell_z = 0.1$, and $\sigma_R^2 = 3.0$ (see (18)). Although these settings shift the correlation lengths and the variance of the covariance function, the resulting fields are produced by the same generative procedure as the training data, i.e., they belong to the same family of exponential-covariance stochastic fields.

Different-family OOD (Unseen generative frameworks) To test robustness against genuinely unseen data, we generate three additional OOD coefficient families using generative processes that differ structurally from the one used for the training fields:

1. **Type 1: Spectral coefficient.** The field is synthesized directly in Fourier space: white noise is filtered by an anisotropic Matérn-type power spectrum $S(k) = [1 + (2\pi q)^2]^{-\beta/2}$, $\beta = 3$, with $q = \sqrt{(\ell_x k_x)^2 + (\ell_y k_y)^2}$, and then inverse-transformed. Unlike the training fields, it is not constructed from a truncated eigendecomposition of the covariance operator, i.e., it involves neither a covariance eigenbasis nor a truncation of the expansion.
2. **Type 2: Cosines coefficient.** The field is a superposition of $M = 300$ random plane-wave cosines with uniformly distributed random phases and wavevectors drawn from Gaussian distributions $k_x \sim \mathcal{N}(0, s_x^2)$, $k_y \sim \mathcal{N}(0, s_y^2)$, whose standard deviations s_x, s_y are set by the correlation lengths. It uses neither a covariance operator nor its eigenfunctions – only random directions, frequencies, and phases.
3. **Type 3: Waved OOD coefficient.** The same-family OOD coefficients are deformed by a vertical coordinate warp $\hat{y} = y + A\sin(2\pi x)$, $A = 0.03$, which produces wavy (non-horizontal) channels (see Figure 5). This family is closer to the training distribution than Types 1 and 2, but it breaks the horizontal-layering geometry of the training fields.

G.3 Spatially Variable Source Term.

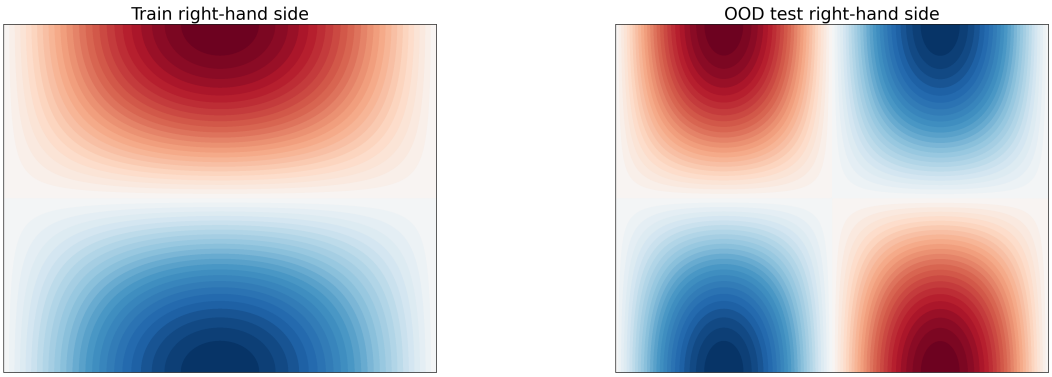
For the 2D case, we consider three right-hand side configurations:

- Uniform unit forcing term
$$f(x) = 1; \tag{19}$$
- Spatially variable forcing term
$$f(x) = \sin(\pi x)\cos(\pi y); \tag{20}$$
- Out-of-distribution spatially variable forcing term
$$f(x) = \sin(2 \cdot \pi x)\cos(\pi y). \tag{21}$$

An example of an input coefficient field $\kappa(x)$ and spatially variable forcing term (20) are shown in Figure 11. ID and OOD right-hand side are shown in Figure 13

For the 3D case, we consider two right-hand side configurations:

- Uniform unit forcing term
$$f(x, y, z) = 1; \tag{22}$$
- Spatially variable forcing term
$$f(x, y, z) = \sin(\pi x)\cos(\pi y)\sin(\pi z). \tag{23}$$



(a) Train right-hand side, $f(x) = \sin(\pi x)\cos(\pi y)$

(b) OOD test right-hand side, $f(x) = \sin(2\pi x)\cos(\pi y)$

Figure 13: Train and OOD test right-hand sides.

H Datasets

H.1 Diffusion equation

We consider the diffusion equation with a heterogeneous coefficient $\kappa(x)$:

$$\begin{aligned} -\nabla \cdot \big(\kappa(x)\nabla u(x)\big) &= f(x), \quad x \in \mathcal{D}, \\ u(x)\big|_{x\in\partial\mathcal{D}} &= 0, \end{aligned} \tag{24}$$

where $\mathcal{D}$ is the unit hypercube, $\partial\mathcal{D}$ denotes its boundary, $f(x) \in L^2(\mathcal{D})$ is a given source term, and $\kappa(x)$ is a heterogeneous diffusion coefficient with high contrast. We assume that $\kappa(x) \geqslant \varepsilon > 0$ is bounded below, while $\kappa(x)$ can exhibit very large variations.

H.2 Diffusion equation with Mixed Boundary Conditions

We consider the diffusion equation with heterogeneous coefficient

$$\begin{aligned} -\nabla \cdot \big(\kappa(x)\nabla u(x)\big) &= f(x), \quad x \in \mathcal{D}, \\ u &= 0, \quad x \in \Gamma_D, \\ -\kappa(x)\frac{\partial u}{\partial n} &= \alpha \cdot \big(u - u_{\text{const}}\big), \quad x \in \Gamma_R, \end{aligned} \tag{25}$$

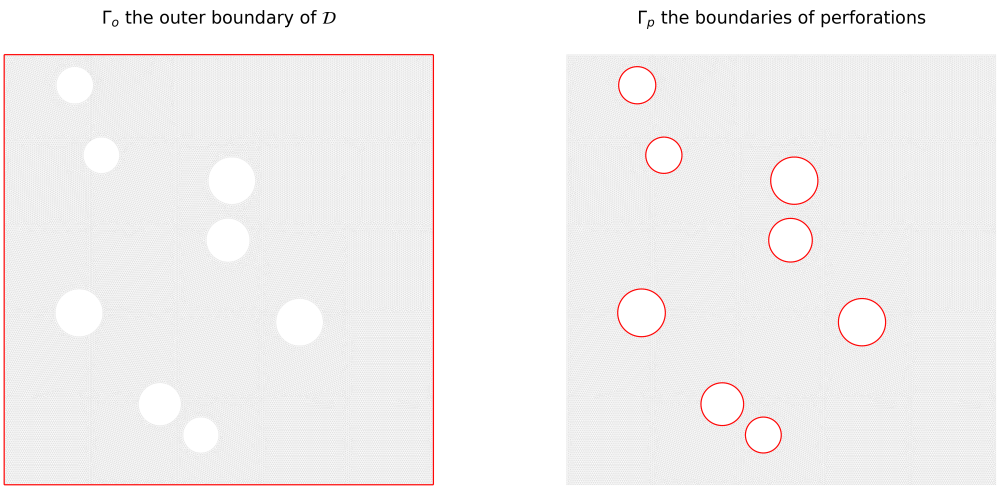


Figure 14: Boundary decomposition of the perforated domain $\mathcal{D}$. Left: the outer boundary Γ_o (red) on which the homogeneous Dirichlet condition $u = 0$ is imposed. Right: the perforation boundaries Γ_p (red) on which the Robin condition $-\kappa\, \partial u/\partial n = \alpha(u - a)$ is imposed.

where the boundary $\partial\mathcal{D}$ is partitioned into a Dirichlet part and a Robin part. The Dirichlet boundary consists of the bottom, left, and right edges:

$$\Gamma_D = \big\{x \in \partial\mathcal{D} \;\big|\; y = 0\big\} \cup \big\{x \in \partial\mathcal{D} \;\big|\; x = 0\big\} \cup \big\{x \in \partial\mathcal{D} \;\big|\; x = 1\big\},$$

while the Robin boundary is the top edge:

$$\Gamma_R = \big\{x \in \partial\mathcal{D} \;\big|\; y = 1\big\},$$

with Robin coefficient $\alpha = 1$, $u_{\text{const}} = 1$.

H.3 Diffusion equation in Perforated Media

We consider the diffusion equation in a perforated domain $\mathcal{D}$ with non-homogeneous Robin boundary conditions on the perforations:

$$\begin{aligned} -\nabla \cdot \big(\kappa(x)\nabla u(x)\big) &= f(x), \quad x \in \mathcal{D}, \\ u &= 0, \quad x \in \Gamma_o, \\ -\kappa(x)\,\frac{\partial u}{\partial n} &= \alpha \cdot \big(u - a\big), \quad x \in \Gamma_p, \end{aligned} \tag{26}$$

where Γ_p denotes the boundaries of the perforations, $\Gamma_o = \partial\mathcal{D} \setminus \Gamma_p$ is the outer boundary of $\mathcal{D}$, n is the outward unit normal, and we set $\alpha = 1.0$ and $a = 0.01$. Figure 14 illustrates the boundary decomposition of the domain: the outer boundary Γ_o on which the Dirichlet condition is enforced, and the perforation boundaries Γ_p on which the Robin condition is enforced.

This formulation follows the standard setup for diffusion in perforated media, with one key difference: the **number** and **spatial arrangement** of the perforations vary across samples, rather than being fixed. As a consequence, the geometry $\mathcal{D}$ (through Γ_p) changes from one sample to another, as illustrated for the 2D case in Figures 15 and 16, which show two samples that differ in both the realization of $\kappa(x)$ and the number and positions of the perforations. Figures 17 and 18 show two examples of the resulting 3D meshes (samples No. 77 and No. 196), together with the projections of the perforations onto the xy-, xz-, and yz-planes. As in the 2D case, the number and positions of the perforations differ between the samples.

H.4 Steady-State Richards Equation

The steady-state Richards equation describes water movement in unsaturated porous media and takes the following form:

$$\begin{aligned} -\nabla \cdot \big(\kappa\big(x, u(x)\big)\nabla u(x)\big) &= f(x), \quad x \in \mathcal{D}, \\ u(x)\big|_{x\in\partial\mathcal{D}} &= 0, \end{aligned} \tag{27}$$

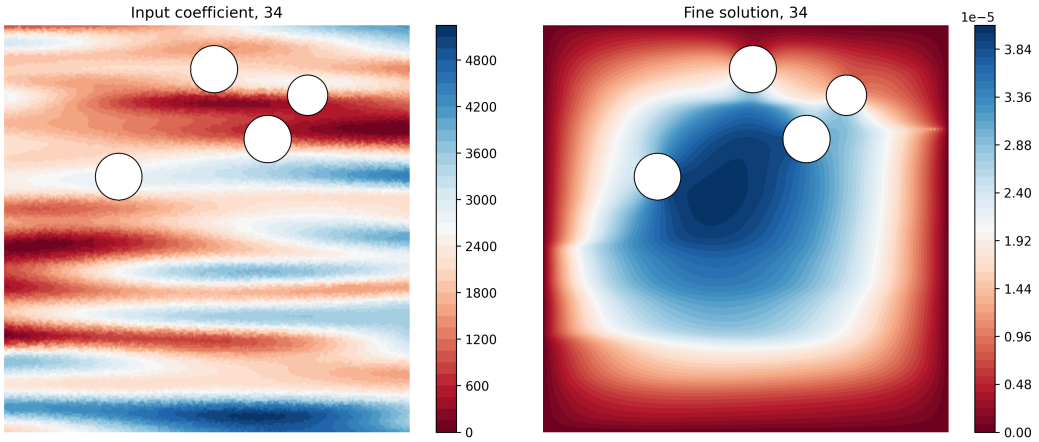


Figure 15: Example (No. 34) of input coefficient $\kappa(x)$ and fine solution $u(x)$.

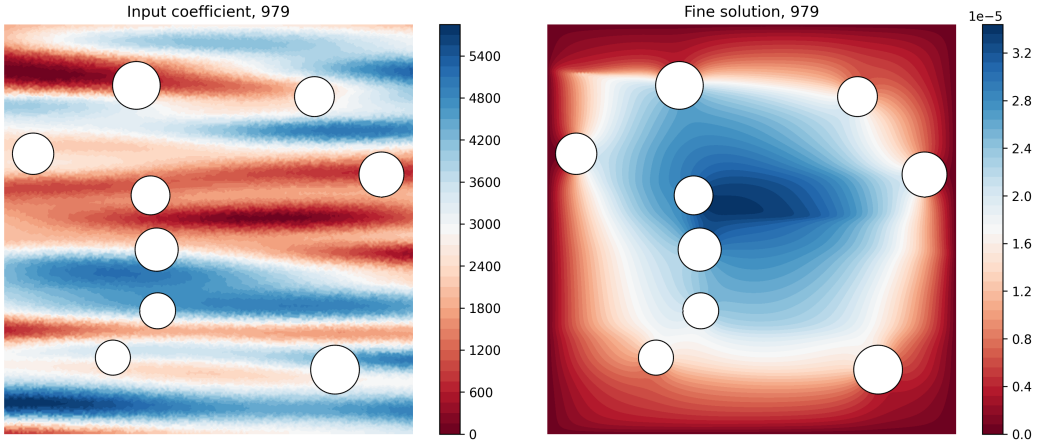


Figure 16: Example (No. 979) of input coefficient $\kappa(x)$ and fine solution $u(x)$.

where $u(x)$ is the pressure head, $\kappa\big(x, u(x)\big)$ is the unsaturated hydraulic conductivity, and $f(x)$ is a source or sink term.

We adopt the Haverkamp model [Haverkamp et al., 1977] to define the conductivity $\kappa\big(x, u(x)\big)$:

$$\kappa\big(x, u(x)\big) = K_s(x)\, K_r(u(x)) = \kappa(x)\, \frac{1}{1+|u|},$$

where $K_s(x) = \kappa(x)$ is the saturated hydraulic conductivity, represented by a heterogeneous high-contrast field that characterizes the permeability of soils, and $K_r(u) = \dfrac{1}{1+|u|}$ is the relative hydraulic conductivity. This formulation decouples the multiscale spatial heterogeneity, encoded in $\kappa(x)$, from the solution-dependent nonlinearity $K_r(u)$, which simplifies the construction of multiscale basis functions.

H.5 Heat Equation

We consider the time-dependent heat equation with a heterogeneous coefficient:

$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla \cdot \big(\kappa(x) \nabla u\big) &= f(x), \quad x \in \mathcal{D},\, t \in (0, T_{\max}], \\ u &= 0, \quad x \in \partial\mathcal{D}, \\ u|_{t=0} &= 0, \quad x \in \mathcal{D}, \end{aligned} \tag{28}$$

where $u(x,t)$ is the temperature, $\kappa(x)$ is the high-contrast thermal conductivity, and $f(x)$ is a source term. We set the time step $\tau = 2.5 \times 10^{-5}$ and the final time $T_{\max} = 5 \times 10^{-4}$ for 2D and $T_{\max} = 2.5 \times 10^{-4}$ for 3D.

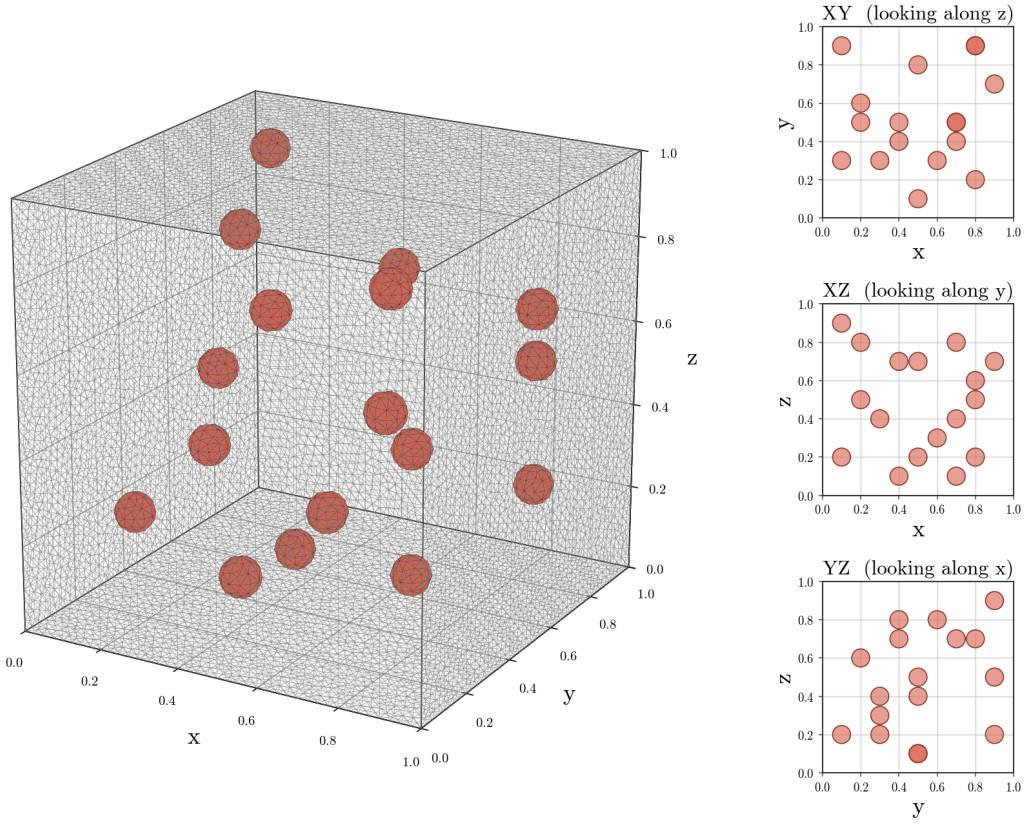


Figure 17: Example (No. 77) of a 3D mesh with spherical perforations and the projections of the perforations onto the coordinate planes.

H.6 Time-Dependent Richards Equation

The time-dependent Richards equation describes transient water movement in variably saturated porous media and takes the following form:

$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla \cdot \big(\kappa\big(x, u(x,t)\big)\nabla u\big) &= f(x), \quad x \in \mathcal{D},\, t \in (0, T_{\max}], \\ u &= 0, \quad x \in \partial\mathcal{D}, \\ u|_{t=0} &= 0, \quad x \in \mathcal{D}, \end{aligned} \tag{29}$$

where $u(x,t)$ denotes the pressure head, $\kappa\big(x, u(x,t)\big)$ is the unsaturated hydraulic conductivity, and $f(x)$ represents a source or sink term.

We adopt a modified Haverkamp model [Haverkamp et al., 1977] for the hydraulic conductivity:

$$\kappa\big(x, u(x,t)\big) = \kappa(x)\,\frac{1}{1 + |u| \cdot 10^5}, \tag{30}$$

where $\kappa(x)$ is the saturated hydraulic conductivity given by a heterogeneous high-contrast field. The relative conductivity $K_r(u) = (1 + |u| \cdot 10^5)^{-1}$ exhibits a sharp decay as the pressure head increases, capturing the rapid reduction in hydraulic conductivity under unsaturated conditions. The large coefficient 10^5 amplifies the nonlinear response, posing significant challenges for numerical solvers.

We set the time step $\tau = 2.5 \times 10^{-5}$ and the final time $T_{\max} = 5 \times 10^{-4}$ for 2D and $T_{\max} = 2.5 \times 10^{-4}$ for 3D.

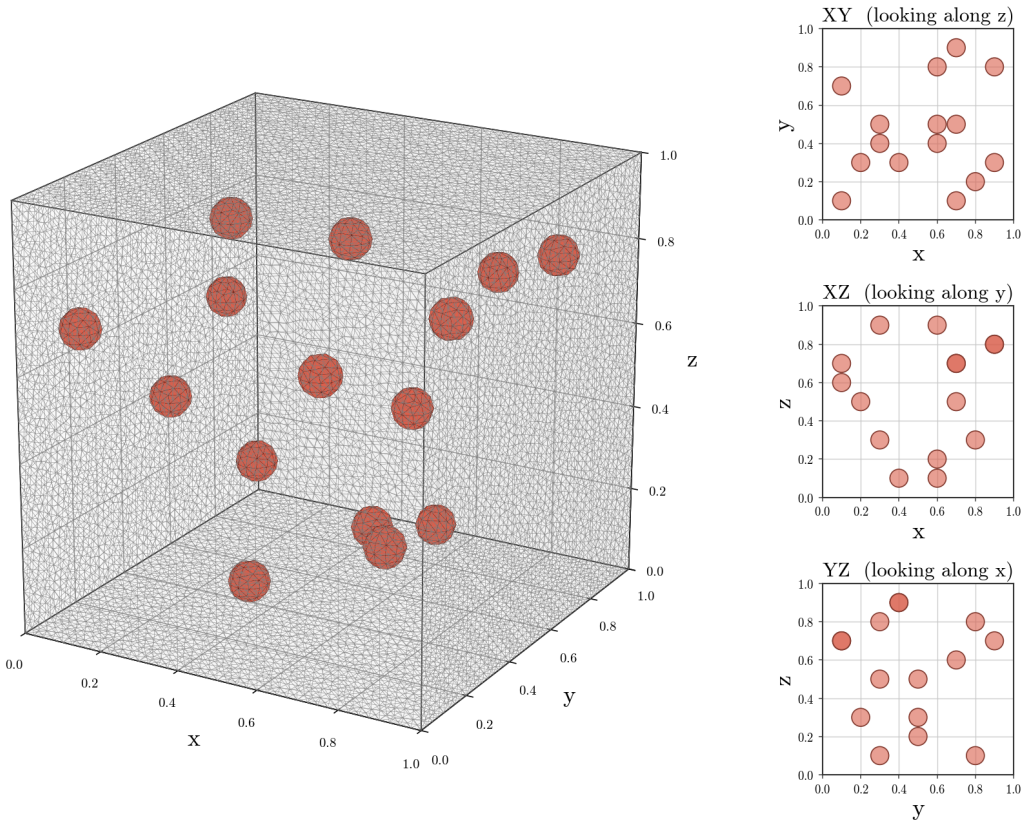


Figure 18: Example (No. 196) of a 3D mesh with spherical perforations and the projections of the perforations onto the coordinate planes.

I Details of the 3D experiments

We utilized a dataset consisting of 3D high-contrast coefficient fields (detailed in Appendix G), partitioned into $N_{\text{train}} = 150$ training, $N_{\text{val}} = 50$ validation, and $N_{\text{test}} = 50$ test samples. To evaluate the robustness of our approach, we also generated 50 out-of-distribution (OOD) coefficient fields, as detailed in Appendix G.2. The structured coarse mesh is $5 \times 5 \times 5$, resulting in $N_c = 216$ nodes, while the irregular fine grid consists of 1,382,363 cells and 238,152 vertices.

We extensively evaluate Transolver-GMsFEM on four benchmark problems: (i) diffusion with Dirichlet boundary conditions (see Appendix H.1), (ii) the steady-state Richards equation (see Appendix H.4), (iii) the heat equation (see Appendix H.5), and (iv) the time-dependent Richards equation (see Appendix H.6). For steady-state problems, we consider two distinct right-hand sides, defined in (22) and (23) (see Appendix G.3). Conversely, for time-dependent problems, we restrict our evaluation to (23).

We compare our results against a Transolver++ baseline, which processes the global coefficient field and right-hand side to directly produce the fine-grid solution. The baseline was optimized for all benchmarks and right-hand sides through a comprehensive grid search of architectural and training hyperparameters, as detailed in Appendix J.5.

J Training Details

J.1 Training Protocol for Transolver-based Models

We employed a consistent training protocol for both the Transolver architecture Wu et al. [2024] and the Transolver++ architecture Luo et al. [2025] adapted for various task settings. All models were trained for 600

epochs using the **AdamW** optimizer paired with a **OneCycle** learning rate scheduler, utilizing a base learning rate of 10^{-3}.

To identify the optimal configurations for 2D and 3D physical problems, we conducted a comprehensive hyperparameter grid search centered around the following values:

1. **Number of layers:** $\{4, 6, 8\}$
2. **Hidden size:** $\{128, 256\}$
3. **MLP layers:** $\{1, 2\}$
4. **Number of heads:** $\{4, 8\}$
5. **Number of slices (M):** $\{32, 64, 128\}$
6. **Gradient clipping:** $\{1, 10, 100\}$

For models trained on 2D local domains (where the number of nodes is less than $4,000$), we tested batch sizes in $\{8, 16, 32\}$. For models trained on the full computational domains ($76,471$ nodes for 2D and $238,152$ vertices for 3D), batch sizes were restricted to $\{2, 4, 8\}$ to accommodate GPU memory limits. The optimal configurations for specific models are detailed in the following subsections.

J.2 Model $\mathcal{G}_1$

Model $\mathcal{G}_1$ performs the mapping $\kappa|_{\omega_i} \mapsto \{\widetilde{\phi}_k^{\omega_i}\}_{k=1}^L$ for $L \in \{4, 8\}$ on local domains ω_i. Based on the grid search described in J.1, the optimal configuration for 2D was: batch size 16, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 128 slices, and a gradient clip of 100. For 3D fields, the optimal configuration was: batch size 4, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 128 slices, and a gradient clip of 100.

J.3 Model $\mathcal{G}_2$

Model $\mathcal{G}_2$ maps $(\kappa|_{\omega_i}, g_{\omega_i}) \mapsto \widetilde{\varepsilon}_i$ on local domains ω_i. Following the grid search in J.1, the optimal 2D configuration was: batch size 16, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 128 slices, and a gradient clip of 1. In 3D, due to the increased complexity of the local source term, the optimal configuration was: batch size 8, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 128 slices, and a gradient clip of 100.

J.4 Model $\mathcal{G}_3$

Model $\mathcal{G}_3$ performs the task $(u_{\text{ms}}^{m,n}, u_{\text{ms}}^{m+1,n-1}, f, \kappa)|_{\omega_i} \mapsto \widetilde{\phi}_n^{\text{on},\omega_i}$ on local domains ω_i. Utilizing the search space defined in J.1, the optimal configuration for both 2D and 3D was: batch size 16, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 64 slices, and a gradient clip of 100.

J.5 Transolver++

This model was evaluated on all benchmarks using the relative L^2 loss. Training was performed on the full high-resolution computational domains. Following the grid search protocol in J.1, the optimal hyperparameters were:

- **2D Steady-state:** Batch size 2, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 64 slices, grad clip 1.
- **2D Time-dependent:** Batch size 4, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 64 slices, grad clip 1.
- **3D (Steady and Time-dependent):** Batch size 4, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 64 slices, grad clip 1.

J.6 Transolver

For the 2D baseline, we utilized the Transolver architecture Wu et al. [2024] trained on the full domain. According to the grid search results in J.1, a hidden size of 256 and $M = 64$ slices provided the best balance between capturing complex physical states and computational efficiency. The optimal parameters for 2D steady-state problems were: batch size 8, 6 layers, hidden size 256, 2 MLP layers, 8 heads, 64 slices, and a gradient clip of 1.

J.7 GNOT

For the GNOT baseline Hao et al. [2023], we performed a hyperparameter grid search centered around default configurations for unstructured meshes:

- Layers: $\{3, 4\}$
- Hidden size (d_{model}): $\{128, 192, 256\}$
- Number of heads: $\{4, 8\}$
- Number of experts (N_{experts}): $\{2, 4, 8\}$
- MLP layers: $\{3, 4\}$
- Learning rate: $\{5 \times 10^{-4}, 1 \times 10^{-3}\}$
- Gradient clipping: $\{1.0, 10.0\}$

The optimal 2D parameters were: batch size 4, 4 layers, hidden size 128, 8 heads, 4 MLP layers, 4 experts, and a learning rate of 5×10^{-4}. Despite this search, the model encountered gradient explosions on the Diffusion dataset with mixed boundary conditions and the spatially variable source term (20), suggesting that the standard GNOT architecture may require additional stabilization for high-resolution meshes with high local variance. The corresponding entries are therefore omitted (marked with "—") in Tables 6, 7, 9, and 13.

J.8 GAOT

We conducted a targeted grid search for the GAOT model Wen et al. [2025] on the 2D domain. The search focused on parameters distinguishing high-resolution setups from default configurations:

- Latent token grid size: $\{128 \times 128, 192 \times 192\}$
- Aggregation radius (r): $\{0.033, 0.022\}$
- Batch size: $\{8, 16\}$
- Max Learning Rate: $\{5 \times 10^{-4}, 8 \times 10^{-4}\}$

The optimal 2D parameters were: batch size 16, 192×192 latent token grid size, aggregation radius 0.022, and a learning rate of 8×10^{-4}. Architectural parameters such as transformer hidden size (256), patch size (2), and MLP layers (3) were fixed to values optimal for high-contrast datasets.

K Loss ablations (absolute vs. relative)

We perform an ablation study comparing absolute and relative formulations of the training losses for both model $\mathcal{G}_2$ (loss $\mathcal{L}_2$) and model $\mathcal{G}_3$ (loss $\mathcal{L}_3$).

Model $\mathcal{G}_2$. The loss $\mathcal{L}_2$ used in our pipeline is absolute:

$$\mathcal{L}_2 = \mathbb{E}_i\Big[\big\|\varepsilon_i - \widetilde{\varepsilon}_i\big\|_2^2 + \big\|\sqrt{\kappa|_{\omega_i}}\,(\nabla\varepsilon_i - \nabla\widetilde{\varepsilon}_i)\big\|_2^2\Big].$$

As an alternative, we consider a relative variant in which each term is normalized:

$$\mathcal{L}_2^{\text{rel}} = \mathbb{E}_i\left[\frac{\big\|\varepsilon_i - \widetilde{\varepsilon}_i\big\|_2^2}{\big\|\varepsilon_i\big\|_2^2} + \frac{\big\|\sqrt{\kappa|_{\omega_i}}\,(\nabla\varepsilon_i - \nabla\widetilde{\varepsilon}_i)\big\|_2^2}{\big\|\sqrt{\kappa|_{\omega_i}}\,\nabla\varepsilon_i\big\|_2^2}\right].$$

As shown in Table 3, the absolute loss $\mathcal{L}_2$ yields lower L_2 and H^1 errors on all steady-state benchmarks. The gap is most pronounced for Diffusion with mixed BCs and the source term (20), where the relative L_2 error increases from 0.65% to 0.93% when $\mathcal{L}_2^{\text{rel}}$ is used. Since the corrections ε_i have comparable magnitudes across local domains, normalization offers no benefit and instead amplifies the contribution of domains with small $\|\varepsilon_i\|_2$.

Model $\mathcal{G}_3$. In contrast, the loss $\mathcal{L}_3$ used for $\mathcal{G}_3$ is relative:

$$\mathcal{L}_3 = \mathbb{E}_{i,n}\left[\frac{\big\|\phi_n^{\text{on},\omega_i} - \widetilde{\phi}_n^{\text{on},\omega_i}\big\|_2^2}{\big\|\phi_n^{\text{on},\omega_i}\big\|_2^2} + \frac{\big\|\sqrt{\kappa|_{\omega_i}}(\nabla\phi_n^{\text{on},\omega_i} - \nabla\widetilde{\phi}_n^{\text{on},\omega_i})\big\|_2^2}{\big\|\sqrt{\kappa|_{\omega_i}}\nabla\phi_n^{\text{on},\omega_i}\big\|_2^2}\right].$$

As an alternative, we consider its absolute counterpart:

$$\mathcal{L}_3^{\text{abs}} = \mathbb{E}_{i,n}\Big[\big\|\phi_n^{\text{on},\omega_i} - \widetilde{\phi}_n^{\text{on},\omega_i}\big\|_2^2 + \big\|\sqrt{\kappa|_{\omega_i}}\,(\nabla\phi_n^{\text{on},\omega_i} - \nabla\widetilde{\phi}_n^{\text{on},\omega_i})\big\|_2^2\Big].$$

Table 4 shows that replacing $\mathcal{L}_3$ with $\mathcal{L}_3^{\text{abs}}$ increases the errors of Transolver-GMsFEM (offline + online) for both time-dependent benchmarks: the L_2 error grows from 0.41% to 0.63% for the Heat equation and from 0.46% to 0.72% for the Richards equation, with a similar degradation in the H^1 norm. This confirms that the relative formulation is essential for $\mathcal{G}_3$, since the online basis functions $\phi_n^{\text{on},\omega_i}$ vary by orders of magnitude across local domains and time steps, and an absolute loss is dominated by the samples with the largest norms.

Table 3: Performance comparison of $\mathcal{L}_2$, and $\mathcal{L}_2^{\text{rel}}$ for 2D problems ($L = 8$, $N_c = 121$).

		$\mathcal{L}_2$		$\mathcal{L}_2^{\text{rel}}$	
L	Dataset	L_2	H^1	L_2	H^1
8	Diffusion, (19)	**0.82**	**17.33**	0.83	17.51
	Diffusion, (20)	**1.68**	**19.66**	1.74	20.38
	Richards, (19)	**1.81**	**18.99**	2.08	20.57
	Richards, (20)	**1.84**	**19.13**	2.00	20.61
	Diffusion + mBCs, (19)	**0.54**	**12.95**	0.63	14.46
	Diffusion + mBCs, (20)	**0.65**	**12.41**	0.93	14.09

Table 4: Performance comparison of $\mathcal{L}_3$, and $\mathcal{L}_3^{\text{abs}}$ for 2D time-dependent problems ($L = 4$).

	Heat, (20)		Richards, (20)	
Method	L_2	H^1	L_2	H^1
Transolver-GMsFEM (offline + online), $\mathcal{L}_3$	**0.34**	**6.70**	**0.34**	**6.35**
Transolver-GMsFEM (offline + online), $\mathcal{L}_3^{\text{abs}}$	0.63	12.25	0.72	13.83

L Predictor ablations (Transolver++ vs. GAOT)

To assess how strongly the proposed framework depends on the choice of the neural predictor, we replace Transolver++ inside the pipeline with GAOT Wen et al. [2025] at two model sizes: a small model with $\sim$1.47M parameters (GAOT-GMsFEM (S)) and a large model with $\sim$2.53M parameters (GAOT-GMsFEM (L)). We compare both variants against Transolver-GMsFEM ($\sim$2.58M parameters) and classical GMsFEM in the 2D setting with $L = 8$ and $N_c = 121$. The relative L_2 and H^1 errors are reported in Table 5. The results shows that GAOT serves as a drop-in replacement for the internal predictor and produces valid multiscale basis functions. Hence, the accuracy of the pipeline does not rely on Transolver++ specifically.

The small GAOT predictor, with about 57% of the parameters of Transolver-GMsFEM, still achieves L_2 errors between 1.05% and 2.34% and outperforms all end-to-end neural baselines on every steady-state benchmark. Increasing the predictor capacity improves the accuracy, so the predictor size can be chosen according to the available computational budget.

Transolver++ is the best choice at a comparable parameter count, but not a requirement. At $\sim$2.5M parameters, Transolver-GMsFEM attains the lowest L_2 error on all benchmarks and the lowest H^1 error on all but one, which empirically justifies our choice of Transolver++. At the same time, GAOT-GMsFEM (L) is close behind, which confirms that the framework tolerates the substitution of the predictor.

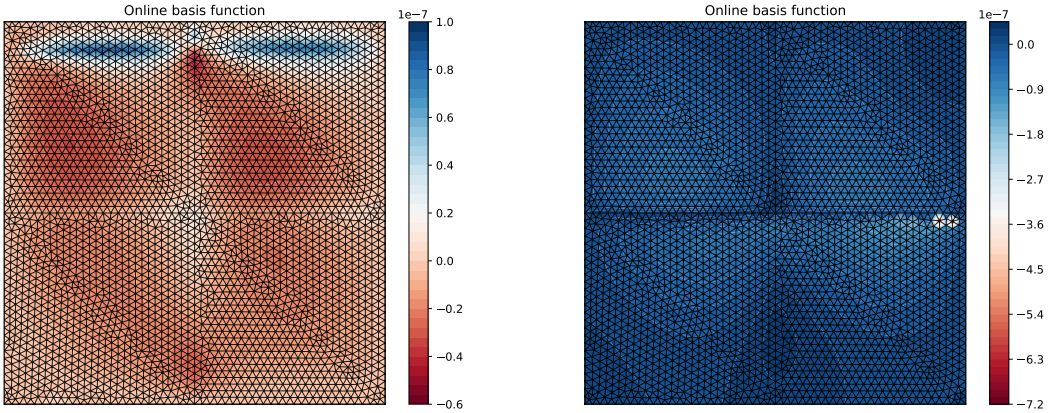
M Offline and Online Basis Functions

In this section, we provide a visual comparison of the high-contrast input coefficients and their corresponding multiscale basis functions on local domains. Figure 12 illustrates the local coefficient fields $\kappa(x)$ for two representative samples at domain ω_{94}, highlighting the complex heterogeneities the model must resolve.

The resulting multiscale basis functions are depicted in the subsequent figures. Figures 20 and 21 display the first $L = 8$ offline eigenvectors. Figure 19 shows the online $\phi_n^{\text{on},\omega_i}$.

Table 5: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline, ~ 2.58M), and GAOT-GMsFEM (~ 1.47M vs. ~ 2.53M) for 2D problems ($L = 8$, $N_c = 121$). Method abbreviations: GM = GMsFEM, Tr-GM = Transolver-GMsFEM, GAOT-GM = GAOT-GMsFEM. Bold and underlined values denote the best and second-best results among the learning-based methods; GMsFEM is shown as a reference.

		GM		Tr-GM (~ 2.58M)		GAOT-GM (~ 1.47M)		GAOT-GM (~ 2.53M)	
L	Dataset	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1
8	Diffusion, (19)	0.79	13.52	**0.82**	**17.33**	1.05	19.66	<u>0.83</u>	<u>17.48</u>
	Diffusion, (20)	1.64	15.02	**1.68**	<u>19.66</u>	1.93	21.95	<u>1.70</u>	**19.22**
	Richards, (19)	1.62	15.14	**1.81**	**18.99**	2.28	22.40	<u>1.92</u>	<u>19.62</u>
	Richards, (20)	1.75	15.42	**1.84**	**19.13**	2.34	22.70	<u>2.11</u>	<u>20.49</u>
	Diffusion + mBCs, (19)	0.44	8.56	**0.54**	**12.95**	1.08	16.97	<u>0.60</u>	<u>13.66</u>
	Diffusion + mBCs, (20)	0.59	10.07	**0.65**	**12.41**	1.40	16.62	<u>0.74</u>	<u>13.26</u>



(a) Online $\phi_n^{\text{on},\omega_i}$ for No. 250 sample

(b) Online $\phi_n^{\text{on},\omega_i}$ for No. 801 sample

Figure 19: Examples of online $\phi_n^{\text{on},\omega_i}$ for two samples: No. 250 and No. 801, on one local domain ω_{94}.

N Results for 2D

N.1 Table results for 2D

Table 6: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver++, GNOT, GAOT, and Transolver for 2D problems ($L = 4$, $N_c = 121$). Diffusion with mixed BCs is abbreviated as Diffusion + mBCs. Method abbreviations: Tr-GMsFEM = Transolver-GMsFEM, Tr++ = Transolver++, Tr = Transolver.

		GMsFEM		Tr-GMsFEM		Tr++		GNOT		GAOT		Tr	
L	Dataset	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1
4	Diffusion, (19)	1.03	14.18	<u>1.43</u>	**21.09**	1.51	27.42	1.56	22.39	1.50	23.10	**1.36**	<u>21.26</u>
	Diffusion, (20)	1.90	15.24	**2.50**	**23.70**	5.15	44.93	2.91	27.77	2.90	27.63	3.96	33.56
	Richards, (19)	2.19	15.81	3.29	**24.04**	3.58	48.53	<u>3.19</u>	32.99	5.11	38.21	**2.88**	<u>32.73</u>
	Richards, (20)	2.13	15.66	**2.93**	**24.38**	4.84	43.84	<u>3.37</u>	<u>27.65</u>	4.12	30.56	4.30	34.12
	Diffusion + mBCs, (19)	0.71	10.40	**1.09**	**18.48**	3.13	53.79	<u>1.52</u>	<u>24.77</u>	3.17	34.39	2.34	36.67
	Diffusion + mBCs, (20)	0.88	11.91	**1.28**	**18.04**	4.84	49.44	—	—	<u>3.94</u>	<u>36.53</u>	5.94	63.07

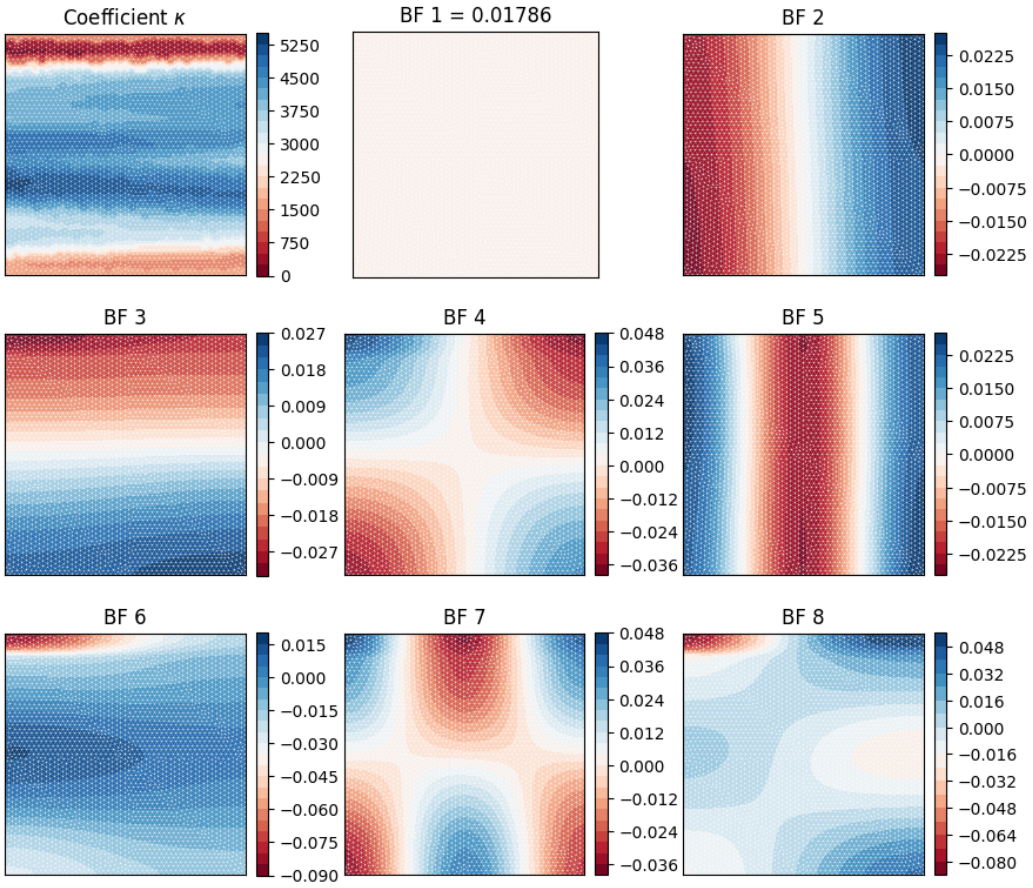


Figure 20: Offline $\{\phi_k^{\omega_i}\}_{k=1}^{L}$ for $\kappa(x)$ for No. 250 sample, local domain ω_{94}, $L = 8$.

Table 7: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver++, GNOT, GAOT, and Transolver for 2D problems ($L = 8$, $N_c = 121$). Diffusion with mixed BCs is abbreviated as Diffusion + mBCs. Method abbreviations: Tr-GMsFEM = Transolver-GMsFEM, Tr++ = Transolver++, Tr = Transolver.

		GMsFEM		Tr-GMsFEM		Tr++		GNOT		GAOT		Tr	
L	Dataset	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1
8	Diffusion, (19)	0.79	13.52	**0.82**	**17.33**	1.51	27.42	1.56	22.39	1.50	23.10	1.36	21.26
	Diffusion, (20)	1.64	15.02	**1.68**	**19.66**	5.15	44.93	2.91	27.77	2.90	27.63	3.96	33.56
	Richards, (19)	1.62	15.14	**1.81**	**18.99**	3.58	48.53	3.19	32.99	5.11	38.21	2.88	32.73
	Richards, (20)	1.75	15.42	**1.84**	**19.13**	4.84	43.84	3.37	27.65	4.12	30.56	4.30	34.12
	Diffusion + mBCs, (19)	0.44	8.56	**0.54**	**12.95**	3.13	53.79	1.52	24.77	3.17	34.39	2.34	36.67
	Diffusion + mBCs, (20)	0.59	10.07	**0.65**	**12.41**	4.84	49.44	—	—	3.94	36.53	5.94	63.07

This section provides a comprehensive numerical results for all 2D benchmarks. We evaluate the performance of Transolver-GMsFEM against the classical GMsFEM and several state-of-the-art neural PDEs solvers, including GNOT, GAOT, Transolver++, and the standard Transolver. Table 6 and Table 7 present the L_2 and H^1 relative errors for 2D steady-state problems using $L = 4$ and $L = 8$ offline basis functions, respectively. Across various datasets, Transolver-GMsFEM consistently achieves errors comparable to the classical GMsFEM. While neural baselines show competitive results in simpler diffusion cases, their performance often degrades in more complex scenarios. Furthermore, as the number of basis functions increases from $L = 4$ to $L = 8$, the performance gap between our method and GMsFEM narrows significantly.

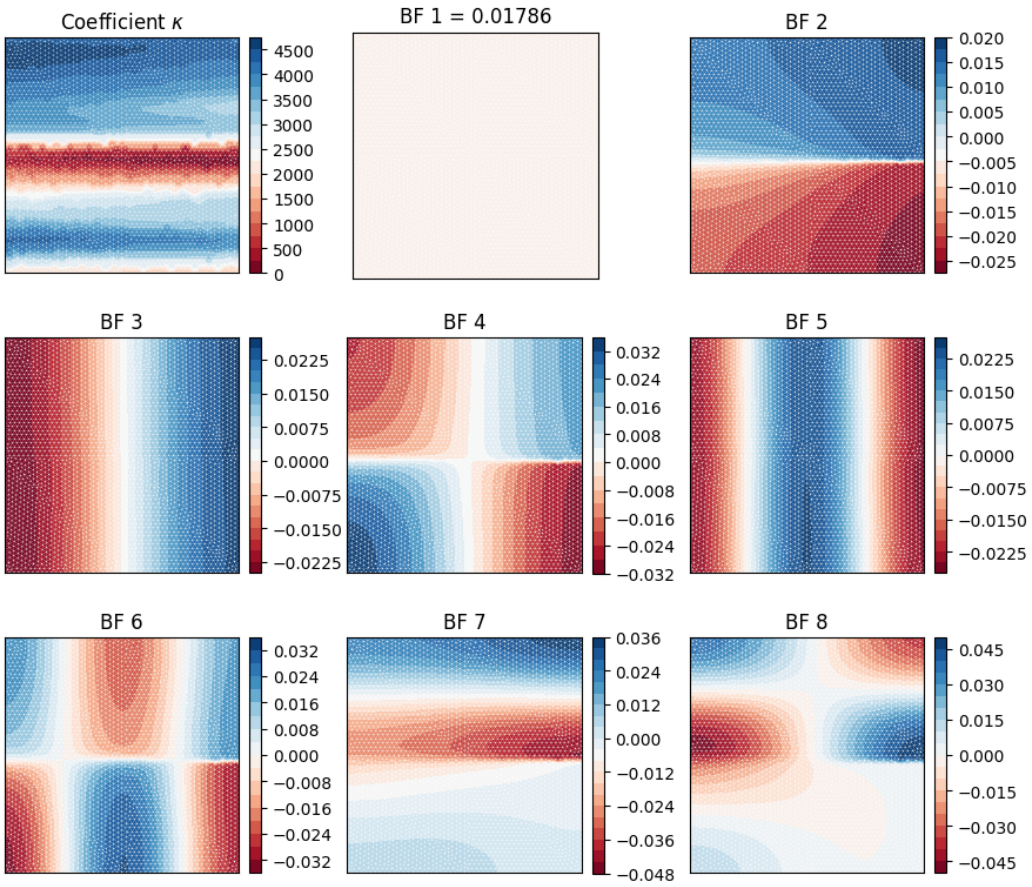


Figure 21: Offline $\{\phi_k^{\omega_i}\}_{k=1}^{L}$ for $\kappa(x)$ for No. 801 sample, local domain ω_{94}, $L = 8$.

Table 8: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline + online), Transolver-GMsFEM (offline + online), Transolver++, GNOT, and GAOT for 2D time-dependent problems ($L = 4$).

	Heat, (20)		Richards, (20)	
Method	L_2	H^1	L_2	H^1
GMsFEM (offline)	1.91	16.68	1.93	16.91
Transolver-GMsFEM (offline)	2.19	23.40	2.44	24.40
Transolver-GMsFEM (offline) + GMsFEM (online)	**0.20**	**3.52**	**0.23**	**3.86**
GMsFEM (offline + online)	0.16	2.41	0.18	2.58
Transolver-GMsFEM (offline + online)	<u>0.34</u>	<u>6.70</u>	<u>0.34</u>	<u>6.35</u>
Transolver++	5.94	42.34	4.76	42.68
GNOT	6.92	43.92	6.76	41.27
GAOT	4.10	26.59	3.16	22.12

The performance for 2D time-dependent Heat and Richards equations is summarized in Table 8. Here, we compare purely offline predictions against configurations that incorporate online GMsFEM components. The results indicate that the combination of Transolver-GMsFEM (offline) + GMsFEM (online) yields the best overall performance among the learning-based methods. This hybrid approach significantly outperforms pure neural baselines such as Transolver++, GNOT, and GAOT, which struggle with the temporal evolution of the H^1

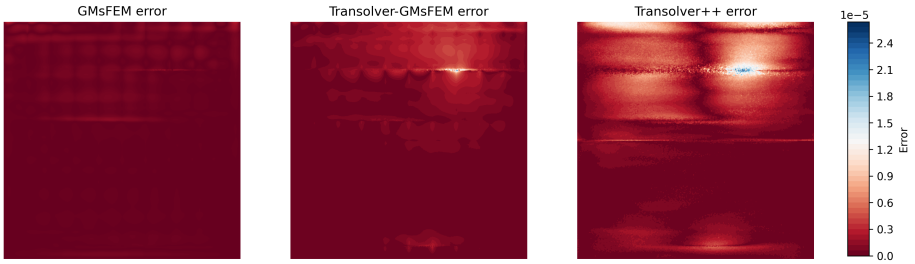


Figure 22: Comparison of absolute error maps for the fine-scale solution (No. 801) in the Diffusion with mixed BCs benchmark with spatial forcing term (20).

norm. In these time-dependent scenarios, standard neural PDE solvers often exhibit relative H^1 errors exceeding 25%, whereas our proposed pipeline maintains much higher structural integrity in the solution.

Throughout these tables, **bold** values indicate the best performance among the learning-based models, while underlined values represent the second-best results. The theoretical GMsFEM baseline, highlighted in blue, is provided as a reference for the upper bound of achievable accuracy.

N.2 Fine-scale solution errors

Figure 22 illustrates the performance of the evaluated methods for Diffusion with mixed BCs benchmark with a spatially variable source term (20) on a challenging test case (No. 801). The input field $\kappa(x)$ (see Figure 11) has a high contrast ratio ranging from 0 to 9600.

A comparative analysis of the absolute error maps (see Figure 22) reveals the following:

- **Classical GMsFEM**: The errors are distributed uniformly across the computational domain without distinct localization.
- **Transolver-GMsFEM (Proposed)**: The error profile is nearly indistinguishable from the classical GMsFEM error. However, slight localized increases in error can be observed near the sharpest transitions of the $\kappa(x)$ field (e.g., the bright spot in the upper right quadrant). This suggests that while the neural approximation is robust, it remains sensitive to extreme local contrast.
- **Transolver++ (Baseline)**: This model exhibits significantly higher and more widespread absolute errors. The most pronounced discrepancies occur in the upper region of the domain, which correlates with areas of extreme coefficient field values. This highlights the difficulty global neural PDEs solvers face when attempting to capture intricate physical interactions in high-contrast media.

N.3 Time-dependent Heat Equation

Figures 23 and 24 show the evolution of the relative L_2 and H^1 errors over time for the time-dependent Heat equation using different methods. Each curve represents the mean error across 200 test samples.

The state-of-the-art baseline, Transolver++, GNOT, and GAOT, performs worst in both error norms. For Transolver++, the mean relative L_2 error increases from 3.72% to 6.58% over 20 time steps, while its mean relative H^1 error rises from 31.43% to 45.47%. GNOT follows a similar downward trend, with its L_2 error climbing from 4.74% to 7.68% and its H^1 error jumping from 34.19% to 46.64%. Even GAOT, which starts with a more competitive 2.51% L_2 error and 18.55% H^1 error, sees these values deteriorate to 4.97% and 29.35%, respectively. Collectively, these results highlight both low accuracy and clear temporal deterioration across all three models.

When the Transolver-GMsFEM offline basis is combined with a Transolver-GMsFEM online stage, the L_2 error increases from 0.130% to 0.412%, and the H^1 error grows from 2.78% to 8.22%, indicating that the fast online stage does not fully suppress error accumulation over time. In contrast, coupling the same Transolver-GMsFEM offline basis with a classical GMsFEM online stage yields stable and accurate results: the L_2 error changes from 0.125% to 0.131%, and the H^1 error from 2.72% to 2.71%.

The full GMsFEM benchmark attains the lowest errors overall, with the L_2 error decreasing from 0.125% to 0.115% and the H^1 error reducing to 1.62%, showing systematic improvement over time. Although GMsFEM (offline, online) remains the most accurate method, the performance gap relative to the Transolver-GMsFEM (offline) + GMsFEM (online) remains modest (for example, H^1: 2.71% versus 1.62%; L_2: 0.131% versus 0.115%), while the proposed method retains substantial offline speedup.

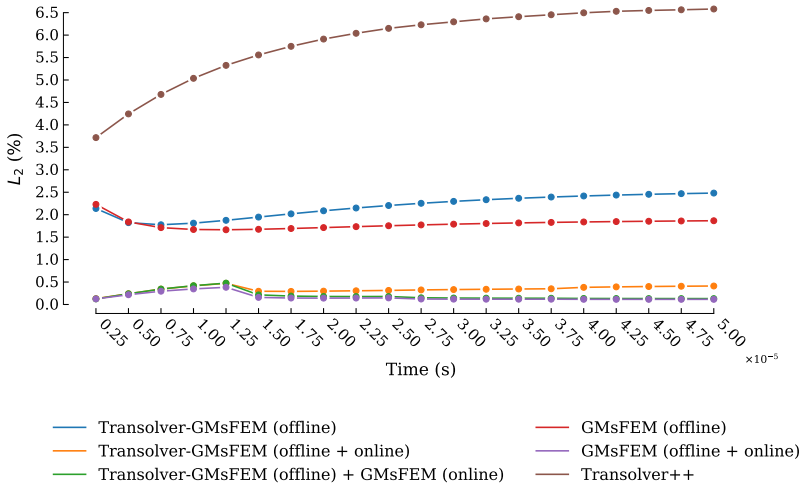


Figure 23: The L_2 error comparison for time-dependent Heat with spatial forcing term (20) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The L_2 error obtained using 4 offline and 1 online basis functions.

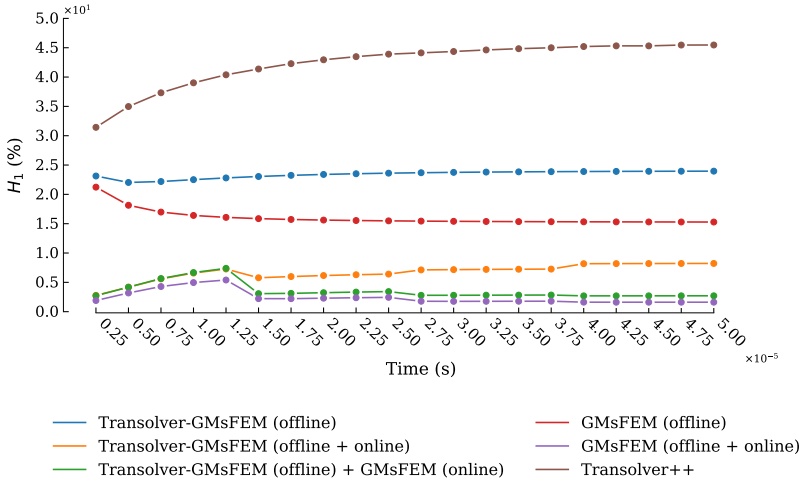


Figure 24: The H^1 error comparison for time-dependent Heat with spatial forcing term (20) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The H^1 error obtained using 4 offline and 1 online basis functions.

While GMsFEM (offline) and Transolver-GMsFEM (offline) exhibit large errors (H^1: 16%–24%; $L_2 \sim 2\%$), these results are notably lower than those of the SOTA baselines discussed previously. Despite this relative advantage, the magnitude of these errors confirms that the online correction stage remains essential for achieving acceptable accuracy.

N.4 Time-dependent Richards Equation

Figures 25 and 26 compare the evolution of the relative L_2 and H^1 errors, respectively, over time for Richards' equation using different methods. The errors shown in the image are the mean values over 200 test samples.

The SOTA baselines – Transolver++, GNOT, and GAOT – are clearly the worst performers in both norms, demonstrating both poor accuracy and a consistent trend of temporal deterioration over the 20 steps. For Transolver++, the mean relative L_2 error grows from 3.64% to 5.30%, while its mean relative H^1 error rises from 34% to 45%. GNOT follows a similar trajectory, with its L_2 error climbing from 4.78% to 7.49% and its

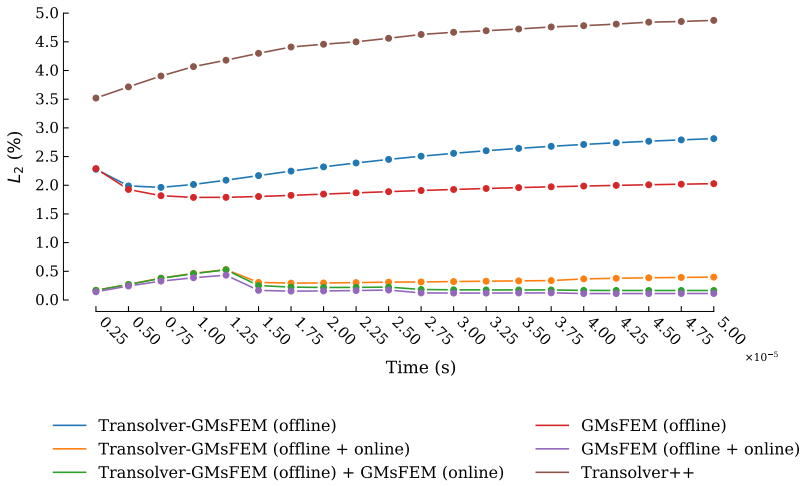


Figure 25: The L_2 error comparison for time-dependent Richards with spatial forcing term (20) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The L_2 error obtained using 4 offline and 1 online basis functions.

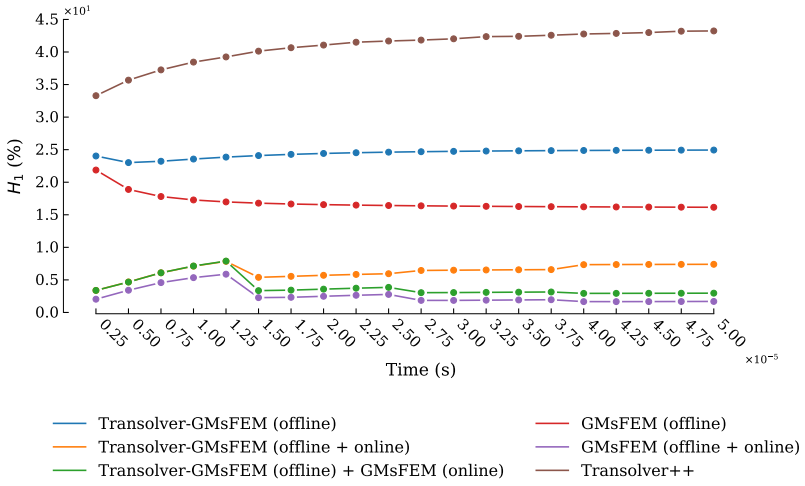


Figure 26: The H^1 error comparison for time-dependent Richards with spatial forcing term (20) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The H^1 error obtained using 4 offline and 1 online basis functions.

H^1 error increasing from 33.73% to 43.11%. Even GAOT exhibits this decline, as its L_2 error increases from 2.52% to 3.53% and its H^1 error shifts from 17.59% to 24.43%, confirming that all three models struggle to maintain stability over time.

With Transolver-GMsFEM online, in both metrics the fast online stage does not fully control error over time. When the same Transolver-GMsFEM offline is combined with the classical GMsFEM online stage, L_2 error stays around 0.170% and ends near 0.166%, and H^1 error decreases from about 3.38% to about 2.96%.

Full GMsFEM achieves the lowest errors overall – L_2 from about 0.14% down to about 0.11%, H^1 down to about 1.69%. The gap between full GMsFEM and the proposed method with GMsFEM online remains modest while the proposed method keeps the large offline and online speedups. The offline-only variants (GMsFEM and Transolver-GMsFEM) sit in the 16–25% range in H^1 and around 2% in L_2, so the online stage is essential.

Even in the fully accelerated configuration, combining Transolver-GMsFEM offline and online stages, the proposed method maintains errors well below those of Transolver++, demonstrating its practical viability

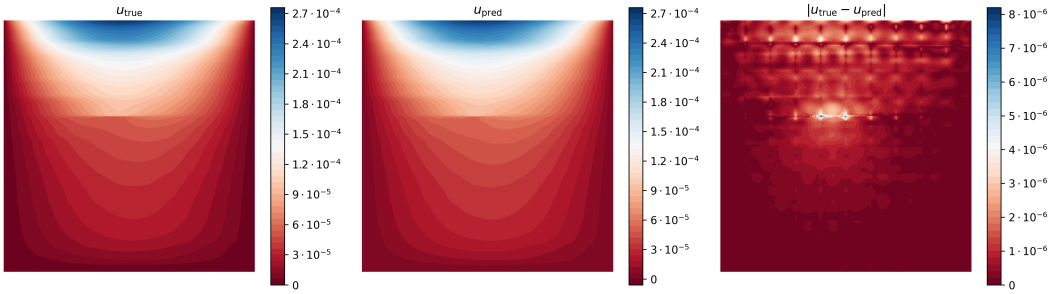


Figure 27: Numerical results for OOD Test Case No. 2 of the Diffusion with mixed BCs benchmark with a spatial forcing term (20). The fine-scale solution is u_{true}, the Transolver-GMsFEM prediction is u_{pred}, and the absolute error is $|u_{\text{true}} - u_{\text{pred}}|$.

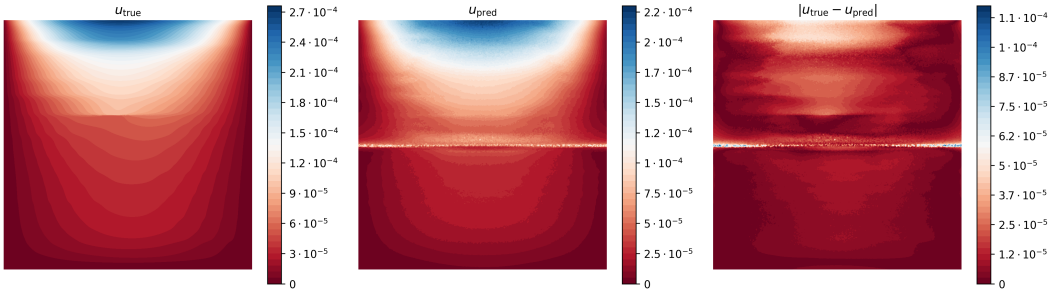


Figure 28: Numerical results for OOD Test Case No. 2 of the Diffusion with mixed BCs benchmark with a uniform forcing term (19). The fine-scale solution is u_{true}, the Transolver++ prediction is u_{pred}, and the absolute error is $|u_{\text{true}} - u_{\text{pred}}|$.

in runtime-critical settings. When higher accuracy comparable to classical GMsFEM is required, the same fast offline basis can be seamlessly paired with a standard GMsFEM online stage, preserving most of the computational speedup while recovering stable, near-GMsFEM accuracy in both L^2 and H^1 norms.

O Out of Distribution Results

This section details the robustness of the Transolver-GMsFEM framework when subjected to out-of-distribution (OOD) scenarios. We evaluate generalization across two primary dimensions: shifts in the statistical parameters of the permeability fields and shifts in the source terms (right-hand sides).

O.1 Robustness to Permeability Field Shifts

Table 9 (2D) and Table 12 (3D) present the steady-state performance when the models are tested on permeability fields with different correlation lengths and variance than those encountered during training. While neural PDE solvers like Transolver++, GNOT, and GAOT suffer from catastrophic accuracy loss—with L_2 errors often exceeding 20% in 2D and reaching over 60% in 3D Richards problems – Transolver-GMsFEM remains remarkably stable. Its performance closely tracks the classical GMsFEM baseline, demonstrating that the learned multiscale basis functions generalize effectively.

To further stress-test generalization, we evaluate the 2D steady-state performance on a more challenging OOD scenario where the test data is generated from entirely different functional frameworks, breaking the underlying covariance structure of the training data. As shown in Table 13, where Types 1, 2, and 3 correspond to Spectral coefficient, Cosines coefficient, and Waved OOD coefficient, respectively, pure neural PDE solvers exhibit catastrophic failures under these structurally shifted distributions. For instance, on the nonlinear Richards problem with source term (19) (Type 3), Transolver++ reaches an L_2 error of 59.30% and an H^1 error of 83.54%. GNOT and GAOT are similarly unstable, with GNOT reaching a staggering 164.45% L_2 error and 268.56% H^1 error on the same task. In contrast, Transolver-GMsFEM maintains remarkable robustness. Its

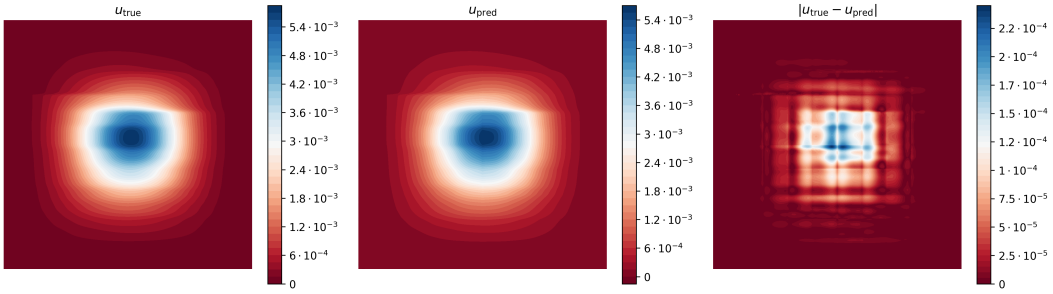


Figure 29: Numerical results for OOD Test Case No. 122 of the Richards benchmark with a uniform forcing term (19). The fine-scale solution is u_{true}, the Transolver-GMsFEM prediction is u_{pred}, and the absolute error is $|u_{\text{true}} - u_{\text{pred}}|$.

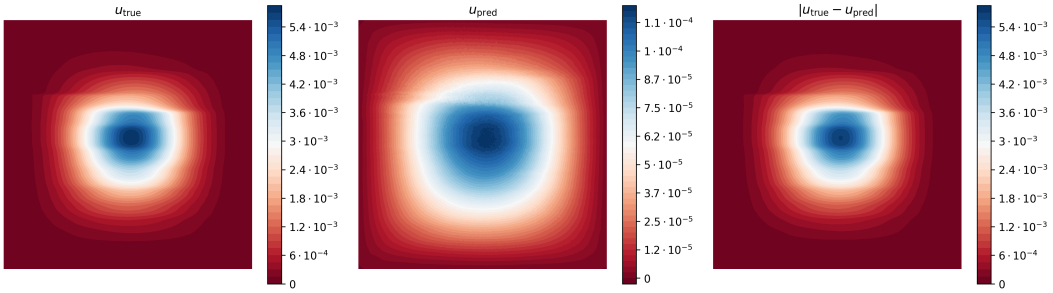


Figure 30: Numerical results for OOD Test Case No. 122 of the Richards benchmark with a spatial forcing term (20). The fine-scale solution is u_{true}, the Transolver++ prediction is u_{pred}, and the absolute error is $|u_{\text{true}} - u_{\text{pred}}|$.

L_2 errors remain consistently low (typically between 1% and 6.8%) and closely track the classical GMsFEM baseline across all unseen generative frameworks.

For 2D time-dependent problems, the errors under OOD conditions are presented in Table 10. Even with shifted permeability statistics, the hybrid Transolver-GMsFEM (offline) + GMsFEM (online) configuration maintains high precision, yielding L_2 errors below 0.3%. The fully learned Transolver-GMsFEM (offline + online) variant also demonstrates strong generalization in these scenarios; with an L_2 error of approximately 0.5%, it consistently secures the second-best performance among learning-based methods.

Visual comparisons of the error maps further illustrate the superior generalization of the proposed method. For the Diffusion with mixed BCs benchmark (Test Case No. 2), Figure 27 shows that the Transolver-GMsFEM prediction is nearly indistinguishable from the fine-scale ground truth u_{true}, with a maximum absolute error of only around 8×10^{-6}. In contrast, the Transolver++ baseline (see Figure 28) exhibits significantly larger errors, reaching 1×10^{-4}. A similar trend is observed in the nonlinear Richards benchmark (Test Case No. 122). As shown in Figure 29, our method maintains a high level of fidelity with a maximum absolute error of 2×10^{-4}. Meanwhile, the Transolver++ baseline (see Figure 30) fails to accurately capture the solution profile, resulting in a much larger maximum error of 5×10^{-3}. These visualizations confirm that our approach prevents the severe performance degradation typical of neural PDE solvers in OOD scenarios.

O.2 Robustness to Source-Term Shifts

For 2D time-dependent problems, Table 11 presents the errors in case of OOD right-hand sides. It is important to note that for steady-state problems, we do not need to test the robustness of our method against OOD right-hand sides; this is because the offline basis functions do not depend on the right-hand sides or boundary conditions. However, in time-dependent scenarios, Transolver-GMsFEM (offline + online) maintains an error margin of around 1%, again taking the second place behind the Transolver-GMsFEM (offline) + GMsFEM (online) configuration. In contrast, the errors of standard neural PDE solvers escalate drastically (up to 180%).

Table 9: OOD stochastic permeability-field. Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver++, GNOT, and GAOT for 2D problems ($L = 8$, $N_c = 121$). Diffusion with mixed BCs is abbreviated as Diffusion+mBCs. Models $\mathcal{G}_1, \mathcal{G}_2$ are trained on generated stochastic permeability fields with parameters $\ell_x = 0.02$, $\ell_y = 0.6$, and $\sigma_R^2 = 1.0$, and tested on generated stochastic permeability fields with parameters $\ell_x = 0.015$, $\ell_y = 0.5$, and $\sigma_R^2 = 2.0$. Method abbreviations: GM = GMsFEM, Tr-GM = Transolver-GMsFEM, Tr++ = Transolver++, Tr = Transolver.

	GM		Tr-GM		Tr++		GNOT		GAOT		Tr	
Dataset	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1
Diff., (19)	0.84	11.71	**0.91**	**16.49**	18.72	68.90	26.41	58.84	23.05	44.66	16.65	53.47
Diff., (20)	1.70	13.48	**1.88**	**19.43**	20.96	65.77	28.75	53.95	15.85	44.67	19.12	56.71
Rich., (19)	4.56	17.85	**5.77**	**22.27**	81.33	92.51	232.52	288.34	77.69	87.47	82.06	90.30
Rich., (20)	2.06	14.28	**2.48**	**17.96**	26.48	67.44	32.26	55.99	24.14	49.34	21.94	51.16
Diff. + mBCs, (19)	0.62	9.99	**0.88**	**16.18**	14.33	69.88	19.26	77.15	39.64	289.74	12.93	53.67
Diff. + mBCs, (20)	0.66	10.86	**0.84**	**13.93**	18.72	212.66	—	—	10.90	130.52	19.14	64.57

Table 10: OOD stochastic permeability-field. Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline + online), Transolver-GMsFEM (offline + online), Transolver++, GNOT, and GAOT for 2D time-dependent problems ($L = 4$). Models $\mathcal{G}_1, \mathcal{G}_2$ are trained on generated stochastic permeability fields with parameters $\ell_x = 0.02$, $\ell_y = 0.6$, and $\sigma_R^2 = 1.0$, and tested on generated stochastic permeability fields with parameters $\ell_x = 0.015$, $\ell_y = 0.5$, and $\sigma_R^2 = 2.0$.

	Heat, (20)		Richards, (20)	
Method	L_2	H^1	L_2	H^1
GMsFEM (offline)	1.84	16.36	1.99	18.01
Transolver-GMsFEM (offline)	2.26	24.19	2.69	26.12
Transolver-GMsFEM (offline) + GMsFEM (online)	**0.25**	**4.36**	**0.27**	**4.62**
GMsFEM (offline + online)	0.16	2.25	0.20	2.77
Transolver-GMsFEM (offline + online)	<u>0.41</u>	<u>8.20</u>	<u>0.46</u>	<u>8.42</u>
Transolver++	12.97	60.84	11.22	64.85
GAOT	14.33	39.54	13.94	44.59
GNOT	10.69	51.83	9.80	50.21

P Results for 3D datasets

To further assess the scalability and practical utility of our framework, we evaluate Transolver-GMsFEM on complex 3D problems. The computational domain is discretized using a high-resolution mesh consisting of $1,382,363$ cells and $238,152$ vertices. The entire 3D inference pipeline, including the prediction of basis functions for 1.38M cells, was executed on a single NVIDIA A100 GPU (80 GB), demonstrating the memory efficiency of Transolver-GMsFEM architecture.

P.1 Steady-state performance

Table 14 summarizes the performance for 3D steady-state problems using $L = 8$ basis functions. Similar to the 2D cases, Transolver-GMsFEM consistently outperforms the pure neural baseline, Transolver++, across both Diffusion and Richards equation benchmarks. Our method achieves L_2 and H^1 errors that are remarkably close to the theoretical GMsFEM baseline (highlighted in blue).

Table 11: OOD right-hand sides. Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline + online), Transolver-GMsFEM (offline + online), Transolver++, GNOT, and GAOT for 2D time-dependent problems ($L = 4$). Model $\mathcal{G}_3$ is trained on (20), and tested on (21).

	Heat, (21)		Richards, (21)	
Method	L_2	H^1	L_2	H^1
GMsFEM (offline)	2.10	16.81	2.13	16.74
Transolver-GMsFEM (offline)	2.38	22.00	2.40	21.72
Transolver-GMsFEM (offline) + GMsFEM (online)	**0.26**	**3.80**	**0.27**	**3.97**
GMsFEM (offline + online)	0.18	2.37	0.20	2.63
Transolver-GMsFEM (offline + online)	<u>1.08</u>	<u>16.06</u>	<u>1.13</u>	<u>14.43</u>
Transolver++	51.49	84.70	48.54	122.67
GAOT	183.00	158.81	181.02	160.05
GNOT	48.97	71.70	57.98	77.73

Table 12: OOD stochastic permeability-field. Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), and Transolver++ for 3D problems ($L = 8$, $N_c = 216$). Models $\mathcal{G}_1, \mathcal{G}_2$ are trained on generated stochastic permeability fields with parameters $\ell_x = 0.02$, $\ell_y = 0.6$, $\ell_z = 0.2$, and $\sigma_R^2 = 4.0$, and tested on generated stochastic permeability fields with parameters $\ell_x = 0.015$, $\ell_y = 0.4$, $\ell_z = 0.1$, and $\sigma_R^2 = 3.0$.

		GMsFEM		Transolver-GMsFEM		Transolver++	
L	Dataset	L_2	H^1	L_2	H^1	L_2	H^1
8	Diffusion, (22)	2.63	17.91	**2.73**	**18.34**	21.11	45.34
	Diffusion, (23)	4.41	20.43	**4.49**	**21.29**	26.82	63.38
	Richards, (22)	3.34	14.18	**4.14**	**16.39**	69.12	77.57
	Richards, (23)	4.27	19.51	**4.58**	**20.65**	44.72	68.32

P.2 Time-dependent performance

For 3D time-dependent Heat and Richards equations, results are presented in Table 15. The hybrid Transolver-GMsFEM (offline) + GMsFEM (online) configuration yields the highest accuracy among learning-based approaches, maintaining L_2 errors below 0.8%. The purely learned Transolver-GMsFEM (offline + online) variant also shows strong performance, consistently taking second place and significantly outperforming the standard Transolver++.

The evolution of relative errors over 10 timesteps for the 3D Heat and Richards equations is illustrated in Figures 31 through 34. These figures compare the error trajectories using 4 offline and 1 online basis functions across various configurations. The hybrid configurations involving online enrichment (Transolver-GMsFEM + GMsFEM online) demonstrate superior stability, with error levels remaining consistently low and close to GMsFEM (offline + online) baseline. In contrast, Transolver++ exhibits a noticeable accumulation of error over time.

P.3 Computational Efficiency

As shown in Table 16, the classical GMsFEM offline phase requires over $5,900$ seconds to compute basis functions for $L = 4$ and over $6,200$ seconds for $L = 8$. In contrast, Transolver-GMsFEM completes the same task in approximately 22 to 27 seconds, representing a speedup of over $230\times$.

Furthermore, the online enrichment step is significantly accelerated. While the classical online GMsFEM takes 94.18 seconds, our method reduces this to just 13.21 seconds.

Table 13: Results for OOD permeability fields generated using entirely different functional frameworks than the training coefficients. Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver++, GNOT, and GAOT for 2D problems ($L = 8$, $N_c = 121$). Diffusion with mixed BCs is abbreviated as Diff.+mBCs. Models $\mathcal{G}_1, \mathcal{G}_2$ are trained on stochastic permeability fields with parameters $\ell_x = 0.02$, $\ell_y = 0.6$, and $\sigma_R^2 = 1.0$, and tested on unseen generative families (Spectral, Cosines and Waved OOD; see Appendix G.2). Method abbreviations: GM = GMsFEM, Tr-GM = Transolver-GMsFEM, Tr++ = Transolver++. Bold and underlined values denote the best and second-best results among the learning-based methods; GMsFEM is shown as a reference.

		GM		Tr-GM		Tr++		GNOT		GAOT	
Type	Dataset	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1
1	Diff., (19)	1.08	12.66	**2.78**	**26.30**	9.82	68.21	9.33	61.83	9.71	55.12
	Diff., (20)	0.87	11.97	**2.38**	**22.84**	20.24	65.46	21.48	80.99	14.49	44.19
	Rich., (19)	2.19	12.87	**5.13**	**24.93**	17.90	79.21	17.96	71.89	18.69	68.80
	Rich., (20)	1.04	12.35	**5.88**	**39.16**	25.84	81.04	25.76	65.09	20.93	52.91
	Diff. + mBCs, (19)	1.25	15.80	**3.80**	**30.22**	17.07	77.13	16.73	67.71	17.14	66.65
	Diff. + mBCs, (20)	1.26	18.79	**3.63**	**29.78**	18.35	78.46	—	—	18.20	65.66
2	Diff., (19)	1.05	12.25	**3.22**	**27.25**	9.65	67.54	9.27	61.66	9.75	57.41
	Diff., (20)	0.87	12.83	**2.34**	**22.99**	21.32	66.69	22.31	86.60	16.77	48.32
	Rich., (19)	2.08	12.40	**6.83**	**28.83**	17.21	78.22	17.36	69.50	18.27	70.32
	Rich., (20)	1.03	13.36	**4.05**	**33.10**	26.37	79.22	26.25	65.77	25.62	59.12
	Diff. + mBCs, (19)	1.33	19.30	**4.64**	**34.40**	18.01	75.90	17.70	70.49	17.81	69.54
	Diff. + mBCs, (20)	1.47	22.93	**3.40**	**28.71**	19.08	76.06	—	—	18.11	65.53
3	Diff., (19)	0.73	13.39	**1.20**	**19.31**	10.70	62.54	17.25	53.38	13.34	40.35
	Diff., (20)	1.49	15.69	**2.37**	**22.89**	13.81	65.08	18.49	54.50	10.26	40.89
	Rich., (19)	2.84	19.15	**5.13**	**24.92**	59.30	83.54	164.45	268.56	18.27	70.32
	Rich., (20)	1.83	17.49	**2.80**	**21.98**	18.64	66.30	22.09	55.17	16.37	45.35
	Diff. + mBCs, (19)	0.48	9.93	**0.88**	**16.86**	10.11	67.30	10.71	58.70	17.96	146.99
	Diff. + mBCs, (20)	0.69	11.53	**1.02**	**15.91**	14.09	186.80	—	—	8.03	75.18

Table 14: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), and Transolver++ for 3D problems ($L = 8$, $N_c = 216$).

		GMsFEM		Transolver-GMsFEM		Transolver++	
L	Dataset	L_2	H^1	L_2	H^1	L_2	H^1
8	Diffusion, (22)	1.69	10.90	**1.98**	**12.32**	3.16	17.16
	Diffusion, (23)	3.73	19.59	**3.91**	**20.58**	8.42	29.73
	Richards, (22)	2.56	17.90	**2.60**	**18.70**	5.79	26.64
	Richards, (23)	4.09	19.23	**4.22**	**20.28**	8.87	30.86

Q Results for Perforated Media

For the perforated benchmarks (see Tables 17 and 18), Transolver-GMsFEM closely matches classical GMsFEM in both 2D and 3D, while all neural baselines degrade substantially. In 2D, Transolver-GMsFEM yields L_2 errors of 0.67 and 0.78 for the two right-hand sides, versus 0.30 and 0.57 for GMsFEM, whereas baselines range from 6.27 to 16.77. In 3D, Transolver-GMsFEM achieves L_2 errors of 1.30 and 3.82 versus 1.10 and 3.39 for GMsFEM, with baselines between 5.04 and 23.46. The same trend holds for H^1 errors. These results confirm that the method handles non-rectangular local domains without sacrificing robustness.

Table 15: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline + online), Transolver-GMsFEM (offline + online), and Transolver++ for 3D time-dependent problems ($L = 4$).

	Heat, (23)		Richards, (23)	
Method	L_2	H^1	L_2	H^1
GMsFEM (offline)	5.79	23.88	6.79	26.87
Transolver-GMsFEM (offline)	6.19	26.10	7.18	29.09
Transolver-GMsFEM (offline) + GMsFEM (online)	**0.67**	**4.53**	**0.71**	**4.64**
GMsFEM (offline + online)	0.63	3.41	0.66	3.55
Transolver-GMsFEM (offline + online)	<u>0.76</u>	<u>5.62</u>	<u>0.81</u>	<u>5.78</u>
Transolver++	9.37	35.22	9.60	38.98

Table 16: Basis generation time: (i) $L = 4$, GMsFEM (offline) vs. Transolver-GMsFEM (offline); (ii) $L = 8$, GMsFEM (offline) vs. Transolver-GMsFEM (offline); (iii) GMsFEM (online) vs. Transolver-GMsFEM (online) for one enrichment step for 3D dataset.

L	Method	Time, s
4	GMsFEM (offline)	5918.20
	Transolver-GMsFEM (offline)	**22.31**
8	GMsFEM (offline)	6230.50
	Transolver-GMsFEM (offline)	**26.81**
4	GMsFEM (online)	94.18
	Transolver-GMsFEM (online)	**13.21**

Table 17: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver++, GNOT, GAOT, and Transolver for 2D problems with perforations ($L = 8$, $N_c = 121$). Method abbreviations: Tr-GMsFEM = Transolver-GMsFEM, Tr++ = Transolver++, Tr = Transolver.

		GMsFEM		Tr-GMsFEM		Tr++		GNOT		GAOT		Tr	
L	Dataset	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1
8	Diffusion + perforations, (19)	0.30	12.76	**0.67**	**16.67**	<u>6.27</u>	43.29	8.31	38.41	6.85	37.86	6.36	<u>34.16</u>
	Diffusion + perforations, (20)	0.57	13.60	**0.78**	**19.12**	<u>12.96</u>	51.71	16.77	45.03	14.59	43.41	13.83	<u>42.42</u>

Table 18: Performance comparison of GMsFEM (offline), Transolver-GMsFEM (offline), Transolver++, GNOT, GAOT, and Transolver for 3D problems with perforations ($L = 8$, $N_c = 216$). Method abbreviations: Tr-GMsFEM = Transolver-GMsFEM, Tr++ = Transolver++, Tr = Transolver.

		GMsFEM		Tr-GMsFEM		Tr++		GNOT		GAOT		Tr	
L	Dataset	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1	L_2	H^1
8	Diffusion + perforations, (22)	1.10	10.06	**1.30**	**12.07**	5.19	18.06	5.82	18.82	5.90	20.40	5.04	17.02
	Diffusion + perforations, (23)	3.39	23.06	**3.82**	**26.05**	20.58	45.98	23.46	50.32	20.94	43.06	20.84	45.09

R Computational Cost Analysis

This section provides a detailed wall-clock breakdown.

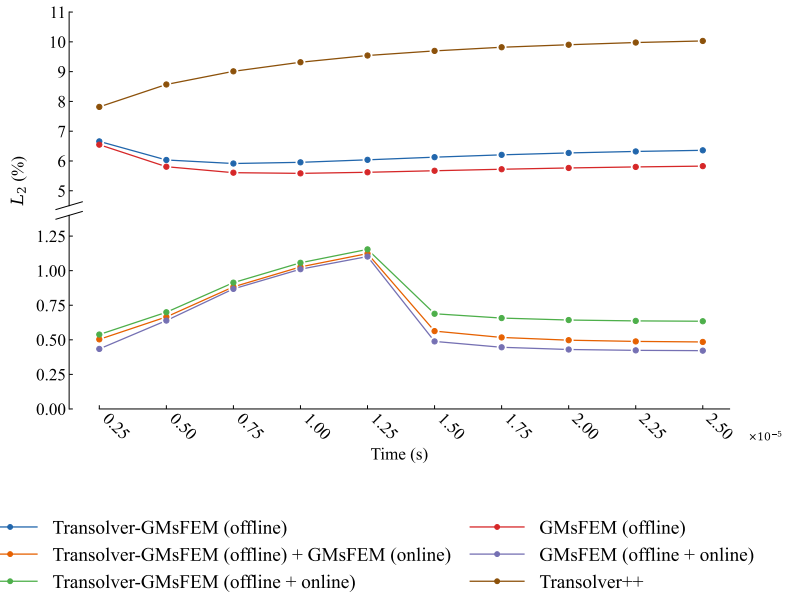


Figure 31: The L_2 error comparison for time-dependent heat 3D with spatial forcing term (23) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The L_2 error obtained using 4 offline and 1 online basis functions.

Hardware and parallelisation. All timings are wall-clock times collected using 16 CPU cores and 1 A100 NVIDIA GPU. Training-data generation – computing local eigenvalue problems and solving fine-scale PDE systems – is a **CPU-only** workload that requires no GPU time and scales linearly with the number of training samples N_{train}. We parallelize this phase across all 16 CPU cores, giving a wall-clock overhead of $\lceil N_{\text{train}}/16 \rceil \cdot T_{\text{data}}$.

Tables 19 and 20 list every primitive timing component used to derive aggregate costs in subsequent subsections.

Table 19: Measured timing primitives for the **steady-state** problem ($L = 8$ offline basis functions per local domain, $N_{\text{train}} = 700$ for 2D, $N_{\text{train}} = 150$ for 3D).

Symbol	Description	2D	3D
GMsFEM			
T_{off}	**Offline** basis generation on *all* local domains (per sample)	650.98 s	6 230.50 s
$T_{\text{sol}}^{\text{off}}$	**Coarse-scale system** solve and projection with offline basis only (per sample)	0.30 s	7.00 s
Transolver-GMsFEM			
$T_{\text{off}}^{\text{extr}}$	**Offline** basis gen. on *extreme* local domains only (per sample)	130.53 s	2 017.60 s
$T_{\mathcal{G}_1}$	Wall time for **training** model $\mathcal{G}_1$ (1 GPU)	17.39 h	22.31 h
$T_{\mathcal{G}_2}$	Wall time for **training** model $\mathcal{G}_2$ (1 GPU)	24.30 h	25.10 h
$T_{\text{off}}^{\mathcal{G}_1,\mathcal{G}_2}$	**Offline** basis prediction by $\mathcal{G}_1$ + $\mathcal{G}_2$ (per sample)	6.12 s	26.81 s
Transolver++			
T_{fine}	**Fine-grid PDE solve** (per sample)	0.50 s	23.00 s
T_{train}	**Model training** wall time (1 GPU)	12.00 h	16.70 h
T_{inf}	**Inference** (per sample)	0.006 s	0.007 s

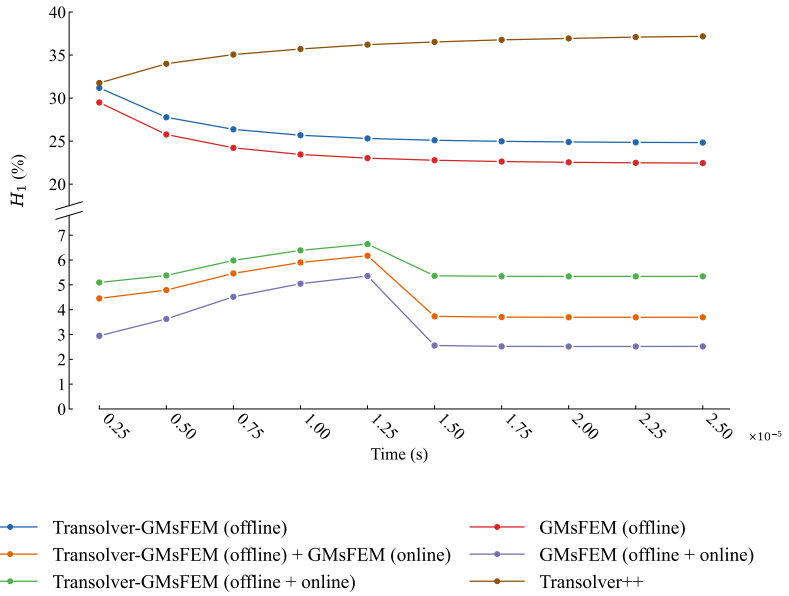


Figure 32: The H^1 error comparison for time-dependent heat 3D with spatial forcing term (23) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The H^1 error obtained using 4 offline and 1 online basis functions.

R.1 Per-Sample Inference Cost

R.1.1 Steady-state equations

For steady-state problems, the per-sample inference cost is determined by the following formula once the model is trained

$$\begin{aligned} I^{\text{ss}}_{\text{GMsFEM}} &= T_{\text{off}} + T^{\text{off}}_{\text{sol}}, \\ I^{\text{ss}}_{\text{Tr-GMsFEM}} &= T^{\mathcal{G}_1,\mathcal{G}_2}_{\text{off}} + T^{\text{off}}_{\text{sol}}, \\ I^{\text{ss}}_{\text{Tr++}} &= T_{\text{inf}}, \end{aligned}$$

where the $I^{\text{ss}}_{\text{GMsFEM}}$, $I^{\text{ss}}_{\text{Tr-G}}$, $I^{\text{ss}}_{\text{T++}}$ are the inference times for steady-state equations for GMsFEM, Transolver-GMsFEM and Transolver++.

R.1.2 Time-dependent equations

For time-dependent problems, the per-sample inference costs are determined by the formulas below

$$\begin{aligned} I^{\text{td}}_{\text{GMsFEM}} &= T_{\text{off}} + T_{\text{on}} \cdot N_{\text{enrich}} + T^{\text{off}}_{\text{sol}} \cdot N_{\text{enrich}} + T^{\text{on}}_{\text{sol}} \cdot N_T, \\ I^{\text{td}}_{\text{Tr-G}} &= T^{\mathcal{G}_1,\mathcal{G}_2}_{\text{off}} + T^{\mathcal{G}_3}_{\text{on}} \cdot N_{\text{enrich}} + T^{\text{off}}_{\text{sol}} \cdot N_{\text{enrich}} + T^{\text{on}}_{\text{sol}} \cdot N_T, \\ I^{\text{td}}_{\text{T++}} &= T_{\text{inf}} \cdot N_T, \end{aligned}$$

where the $I^{\text{td}}_{\text{GMsFEM}}$, $I^{\text{td}}_{\text{Tr-G}}$, $I^{\text{td}}_{\text{T++}}$ are the inference times for time-dependent equations for GMsFEM, Transolver-GMsFEM and Transolver++.

The coarse-system solve terms $T^{\text{off}}_{\text{sol}} \cdot N_{\text{enrich}} + T^{\text{on}}_{\text{sol}} \cdot N_T$ are identical for GMsFEM and Transolver-GMsFEM; the only difference between the two methods is the time spent computing basis functions (T_{off} vs. $T^{\mathcal{G}_1,\mathcal{G}_2}_{\text{off}}$ and $T_{\text{on}} \cdot N_{\text{enrich}}$ vs. $T^{\mathcal{G}_3}_{\text{on}} \cdot N_{\text{enrich}}$).

Table 21 gives the resulting numerical values.

R.2 Training Cost and Data Generation

The initial computational cost of Transolver-GMsFEM and Transolver++ consists of two stages: generating training data on the CPU and performing model training on the GPU.

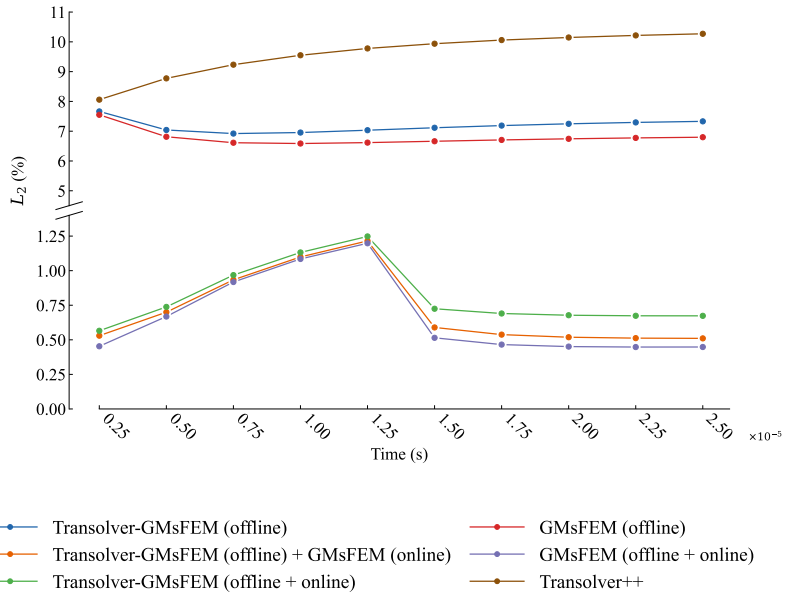


Figure 33: The L_2 error comparison for time-dependent Richards 3D with spatial forcing term (23) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The L_2 error obtained using 4 offline and 1 online basis functions.

Transolver-GMsFEM. For the steady-state problem, training data for $\mathcal{G}_1$ and $\mathcal{G}_2$ requires offline basis functions computed only on the **extreme** local domains:

$$T^{\text{ss}}_{\text{data}} = \lceil N_{\text{train}}/16 \rceil \cdot T^{\text{extr}}_{\text{off}}, \qquad T^{\text{ss}}_{\text{model}} = T_{\mathcal{G}_1} + T_{\mathcal{G}_2}.$$

For the time-dependent PDEs problem, offline and online basis functions are both required as training targets for $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$:

$$T^{\text{td}}_{\text{data}} = \lceil N_{\text{train}}/16 \rceil \cdot \big(T_{\text{off}} + T_{\text{on}} \cdot N_{\text{enrich}}\big), \quad T^{\text{td}}_{\text{model}} = T_{\mathcal{G}_1} + T_{\mathcal{G}_2} + T_{\mathcal{G}_3}.$$

Transolver++. Training data consists of fine-grid PDE solutions:

$$T^{\text{ss}}_{\text{data}} = \lceil N_{\text{train}}/16 \rceil \cdot T_{\text{fine}}, \quad T^{\text{td}}_{\text{data}} = \lceil N_{\text{train}}/16 \rceil \cdot T_{\text{fine}} \cdot N_T, \quad T^{\text{ss/td}}_{\text{model}} = T^{\text{ss/td}}_{\text{train}}.$$

R.3 Break-Even Analysis

We define the **break-even count** N^*_{test} as the minimum number of distinct test samples at which the total cost of the surrogate (one-time training plus per-sample inference) equals the cost of running the classical solver from scratch for each sample:

$$N^*_{\text{test}} = \left\lceil \frac{T_{\text{data}} + T_{\text{model}}}{I^{\text{ss / td}}_{\text{GMsFEM}} - I^{\text{ss / td}}_{\text{Tr-G}}} \right\rceil.$$

For $N_{\text{test}} > N^*_{\text{test}}$, the surrogate is cheaper overall.

The break-even analysis (Table 23) shows that Transolver-GMsFEM becomes cheaper overall after 242 samples for 2D steady-state problems and after 31 samples for 3D steady-state problems.

R.4 Comparison with a parallelized GMsFEM baseline

The local spectral problems (5) and (7) are independent across coarse neighborhoods ω_i and can therefore be solved in parallel. Since a comparison against serial GMsFEM alone may understate the performance of the classical approach, we also measure the offline basis generation time of a parallelized GMsFEM implementation. For one coefficient field, offline basis generation consists of $N_c = 121$ independent local problems ($L = 8$ basis functions each), which we distribute across CPU cores on a single shared-memory node. The resulting wall-clock times are reported in Table 24.

Table 20: Measured timing primitives for the **time-dependent** problem ($L = 4$, $N_T = 20\,/\,10$ in 2D / 3D, online enrichment every 5 steps, $N_{\text{enrich}} = N_T/5$, $N_{\text{train}} = 700$ for 2D, $N_{\text{train}} = 150$ for 3D).

Symbol	Description	2D	3D
GMsFEM			
T_{off}	**Offline** basis gen. on all local domains (per sample)	476.74 s	5 918.20 s
T_{on}	**Online** basis gen. on all local domains (per enrichment step)	6.66 s	94.18 s
$T_{\text{sol}}^{\text{off}}$	**Coarse-scale system** solve with offline basis only (per timestep)	0.30 s	7.00 s
$T_{\text{sol}}^{\text{on}}$	**Coarse-scale system** solve with offline+online basis (per timestep)	0.45 s	11.00 s
Transolver-GMsFEM			
$T_{\mathcal{G}_1}$	Wall time for **training** $\mathcal{G}_1$ (1 GPU)	17.39 h	22.31 h
$T_{\mathcal{G}_2}$	Wall time for **training** $\mathcal{G}_2$ (1 GPU)	24.30 h	25.10 h
$T_{\mathcal{G}_3}$	Wall time for **training** $\mathcal{G}_3$ (1 GPU)	30.10 h	47.20 h
$T_{\text{off}}^{\mathcal{G}_1,\mathcal{G}_2}$	**Offline** basis prediction by $\mathcal{G}_1 + \mathcal{G}_2$ (per sample)	5.32 s	22.31 s
$T_{\text{on}}^{\mathcal{G}_3}$	**Online** basis prediction by $\mathcal{G}_3$ (per enrichment step)	1.28 s	13.21 s
Transolver++			
T_{fine}	**Fine-grid PDE solve** per timestep (per sample)	0.50 s	23.00 s
$T_{\text{train}}^{\text{td}}$	**Model training** wall time (1 GPU)	36.00 h	47.00 h
T_{inf}	**Inference** per timestep (per sample)	0.006 s	0.007 s

Table 21: Per-sample basis generation wall-clock time and speedup factors.

	$L = 4$		$L = 8$		Online	
Method	2D	3D	2D	3D	2D	3D
GMsFEM	476.7 s	5 918.2 s	651.0 s	6 230.5 s	6.7 s	94.2 s
Transolver-GMsFEM	5.32 s	22.3 s	6.12 s	26.8 s	1.3 s	13.2 s
speedup over GMsFEM	×**90**	×**265**	×**106**	×**232**	×**5**	×**7**

Table 22: One-time training cost (data generation: 16 CPU cores; model training: 1 GPU).

			Steady-state		Time-dependent	
Method	Phase	Resources	2D	3D	2D	3D
Transolver-GMsFEM	Data generation	16 CPU cores	1.6 h	5.6 h	6.1 h	17.0 h
	Model training	1 GPU	41.7 h	47.4 h	71.8 h	94.6 h
	Total		**43.3 h**	**53.0 h**	**77.9 h**	**111.6 h**
Transolver++	Data generation	16 CPU cores	<0.1 h	0.06 h	0.1 h	0.64 h
	Model training	1 GPU	12.0 h	16.7 h	36.0 h	47.0 h
	Total		**12.0 h**	**16.8 h**	**36.1 h**	**47.6 h**

Table 23: Break-even sample count N_{test}^*. For $N_{\text{test}} > N_{\text{test}}^*$ test samples, the surrogate is cheaper in total than running the classical reference solver independently for each sample.

	Steady-state		Time-dependent	
Comparison	2D	3D	2D	3D
Transolver-GMsFEM vs. GMsFEM	242	31	569	67

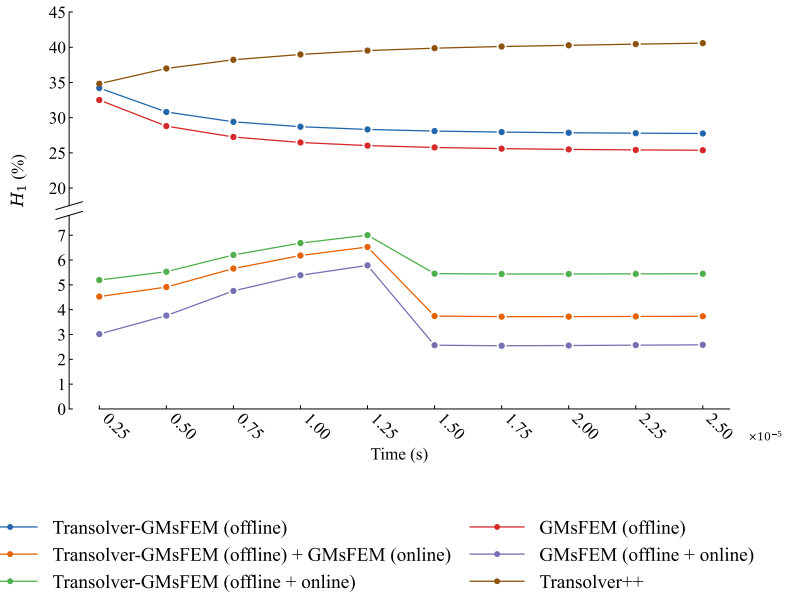


Figure 34: The H^1 error comparison for time-dependent Richards 3D with spatial forcing term (23) between Transolver-GMsFEM (offline), Transolver-GMsFEM (offline + online), Transolver-GMsFEM (offline) + GMsFEM (online), GMsFEM (offline), GMsFEM (offline + online), Transolver++. The H^1 error obtained using 4 offline and 1 online basis functions.

Table 24: Strong scaling of the parallelized GMsFEM offline basis generation (2D, $L = 8$, $N_c = 121$). The speedup is computed relative to the serial (single-core) run.

Cores	Offline time per sample, s	Speedup
1	650.98	1.0×
4	162.70	4.0×
8	79.90	8.1×
16	40.60	16.0×
32	21.80	29.9×
48	20.80	31.3×
64	24.00	27.1×
96	30.60	21.3×

The parallelization is nearly ideal up to 32 cores, reducing the offline cost from 650.98 s to about 21 s per coefficient field. The speedup peaks at approximately $31\times$ with 48 cores; beyond this point, the wall-clock time no longer decreases and even increases.

Table 25 compares the best parallelized GMsFEM configuration with serial GMsFEM and Transolver-GMsFEM. Even relative to the parallelized classical solver at its best configuration, Transolver-GMsFEM is approximately $3.4\times$ faster per coefficient field at inference.

Table 25: Offline basis generation time per sample for serial GMsFEM, parallelized GMsFEM (48 CPU cores), and Transolver-GMsFEM (1 GPU) in 2D ($L = 8$, $N_c = 121$). The speedup is computed relative to serial GMsFEM.

Method	Time per sample, s	Speedup
GMsFEM, serial	650.98	1×
GMsFEM, parallelized (48 cores)	20.80	31×
Transolver-GMsFEM (GPU)	6.12	106×

R.5 Discussion

Data generation cost is modest and GPU-free. The training-data generation wall-clock time for Transolver-GMsFEM ranges from 1.6 h (steady-state equations, 2D) to 17.0 h (time-dependent equations, 3D) when parallelised over 16 CPU cores (see Table 22). The model training phase (41–95 h) is the dominant cost, not the data generation.

Practical break-even for Transolver-GMsFEM. Transolver-GMsFEM reaches break-even after only **31 samples** (3D steady-state) or **67 samples** (3D time-dependent). Even the most demanding 2D cases require only 242 and 569 samples, respectively. Following this logic, as we move to even larger grids or more complex physics:

- Classical Cost (T_{off}): Increases significantly due to the increased degrees of freedom (DoF) in the local fine-scale problems.
- Neural Inference ($T_{\text{off}}^{\mathcal{G}_1,\mathcal{G}_2}, T_{\text{on}}^{\mathcal{G}_3}$): Increases much more slowly, as it is dominated by the forward pass complexity of the Transolver architecture rather than the iterative/matrix-inversion costs of a PDE solver.

Therefore, we can state that the more 'difficult' the problem (higher dimensionality or finer meshes), the more practical Transolver-GMsFEM becomes.

Comparison with parallelized GMsFEM. The local spectral problems are independent across coarse neighborhoods, so GMsFEM can be parallelized. Even the best parallel configuration (about $31\times$ speedup on 48 cores) is still about $3.4\times$ slower per coefficient field at inference than Transolver-GMsFEM. Parallel GMsFEM scales nearly ideally up to 32 cores but saturates beyond that, reducing the offline cost from 650.98 s to about 21 s per field. Transolver-GMsFEM remains faster than a well-tuned parallel classical solver.

Out-of-distribution fragility for Transolver++. Transolver++ must be fully retrained whenever the right-hand side profile, boundary conditions, or the coefficient distribution changes. Transolver-GMsFEM does not carry this restriction: its outputs are multiscale basis functions whose quality is grounded in the local physics of the coarse problem, and the coarse solver remains numerically exact. Moderate distribution shifts therefore degrade accuracy gracefully rather than catastrophically, with no retraining required. In any realistic simulation workflow—where problem parameters change across design iterations or physical scenarios – the **effective** time for data generation and training for Transolver++ or any baseline model is multiplied by the number of distinct distributions encountered, substantially eroding its practical advantage over both GMsFEM and Transolver-GMsFEM.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper explicitly states its contributions in the Introduction, including the development of a hybrid multiscale framework, robust OOD generalization, a selective sampling strategy, and dual-stage acceleration. These claims are supported by experimental results detailed in Section 7 and Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed limitations in the final paragraph of the Conclusion (Section 8).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper contains one theoretical result, the uniform error bound of Theorem 6.1 in Section 6.2, which states its assumptions explicitly and refers to them. The full set of assumptions (Assumptions E.1–E.3), all auxiliary lemmas and propositions, and the complete proof are given in Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experimental pipeline is thoroughly described in Sections 4 through 6, and all specific training parameters, hyperparameters, and dataset configurations are provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code and data are not publicly released, but are available from the corresponding author upon reasonable request. All algorithmic details, model configurations, hyperparameters, and experimental protocols are described in sufficient detail in the manuscript to reproduce the reported results from the text.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.

- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed information regarding data splits, optimizers, hyperparameters, and the choice of architecture (Transolver++) is included in Section 7 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports only mean L_2 and H^1 errors across 200 test samples for 2D problems and 50 test samples for 3D problems.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The detailed Computational Cost Analysis is in Appendix. The paper specifies that all numerical experiments were performed using a single Nvidia Tesla A100 GPU with 80 GB memory and 16 CPU cores.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The authors have reviewed the code of ethics and confirmed that the research adheres to ethical standards.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper briefly discusses the positive impact of improving PDE solvers for industrial and engineering applications while noting no immediate direct negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The research focus is on numerical solvers for PDEs in scientific computing, which does not involve high-risk assets like large-scale language models, image generators, or sensitive scraped datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper explicitly cites the original works for the methods and models used, such as GMsFEM , GNOT, GAOT, Transolver , and Transolver++.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The primary contribution is a new hybrid framework (Transolver-GMsFEM) and methodology. While the code and data used for validation will be released upon publication, the paper does not introduce a standalone general-purpose asset (such as a new benchmark dataset or software library) that requires separate documentation beyond the descriptions provided in the main text and Appendix.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The research does not involve human subjects or crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: IRB approval is not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were only utilized for basic grammar checking and did not impact the core methodology or scientific rigor of the work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.