ALIOTH: NEURAL MODULATORS FOR SCIENTIFIC COMPUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scientific computing underpins modern science and engineering, and solving partial differential equations (PDEs) lies at its core. For decades, the field has pursued PDE solvers that are both accurate and generalizable. However, existing approaches rarely achieve both. Classical numerical algorithms guarantee accuracy but solve each problem anew, whereas neural operators generalize but cannot guarantee accuracy. More recently, neural preconditioners embed learning into numerical solvers but typically train for each problem or problem family. In this work, we introduce the **neural modulator**, a single pretrained network that generates solver components across problems and solvers. We present Alioth, a neural modulator that generates coarse spaces, the solver components that capture slow-to-converge error modes. Coarse spaces are **transferable**, as these modes follow local operator structure shared across PDEs, and **error-tolerant**, as imperfect ones only slow convergence. Pretrained once on synthetic diffusion problems with a label-free spectral objective, Alioth serves domain decomposition and algebraic multigrid, the two workhorse solvers for large-scale PDEs. With domain decomposition on 13 scalar SPD systems spanning reservoir simulation, heat transfer, fluid dynamics, structural mechanics, and electromagnetics, it needs 48% fewer iterations than the classical Nicolaides coarse space and 20% fewer than unlearned subspace iteration with the same number of matrix products. Because accuracy is set by the solver, five iterations on Darcy flow reach errors over 100× below neural operators and PDE foundation models, in and out of their training distribution. These results point to a new direction for AI-enabled scientific computing: *learn the solver, not the solution*.

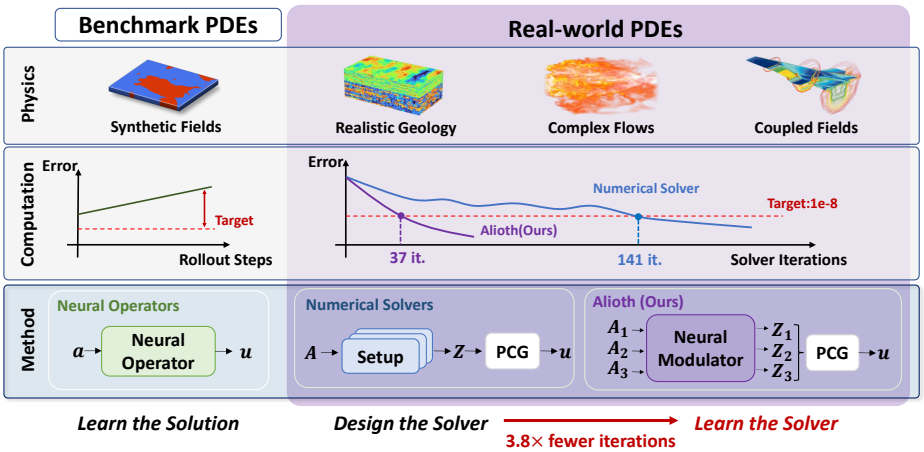


Figure 1: **Motivation of Alioth.** Left: neural operators learn solutions in one pass, but errors accumulate over autoregressive rollouts. Middle: numerical solvers reach high accuracy but set up their components anew for every system. Right: Alioth learns the solver once, generating coarse spaces for different systems with a shared neural modulator, and PCG reaches the same tolerance in 3.8× fewer iterations. Curves are schematic; iterations on tmt_sym are measured against Nicolaides.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Partial differential equations (PDEs) govern much of the physical world (e.g., fluid flow, heat transfer, elasticity, and electromagnetism), and solving them is central to scientific computing. In practice, a PDE is discretized into sparse linear systems, whose solution dominates the computational cost (Saad, 2003). For industrial simulations, a single run may solve thousands of these systems. Such workloads require solvers that are accurate on every system and generalizable across them, rather than re-tuned for each problem.

Existing approaches largely follow two paradigms. On the one hand, classical numerical solvers are **accurate but problem-specific**. Multigrid and domain decomposition guarantee accuracy through residual-controlled iteration (Xu, 1992; Toselli & Widlund, 2005). Their efficiency, however, relies on solver components that must be rebuilt from scratch for every new system (Brezina et al., 2005; Spillane et al., 2014). Recent LLM agents automate this setup (Liu et al., 2025; Dong et al., 2026), but they can only select and tune existing numerical methods, through costly trial and error on every problem. On the other hand, neural operators are **generalizable but accuracy-limited**. They learn solution maps across problem instances (Lu et al., 2021; Li et al., 2021), and PDE foundation models extend this generalization through large-scale pretraining (Hao et al., 2024; Herde et al., 2024; Chen et al., 2025a;b). Yet their relative errors typically plateau around 10^{-2} to 10^{-3} (de Hoop et al., 2022; McGreivy & Hakim, 2024), orders of magnitude above what numerical solvers routinely reach. More recently, neural preconditioners bridge the two paradigms by embedding learning into numerical solvers (Luz et al., 2020; Chen, 2025; Zhang et al., 2026). They retain accuracy but remain problem-specific, as each network is typically trained for a single problem. What remains missing is a component that is learned once and transfers across problems.

We find that such a component can be learned by *predicting the coarse space of iterative numerical solvers*, for two reasons. First, the coarse space is **transferable**. It captures the few slow-to-converge error modes that limit convergence in both multigrid and domain decomposition (Xu, 1992; Xu & Zikatanov, 2017), and these modes are determined by local operator structure that is largely shared across PDE problems (Figure 1, right). Second, the coarse space is **error-tolerant**. An inaccurate coarse space slows convergence but leaves the attainable accuracy intact, whereas errors in a predicted solution directly bound its accuracy (Figure 1, middle and right). These properties suggest a simple principle: *learn the solver, not the solution*.

Following this principle, we introduce a **neural modulator** named *Alioth*, a pretrained network that generates solver components rather than solutions. For each new problem, Alioth produces the required coarse space without retraining, much as hypernetworks generate LoRA weights for new tasks (Charakorn et al., 2025). Alioth consists of a learned encoder and deterministic assembly heads (Figure 2). To transfer across PDE problems, the encoder applies learned matrix polynomials directly to local blocks of the linear system, so that one set of weights serves systems of any size. To preserve accuracy, the heads assemble the resulting slow modes into coarse spaces for domain decomposition or algebraic multigrid, leaving convergence control to the numerical iteration. To avoid labels, the encoder is pretrained by minimizing Rayleigh quotients, which drives its outputs toward the slowest modes without precomputed eigenvectors.

We evaluate Alioth on two widely used real-world benchmarks: the SPE10 reservoir model, an industry standard for high-contrast subsurface flow (Christie & Blunt, 2001), and PDE matrices from the SuiteSparse Matrix Collection, drawn from thermal, fluid, structural, and electromagnetic applications (Davis & Hu, 2011). Pretrained only on synthetic diffusion problems, Alioth reduces domain-decomposition iterations by 48% relative to the classical Nicolaides coarse space and by 20% relative to unlearned subspace iteration with the same number of matrix products, across 13 systems with up to 1.23 million degrees of freedom. With algebraic multigrid, it performs on par with smoothed aggregation. Because accuracy comes from solver iterations, five iterations on Darcy flow reach errors over 100$\times$ below seven neural operators and PDE foundation models, in and out of their training distribution.

Our contributions are threefold:

- **Neural modulators for PDE solving.** We define neural modulators, networks that are fixed after pretraining and generate solver components rather than solutions, and identify the coarse space as a transferable and error-tolerant target.

- **Label-free spectral pretraining.** Alioth learns the slowest modes of PDE operators without eigenvector labels, using an objective whose excess bounds the subspace error.
- **Zero-shot on real-world PDEs.** Pretrained only on synthetic diffusion, a single 36.6K-parameter network transfers to 13 systems across five application domains and reduces domain-decomposition iterations by 48% relative to the classical Nicolaides coarse space.

2 Related Work

Numerical solvers and coarse spaces. Multigrid and domain decomposition achieve scalable convergence by pairing local smoothing with a coarse space that removes slow-to-converge error modes (Xu, 1992; Toselli & Widlund, 2005; Xu & Zikatanov, 2017). Constructing this coarse space for a new operator is the central difficulty. Adaptive smoothed aggregation and bootstrap AMG discover slow modes through test iterations (Brezina et al., 2005; Brandt et al., 2011), GenEO extracts them from local generalized eigenproblems (Spillane et al., 2014), and Krylov recycling reuses them across a sequence of related systems (Parks et al., 2006). LLM agents now automate solver selection and configuration (Liu et al., 2025; Dong et al., 2026). All of these methods rebuild or reselect components for each problem, whereas Alioth learns to construct coarse spaces and amortizes this discovery across problems.

Neural operators and PDE foundation models. Neural operators such as DeepONet (Lu et al., 2021), FNO (Li et al., 2021), and physics-informed variants (Li et al., 2024) learn mappings from PDE inputs to solutions, and PDE foundation models scale them through pretraining on heterogeneous data (Hao et al., 2024; Herde et al., 2024; McCabe et al., 2024; Chen et al., 2025a;b). Their output is a solution whose error is fixed by the network, and reported relative errors typically remain above 10^{-3} (de Hoop et al., 2022; McGreivy & Hakim, 2024). Alioth shares their pretrain-once philosophy but learns solver components, so its accuracy is controlled by numerical iteration rather than by the network.

Learning inside numerical solvers. A growing line of work embeds learning into iterative solvers. Learned multigrid methods predict prolongation operators or smoothers (Greenfeld et al., 2019; Luz et al., 2020), learned domain decomposition methods predict where or how to enrich coarse spaces (Heinlein et al., 2021; Klawonn et al., 2026), and neural preconditioners learn the action of a preconditioner inside Krylov iterations (Häusner et al., 2024; Chen, 2025; Zhang et al., 2026). HINTS alternates neural corrections with relaxation (Zhang et al., 2024). These methods retain the accuracy of numerical iteration, but each network is typically trained for one problem family or one matrix and serves one solver. Alioth is instead pretrained once without labels, generalizes zero-shot across PDE problems, and serves both multigrid and domain decomposition.

3 Preliminaries

This section formalizes the two observations in Section 1. After stating the problem, we introduce coarse spaces and show that they are error-tolerant (Proposition 1) and admit a local spectral target (Proposition 2).

Problem setup. Our goal is to solve PDEs to a prescribed accuracy. Discretizing a PDE with n degrees of freedom yields a sparse linear system $Au = b$ with $A \in \mathbb{R}^{n\times n}$ symmetric positive definite (SPD), as arises from diffusion, elasticity, and many electromagnetic problems (Appendix A.1). An iterative solver produces iterates u_k and stops once $\|b - Au_k\|_2 \le \varepsilon\|b\|_2$. We seek a single learned map that, fixed after pretraining, equips such solvers with effective coarse spaces across different PDEs.

Coarse spaces. Let N overlapping subdomains cover the degrees of freedom, with restrictions R_i and local matrices $A_i = R_i A R_i^\top$. Given a coarse basis $Z \in \mathbb{R}^{n\times m}$, the two-level Schwarz preconditioner is

$$M^{-1} = Z(Z^\top AZ)^{-1}Z^\top + \sum_{i=1}^{N} R_i^\top A_i^{-1} R_i. \tag{1}$$

Local solves exchange information only between neighboring subdomains, so error spanning many subdomains decays slowly. The coarse term acts as a global shortcut: it removes the best approximation of the error within $\mathrm{range}(Z)$ in the energy norm $\|v\|_A = (v^\top A v)^{1/2}$ (Toselli & Widlund, 2005). We use M^{-1} in the preconditioned conjugate gradient method (PCG; Saad, 2003), whose convergence is governed by the condition number $\kappa = \lambda_{\max}(M^{-1}A)/\lambda_{\min}(M^{-1}A)$.

Proposition 1 (Error tolerance). *For any full-rank Z, M^{-1} is SPD, and the PCG error $e_k = u - u_k$ satisfies $\|e_k\|_A \le 2\big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\big)^k \|e_0\|_A$. Hence Z affects the number of iterations through κ, but not the attainable accuracy.*

Choosing the coarse space. Proposition 1 leaves open which Z to use. GenEO answers this question through local generalized eigenproblems (Spillane et al., 2014). Each subdomain i defines a symmetric pencil (H_i, B_i) from its local matrices, whose low eigenvectors are the error modes that local solves remove slowly.

Proposition 2 (Local spectral target; adapted from Spillane et al. (2014)). *Let $\lambda_{i,1} \le \lambda_{i,2} \le \cdots$ be the generalized eigenvalues of (H_i, B_i). If Z spans the partition-of-unity extensions of the first m_i eigenvectors of every pencil, then*

$$\kappa \le C_{k_0}\Big(1 + \max_i \lambda_{i,m_i+1}^{-1}\Big), \tag{2}$$

where C_{k_0} depends only on the maximal number k_0 of subdomains sharing a degree of freedom.

Algebraic multigrid imposes an analogous requirement: its interpolation must reproduce low-energy modes (Vaněk et al., 1996; Brezina et al., 2005). Alioth therefore learns a map from local pencils to their low eigenspaces, as described next.

4 Method

Following Section 3, Alioth learns a map from local pencils to their low eigenspaces and leaves accuracy to the numerical solver (Figure 2). Given a system $Au = b$, Alioth partitions the matrix graph into N overlapping subdomains and forms a local pencil (H_i, B_i) for each (Appendix B.1). A learned encoder maps each pencil to k ordered local modes $V_i \in \mathbb{R}^{n_i \times k}$, and a deterministic assembly head combines $\{V_i\}$ into a coarse space for domain decomposition or algebraic multigrid. We describe the encoder, the assembly heads, and the pretraining objective.

4.1 Encoder: Learned Subspace Iteration

The encoder generalizes subspace iteration, the classical method for approximating low eigenvectors (Saad, 2011). Subspace iteration repeatedly applies a smoothing step $G_i = I - \omega J_i^{-1} H_i$ with diagonal scaling J_i to random vectors, attenuating high-energy components until the low modes dominate. Alioth makes this process learnable at the same cost. Starting from s random vectors $X_i^{(0)} \in \mathbb{R}^{n_i \times s}$, each of T stages applies a learned matrix polynomial followed by pointwise channel mixing:

$$Y_i^{(t)} = \sum_{p=0}^{P} c_{t,p}\, G_i^p\, X_i^{(t-1)}, \qquad X_i^{(t)} = Y_i^{(t)} + \phi_t\big(\big[\, Y_i^{(t)},\ H_i Y_i^{(t)},\ B_i Y_i^{(t)},\ d_i \,\big]\big), \tag{3}$$

where the coefficients $c_{t,p}$ and the damping ω are learned, ϕ_t is an MLP applied independently at each degree of freedom, and d_i contains partition weights. The polynomial acts as a learned spectral filter that amplifies low modes, while the MLP learns how to combine channels. Finally, spectral normalization B_i-orthonormalizes $X_i^{(T)}$ and applies a Rayleigh–Ritz rotation, returning k modes ordered by ascending Rayleigh quotient.

This design supports transfer in two ways. First, the encoder accesses the system only through products with local matrices, so the same weights apply to subdomains of any size and origin. Second, both the polynomial and the pointwise MLP commute with reorderings of the degrees of freedom, so the encoder is independent of how a mesh is numbered. We use $T = 3$ stages of degree $P = 32$, with $s = 32$ input vectors and $k = 16$ output modes. The resulting propagation budget of 96 matrix products equals that of the unlearned baseline SubIter in Section 5, so any gain comes from learning rather than extra matrix products.

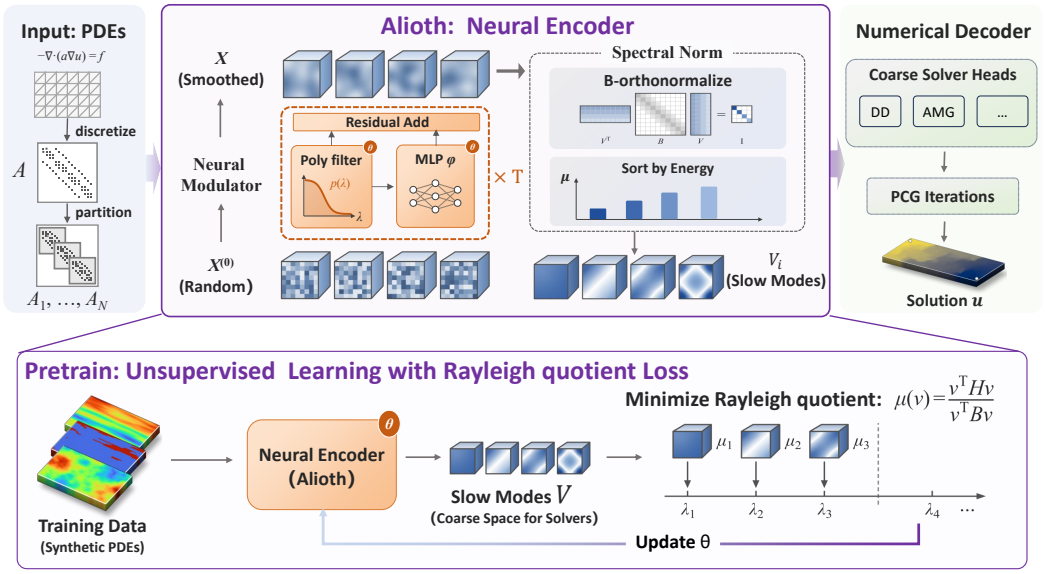


Figure 2: **Overview of Alioth.** A PDE is discretized into a sparse system and partitioned into local blocks A_i. The neural encoder turns random vectors into slow modes V_i through T stages of a learnable polynomial filter and a pointwise MLP, then a parameter-free spectral normalization orthonormalizes and sorts them by energy μ. The numerical decoder uses V_i as the coarse space of DD, AMG, or other solvers, and PCG iterates to the solution u. Pretraining on synthetic PDEs minimizes the Rayleigh quotients $\mu(v) = v^\top Hv / v^\top Bv$ of the outputs, pushing each μ_j toward λ_j.

4.2 Assembly: Solver-Agnostic Coarse Spaces

The assembly heads turn the same local modes into coarse spaces for different solvers. They contain no learnable parameters, so accuracy is controlled by the numerical iteration rather than by the network. We present two example heads below. Other solvers that rely on coarse spaces can be supported by adding new heads, without retraining the encoder.

Example head 1: Domain decomposition. The Schwarz head follows Proposition 2. It combines the partition-of-unity constant used by the classical Nicolaides coarse space with the first m learned modes of each subdomain, and extends them to the global space:

$$Z = \left[R_1^\top D_1[\mathbf{1}, V_1^{(m)}],\ \ldots,\ R_N^\top D_N[\mathbf{1}, V_N^{(m)}] \right], \tag{4}$$

where $V_i^{(m)}$ denotes the first m learned modes, orthogonalized against the constant. The constant captures the near-kernel of diffusion operators, and the learned modes capture what it misses. This Z enters the two-level preconditioner in Equation (1).

Example head 2: Algebraic multigrid. AMG requires global candidate vectors that its interpolation must reproduce, rather than local bases. The AMG head aligns the local modes on subdomain overlaps and blends them with the partition of unity into global candidates, which are appended to the constant vector. As in adaptive AMG (Brezina et al., 2005), a few Gauss–Seidel sweeps on $Ax = 0$ smooth each learned candidate before an energy-minimizing interpolation reproduces all candidates on every level (Olson et al., 2011). The resulting V-cycle preconditions PCG (Appendix B.5).

Accuracy. In both heads, the learned modes determine only the coarse space. By Proposition 1, any full-rank Z yields an SPD two-level preconditioner. Likewise, a symmetric V-cycle with Galerkin coarse operators and a convergent smoother remains SPD for any full-rank interpolation

(Trottenberg et al., 2001). PCG therefore reaches the prescribed tolerance regardless of the quality of the learned modes, which affects only the number of iterations.

4.3 Pretraining: Label-Free Spectral Objective

Exact eigenvectors make poor labels: each requires a dense eigendecomposition and is defined only up to sign and rotation. Alioth is instead trained on the Rayleigh quotients $\mu_{i,j} = v_{i,j}^\top H_i v_{i,j}$ of its B_i-orthonormal outputs, sorted in ascending order:

$$\mathcal{L} = \mathbb{E}\Big[\sum_{j=1}^{k} w_j \, \mu_{i,j} + \beta \, \mathcal{R}_i \Big], \qquad w_j = \frac{k+1-j}{k}, \tag{5}$$

where $\mathcal{R}_i$ penalizes redundant channels (Appendix C.2). The decreasing weights prioritize the lowest modes, which the assembly heads use first.

Proposition 3 (Subspace error bound). *Let (H, B) have eigenvalues $\lambda_1 \le \lambda_2 \le \cdots$, let U_k span the first k eigenvectors, and let $F^\star = \sum_j w_j \lambda_j$. For any B-orthonormal V, $F(V) = \sum_j w_j \mu_j(V) \ge F^\star$. If $\delta = \lambda_{k+1} - \lambda_k > 0$, then $\| \sin \Theta_B(V, U_k) \|_F^2 \le (F(V) - F^\star)/(w_k \delta)$, where Θ_B denotes the principal angles in the B-inner product.*

Minimizing $\mathcal{L}$ therefore drives the outputs toward the target of Proposition 2 without computing any eigenvector (proof in Appendix A.4). Pretraining uses only synthetic diffusion problems, and no evaluation system is seen during training (Appendix C.1).

5 Experiments

We design experiments to answer four questions. **Q1 (Transfer):** Can a single pretrained Alioth serve different PDE problems without retraining? **Q2 (Accuracy):** How does it compare with methods that learn solutions or train for each problem? **Q3 (Learning):** Do the gains come from learning rather than extra computation, and how close are they to exact local eigenspaces? **Q4 (Cost):** What does Alioth cost, how does it scale, and where is the bottleneck?

5.1 Setup

Benchmarks. We evaluate on 13 SPD systems with 13K to 1.23M degrees of freedom. Five come from the SPE10 reservoir model (Christie & Blunt, 2001), including four high-contrast 2D layers and the full 3D model. Six come from the SuiteSparse Matrix Collection (Davis & Hu, 2011), covering thermal, fluid, structural, and electromagnetic applications. The remaining two are 2D and 3D Poisson problems. For comparison with neural operators, we use 2D Darcy flow on a 256^2 grid with in-distribution coefficients and two out-of-distribution families, lognormal and channelized. A circuit-simulation matrix outside the PDE setting is reported in Appendix E.2 (the `G3_circuit` row of Table 9).

Metrics. We report three metrics. (1) *Iterations*, the number of PCG steps to reach a relative residual of 10^{-8}, compare coarse spaces within the same solver. (2) *Relative error*, $\|u - u_{\text{ref}}\|_2 / \|u_{\text{ref}}\|_2$, compares Alioth with neural operators, which produce a solution in a single forward pass. (3) *Setup time* compares the encoding cost of Alioth with the per-problem training cost of GNP. Alioth results are medians over three pretraining seeds.

Baselines and ablation. We compare Alioth with two groups of baselines and one ablation. (1) The numerical baselines are one-level Schwarz, the classical Nicolaides coarse space, and smoothed aggregation with the constant candidate, while exact local eigenspaces (Exact) serve as a spectral reference (Spillane et al., 2014). (2) The learned baselines are four neural operators (DeepONet, FNO, U-Net, Transolver) and three fine-tuned PDE foundation models (Poseidon-T, DPOT-Ti, DPOT-S), all trained on 1,000 in-distribution Darcy fields, as well as GNP, which is trained separately for each problem. (3) The ablation, SubIter and SubIter-2×, replaces the learned encoder with unlearned subspace iteration of the same and twice the propagation budget. Production AMG solvers such as hypre BoomerAMG (Henson & Yang, 2002) couple the coarse space with many engineering components, so we compare them in terms of end-to-end cost in Section 5.5.

Table 1: PCG iterations to a relative residual of 10^{-8} (lower is better). Alioth reports the median over three seeds. SubIter and SubIter-2× are unlearned ablations at the same and twice the propagation budget. Exact uses exact local eigenspaces as a spectral reference and is excluded from bold. TM = thermomech_dM, TH = thermal2, PF = parabolic_fem, AP = apache2, EM = tmt_sym, EC = ecology2, P2/P3 = Poisson 2D/3D.

	Subsurface flow (SPE10)					Heat		Fluid	Struct.	EM	Diffusion		
Method	L5	L26	L47	L68	3D	TM	TH	PF	AP	EM	EC	P2	P3
DoF	13K	13K	13K	13K	1.12M	204K	1.23M	526K	715K	727K	1.00M	1.05M	885K
Domain decomposition													
One-level	43	49	52	58	552	**13**	1563	389	536	591	861	718	715
Nicolaides	48	53	58	65	350	**13**	912	135	188	141	162	196	391
SubIter (ablation)	37	40	58	57	334	**13**	799	55	165	45	50	71	303
SubIter-2× (ablation)	**26**	**25**	49	**42**	312	**13**	675	44	147	38	**43**	**48**	213
Exact (reference)	24	23	24	28	184	13	152	42	81	37	43	48	115
Alioth (ours)	29	29	**47**	46	**307**	15	**630**	**43**	**135**	**37**	44	**48**	**159**
Reduction vs. Nicolaides	40%	45%	19%	29%	12%	−15%	31%	68%	28%	74%	73%	76%	59%
Algebraic multigrid													
Smoothed aggregation	10	10	**22**	**20**	**21**	**5**	**12**	61	**11**	**10**	**11**	**11**	**9**
SubIter (ablation)	10	10	24	**20**	26	**5**	13	**12**	87	11	**11**	13	10
Alioth (ours)	**9**	**9**	23	**20**	23	**5**	**12**	**12**	91	11	12	13	11

Protocol. All results use the final checkpoint of one pretrained encoder per seed; no evaluation system is used to train or select the encoder. Domain decomposition serves as a controlled testbed for coarse spaces, not as a competitive solver. Both heads include the constant vector, so the classical baselines, Nicolaides and smoothed aggregation, differ from Alioth only in the learned modes. All AMG solvers share the same aggregation, smoothing, and interpolation settings (Appendix D).

5.2 Q1: Zero-Shot Transfer Across PDE Problems

Table 1 reports PCG iterations on the 13 systems. With domain decomposition, Alioth needs fewer iterations than the Nicolaides coarse space on 12 of 13 systems, reducing the count by 48% in geometric mean and by 71% relative to one-level Schwarz. The largest reductions occur beyond the diffusion problems used in pretraining, for example from 141 to 37 iterations on the electromagnetic system tmt_sym and from 135 to 43 on the convection–diffusion system parabolic_fem. The only loss is thermomech_dM, where all methods need 13 to 15 iterations. With algebraic multigrid, the same encoder performs on par with smoothed aggregation, whose constant candidate already captures the near-kernel of most diffusion systems (geometric-mean ratio 1.08, at 2.5× its operator complexity). Local spectral candidates help most when the constant is insufficient, reducing iterations from 61 to 12 on parabolic_fem, but they degrade convergence on apache2. Neural operators do not apply here, since each system is a single matrix without a training family. These results answer Q1: a single pretrained encoder transfers across PDE problems without retraining, with the largest gains for domain decomposition.

5.3 Q2: Accuracy Compared with Learned Methods

Table 2 compares Alioth with learned solution maps on Darcy flow. In distribution, their median errors range from 4.6×10^{-3} for DPOT-S to 3.0×10^{-2} for DeepONet. Out of distribution, every method fails, including the fine-tuned foundation models, with median errors between 0.75 and 2.1. Although Alioth is not trained on Darcy data, five PCG iterations reduce its error to 1.3×10^{-5} in distribution and 3.6×10^{-3} on channelized fields, 359× and 209× below the best learned method on each set. Further iterations drive the error below 10^{-10}. This contrast illustrates Proposition 1: an imperfect solver component costs iterations, whereas an imperfect solution caps accuracy. Alioth and smoothed aggregation, which share the same AMG settings, reach comparable accuracy on all three sets. The advantage over learned solution maps therefore comes from iterating a numerical solver rather than from a particular preconditioner.

We next compare with GNP, which trains a neural preconditioner for each problem. On the first 10 fields of each Darcy set, GNP spends about 142 s of training per field, yet converges on none of the 30 fields within 100 FGMRES(10) iterations. Under the same solver, Alioth sets up in about 1 s

Table 2: **Relative error on 2D Darcy flow (256^2 grid), median over 100 test fields per set.** OOD-L and OOD-C use lognormal and channelized out-of-distribution coefficients. Learned solution maps are trained or fine-tuned on 1,000 in-distribution fields, whereas Alioth is not trained on Darcy data. †Trained for each field and run with FGMRES(10) on the first 10 fields of each set.

Method	Trained on Darcy	ID	OOD-L	OOD-C
Neural operators				
DeepONet (Lu et al., 2021)	✓	3.0×10^{-2}	9.8×10^{-1}	2.1
FNO (Li et al., 2021)	✓	7.0×10^{-3}	9.7×10^{-1}	9.8×10^{-1}
U-Net (Ronneberger et al., 2015)	✓	5.3×10^{-3}	9.9×10^{-1}	1.2
Transolver (Wu et al., 2024)	✓	1.0×10^{-2}	9.9×10^{-1}	9.4×10^{-1}
PDE foundation models (fine-tuned)				
Poseidon-T (Herde et al., 2024)	✓	1.3×10^{-2}	9.9×10^{-1}	1.1
DPOT-Ti (Hao et al., 2024)	✓	5.4×10^{-3}	9.9×10^{-1}	9.1×10^{-1}
DPOT-S (Hao et al., 2024)	✓	4.6×10^{-3}	1.0	7.5×10^{-1}
Iterative solvers after 5 iterations				
Smoothed aggregation	–	1.6×10^{-5}	3.9×10^{-5}	$\mathbf{3.4\times10^{-3}}$
GNP† (Chen, 2025)	per field	9.9×10^{-1}	1.0	1.0
Alioth (ours)	×	$\mathbf{1.3\times10^{-5}}$	$\mathbf{2.0\times10^{-5}}$	3.6×10^{-3}
Δ vs. best learned method		359×	4.9×10^{4}×	209×
Iterations to reach a relative residual of 10^{-8}				
Smoothed aggregation	–	**12**	**12**	21
GNP† (Chen, 2025)	per field	>100	>100	>100
Alioth (ours)	×	**12**	**12**	**20**

per field without any training and converges on all 30 fields. On six matrices from Table 1, GNP reaches the 100-iteration cap on five and needs 29 iterations on thermomech_dM, versus 5 for Alioth (Appendix E.1). This comparison reflects the reported training budget rather than an intrinsic limit of neural preconditioners. These results answer Q2: Alioth attains the accuracy of numerical solvers, which learned solution maps cannot, without the per-problem training that neural preconditioners require.

5.4 Q3: Learning Rather Than Computation

Two comparisons separate the effect of learning from that of computation. First, learning is worth a doubling of the matrix products, although the pointwise MLP adds dense work (Appendix F). At the same propagation budget, Alioth needs 20% fewer iterations than SubIter in geometric mean and wins on 12 of 13 systems. SubIter needs 194 propagation steps to match Alioth at 96 (Figure 4d), so learning halves the propagation needed for a coarse space of the same quality.

Second, exact local eigenspaces indicate how close Alioth comes to exact local eigenspaces. Measured as the fraction of the gap between SubIter and Exact that Alioth closes, the median over 12 systems is 63%. Alioth closes the gap entirely on tmt_sym and Poisson 2D and closes 92% on parabolic_fem, but only 18–38% on five harder systems: SPE10 3D, L47, L68, thermal2, and apache2.

Figure 3 illustrates this behavior on SPE10 layer 68. On a representative subdomain, the learned mode reproduces the exact local eigenvector, whereas the unlearned mode carries spurious lobes. After 30 iterations, Alioth reduces the error to 2.8×10^{-6}, compared with 8.0×10^{-4} for Nicolaides and 1.4×10^{-4} for SubIter. These results answer Q3: the gains come from learning rather than extra computation, and Alioth approaches exact local eigenspaces on most systems while leaving room on high-contrast problems.

5.5 Q4: Cost and Scalability

Figure 4 summarizes the cost of Alioth. Encoding is cheap. On Poisson 2D with one million degrees of freedom, it takes 3.3 s on a GPU, below 1% of the 389 s required by exact local eigensolves and comparable to the 5.3 s setup of smoothed aggregation (Figure 4a). Pretraining is paid once. At about 1 s per new system, Alioth recovers its 1,072 s of pretraining after 8 systems relative to GNP,

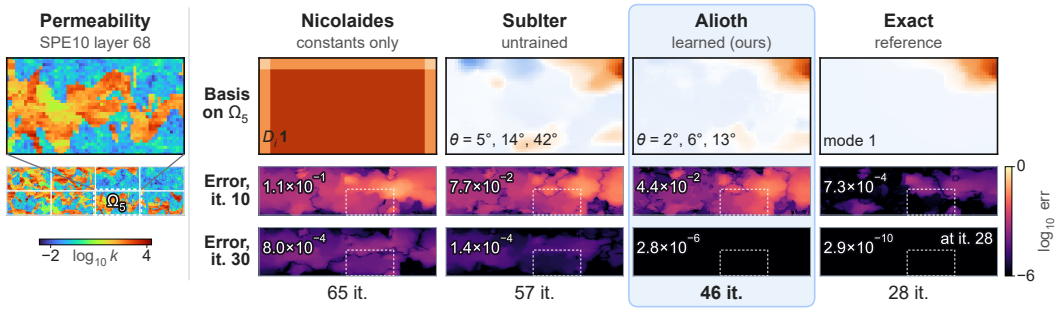


Figure 3: **Case study on SPE10 layer 68.** All columns except Nicolaides add eight modes per subdomain to the constant, and Alioth uses its median seed. Top: the first mode on subdomain Ω_5 with the constant removed, and its principal angles to the first three exact eigenvectors. Middle and bottom: PCG error $\log_{10}(|e_k|/\|e_0\|_\infty)$ after 10 and 30 iterations, labeled with the relative L_2 error. Iteration counts appear below each column.

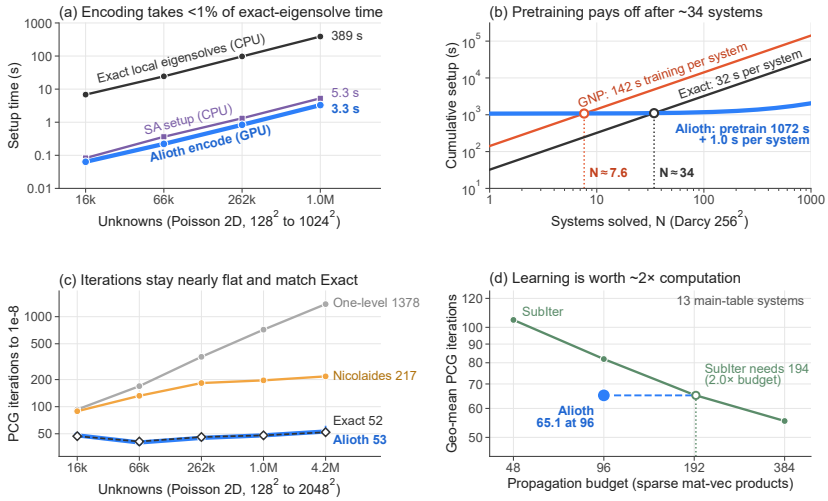


Figure 4: **Cost and scalability.** (a) Setup time on Poisson 2D, with exact eigensolves and smoothed aggregation on one CPU thread and Alioth on a GPU. (b) Cumulative setup time over N Darcy systems (256^2 grid), including pretraining for Alioth and per-system training for GNP. (c) PCG iterations on Poisson 2D from 128^2 to 2048^2 grids. (d) Geometric-mean iterations over the 13 systems of Table 1 for SubIter at increasing propagation budgets, with Alioth at a budget of 96.

which trains for 142 s per system, and after 34 systems relative to exact eigensolves (Figure 4b). Alioth also scales with problem size. From 16K to 4.2M degrees of freedom, its iteration count grows only from 48 to 53 and tracks exact eigenspaces, whereas Nicolaides grows from 89 to 217 and one-level Schwarz from 93 to 1,378 (Figure 4c). Production AMG nevertheless remains faster end to end. On four timed systems, hypre BoomerAMG takes 1.0–4.7 s, whereas our prototype takes 17.3–154.4 s, of which graph partitioning accounts for 42–82% (Appendix E.3). These results answer Q4: the learned component is cheap to evaluate and amortizes quickly, and the remaining end-to-end cost lies outside the network.

6 Conclusion

We introduced neural modulators, which learn solver components rather than solutions, and instantiated them as Alioth, a pretrained generator of coarse spaces. Unlike neural operators, whose errors plateau around 10^{-2} to 10^{-3}, it converges to the prescribed solver tolerance. Unlike neural preconditioners, which are trained for each matrix, it is pretrained once and transfers zero-shot to 13 PDE systems across five domains and two solvers. On tmt_sym and Poisson 2D, it matches exact local eigenspaces at a fraction of their setup cost. These results position learned solver components as reusable building blocks for scientific computing. Appendix F discusses its limitations.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting, Chinese–English translation, literature and citation checking, mathematical exposition, figure layout, and scripts for running experiments and producing tables. Conceptual diagrams used AI-assisted image generation, and all data plots were rendered from numerical outputs. We checked AI-assisted text, derivations, and code against the implementation, experiment records, cited sources, and numerical residuals. The authors take full responsibility for the manuscript and its artifacts.

REPRODUCIBILITY STATEMENT

Appendix A states the mathematical assumptions and proofs. Appendix B specifies the local operators, the encoder, and both assembly heads. Appendix C gives the pretraining data, the optimization settings, and the reported checkpoint. Appendix D lists the evaluation systems, the baseline configurations, the Darcy protocol, and the software, hardware, and timing conditions. Appendix E reports the comparison with GNP, per-seed AMG results, runtimes, the Darcy two-phase subset, the SPE10 case study, and per-seed domain-decomposition results. Appendix F discusses the limitations of the method and evaluation.

REFERENCES

Hussam Al Daas and Laura Grigori. A class of efficient locally constructed preconditioners based on coarse spaces. *SIAM Journal on Matrix Analysis and Applications*, 40(1):66–91, 2019.

Achi Brandt, James Brannick, Karsten Kahl, and Ira Livshits. Bootstrap AMG. *SIAM Journal on Scientific Computing*, 33(2):612–632, 2011. doi: 10.1137/090752973.

Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*, volume 15 of *Texts in Applied Mathematics*. Springer, 3 edition, 2008. doi: 10.1007/978-0-387-75934-0.

Marian Brezina, Robert D. Falgout, Scott MacLachlan, Thomas A. Manteuffel, Stephen F. McCormick, and John W. Ruge. Adaptive smoothed aggregation (αsa) multigrid. *SIAM Review*, 47 (2):317–346, 2005. doi: 10.1137/050626272.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7485–7514, 2025. URL https://proceedings.mlr.press/v267/charakorn25a.html.

Jie Chen. Graph neural preconditioners for iterative solutions of sparse linear systems. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Tkkrm3pA35.

Tianyu Chen, Haoyi Zhou, Ying Li, Hao Wang, Chonghan Gao, Rongye Shi, Shanghang Zhang, and Jianxin Li. OmniArch: Building foundation model for scientific computing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 9860–9887, 2025a. URL https://proceedings.mlr.press/v267/chen25cp.html.

Tianyu Chen, Haoyi Zhou, Ying Li, Hao Wang, Zhenzhe Zhang, Tianchen Zhu, Shanghang Zhang, and Jianxin Li. FreqMoE: Dynamic frequency enhancement for neural PDE solvers. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 2025b. URL https://arxiv.org/abs/2505.06858.

Michael A. Christie and Martin J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, 4(4):308–317, 2001. doi: 10.2118/72469-PA.

Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1:1–1:25, 2011. doi: 10.1145/2049662.2049663.

Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart. The cost-accuracy trade-off in operator learning with neural networks. *Journal of Machine Learning*, 1(3): 299–341, 2022. doi: 10.4208/jml.220509.

Zhijie Deng, Jiaxin Shi, and Jun Zhu. NeuralEF: Deconstructing kernels by deep neural networks. In *International Conference on Machine Learning*, 2022.

Clark R. Dohrmann and Olof B. Widlund. On the design of small coarse spaces for domain decomposition algorithms. *SIAM Journal on Scientific Computing*, 39(4):A1466–A1488, 2017.

Clark R. Dohrmann, Axel Klawonn, and Olof B. Widlund. Domain decomposition for less regular subdomains: Overlapping Schwarz in two dimensions. *SIAM Journal on Numerical Analysis*, 46 (4):2153–2168, 2008.

Huanshuo Dong, Keyao Zhang, Hong Wang, Zhezheng Hao, Zhiwei Zhuang, Ziyan Liu, Jiacong Wang, Gengyuan Liu, and Xin Jin. AutoPDE: Reliable agentic PDE solving via explicitly represented solver strategies, 2026. URL https://arxiv.org/abs/2606.10752.

Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid pde solvers. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2415–2423, 2019.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-regressive denoising operator transformer for large-scale PDE pre-training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17616–17635, 2024. URL https://proceedings.mlr.press/v235/hao24d.html.

Paul Häusner, Ozan Öktem, and Jens Sjölund. Neural incomplete factorization: Learning preconditioners for the conjugate gradient method. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=FozLrZ3CI5.

Alexander Heinlein, Axel Klawonn, Martin Lanser, and Janine Weber. Combining machine learning and adaptive coarse spaces—a hybrid approach for robust FETI-DP methods in three dimensions. *SIAM Journal on Scientific Computing*, 43(5):S816–S838, 2021. doi: 10.1137/20M1344913.

Van Emden Henson and Ulrike Meier Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002. doi: 10.1016/S0168-9274(01)00115-5.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/84e1b1ec17bb11c57234e96433022a9a-Abstract-Conference.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Axel Klawonn, Martin Lanser, and Janine Weber-Hamacher. Learning adaptive coarse spaces using transferable neural network models for linear and nonlinear overlapping domain decomposition methods. *arXiv preprint arXiv:2607.06261*, 2026. URL https://arxiv.org/abs/2607.06261.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024. doi: 10.1145/3648506.

Xin Liang, Li Wang, Lei-Hong Zhang, and Ren-Cang Li. On generalizing trace minimization principles. *Linear Algebra and Its Applications*, 656:483–509, 2023.

Jianming Liu, Ren Zhu, Jian Xu, Kun Ding, Xu-Yao Zhang, Gaofeng Meng, and Cheng-Lin Liu. PDE-Agent: A toolchain-augmented multi-agent framework for PDE solving, 2025. URL https://arxiv.org/abs/2512.16214.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Ilay Luz, Meirav Galun, Haggai Maron, Ronen Basri, and Irad Yavneh. Learning algebraic multigrid using graph neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6489–6499, 2020.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d7cb9db5ade2db7814fbd01ee59f4c7b-Abstract-Conference.html.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6: 1256–1269, 2024. doi: 10.1038/s42256-024-00897-5.

Luke N. Olson, Jacob B. Schroder, and Raymond S. Tuminaro. A general interpolation strategy for algebraic multigrid using energy minimization. *SIAM Journal on Scientific Computing*, 33(2): 966–991, 2011. doi: 10.1137/100803031.

Michael L. Parks, Eric de Sturler, Greg Mackey, Duane D. Johnson, and Spandan Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5): 1651–1674, 2006. doi: 10.1137/040607277.

David Pfau, Stig Petersen, Ashish Agarwal, David G. T. Barrett, and Kimberly L. Stachenfeld. Spectral inference networks: Unifying deep and spectral learning. In *International Conference on Learning Representations*, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28.

J. Jon Ryu, Xiangxiang Xu, H. S. Melihcan Erol, Yuheng Bu, Lizhong Zheng, and Gregory W. Wornell. Operator SVD with neural networks via nested low-rank approximation. In *International Conference on Machine Learning*, 2024.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2 edition, 2003. doi: 10.1137/1.9780898718003.

Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. SIAM, revised edition, 2011. doi: 10.1137/1.9781611970739.

Nicole Spillane, Victorita Dolean, Patrice Hauret, Frederic Nataf, Clemens Pechstein, and Robert Scheichl. Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. *Numerische Mathematik*, 126(4):741–770, 2014. doi: 10.1007/s00211-013-0576-y.

Andrea Toselli and Olof B. Widlund. *Domain Decomposition Methods—Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer, 2005. doi: 10.1007/b137868.

Ulrich Trottenberg, Cornelius W. Oosterlee, and Anton Schuller. *Multigrid*. Academic Press, 2001.

Petr Vaněk, Jan Mandel, and Marian Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996. doi: 10.1007/BF02238511.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53681–53705, 2024. URL https://proceedings.mlr.press/v235/wu24r.html.

Jinchao Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34 (4):581–613, 1992. doi: 10.1137/1034116.

Jinchao Xu and Ludmil T. Zikatanov. Algebraic multigrid methods. *Acta Numerica*, 26:591–721, 2017. doi: 10.1017/S0962492917000083.

Enrui Zhang, Adar Kahana, Alena Kopaničáková, Eli Turkel, Rishikesh Ranade, Jay Pathak, and George Em Karniadakis. Blending neural operators and relaxation methods in PDE numerical solvers. *Nature Machine Intelligence*, 6(11):1303–1313, 2024. doi: 10.1038/s42256-024-00910-x.

Zechen Zhang, Rui Peng Li, and Yousef Saad. Graph neural multilevel preconditioners for iterative solvers. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026. doi: 10.1145/3770855.3818863. URL https://arxiv.org/abs/2607.28456.

APPENDIX CONTENTS

The appendix provides proofs, implementation details, pretraining details, evaluation protocols, additional results, and a discussion of limitations.

A Mathematical Foundations and Proofs 15

A.1 From a PDE to a Sparse SPD System 15

A.2 Coarse Correction as Energy Projection 15

A.3 An Idealized Two-Level Convergence Calculation 15

A.4 Weighted Ritz Objective and Subspace Error 16

A.5 From Propositions to Implementation 16

B Architecture and Numerical Integration 17

B.1 Local Matrix Construction and Scaling 17

B.2 Propagation and Pointwise Mixing 17

B.3 Orthogonalization, Ritz Ordering, and Numerical Precision 17

B.4 Schwarz Assembly 18

B.5 AMG Candidate Assembly 18

C Pretraining Details 19

C.1 Data Recipe and Optimization 19

C.2 Complete Loss and Interpretation 19

D Experimental Protocols and Reproducibility 20

D.1 Problem Families and Statistics 20

D.2 Baseline Configurations 21

D.3 Darcy Flow Protocol 22

D.4 Implementation and Timing 23

E Additional Results 24

E.1 Comparison with a Per-Problem Neural Preconditioner 24

E.2 AMG Head: Per-Seed Results and Operator Complexity 25

E.3 Runtime 26

E.4 Darcy Generator and Two-Phase Subset 26

E.5 SPE10 Case Study 27

E.6 Per-Seed Robustness 28

F Limitations 28

A Mathematical Foundations and Proofs

This appendix derives the algebraic problem and the spectral statements used in Sections 3 and 4. Appendix B specifies the implementation, Appendix C describes pretraining, and Appendices D and E give evaluation protocols and additional results.

A.1 From a PDE to a Sparse SPD System

Consider a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$ and the scalar diffusion–reaction equation

$$-\nabla \cdot (K_\mu \nabla u) + c_\mu u = f_\mu, \qquad u|_{\partial\Omega} = 0. \tag{6}$$

Assume $K_\mu \in L^\infty(\Omega; \mathbb{R}^{d\times d})$ is symmetric and uniformly positive definite, $c_\mu \in L^\infty(\Omega)$ is nonnegative, and $f_\mu \in L^2(\Omega)$. Multiplying by $v \in H_0^1(\Omega)$ and integrating by parts gives

$$\begin{aligned} &a_\mu(u, v) = \ell_\mu(v) \quad \text{for all } v \in H_0^1(\Omega), \\ &a_\mu(u, v) = \int_\Omega (\nabla v^\top K_\mu \nabla u + c_\mu uv)\, dx, \qquad \ell_\mu(v) = \int_\Omega f_\mu v\, dx. \end{aligned} \tag{7}$$

For a conforming finite-element basis $\{\varphi_j\}_{j=1}^n$, write $u_h = \sum_j \mathbf{u}_j \varphi_j$. Testing with each basis function yields

$$A\mathbf{u} = \mathbf{b}, \qquad A_{ij} = a_\mu(\varphi_j, \varphi_i), \qquad \mathbf{b}_i = \ell_\mu(\varphi_i). \tag{8}$$

Symmetry follows from $K_\mu = K_\mu^\top$, while coercivity and linear independence of the basis imply positive definiteness. Local support yields sparsity (Brenner & Scott, 2008). Other discretizations can produce the same algebraic task; this derivation does not assert that every PDE or linearization is SPD. The continuous field u and its coefficient vector $\mathbf{u}$ are distinct, and reducing the algebraic residual does not remove discretization error.

A.2 Coarse Correction as Energy Projection

Let $A \succ 0$ and let Z have independent columns. Then $A_c = Z^\top AZ \succ 0$, and

$$\Pi_Z = ZA_c^{-1}Z^\top A, \qquad \mathbf{e}^+ = (I - \Pi_Z)\mathbf{e}. \tag{9}$$

Since $\Pi_Z^2 = \Pi_Z$ and $Z^\top A(I - \Pi_Z) = 0$, Π_Z is the A-orthogonal projector onto $\mathrm{range}(Z)$. Minimizing $\|\mathbf{e} - Zc\|_A^2$ gives the normal equations $A_c c = Z^\top A\mathbf{e} = Z^\top \mathbf{r}$. Orthogonality gives the Pythagorean identity

$$\|\mathbf{e}^+\|_A^2 = \min_c \|\mathbf{e} - Zc\|_A^2 = \|\mathbf{e}\|_A^2 - \|\Pi_Z\mathbf{e}\|_A^2. \tag{10}$$

The correction depends on the span, not the chosen basis: replacing Z by ZT for invertible T leaves Π_Z unchanged. The result assumes an exact coarse solve; finite-precision or inexact solvers introduce additional errors.

A.3 An Idealized Two-Level Convergence Calculation

Let $R \succ 0$ and assume $0 < \lambda_1 \le \cdots \le \lambda_n \le 1$ for the eigenvalues of RA. The operator RA is self-adjoint in the A inner product because $(RA)^\top A = ARA$. It therefore admits an A-orthonormal eigenbasis v_j satisfying

$$Av_j = \lambda_j R^{-1} v_j. \tag{11}$$

Suppose Z spans $v_1, \ldots, v_m$, $m < n$, and expand $\mathbf{e} = \sum_j a_j v_j$. Smoothing followed by coarse correction yields

$$E_Z\mathbf{e} = (I - \Pi_Z)(I - RA)\mathbf{e} = \sum_{j=m+1}^{n} (1 - \lambda_j) a_j v_j. \tag{12}$$

Consequently $\|E_Z\mathbf{e}\|_A^2 = \sum_{j>m}(1 - \lambda_j)^2 a_j^2$. The largest factor is $(1 - \lambda_{m+1})^2$, attained for $\mathbf{e} = v_{m+1}$, so

$$\|E_Z\|_A = 1 - \lambda_{m+1}. \tag{13}$$

This calculation isolates the benefit of representing slow directions. It requires the stated scaling, exact eigenvectors, and exact coarse correction; it is not a convergence factor claimed for the implemented learned preconditioner. General AMG theory connects coarse approximation with smoother-dependent spectra (Xu & Zikatanov, 2017).

A.4 Weighted Ritz Objective and Subspace Error

Let $H = H^\top$, $B \succ 0$, $k < n$, and $V^\top BV = I_k$. Set $C = B^{-1/2}HB^{-1/2}$ and $Q = B^{1/2}V$, so $Q^\top Q = I_k$. The generalized eigenvalues of (H, B) are the ordinary eigenvalues λ_j of C, and the Ritz values $\mu_j(V)$ are the eigenvalues of $Q^\top CQ$. The min–max principle gives $\mu_j \geq \lambda_j$ for $j \leq k$. Therefore, for positive weights $w_j = (k + 1 - j)/k$,

$$F(V) = \sum_{j=1}^{k} w_j \mu_j \geq \sum_{j=1}^{k} w_j \lambda_j = F_\star. \tag{14}$$

Equality is attained by the lowest generalized eigenspace, consistent with weighted trace minimization (Liang et al., 2023) and with neural objectives that learn ordered eigenfunctions (Pfau et al., 2019; Deng et al., 2022; Ryu et al., 2024). Counting how many prefixes contain each μ_j also proves

$$\sum_{j=1}^{k} \frac{k+1-j}{k} \mu_j = \frac{1}{k} \sum_{\ell=1}^{k} \sum_{j=1}^{\ell} \mu_j. \tag{15}$$

This explains the emphasis on useful leading prefixes when the backend retains fewer than k directions.

For the angle bound, let $p_1, \ldots, p_n$ be orthonormal eigenvectors of C and define $\alpha_j = \|p_j^\top Q\|_2^2$. Then $0 \leq \alpha_j \leq 1$, $\sum_j \alpha_j = k$, and the mass outside the target space is

$$\eta = \sum_{j>k} \alpha_j = \sum_{j \leq k} (1 - \alpha_j) = \| \sin \Theta_B(V, U_k) \|_F^2. \tag{16}$$

The trace excess satisfies

$$\begin{aligned} \sum_{j=1}^{k} (\mu_j - \lambda_j) &= \sum_{j>k} \lambda_j \alpha_j - \sum_{j \leq k} \lambda_j (1 - \alpha_j) \\ &\geq (\lambda_{k+1} - \lambda_k)\eta = \delta\eta. \end{aligned} \tag{17}$$

Since each $\mu_j - \lambda_j \geq 0$ and $w_j \geq w_k$, $F(V) - F_\star \geq w_k \delta\eta$, proving Proposition 3. Thus spectral excess tending to zero implies convergence of the subspace when $\delta > 0$.

Scope of the guarantee. The result remains valid for symmetric indefinite H, but negative local modes are not automatically the slow positive modes in Appendix A.3. Without a gap, the lowest k-dimensional target may not be unique. The theorem assumes exact normalization and concerns F, not the regularized neural objective or the optimizer trajectory. It does not prove that Adam reaches $F_\star$, that each learned stage is contractive, or that a learned local space produces globally optimal solver convergence.

A.5 From Propositions to Implementation

The three propositions motivate the design. They do not certify the trained network, and the implementation departs from them in three ways.

Error tolerance. Proposition 1 is the standard PCG bound and holds for any full-rank coarse basis. We use it only to separate the two effects of a coarse space: it changes the number of iterations, not the attainable accuracy.

Local spectral target. Proposition 2 assumes that each H_i is a positive semidefinite local Neumann matrix. The implementation uses the algebraic surrogate of Appendix B.1, whose diagonal flooring preserves symmetry but does not guarantee $H_i \succeq 0$. It also keeps a fixed number $m = 8$ of modes per subdomain instead of selecting m_i by an eigenvalue threshold. The Exact column of Table 1 uses exact eigenvectors of the same pencils with the same m. It is therefore a spectral reference for Alioth, not the best coarse space of that dimension.

Leading directions. The encoder outputs $k = 16$ directions, and the Schwarz head uses the first $m = 8$. Proposition 3 controls the full k-dimensional output, but the weights also control every leading block. The outputs are Ritz vectors sorted by energy, so the first ℓ of them span a B-orthonormal subspace $V_{1:\ell}$ with Ritz values $\mu_1, \dots, \mu_\ell$. Repeating the angle argument above with k replaced by ℓ gives $\sum_{j\le\ell}(\mu_j - \lambda_j) \ge (\lambda_{\ell+1} - \lambda_\ell)\,\|\sin\Theta_B(V_{1:\ell}, U_\ell)\|_F^2$. Since every term $\mu_j - \lambda_j$ is nonnegative and $w_j \ge w_\ell$ for $j \le \ell$,

$$\|\sin\Theta_B(V_{1:\ell}, U_\ell)\|_F^2 \le \frac{k\,\big(F(V) - F_\star\big)}{(k+1-\ell)\,(\lambda_{\ell+1} - \lambda_\ell)}. \tag{18}$$

With $k = 16$ and $\ell = m = 8$, lowering the objective therefore moves the eight retained modes toward the lowest local eigenspace whenever $\lambda_9 > \lambda_8$. The bound concerns the objective on a given pencil. It is not a generalization guarantee for matrices outside the pretraining pool, which Section 5 addresses empirically.

B Architecture and Numerical Integration

B.1 Local Matrix Construction and Scaling

Let R_i restrict vectors to an overlapping subdomain, $A_i = R_i A R_i^\top$, and let the diagonal entry of D_i be the reciprocal of the number of subdomains covering that degree of freedom. Thus $\sum_i R_i^\top D_i R_i = I$. Define

$$S_i = \operatorname{diag}(A_i)^{-1/2}, \quad B_i = S_i D_i A_i D_i S_i, \quad s_i = R_i A\mathbf{1} - A_i\mathbf{1}. \tag{19}$$

Before safeguarding, the scaled algebraic Neumann operator is $\widehat{H}_i = S_i[A_i + \operatorname{diag}(s_i)]S_i$. The implementation sets $(H_i)_{jj} = \max\{(\widehat{H}_i)_{jj}, 0.05\}$ and retains the off-diagonals. This is an algebraic local construction inspired by generalized spectral coarse spaces (Spillane et al., 2014). For SPD A, positive coverage weights imply $B_i \succ 0$. Diagonal flooring preserves symmetry but does not establish $H_i \succeq 0$ for an arbitrary application matrix.

For graph-based partitioning, the implementation includes breadth-first traversal, division into target-size cores, and enlargement by graph-neighbor layers. The four SPE10 2D layers and the Darcy fields use regular-grid partitions, and all other systems use graph partitions (Appendix D.1). The shared encoder requires only the resulting sparse matrices and weights, not a fixed global grid size.

B.2 Propagation and Pointwise Mixing

Let $h_j = (H_i)_{jj}$ and $\gamma_i = \max_j \sum_\ell |(H_i)_{j\ell}| / \max(h_j, 10^{-8})$. The diagonal in Equation (3) is

$$J_i = \max(1, \gamma_i/2)\,\operatorname{diag}(\max(h_j, 10^{-8})). \tag{20}$$

The damping is a sigmoid of one learned scalar, initialized to 0.67. Polynomial coefficients start at $c_{t,P} = 1$ and zero otherwise, giving G_i^P initially. Powers are applied recursively using sparse matrix–matrix products. Learned coefficients are unconstrained, so this parameterization alone does not guarantee contraction.

The pointwise update of Equation (3) is, with q_t the width of stage t,

$$X_i^{(t)} = Y_{i,1:q_t}^{(t)} + \phi_t\big([\,Y_i^{(t)},\ H_i Y_i^{(t)},\ B_i Y_i^{(t)},\ d_i\,]\big). \tag{21}$$

Each MLP has two hidden layers of width 64 with GELU activations. Intermediate widths are 32 and the final width is 16. Parameters are shared over points and subdomains, and each stage has its own channel mixer and polynomial coefficients. A residual connection preserves a direct computational path, not a mathematical full-rank guarantee.

B.3 Orthogonalization, Ritz Ordering, and Numerical Precision

After the mixing step of each stage, the features $X = X_i^{(t)}$ are orthonormalized. Form $G = X^\top B_i X$ and symmetrize it. Normalize its diagonal using $d_j = \max(G_{jj}, 10^{-30})^{-1/2}$ and $C =$

$\mathrm{diag}(d)G\,\mathrm{diag}(d)$. Cholesky factorization of $C+\epsilon I = LL^\top$ gives the update $X \leftarrow X\,\mathrm{diag}(d)L^{-\top}$. Two passes are applied. The initial jitter is 10^{-8} and is increased on factorization failure. The implementation safeguards nonfinite entries; such safeguards are numerical procedures, not assumptions of the exact theory.

For the resulting basis Q_i, symmetrize the projected matrix and compute

$$Q_i^\top H_i Q_i = O_i\,\mathrm{diag}(\mu_{i,1},\ldots,\mu_{i,q_t})O_i^\top, \qquad X_i^{(t)} \leftarrow Q_i O_i, \tag{22}$$

with ascending eigenvalues. Sparse features use single precision by default, while Gram matrices and small factorizations use double precision. Ritz eigenvalues retain gradients; the eigenvector rotation is evaluated without gradients to avoid unstable derivatives at repeated eigenvalues. Thus the actual training loop uses this gradient convention, rather than differentiating through every eigenvector operation. The encoder avoids a full local eigensolve but does use small factorizations of size at most 32.

B.4 Schwarz Assembly

The Schwarz head implements Eq. (4). Let $\widetilde{V}_i$ be the encoder output on subdomain i in scaled coordinates, and let $V_i = S_i\widetilde{V}_i$ be the same modes in original coordinates, with S_i from Eq. (19). For a subdomain with n_i degrees of freedom, we collect the local constant and the first eight modes,

$$W_i = \left[\, n_i^{-1/2}\mathbf{1},\ v_{i,1}/\|v_{i,1}\|_2,\ \ldots,\ v_{i,8}/\|v_{i,8}\|_2 \,\right] \in \mathbb{R}^{n_i\times 9}, \tag{23}$$

where $v_{i,j}$ is the j-th column of V_i. A QR factorization with column pivoting, $W_i\Pi_i = Q_iT_i$, gives an orthonormal basis of $\mathrm{span}(W_i)$. We keep the first r_i columns of Q_i, where r_i is the number of diagonal entries with $|(T_i)_{jj}| > 10^{-8}\,|(T_i)_{11}|$. The coarse basis and the two-level preconditioner are

$$Z = \left[\, R_1^\top D_1 Q_{1,1:r_1},\ \ldots,\ R_N^\top D_N Q_{N,1:r_N} \,\right], \qquad M^{-1} = Z\big(Z^\top AZ\big)^{-1}Z^\top + \sum_{i=1}^{N} R_i^\top A_i^{-1} R_i. \tag{24}$$

This is Eq. (1) with the coarse space of Eq. (4). Since $Q_{i,1:r_i}$ spans the constant together with the learned modes, the learned modes contribute to Z only through their components orthogonal to the constant. This is the projection stated after Eq. (4). Column pivoting keeps this span exact when a mode is nearly parallel to the constant.

Baselines with the same assembly. Nicolaides uses $W_i = [\, n_i^{-1/2}\mathbf{1}\,]$, so its coarse space has N columns. SubIter, SubIter-2×, and Exact use Eq. (23) and Eq. (24) unchanged and replace only V_i. SubIter and SubIter-2× take the first eight vectors of unlearned subspace iteration with 96 and 192 sparse matrix products. Exact takes the eight lowest generalized eigenvectors of the local pencil (H_i, B_i) of Appendix B.1, mapped back by S_i. For Alioth, SubIter, and SubIter-2×, every subdomain of every system keeps $r_i = 9$ columns, so their coarse spaces have $9N$ columns (Table 4). For Exact, a few subdomains keep fewer columns on SPE10 3D, thermal2, ecology2, and Poisson 2D, because an exact eigenvector already contains the constant direction. One-level Schwarz drops the coarse term in Eq. (24).

Solver settings. Local solves use a sparse LU factorization of each A_i. The coarse matrix $Z^\top AZ$ is factored by dense Cholesky when its dimension is at most 6,000 and by sparse LU otherwise. The preconditioner is additive. PCG starts from zero with a right-hand side b drawn from a standard normal distribution (seed 0) and stops at a relative residual of 10^{-8}, with a cap of 5,000 iterations. All entries in Table 1 converge within the cap.

B.5 AMG Candidate Assembly

The AMG head turns the first local mode of every subdomain into one global candidate vector and pairs it with the constant vector. It proceeds in four steps.

Step 1: Local projection. For each subdomain i, we take the first mode $v_{i,1}$ in original coordinates and remove its component along the normalized local constant $c_i = n_i^{-1/2}\mathbf{1}$, which gives $\hat{v}_i = v_{i,1} - c_i c_i^\top v_{i,1}$.

Step 2: Gluing. We visit the subdomains in order and assemble a global vector x. On the overlap with previously visited subdomains, a scalar α_i is chosen by least squares so that $\alpha_i \hat{v}_i$ matches the values already assembled there. Without such an overlap, α_i only fixes the sign. We then add $R_i^\top D_i \, \alpha_i \hat{v}_i$ to x. This step aligns signs and amplitudes. It does not rotate local subspaces. The columns of $[\mathbf{1}, x]$ are normalized, and a QR factorization removes a dependent column if one occurs.

Step 3: Smoothing. Following adaptive AMG (Brezina et al., 2005), we apply four symmetric Gauss–Seidel sweeps to $Ax = 0$, starting from x. Each sweep is one forward and one backward pass. The smoothed vector is orthogonalized against the normalized constant $c = n^{-1/2}\mathbf{1}$ and normalized, which gives x_s. The candidate matrix is $B = [\, c, \; x_s \,]$.

Step 4: Hierarchy. B initializes a smoothed-aggregation hierarchy (Vaněk et al., 1996) built with PyAMG. We use the symmetric strength measure with $\theta = 0.1$, energy-minimizing prolongation smoothing (Olson et al., 2011) with conjugate gradients of degree 2, and a maximum coarsest-level size of 50. All other options keep their PyAMG defaults. One V-cycle preconditions PCG, which starts from zero and stops at a relative residual of 10^{-8}, with a cap of 1,000 iterations.

Baselines with the same settings. Smoothed aggregation in Tables 1 and 2 calls the same solver with the same options and no user candidates, so PyAMG uses the constant vector alone. Smoothed aggregation and Alioth therefore differ only in the learned candidate. SubIter builds the candidate with the same four steps from unlearned subspace iteration with 96 sparse matrix products. Because B has two columns, each aggregate carries two coarse degrees of freedom. The operator complexity of the Alioth head is thus 2.48 times that of smoothed aggregation in geometric mean over the 13 systems (Appendix E.2). On Darcy flow, Alioth and SubIter encode regular 64×32 subdomain cores with overlap 2 on the 256^2 grid.

C Pretraining Details

C.1 Data Recipe and Optimization

The training pool consists of synthetic scalar-diffusion matrices in two and three dimensions, with lognormal and channelized coefficients and coefficient and mesh anisotropy. Each seed draws ten matrices from this corpus and removes the synthetic elasticity matrices, which leaves seven matrices per seed. Seeds 0, 1, and 2 contain $(5, 2)$, $(4, 3)$, and $(7, 0)$ two- and three-dimensional matrices, respectively. The pool contains no Darcy field. At each update, we sample one pool matrix and up to 32 of its subdomains, and draw fresh Gaussian probes. Training uses no solution or eigenvector labels. We minimize Eq. (5) with $\beta = 0.1$ using Adam (Kingma & Ba, 2015) with learning rate 10^{-3} and gradient-norm clipping at 1 for 3,000 updates, with no early stopping. Table 3 lists the full configuration.

Pretraining one seed takes 1,048 to 1,078 s on one RTX 5090 GPU (median 1,072 s). This time includes periodic logging of held-out curves; for seed 0, the updates alone take about 776 s. Figure 4b uses the median of 1,072 s.

Evaluation matrices are not in this pool. We report the final checkpoint after 3,000 updates; iteration curves on six evaluation systems were logged every 250 updates but not used for checkpoint or hyperparameter selection.

C.2 Complete Loss and Interpretation

For the intermediate features of stage t before orthogonalization, let $C_i^{(t)}$ be the B_i-Gram matrix after each column is normalized. The regularizer in Eq. (5) is

$$\mathcal{R}_i = \frac{1}{T} \sum_{t=1}^{T} \frac{\|C_i^{(t)} - I\|_F^2}{q_t}, \qquad \mathcal{L}_{\text{spec}} = \mathbb{E}_{A,i,\Omega} \sum_{j=1}^{k} w_j \mu_{i,j}, \tag{25}$$

where q_t is the width of stage t and Ω denotes the Gaussian probes. The penalty discourages redundant channels before normalization. It is not the orthogonality residual of the final output. Adam

Table 3: Encoder and pretraining configuration used for all reported results.

Quantity	Value
Input probes s / output modes k	32 / 16
Stages T / polynomial degree P per stage	3 / 32
Propagation budget	96 sparse matrix products
MLP hidden width / trainable parameters	64 / 36,596
Schwarz modes per subdomain	8, plus the constant
AMG learned candidates	1, plus the constant
Training seeds / matrices per seed	3 / 7
Matrices / maximum subdomains per update	1 / 32
Optimizer / updates	Adam / 3,000
Learning rate / gradient-norm clipping	10^{-3} / 1
Gram penalty coefficient β	0.1
Reported checkpoint	final update (3,000)

optimizes $\mathcal{L}_{\mathrm{spec}} + \beta\,\mathbb{E}[\mathcal{R}_i]$ with the gradient convention of Appendix B.3. Proposition 3 characterizes the exact spectral objective for a fixed pencil. It does not cover the regularized stochastic optimization, and we make no claim that training attains the optimum.

D Experimental Protocols and Reproducibility

D.1 Problem Families and Statistics

The main suite in Table 1 contains 13 SPD systems from partial differential equations. Five come from the SPE10 reservoir model (Christie & Blunt, 2001): four two-dimensional layers and the full three-dimensional model. Six come from the SuiteSparse Matrix Collection (Davis & Hu, 2011). The remaining two are uniform Poisson problems on a 1024^2 grid and a 96^3 grid. The circuit-simulation matrix G3_circuit does not come from a PDE. We report it separately and exclude it from every aggregate. With domain decomposition, it needs 619 (one-level), 325 (Nicolaides), 235 (SubIter), 195 (SubIter-2×), 138 (Exact), and 148 (Alioth) iterations; Table 9 gives its AMG results. Table 4 lists sizes, partitions, and coarse dimensions. None of these matrices is in the pretraining pool (Appendix C.1).

Selection. The suite combines a standard benchmark, public matrices, and controlled model problems. We fixed it before evaluating Alioth. SPE10 is the standard high-contrast benchmark for reservoir simulation. The six SuiteSparse matrices are large SPD matrices from PDE discretizations, with 0.2 to 1.2 million degrees of freedom, and are commonly used to evaluate algebraic preconditioners. The two Poisson problems isolate the effect of problem size. All thirteen systems are scalar problems, matching the scalar diffusion problems used for pretraining. Vector-valued systems such as elasticity are outside this scope (Appendix F).

Aggregation. Alioth entries are medians over three pretraining seeds. For a baseline with N_i^{base} iterations and Alioth with N_i^{Alioth} iterations on system i, we report the geometric-mean ratio over the $M = 13$ systems,

$$G = \exp\Big[\frac{1}{M}\sum_{i=1}^{M}\log\frac{N_i^{\mathrm{Alioth}}}{N_i^{\mathrm{base}}}\Big], \tag{26}$$

so that $G < 1$ means fewer iterations for Alioth, and a reduction of $1 - G$. Each system has equal weight, independent of its size or runtime. Wins, losses, and ties compare integer iteration counts. The gap closed relative to exact local eigenspaces is $(N^{\mathrm{SubIter}} - N^{\mathrm{Alioth}})/(N^{\mathrm{SubIter}} - N^{\mathrm{Exact}})$, computed on the 12 systems with a nonzero denominator (thermomech_dM is excluded). Relative solution error is $\|u - u_{\mathrm{ref}}\|_2/\|u_{\mathrm{ref}}\|_2$, which differs from the relative residual $\|b - Au\|_2/\|b\|_2$ used as the stopping criterion.

Table 4: Evaluation systems. N is the number of overlapping subdomains, which equals the Nicolaides coarse dimension. Alioth, SubIter, and SubIter-2$\times$ keep nine columns per subdomain (Appendix B.4), so their coarse dimension is $9N$. SPE10 2D layers use regular 55×30 subdomain cores with overlap 2. All other systems use graph partitions with cores of about 2,000 unknowns and one layer of overlap.

System	Application	DoF	N	Coarse dim. ($9N$)
SPE10 layer 5	Subsurface flow	13K	8	72
SPE10 layer 26	Subsurface flow	13K	8	72
SPE10 layer 47	Subsurface flow	13K	8	72
SPE10 layer 68	Subsurface flow	13K	8	72
SPE10 3D	Subsurface flow	1.12M	562	5,058
thermomech_dM	Heat transfer	204K	104	936
thermal2	Heat transfer	1.23M	614	5,526
parabolic_fem	Fluid dynamics	526K	268	2,412
apache2	Structural	715K	359	3,231
tmt_sym	Electromagnetics	727K	364	3,276
ecology2	Diffusion	1.00M	502	4,518
Poisson 2D	Diffusion	1.05M	528	4,752
Poisson 3D	Diffusion	885K	441	3,969
G3_circuit (not a PDE)	Circuit simulation	1.59M	792	7,128

D.2 Baseline Configurations

Numerical baselines, reference, and ablations. All domain decomposition methods share the partition, the local LU factorizations, the right-hand side, and the additive two-level form of Appendix B.4. They differ only in the coarse space.

- **One-level**: additive Schwarz without a coarse space.
- **Nicolaides**: one partition-of-unity constant per subdomain.
- **SubIter** (ablation): the constant plus the first eight vectors of unlearned subspace iteration per subdomain. It uses three stages of 32 smoothing steps each, which is the propagation budget of Alioth (96 sparse matrix products).
- **SubIter-2$\times$** (ablation): as SubIter, with 64 steps per stage (192 products).
- **Exact** (reference): the constant plus the eight lowest generalized eigenvectors of each local pencil (H_i, B_i), computed by a dense eigensolver. This construction follows GenEO (Spillane et al., 2014). It serves as a spectral reference and not as a baseline of equal cost.

For algebraic multigrid, **smoothed aggregation** uses the constant vector as its only candidate, and **SubIter** adds one candidate from unlearned subspace iteration. Both share every setting with the Alioth head (Appendix B.5). Production AMG (hypre BoomerAMG) appears only in the runtime comparison of Appendix E.3.

Learned solution maps. All seven learned baselines in Table 2 are trained or fine-tuned on the same 1,000 in-distribution Darcy fields (Appendix D.3). Every model takes $\log a$ as input, normalized by the global mean and standard deviation of the training set, and its output is de-normalized with the training statistics of u. We evaluate the weights after the last epoch and select no checkpoint on any test set. Each model runs on one RTX 5090 GPU. Table 5 lists the configurations.

- **FNO** follows the original recipe: Adam with learning rate 10^{-3} and weight decay 10^{-4}, step decay by 0.5 every 100 epochs, and batch size 20.
- **DeepONet** uses a convolutional branch with six stride-2 convolutions and two dense layers, and a four-layer MLP trunk of width 256 on (x, y), with 256 basis functions. We first trained it with the FNO schedule for 500 epochs. Its training error was still decreasing, so Table 2 reports a 3,000-epoch run with cosine decay and gradient clipping at 1, whose wall clock is comparable to FNO. The 500-epoch run is reported in Appendix E.4.

Table 5: Learned baselines on Darcy flow (256^2). Epochs count passes over the 1,000 training fields. Wall clock is the training or fine-tuning time on one RTX 5090.

Method	Parameters	Epochs	Wall clock	Note
DeepONet (Lu et al., 2021)	4.00M	3,000	13.9 min	CNN branch, MLP trunk
FNO (Li et al., 2021)	1.19M	500	20.2 min	12 modes, width 32, 4 layers
U-Net (Ronneberger et al., 2015)	7.77M	500	21.3 min	4 downsampling levels
Transolver (Wu et al., 2024)	2.83M	100	103.4 min	official Darcy configuration
Poseidon-T (Herde et al., 2024)	20.8M	200	20.6 min	official steady-state fine-tuning
DPOT-Ti (Hao et al., 2024)	5.54M	200	6.1 min	our steady-state adaptation
DPOT-S (Hao et al., 2024)	22.1M	200	17.2 min	our steady-state adaptation

- **U-Net** has channel widths 32 to 512 with group normalization and GELU, takes $(\log a, x, y)$ as input, and uses the FNO optimizer and schedule.
- **Transolver** uses the official structured-mesh Darcy configuration (hidden width 128, 8 layers, 8 heads, 64 slices), AdamW with a one-cycle schedule, batch size 4, and a relative L^2 loss plus 0.1 times a derivative regularizer. One epoch takes 61 to 65 s at 256^2, so we train 100 epochs instead of the official 500, which were set for an 85^2 grid. We do not zero the predictions on the boundary ring as the official code does, because our cell-centered grid has nonzero boundary values.
- **Poseidon-T** follows the official scOT steady-state fine-tuning path. The embedding and recovery layers are re-initialized, and the pretrained backbone is loaded. We use AdamW with learning rate 5×10^{-5} for the backbone and 5×10^{-4} for the new layers, cosine decay, batch size 40, and the official normalized L1 loss.
- **DPOT-Ti and DPOT-S**. The official DPOT code fine-tunes only time-dependent autoregressive tasks, so the steady-state setting requires our own adaptation. We load all AFNO blocks and the positional embedding, interpolated to a 32^2 patch grid, and re-initialize the patch embedding, temporal aggregation, and output layer for one input channel, one frame, and patch size 8. Hyperparameters follow the official fine-tuning configuration: Adam, one-cycle schedule with peak learning rate 10^{-3} and 40 warm-up epochs, and batch size 32.

Per-problem neural preconditioner (GNP). GNP (Chen, 2025) uses the authors' code and training recipe without modification: a residual graph convolutional network with 8 layers, embedding width 16, and hidden width 32 (10,001 parameters), trained for 2,000 steps with Adam (learning rate 10^{-3}), batch size 16, and an L1 residual loss on mixed Arnoldi and Gaussian training vectors. The matrix is scaled by a Gershgorin estimate of its spectral radius. After training, the checkpoint with the lowest training loss is restored, as in the authors' code. The solver is the authors' flexible GMRES with restart 10, a cap of 100 iterations, and a relative residual tolerance of 10^{-8}. Alioth enters the same FGMRES as a fixed preconditioner. The right-hand side is a standard normal vector (seed 0) on the six matrices of Appendix E.1 and the Darcy source term on the Darcy fields. In these comparisons, Alioth uses the AMG head with the constant and one learned candidate from seed 0, without the smoothing step of Appendix B.5.

D.3 Darcy Flow Protocol

Problem and data. We solve $-\nabla \cdot (a \nabla u) = 1$ on the unit square with homogeneous Dirichlet conditions. The discretization is the cell-centered five-point two-point-flux scheme on a 256×256 grid with harmonic averaging of face coefficients, which gives 65,536 unknowns. All coefficient fields derive from a periodic Gaussian random field g with spectral coefficients $\tau^{\alpha-1}(4\pi^2|k|^2 + \tau^2)^{-\alpha/2}$, $\alpha = 2$, and $\tau = 3$.

- **In distribution (ID)**: $a = 12$ where $g \geq 0$ and $a = 3$ elsewhere, following the FNO recipe (Li et al., 2021). We generate 1,000 training fields and 100 test fields.
- **OOD-L**: $a = \exp(\sigma g)$, with σ chosen so that the ratio of the 99th to the 1st percentile of a equals 10^4. We generate 100 fields.

- **OOD-C**: g is stretched by a factor of 8 along x, and $a = 10^3$ on the top 30% of its values and $a = 1$ elsewhere. We generate 100 fields.

ID and OOD are defined relative to the training set of the learned baselines. Alioth is not trained on any Darcy field. Its synthetic pretraining pool does contain lognormal and channelized scalar-diffusion matrices (Appendix C.1). Appendix E.4 discusses a deviation of the ID generator from the original recipe.

Reference and metrics. The reference solution u_{ref} is a sparse direct solve of the same A and b. The stored data labels differ from it by at most 1.7×10^{-9} in relative norm. Every error in Table 2 and in this appendix is $\|u - u_{\mathrm{ref}}\|_2 / \|u_{\mathrm{ref}}\|_2$ over all 256^2 cells. Learned solution maps predict u in one forward pass. Iterative methods run PCG from a zero initial guess with the AMG settings of Appendix B.5. We record the error after five iterations and the number of iterations to relative residuals of 10^{-2} and 10^{-8}, with a cap of 1,000. For Alioth, we first take the median over the three seeds on each field and then summarize over fields. Quartiles use linear interpolation. The GNP comparison runs in FGMRES on the first 10 fields of each set and is described in Appendix E.1.

Table 6: Iterative solvers on Darcy flow, all 100 fields per set, median [first quartile, third quartile]. All three methods share the AMG settings of Appendix B.5. W/L/T counts the fields on which Alioth needs fewer, more, or equally many iterations than smoothed aggregation to reach 10^{-8}.

Set	Method	Error after 5 it.	It. to 10^{-2}	It. to 10^{-8}	Op. complexity
ID	Smoothed aggregation	1.62×10^{-5} [1.53, 2.27]	5 [5, 5]	12 [12, 12]	1.85
	SubIter	1.74×10^{-5} [1.69, 1.96]	5 [5, 5]	12 [12, 12]	4.33
	Alioth	1.27×10^{-5} [1.27, 1.41]	5 [5, 5]	12 [12, 12]	4.32
OOD-L	Smoothed aggregation	3.89×10^{-5} [2.69, 5.62]	5 [5, 6]	12 [12, 13]	1.85
	SubIter	2.71×10^{-5} [1.90, 4.00]	5 [5, 5]	12 [12, 12]	4.33
	Alioth	1.96×10^{-5} [1.54, 2.47]	5 [5, 5]	12 [12, 12]	4.33
OOD-C	Smoothed aggregation	3.36×10^{-3} [1.38, 8.22]	10 [9, 11]	21 [18, 22]	1.90
	SubIter	3.25×10^{-3} [1.39, 7.73]	9 [8, 10]	20 [18, 22]	4.57
	Alioth	3.59×10^{-3} [1.59, 10.2]	9 [8, 11]	20 [18, 21]	4.58
Alioth vs. smoothed aggregation, W/L/T over 100 fields: ID 8/0/92, OOD-L 49/1/50, OOD-C 52/23/25.					

Quartiles of the error column share the power of ten of the median. Alioth has the lowest median error after five iterations on ID and OOD-L. On OOD-C its median error is slightly higher than that of smoothed aggregation (3.59×10^{-3} versus 3.36×10^{-3}), although it needs fewer iterations on 52 of the 100 fields. Per seed, the median iterations to 10^{-8} are 12, 12, and 12 on ID and OOD-L, and 19, 20, and 20 on OOD-C. At a relative residual of 10^{-8}, every run of the three iterative methods on all 300 fields reaches an error below 7.4×10^{-11}. The largest values are 5.9×10^{-11} for smoothed aggregation, 6.6×10^{-11} for SubIter, and 7.4×10^{-11} for a single Alioth seed (4.2×10^{-11} for the per-field median over seeds).

D.4 Implementation and Timing

Software and hardware. The implementation uses Python 3.12, PyTorch 2.11.0 with CUDA 12.8, SciPy 1.18.1, and PyAMG 5.3.0. The hypre BoomerAMG rows use petsc4py 3.19 with hypre. Experiments run on NVIDIA RTX 5090 GPUs with 32 GB of memory on a 320-core host. The encoder runs on one GPU. Partitioning, local factorizations, AMG setup, and all Krylov iterations run on the CPU.

Timing conditions. Iteration counts do not depend on timing, and the main-table runs were executed with several processes in parallel. We therefore report no times from those runs. All reported times come from dedicated runs with one CPU thread (`OMP_NUM_THREADS=1`).

- **Figure 4a** times, on Poisson 2D grids from 128^2 to 1024^2, the exact local generalized eigensolves (dense LAPACK, eight modes per subdomain) and the setup of smoothed aggregation on one CPU thread, and the Alioth encoding as one forward pass on one GPU,

including host–device transfer. Each point is the median of three repetitions. For Alioth, it is the median over the three seeds.

- **Figure 4b** uses one ID Darcy test field. The per-system cost of Alioth is 1.0 s, consisting of 0.22 s of encoding and 0.76 s for assembling the AMG head. Exact eigensolves take 32 s per field. GNP training takes 142 s on this field, which equals the median over the 30 fields of Appendix E.1. Pretraining costs 1,072 s once (Appendix C.1).
- **The runtime comparison** in Appendix E.3 runs each method serially on an otherwise idle machine with one CPU thread. hypre is run twice per matrix, and the second run is reported.

E Additional Results

E.1 Comparison with a Per-Problem Neural Preconditioner

We compare Alioth with GNP (Chen, 2025) inside the same flexible GMRES with restart 10, tolerance 10^{-8}, and a cap of 100 iterations. Appendix D.2 gives the GNP recipe. Alioth uses the AMG head with the constant and one learned candidate from seed 0, without the smoothing step of Appendix B.5. Alioth setup includes partitioning, encoding, candidate assembly, and the AMG setup.

Darcy flow. GNP is trained separately on each of the first 10 fields of every Darcy set. Table 7 reports the results. GNP converges on none of the 30 fields. After five iterations its error stays between 0.985 and 1.000, so the iterate has barely moved from the zero initial guess. Alioth converges on all 30 fields. In ID, fields 2, 3, and 8 take the same single coefficient value everywhere and yield the same matrix, so each method gives identical results on these three fields (Appendix E.4).

Table 7: GNP versus zero-shot Alioth on the first 10 fields of each Darcy set, both in FGMRES(10) with a cap of 100 iterations. Entries are median [minimum, maximum] over the 10 fields. Conv. counts converged fields, and Res. at 100 is the GNP relative residual after 100 iterations. Errors after five FGMRES iterations are relative to the direct solution.

	GNP (trained per field)			Alioth (zero-shot)		
Set	Training (s)	Conv.	Res. at 100	Setup (s)	Conv.	It. to 10^{-8}
ID	141.5 [141.2, 143.8]	0/10	0.684 [0.584, 0.739]	1.0 [1.0, 1.1]	10/10	12 [12, 12]
OOD-L	143.2 [142.2, 144.9]	0/10	0.994 [0.987, 1.000]	1.1 [1.0, 1.2]	10/10	12 [12, 13]
OOD-C	142.2 [141.6, 143.9]	0/10	0.913 [0.875, 0.925]	1.0 [1.0, 1.1]	10/10	22 [20, 27]

Median error after 5 iterations, GNP / Alioth: ID 0.99 / 6.0×10^{-5}; OOD-L 1.0 / 1.2×10^{-4}; OOD-C 1.0 / 3.8×10^{-2}.

These Alioth numbers come from FGMRES and differ from the PCG numbers in Table 2. GMRES minimizes the residual rather than the energy-norm error, so its error after five iterations is 2 to 7 times larger than that of PCG on the same fields. They should be compared only with the GNP rows of the same table.

Two checks indicate that the GNP result reflects a weak learned preconditioner rather than a failed setup. First, unpreconditioned FGMRES(10) reaches median residuals of 0.82, 0.97, and 0.96 after 100 iterations on ID, OOD-L, and OOD-C. GNP improves on it slightly on all 10 ID and all 10 OOD-C fields, and it is worse on all 10 OOD-L fields. Its training loss decreases by factors of about 17 to 34 on ID and 8 to 12 on OOD-L. Second, we trained GNP for 10,000 steps, five times the recipe, on ID field 0 and OOD-C field 0 (709 s and 704 s). Neither converges. The residual after 100 iterations drops from 0.62 to 0.40 on ID field 0 and stays at 0.87 on OOD-C field 0, and the error after five iterations stays above 0.98. These results hold for the reported training budget and do not establish an intrinsic limit of neural preconditioners.

Matrices from Table 1. Table 8 repeats the comparison on six matrices with a standard normal right-hand side. GNP reaches the cap on five of them and converges on thermomech_dM in 29 iterations, where Alioth needs 5. On the eight evaluation matrices with more than 0.7 million unknowns, GNP with the authors' batch size of 16 runs out of memory on a 32 GB GPU. With batch size 4, it converges on none of the eight within the cap.

Table 8: GNP versus zero-shot Alioth on six matrices of Table 1, FGMRES(10), tolerance 10^{-8}, cap 100. "$>$100" means no convergence within the cap, and the parenthesis gives the final relative residual. GNP training and Alioth setup are different cost components.

System	GNP training (s)	GNP iterations	Alioth iterations	Alioth setup (s)
SPE10 layer 5	59	>100 (2×10^{-1})	9	0.4
SPE10 layer 26	54	>100 (1×10^{-1})	9	0.3
SPE10 layer 47	57	>100 (7×10^{-1})	33	0.3
SPE10 layer 68	57	>100 (5×10^{-1})	31	0.3
thermomech_dM	489	29	5	4.2
parabolic_fem	1,317	>100 (1×10^{-2})	11	14.3

E.2 AMG Head: Per-Seed Results and Operator Complexity

Table 9 expands the AMG block of Table 1 with per-seed iterations, operator complexities, and the head without the smoothing step of Appendix B.5. All columns share the same SA settings and include the constant vector.

Table 9: AMG iterations to a relative residual of 10^{-8} and operator complexity (op.). Alioth entries give the median and, in brackets, seeds 0, 1, and 2. "No smoothing" is the Alioth head without the four Gauss–Seidel sweeps on the candidate. G3_circuit is not a PDE and is excluded from all aggregates.

	Smoothed aggregation		SubIter		Alioth	Alioth, no smoothing
System	it.	op.	it.	op.	it. [s0, s1, s2] / op.	it. [s0, s1, s2]
SPE10 layer 5	10	2.43	10	6.65	9 [9, 10, 9] / 6.64	9 [9, 10, 9]
SPE10 layer 26	10	2.41	10	6.64	9 [9, 10, 9] / 6.63	9 [9, 9, 9]
SPE10 layer 47	22	2.47	24	6.85	23 [22, 24, 23] / 6.83	23 [23, 23, 22]
SPE10 layer 68	20	2.46	20	6.83	20 [20, 21, 20] / 6.77	21 [20, 21, 21]
SPE10 3D	21	3.74	26	11.90	23 [23, 23, 30] / 11.91	29 [30, 24, 29]
thermomech_dM	5	1.30	5	2.20	5 [5, 5, 5] / 2.21	5 [5, 5, 5]
thermal2	12	1.45	13	2.81	12 [13, 12, 12] / 2.81	13 [13, 13, 13]
parabolic_fem	61	1.53	12	2.91	12 [12, 11, 12] / 2.90	12 [12, 11, 12]
apache2	11	2.65	87	8.16	91 [90, 91, 91] / 8.18	91 [91, 91, 91]
tmt_sym	10	1.63	11	3.49	11 [11, 11, 11] / 3.49	15 [15, 15, 15]
ecology2	11	1.86	11	4.37	12 [12, 13, 11] / 4.39	12 [12, 12, 11]
Poisson 2D	11	2.12	13	5.54	13 [13, 13, 13] / 5.54	14 [14, 13, 14]
Poisson 3D	9	2.53	10	7.38	11 [11, 11, 11] / 7.36	12 [12, 12, 12]
G3_circuit (not a PDE)	16	3.59	16	11.42	17 [17, 16, 17] / 11.45	16 [16, 16, 17]

Over the 13 systems, the geometric-mean iteration ratio of Alioth to smoothed aggregation is 1.077, with 3 wins, 7 losses, and 3 ties. The ratio of Alioth to SubIter is 0.983 (5 wins, 3 losses, 5 ties), and that of SubIter to smoothed aggregation is 1.096. Per seed, the ratio of Alioth to smoothed aggregation is 1.079, 1.102, and 1.092. Both learned and unlearned candidates double the coarse degrees of freedom per aggregate, and the operator complexity rises by a factor of 2.48 in geometric mean. Smoothing the candidate lowers iterations relative to the head without smoothing, with a geometric-mean ratio of 0.938 (6 wins, 0 losses, 7 ties), and leaves the complexity essentially unchanged.

The constant vector already captures the near-kernel of most of these systems, so a second candidate rarely helps. The exception is parabolic_fem, where both learned and unlearned candidates reduce iterations from 61 to 12. On apache2, both degrade convergence, from 11 to 87 (SubIter) and 91 (Alioth). The loss therefore comes from turning local modes into a global candidate, not from learning itself.

Candidate filtering. We also tested keeping the learned candidate only when its Rayleigh quotient is at most twice that of the constant. On systems with Dirichlet boundaries or near-singular operators, the constant has a very small Rayleigh quotient (3.8×10^{-8} on tmt_sym and 3.8×10^{-6}

on parabolic_fem), so this rule rejects the learned candidate on most systems. It also rejects the candidate that reduces parabolic_fem from 61 to 12 iterations. We therefore keep the candidate unconditionally.

E.3 RUNTIME

Table 10 reports end-to-end setup and solve times on four SuiteSparse systems with dedicated timing runs (conditions in Appendix D.4). hypre BoomerAMG with default settings is the fastest method on every system. Alioth with GPU encoding takes 16 to 33 times as long as hypre. Graph partitioning, implemented as a pure-Python breadth-first traversal, accounts for 42% to 82% of the Alioth total. Without it, Alioth still takes 10.0 to 27.3 s. On these systems, Alioth does not yet match a production AMG code in either iterations or wall-clock time.

Table 10: End-to-end setup plus solve time to a relative residual of 10^{-8}, in seconds, with one CPU thread; it. is the number of iterations. Alioth encodes on one GPU (or on one CPU thread, last row), and its totals include partitioning. The timed Alioth head uses seed 0 and the constant plus one learned candidate without the smoothing step of Appendix B.5, so its iteration counts differ from Table 1. SA here uses PyAMG defaults, unlike the smoothed aggregation of Table 1. Adaptive SA uses two candidates with five candidate iterations.

	parabolic_fem 526K		tmt_sym 727K		ecology2 1.00M		thermal2 1.23M	
Method	it.	s	it.	s	it.	s	it.	s
hypre BoomerAMG (defaults)	5	1.0	7	1.9	7	1.4	8	4.7
SA (PyAMG defaults)	21	1.4	25	4.0	12	2.9	25	8.2
Adaptive SA (2 candidates)	65	34.1	16	35.5	17	43.7	11	55.4
Alioth, GPU encoding	13	17.3	14	30.1	11	34.3	13	154.4
partitioning		7.3		13.9		14.5		127.1
partitioning share		42%		46%		42%		82%
total without partitioning		10.0		16.2		19.8		27.3
Alioth, CPU encoding	13	22.4	14	36.7	11	40.5	13	175.8

Encoding on one CPU thread instead of the GPU adds 5.1 to 21.4 s. Partitioning is part of the practical cost of the current implementation, and we include it in every Alioth total.

E.4 DARCY GENERATOR AND TWO-PHASE SUBSET

Our ID generator keeps the mean mode ($k = 0$) of the Gaussian random field. Common implementations of the FNO recipe set this mode to zero. Its spectral weight is $1/\tau = 1/3$, which is 5.4 times the weight at $|k| = 1$, so the field mean often dominates, and after thresholding some fields take a single value everywhere. Of the 100 ID test fields, 32 have a single coefficient (14 with $a = 3$ and 18 with $a = 12$). Of the 1,000 training fields, 266 do (141 and 125). The OOD-L and OOD-C sets contain no such field.

A single-coefficient field gives a scaled Poisson problem. On these fields, learned solution maps are about an order of magnitude more accurate than on two-phase fields, which pulls down their ID medians and first quartiles. Iterative solvers see only two distinct matrices among them, which is why the median and the first quartile of Alioth coincide in Table 6. We therefore also report the 68 two-phase ID fields separately.

On the two-phase subset, the best learned map is DPOT-S with a median error of 7.22×10^{-3}, and FNO reaches 1.09×10^{-2}. After five iterations, the iterative solvers reach 1.35×10^{-5} (Alioth) to 2.06×10^{-5} (smoothed aggregation), and all three need a median of 12 iterations to reach 10^{-8}, with both quartiles also at 12. The gap between Alioth and the best learned map grows from 359 times on all ID fields to 535 times on the two-phase subset. The conclusions of Table 2 therefore do not depend on the single-coefficient fields.

Table 11: Relative error on Darcy flow, median [first quartile, third quartile]; quartiles share the power of ten of the median. "Two-phase" is the ID subset of 68 fields with both coefficient values, and "single" gives the median over the other 32 ID fields. Learned maps predict in one forward pass. Iterative solvers report the error after five PCG iterations, with Alioth taking the per-field median over three seeds. Table 2 reports DeepONet at 3,000 epochs.

	ID				
Method	All 100	Two-phase (68)	Single (32)	OOD-L (100)	OOD-C (100)
Learned solution maps					
DeepONet (500 epochs)	$4.25[0.670, 7.71]\times10^{-2}$	$6.01[4.20, 8.63]\times10^{-2}$	5.44×10^{-3}	$9.82[8.99, 13.9]\times10^{-1}$	1.85 [1.06, 2.88]
DeepONet (3,000 epochs)	$3.00[0.190, 6.44]\times10^{-2}$	$5.49[2.84, 8.16]\times10^{-2}$	1.90×10^{-3}	$9.83[8.25, 24.9]\times10^{-1}$	2.10 [1.28, 3.08]
FNO	$7.01[1.43, 13.3]\times10^{-3}$	$1.09[0.688, 1.51]\times10^{-2}$	1.43×10^{-3}	$9.67[8.14, 13.4]\times10^{-1}$	$9.75[5.90, 13.5]\times10^{-1}$
U-Net	$5.25[1.53, 9.62]\times10^{-3}$	$8.51[5.21, 11.1]\times10^{-3}$	1.53×10^{-3}	$9.94[9.33, 20.2]\times10^{-1}$	1.21 [0.783, 1.86]
Transolver (100 epochs)	$1.03[0.0984, 1.68]\times10^{-2}$	$1.50[1.02, 1.97]\times10^{-2}$	9.84×10^{-4}	$9.90[8.85, 15.1]\times10^{-1}$	$9.39[6.61, 13.6]\times10^{-1}$
Poseidon-T	$1.29[1.02, 1.69]\times10^{-2}$	$1.55[1.28, 1.83]\times10^{-2}$	1.02×10^{-2}	$9.88[8.70, 13.9]\times10^{-1}$	1.13 [0.760, 1.69]
DPOT-Ti	$5.36[0.306, 11.1]\times10^{-3}$	$8.79[5.27, 11.9]\times10^{-3}$	3.06×10^{-4}	$9.92[9.14, 15.4]\times10^{-1}$	$9.12[6.89, 13.8]\times10^{-1}$
DPOT-S	$4.56[0.255, 9.25]\times10^{-3}$	$7.22[4.07, 10.3]\times10^{-3}$	2.55×10^{-4}	$9.95[9.31, 12.9]\times10^{-1}$	$7.50[6.62, 11.0]\times10^{-1}$
Iterative solvers after five PCG iterations					
Smoothed aggregation	$1.62[1.53, 2.27]\times10^{-5}$	$2.06[1.59, 2.62]\times10^{-5}$	1.53×10^{-5}	$3.89[2.69, 5.62]\times10^{-5}$	$3.36[1.38, 8.22]\times10^{-3}$
SubIter	$1.74[1.69, 1.96]\times10^{-5}$	$1.79[1.62, 2.12]\times10^{-5}$	1.74×10^{-5}	$2.71[1.90, 4.00]\times10^{-5}$	$3.25[1.39, 7.73]\times10^{-3}$
Alioth	$1.27[1.27, 1.41]\times10^{-5}$	$1.35[1.21, 1.47]\times10^{-5}$	1.27×10^{-5}	$1.96[1.54, 2.47]\times10^{-5}$	$3.59[1.59, 10.2]\times10^{-3}$

E.5 SPE10 Case Study

Figure 3 uses SPE10 layer 68 on its 220×60 grid, discretized by two-point flux approximation and split into eight overlapping subdomains (regular 55×30 cores, overlap 2). All four columns share the matrix, the partition, the local factorizations, and the right-hand side, and they use the additive two-level preconditioner of Appendix B.4. Nicolaides has a coarse space of dimension 8. SubIter, Alioth, and Exact add eight modes per subdomain to the constant, for dimension 72. The Alioth column shows seed 0, whose count of 46 iterations is the median of the three seeds and equals the entry in Table 1. Every count in Table 12 matches Table 1.

The top row of Figure 3 shows the first mode on subdomain Ω_5 after the constant is removed. The principal angles are measured in the B_5-inner product between the first three modes and the first three exact generalized eigenvectors. The middle and bottom rows show $\log_{10}(|e_k|/\|e_0\|_\infty)$ after 10 and 30 iterations from a zero initial guess. The lower end of the color scale saturates, so the darkest color does not mean zero error.

Table 12: Case study on SPE10 layer 68. Relative L^2 errors after 10 and 30 PCG iterations, iterations to a relative residual of 10^{-8}, and principal angles on subdomain Ω_5 between the first three modes and the first three exact eigenvectors. Exact converges at iteration 28, so its last error is at that iteration.

Method	Error, it. 10	Error, it. 30	It. to 10^{-8}	Angles on Ω_5
Nicolaides	1.12×10^{-1}	8.00×10^{-4}	65	–
SubIter	7.69×10^{-2}	1.40×10^{-4}	57	$5.4^\circ, 14.1^\circ, 41.6^\circ$
Alioth, seed 0	4.44×10^{-2}	2.81×10^{-6}	46	$2.2^\circ, 6.0^\circ, 13.0^\circ$
Alioth, seed 1	1.59×10^{-2}	1.44×10^{-5}	49	$2.7^\circ, 9.2^\circ, 23.8^\circ$
Alioth, seed 2	8.76×10^{-3}	1.99×10^{-6}	44	$3.4^\circ, 6.1^\circ, 7.1^\circ$
Exact	7.26×10^{-4}	2.85×10^{-10} (it. 28)	28	–

For all three seeds, the first learned mode lies within 3.4° of the first exact eigenvector, compared with 5.4° for SubIter. Every seed also has a lower error than SubIter after 30 iterations.

Other SPE10 layers. The same pattern holds across the four SPE10 layers of Table 1. Nicolaides needs 48, 53, 58, and 65 iterations on layers 5, 26, 47, and 68, SubIter needs 37, 40, 58, and 57, Alioth needs 29, 29, 47, and 46, and Exact needs 24, 23, 24, and 28. Alioth closes 62% and 65% of the gap between SubIter and Exact on layers 5 and 26, but only 32% and 38% on layers 47 and 68. The remaining gap to exact local eigenspaces is therefore largest on layers 47 and 68.

E.6 Per-Seed Robustness

Table 13 repeats the domain-decomposition aggregates of Table 1 for each pretraining seed, using the geometric-mean ratio of Eq. (26) over the 13 systems.

Table 13: Per-seed geometric-mean iteration ratio of Alioth to each domain-decomposition method over the 13 systems ($G < 1$ means fewer iterations for Alioth), with Alioth wins/losses/ties in integer iteration counts. The median row uses the per-system median over the three seeds, as in Table 1.

	vs. Nicolaides		vs. SubIter		vs. SubIter-2×		vs. One-level	
Alioth	G	W/L/T	G	W/L/T	G	W/L/T	G	W/L/T
Seed 0	0.518	12/1/0	0.789	12/1/0	0.989	6/5/2	0.287	12/1/0
Seed 1	0.544	12/1/0	0.828	12/1/0	1.038	4/7/2	0.301	12/1/0
Seed 2	0.517	12/1/0	0.787	12/1/0	0.986	7/5/1	0.286	12/1/0
Median over seeds	0.522	12/1/0	0.795	12/1/0	0.997	7/5/1	0.289	12/1/0

Every seed needs fewer iterations than Nicolaides and SubIter on 12 of the 13 systems, and the single loss is always thermomech_dM. Against SubIter-2×, which uses twice the propagation budget, the per-seed ratios lie between 0.986 and 1.038, so the two are roughly on par.

F Limitations

We summarize the limitations revealed by the experiments.

Scope. All evaluation systems are scalar SPD problems, and pretraining uses only scalar diffusion. Vector-valued systems such as elasticity, whose near-kernel contains rigid-body modes, and nonsymmetric or indefinite systems remain untested. Extending Alioth to them requires multi-component probes and pretraining data, and, for nonsymmetric systems, a different spectral objective.

Algebraic multigrid. With AMG, the learned candidate performs on par with smoothed aggregation rather than improving on it. The geometric-mean iteration ratio is 1.08, with a large gain on parabolic_fem (61 to 12 iterations) and a large loss on apache2 (11 to 91). Unlearned candidates show the same pattern (Appendix E.2), so the bottleneck lies in assembling local modes into a global candidate rather than in the encoder.

Gap to exact local eigenspaces. Alioth closes a median of 63% of the gap between SubIter and exact local eigenspaces. The gap closed is smaller on the high-contrast SPE10 systems (18–38%), thermal2 (26%), and apache2 (36%). Pretraining on higher-contrast coefficients may narrow it.

Runtime. Our prototype is slower end to end than production AMG. On four timed systems, hypre BoomerAMG takes 1.0 to 4.7 s, whereas the prototype takes 17.3 to 154.4 s. Graph partitioning, implemented as a pure-Python traversal, accounts for 42% to 82% of this time, and even without it the prototype takes 6 to 14 times as long as hypre (Appendix E.3). The encoder itself takes 3.3 s for one million degrees of freedom on a GPU. An optimized partitioner and a compiled solve phase are required before Alioth can compete with production AMG in wall-clock time.

Baselines and cost accounting. We compare with classical coarse spaces and with exact local eigenspaces across the full suite. Adaptive smoothed aggregation, which discovers slow modes without training, appears only in the runtime study on four systems (Appendix E.3). There, Alioth needs fewer iterations on three of them, but bootstrap AMG, algebraic coarse spaces such as GDSW, RGDSW, and algebraic GenEO (Dohrmann et al., 2008; Dohrmann & Widlund, 2017; Al Daas & Grigori, 2019), and a suite-wide comparison with adaptive methods remain open. The neural preconditioner GNP did not converge on our Darcy fields, and five times more training on two of them did not change this (Appendix E.1). Per-problem tuning might. Finally, the equal-budget comparison with SubIter counts sparse matrix products. The pointwise MLP adds dense per-node work that this count omits, although the measured encoding time in Figure 4a includes it.