
Self-Supervised Coarse Spaces for Structured Multigrid Failure Modes

Reza Akbarian Bafghi¹, Rahul Sharma², John L. Davenport², Maziar Raissi³
¹University of Colorado Boulder ²Vinci4D ³University of California Riverside
reza.akbarianbafghi@colorado.edu rahul.sharma@vinci4d.ai
johnny@vinci4d.ai maziar.raissi@ucr.edu

Abstract

Can a learned representation improve a structured multigrid solver after strong classical tuning? We train a Fourier Neural Operator to produce a low-dimensional coarse space using an unsupervised Rayleigh-quotient loss, which consumes neither solution labels nor eigensolves. We evaluate two failure modes. For rotated anisotropic diffusion in 2D, default Ruge-Stüben algebraic multigrid (AMG) takes 106.5 iterations, versus 46.7 for geometric multigrid (GMG); the learned space reduces AMG to 55.4. A stronger classical baseline changes the conclusion: evolution strength-of-connection reduces AMG to 32.6 without learning, and the same learned checkpoint gives a further 18.4% reduction to 26.6. For topologically complex 3D mazes, a held-out, multi-seed study finds a 20.2% mean reduction for GMG3D. Transfer is sharply scoped. Rotated anisotropy is negative in 3D on average, and the maze-trained network at its original rank is harmful on real Bentheimer micro-CT data. Increasing rank and distilling separately on each real operator recovers a 75.5% reduction relative to GMG3D, but still needs $2.1\times$ the AMG iterations. Paired end-to-end timing with a separate operator-only checkpoint shows no single-solve speedup over evolution AMG; repeated solves only approach break-even. The representation complements classical solver design rather than replacing it.

1 Introduction

Structured grids make geometric multigrid (GMG) inexpensive, but neither known geometry nor smooth benchmark coefficients guarantee a good coarse space. We study two distinct gaps: oblique material anisotropy, which can break classical algebraic multigrid (AMG) strength heuristics, and 3D topological obstacles, which can break GMG’s geometric hierarchy. The question is deliberately practical: does a learned coarse-space representation still add value after the relevant classical alternatives are tested?

Our answer is qualified but useful. On the 2D anisotropy family, learning substantially improves default AMG, yet a classical evolution-strength hierarchy is stronger; the learned space remains complementary and cuts a further 18.4%. On held-out 3D mazes, it gives a smaller but consistent 20.2% mean cut. The appendix reports the unsuccessful extensions with the same prominence: average 3D anisotropy, large-resolution transfer, and zero-shot Bentheimer transfer. This separation between a mechanism’s potential, a generalizing checkpoint, and a competitive solver is central to the evaluation.

Related work. Classical AMG and GMG are the starting points [Ruge and Stüben, 1987, Trottenberg et al., 2000]; rotated anisotropy exposes limits of AMG strength heuristics, motivating evolution

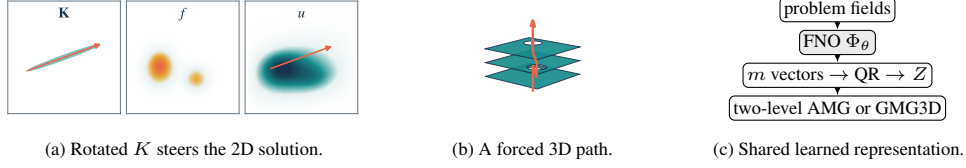


Figure 1: Problem families and method. Only Φ_θ is learned in the default path; QR and two-level compositing are fixed.

strength [Olson et al., 2010]. Learned multilevel components include prolongation [Greenfeld et al., 2019, Luz et al., 2020], sparse coarse operators [Fink et al., 2026], and domain-decomposition coarse spaces [Klawonn et al., 2026]. Operator-learned preconditioning and deflation are related but distinct [Kopaničáková and Karniadakis, 2025, Kopaničáková et al., 2025]. NeurKItt [Luo et al., 2024] is closest in architecture, using an FNO and QR but training against eigensolved invariant subspaces. GenEO [Spillane et al., 2014] is our classical coarse-space control. Appendix A details the closest multilevel comparisons.

2 Method and evaluation

We discretize $-\nabla \cdot (K \nabla u) = f$ by a cell-centered finite-volume method, producing an SPD system $Ax = b$. An FNO Φ_θ maps the discretized problem fields to m vectors; diagonal rescaling and thin QR produce $Z \in \mathbb{R}^{N \times m}$. The default objective is

$$\mathcal{L}(Q; A) = \frac{1}{m} \text{tr} \left(Q^\top D^{-1/2} A D^{-1/2} Q \right), \quad D = \text{diag}(A),$$

which targets low-energy error without consuming eigenvectors or solution labels. Low energy under A is not necessarily slow to converge under the chosen AMG or GMG preconditioner; this objective is a proxy for, not a direct optimization of, the wrapped solver’s convergence. At test time, Z enters a balancing two-level deflation correction [Nicolaidis, 1987, Mandel, 1993, Tang et al., 2009] around AMG in 2D and GMG3D in 3D. Appendix B specifies all channels, architectures, ranks, and training schedules.

Pillar 1 uses constant rotated tensors $K = R(\theta) \text{diag}(k_1, k_2) R(\theta)^\top$, with $\theta \in [16, 25]^\circ$, $k_1/k_2 \in [200, 5000]$, $n = 256$, and $m = 32$. Pillar 2 uses near-impermeable barrier planes with randomized openings, contrast 10^4 , $n = 48$ ($N = 110,592$), and $m = 16$. We report PCG iterations to relative residual 10^{-8} and separately report setup plus solve time. Every displayed number is checked against its frozen result record by the provenance tests described in Appendix C.

3 Results

Pillar 1: the baseline gap and the stronger classical test. Across 12 held-out operators, default Ruge-Stüben AMG needs 106.5 iterations and GMG needs 46.7. The learned space reduces default AMG to 55.4, a 48.0% cut that ranges from 47.03% to 48.04% across three training seeds. This is not the strongest classical comparison. Changing only AMG’s strength measure to evolution strength [Olson et al., 2010] reduces the count to 32.6, a 69.4% cut with no learning. Applying the unchanged learned checkpoint on top reduces it again to 26.6, a further 18.4%, with a positive cut on all 12 operators. Figure 2a makes both denominators explicit.

Pillar 1: robustness and limits. Smoothed-aggregation AMG improves the default by 22.5%, and the same learned checkpoint adds 16.6%. Fixed-rank geometric spaces add only 6.9% to 10.2%, and our fixed-configuration GenEO-style control reaches 29.0%; our GenEO-style adaptive control instead reaches 53.8% to 75.1% with more vectors. Its local eigenproblem approximates, rather than exactly reproduces, GenEO’s overlap-restricted construction, so the rank ratios are not a general efficiency claim. An independent PyAMG generator reproduces the AMG-worse-than-GMG ordering for its finite-difference stencil, but not its finite-element stencil. NeurKItt-style eigensolve supervision gives a statistical tie, 48.9% versus 48.0%. The headline architecture loses slightly on wall-clock, while a smaller architecture wins by 9.6%. The default AMG/GMG ratio also grows from $1.53\times$ at

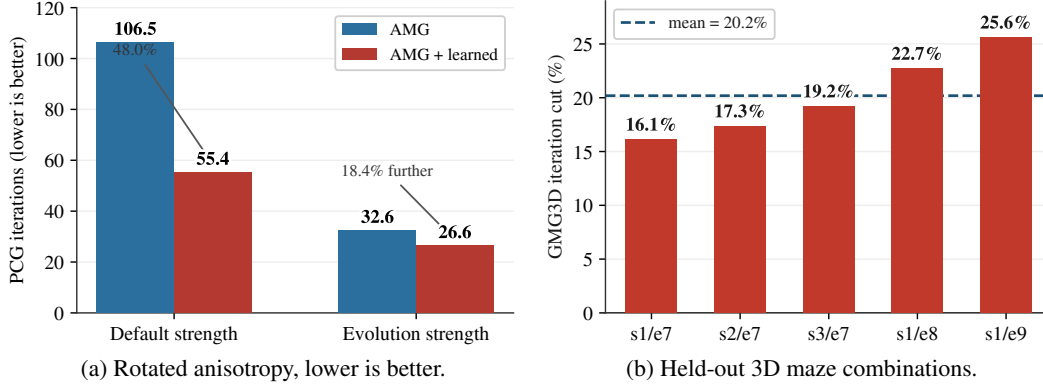


Figure 2: The two central tests. (a) Mean PCG iterations on 12 operators: evolution strength is a much stronger AMG baseline, while the learned space remains complementary. (b) GMG3D iteration cut for five training seed and evaluation seed combinations; the dashed line is their 20.2% mean. Values come directly from the records behind Tables 3 and 7.

$n = 128$ to $3.38\times$ at $n = 512$ (Figure 3a). Full costs, uncertainty, and discretization controls appear in Appendix D.

Pillar 1: 3D does not generalize broadly. AMG beats GMG3D on average in the synthetic sweep and real-rock tests, the reverse of the 2D ordering. The appendix documents a narrow exception found by targeted search, not a generalizing 3D checkpoint.

Pillar 2: held-out 3D topology. On one fixed maze operator, an oracle space establishes a 37.1% GMG3D cut. The trained network is evaluated on six held-out instances for each of five training/evaluation seed combinations. All five cuts are positive, from 16.1% to 25.6%, with mean 20.2% (Figure 2b). Increasing the training set from 64 to 200 instances raises the matched seed result from 16.1% to 24.1%. Resolution transfer is weaker: a network trained only at $n = 48$ has no AMG wall-clock crossover and becomes harmful at $n = 256$ (Appendix E).

Pillar 2: real topology requires adaptation. On three Bentheimer sandstone subvolumes, GMG3D is already much worse than AMG, 111.4 versus 12.8 iterations. The maze-trained $m = 16$ checkpoint worsens GMG3D by 14.6%. An oracle rank sweep on one subvolume is also negative at $m = 16$, but cuts iterations by 64.7% at $m = 32$ and 81.5% at $m = 64$. Directly retraining at $m = 64$ remains negative; per-operator distillation toward the oracle subspace gives a 75.5% mean cut across all three. This is adaptation, not zero-shot generalization, and the resulting method still takes $2.1\times$ the iterations of plain AMG. On synthetic mazes too, learned plus GMG3D remains slower than plain AMG in the measured wall-clock comparison.

Offline transient replay. To test amortization, we solve precomputed right-hand sides from a backward-Euler trajectory as one block. The learned correction cuts GMG iterations by 9% to 16%; the timing combines batching and learning, so it does not establish faster online time stepping. The eight-step case is slower; details appear in Appendix E.

4 Discussion

Solver selection depends on wall-clock, not only rank or iterations. In the dedicated strength-measure sweep, evolution AMG costs 0.74s in setup plus solve versus 1.28s for default AMG, so the stronger classical choice is also 42% faster. A separate paired study of a retrained operator-only checkpoint includes inference, QR, setup, and solves on 12 operators with three alternating-order repeats per machine. Relative to evolution AMG alone, learned plus evolution is slower for one right-hand side on both machines. At 64 right-hand sides the geometric speed ratios are 1.01 and 1.03: approximately break-even to a small machine-dependent gain, not an established general speedup (Appendix D). With default AMG, the large network loses 2.2% to 3.2% wall-clock across three

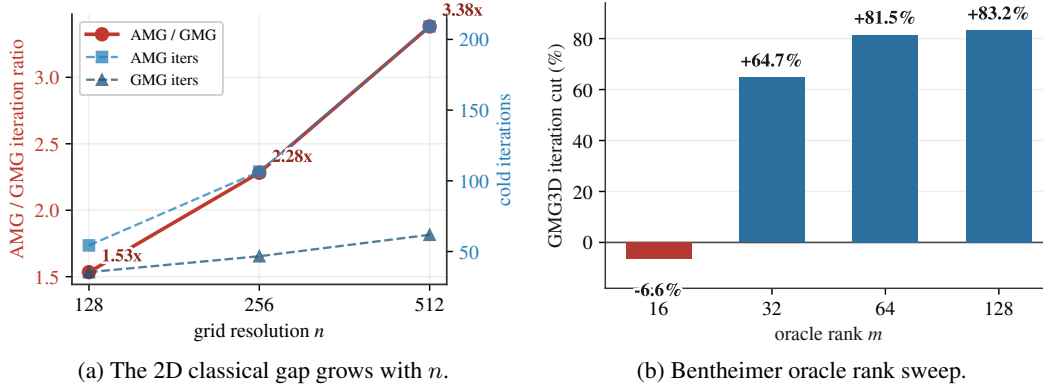


Figure 3: Scope diagnostics from frozen records. (a) Default AMG becomes progressively weaker than GMG as 2D resolution increases. (b) On one real Bentheimer subvolume, the oracle correction changes from harmful at $m = 16$ to strongly positive at $m = 32$, then saturates. Panel (b) establishes representational headroom; it is a per-operator oracle, not a generalizing learned result.

sessions, while a smaller network wins by 9.6%; wrapping GMG is 1.6 to 2.1 $\times$ slower. Learning is credible only when inference and QR can be amortized over repeated solves.

The classical controls identify two different failure mechanisms. For axis-aligned anisotropy, semi-coarsening behaves as textbook theory predicts. On the oblique 2D family, an axis-aligned line smoother is 3.59 $\times$ worse and semi-coarsening is 35.7% worse, because neither tracks the rotated direction. Evolution strength instead changes how AMG infers strong connections. We have not isolated which residual modes explain the further 18.4% learned cut, so complementarity is empirical, not yet a mechanistic claim.

Oracle capacity is not deployment performance. At rank 32, a converged 2D oracle cuts default-AMG iterations by 48.45%, only 0.45 points beyond the trained 48.0% (Appendix D); evolution AMG remains stronger. On Bentheimer, rank-64 oracle headroom is large, but direct rank-64 training fails. Capacity alone does not ensure a useful predictor.

Real-data conclusions remain limited. The Bentheimer volumes are downsampled from 75^3 to the trained 48^3 resolution before Otsu segmentation. The resulting pore fractions are 18% to 29% above the dataset’s documented range, and alternative resampling rules were not tested, leaving a topology-distortion confound. The positive distillation result also uses a separate oracle target for each operator. It establishes capacity at sufficient rank, not self-supervised transfer or superiority to AMG on real rock data.

Uncertainty and scope. The three 2D checkpoints span 47.03% to 48.04% for the default-AMG cut. The five 3D combinations are not a full factorial seed grid, and Bentheimer has only three subvolumes. These results concern scalar elliptic diffusion with PCG and a two-level correction, not general multilevel, nonsymmetric, or system-PDE behavior.

5 Conclusion

Learning does not replace multigrid design: evolution strength closes most of the original 2D AMG gap, after which the learned space adds a smaller, consistent improvement. The 3D maze result supports within-family generalization, while large-resolution and real-topology tests expose its limits. The most credible use case is a correction reused for many solves on one operator. Rank-64 cross-operator training on native-resolution real topology without oracle labels remains untested, as does a robust wall-clock win over evolution AMG.

References

- Zhijie Deng, Jiaxin Shi, and Jun Zhu. NeuralEF: Deconstructing kernels by deep neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4976–4992. PMLR, 2022.
- Yali Fink, Ido Ben-Yair, Lars Ruthotto, and Eran Treister. RAPNet: Accelerating algebraic multigrid with learned sparse corrections. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 31125–31145. PMLR, 2026.
- Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. EigenGame: PCA as a Nash equilibrium. In *International Conference on Learning Representations*, 2021.
- 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*, pages 2415–2423. PMLR, 2019.
- 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.
- Alena Kopaničáková and George Em Karniadakis. DeepONet based preconditioning strategies for solving parametric linear systems of equations. *SIAM Journal on Scientific Computing*, 47(1): C151–C181, 2025. doi: 10.1137/24M162861X.
- Alena Kopaničáková, Youngkyu Lee, and George Em Karniadakis. Leveraging operator learning to accelerate convergence of the preconditioned conjugate gradient method. *Machine Learning for Computational Science and Engineering*, 1(2):39, 2025. doi: 10.1007/s44379-025-00039-7.
- Jian Luo, Jie Wang, Hong Wang, Huanshuo Dong, Zijie Geng, Hanzhu Chen, and Yufei Kuang. Neural Krylov iteration for accelerating linear system solving. In *Advances in Neural Information Processing Systems*, volume 37, 2024.
- 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*, pages 6489–6499. PMLR, 2020.
- Jan Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(3):233–241, 1993.
- R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24(2):355–365, 1987. doi: 10.1137/0724027.
- Luke N Olson, Jacob Schroder, and Raymond S Tuminaro. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications*, 17(4):713–733, 2010.
- 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.
- John W Ruge and Klaus Stüben. Algebraic multigrid. In *Multigrid Methods*, pages 73–130. SIAM, 1987.
- Nicole Spillane, Victorita Dolean, Patrice Hauret, Frédéric 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.
- Jok M. Tang, Reinhard Nabben, Cornelis Vuik, and Yogi A. Erlangga. Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *Journal of Scientific Computing*, 39(3):340–370, 2009.
- Ulrich Trottenberg, Cornelis W Oosterlee, and Anton Schuller. *Multigrid*. Academic Press, 2000.

Guide to this supplementary material. The main text (4 pages) stands on its own; the sections below give full detail for readers who want it, organized to mirror the main text’s own structure. App. A backs §1’s RAPNet differentiation; App. B gives the architecture and training details behind Figure 1’s shared recipe (§2); App. C states the reproducibility/provenance mechanism cited throughout; App. D gives the full detail behind the Pillar 1 core result and robustness discussion, including wall-clock nuance, the GenEO-adaptive derivation, the $\cot^2 \theta$ derivation, and per-line 3D-negative-result detail; App. E gives the maze construction and full detail behind the Pillar 2 synthetic result, including oracle-vs-trained methodology, full scale-up tables, Bentheimer segmentation methodology, the retraining control, and the transient-PDE caveat; App. F lists dataset licenses.

A Related-work verification detail

RAPNet’s [Fink et al., 2026] 3D anisotropic diffusion experiment (its Appendix, “3D FEM Anisotropic Diffusion”) uses $\sigma(x, y, z) = R(\theta, \phi) \text{diag}(1, 10^{-4}, 10^{-4}) R(\theta, \phi)^\top$, $R(\theta, \phi) = R_z(\theta) R_y(\phi)$, per-point θ, ϕ aligning the diffusion axis with a prescribed velocity field, a genuine, spatially-varying rotated tensor, ratio 10^4 (higher than this paper’s tested 200–5000 range). Its Table 1 (variables; AGG/SpSA/RAPNet standalone; AGG/SpSA/RAPNet as GMRES preconditioner), verified against the source PDF: 227K: $228 \pm 21/132 \pm 10/202 \pm 17$; 63 $\pm 4/60 \pm 4/59 \pm 4$. 355K: $262 \pm 25/151 \pm 12/231 \pm 20$; 70 $\pm 5/66 \pm 4/65 \pm 4$. 524K: $285 \pm 29/164 \pm 14/251 \pm 24$; 74 $\pm 6/71 \pm 5/69 \pm 5$. Standalone RAPNet beats AGG but loses to SpSA at every size; as a GMRES preconditioner it gives a small, consistent 1–5-iteration edge over both. Differentiation rests on mechanism and baselines (§1), not non-overlapping problem classes.

B Learned coarse space: architecture and training details

This section gives the exact architecture, training schedule, and test-time correction behind Figure 1’s shared recipe (§2).

Architecture. An FNO maps discretized problem fields to m coarse vectors: 16 modes, width 64, 4 layers, unless stated otherwise. Pillar 1 uses 6 channels (K_{xx}, K_{xy}, K_{yy} , one sampled source field, and two coordinate channels); Pillar 2 uses 4 channels (the coefficient field and three coordinate channels). The Pillar 1 source channel is inherited from the data pipeline, but is not the PCG right-hand side used for evaluation and does not enter the Rayleigh loss. We therefore describe the input as problem fields rather than claiming it contains the coefficient alone. The coarse-space size is $m=32$ for Pillar 1 and $m=16$ for Pillar 2.

Training. For the default checkpoints, the only training signal is the unsupervised Rayleigh quotient $\text{tr}(Q^\top D^{-1/2} A D^{-1/2} Q)/m$, D the diagonal of A and Q the QR factor of the (diagonally rescaled) network output: the optimization consumes no eigendecomposition of A or solution label. The inherited Pillar 1 dataset builder does compute and store an unused solution field; neither the training loop nor the loss reads it. This is avoidable data-preparation overhead, not supervision or a requirement of the method. The thin $N \times m$ QR above orthonormalizes the network’s own output and is not a factorization of A . This eigendecomposition-free approach to a spectral subspace parallels Rayleigh-quotient eigenspace learning elsewhere [Gemp et al., 2021, Pfau et al., 2019, Deng et al., 2022] (there, PCA and kernel eigenfunctions, not multigrid coarse spaces). Concretely: 200 (Pillar 1) or 64 (Pillar 2) randomly generated training operators (disjoint seed from the held-out test operators), Adam (lr = 10^{-3} , cosine-annealed) for 80 (Pillar 1) or 250 (Pillar 2) epochs, one operator per optimizer step, gradient norm clipped at 1.0; 3 independently trained seeds per pillar unless stated otherwise (App. C).

Test-time correction. The coarse space Z enters a two-level deflation/additive coarse-grid correction in the balancing Neumann-Neumann family [Nicolaidis, 1987, Mandel, 1993, Tang et al., 2009] (“BNN” throughout), using learned vectors rather than GenEO’s local generalized eigensolves [Spillane et al., 2014]. Unlike classical BNN, we wrap a generic first-level preconditioner (AMG for Pillar 1, GMG3D for Pillar 2) rather than subdomain Dirichlet/Neumann solves.

C Reproducibility

All coefficient fields, sources, and train/test splits are generated by seeded pseudo-random-number generators; every table states the number of operators, seeds, or independent timing sessions it averages over. An automated correctness suite (symmetry, M-matrix property where applicable, SPD, PCG parity against a reference solver, coarse-space/BNN invariants) runs before any milestone, and a separate script re-derives every numeric claim in this paper from its producing result record via a provenance manifest mapping each paper element to the script and record that produced it. An independent verification module additionally recomputes the headline numbers from raw records rather than the frozen summary alone. Every reported percentage is computed from the full-precision internal iteration counts and timings, not from the 1-decimal values a table displays; recomputing a percentage from a table’s own displayed values instead of the underlying record can therefore differ from the reported figure by a few hundredths of a point, a rounding artifact, not a discrepancy. All GPU experiments pin a single device in a fixed PCI-bus order; where a quantity varies session to session (machine load, thermal state, since this project’s experiments frequently ran on a shared, contended machine, checked to be otherwise idle before every timed run), we report the number of independent sessions and the resulting spread rather than a single uncontrolled measurement. Every result record is frozen; a later correction changes the paper’s text or the provenance manifest, never the JSON itself.

D Pillar 1 supplementary material

Table 1: Core result, full per-seed breakdown (Pillar 1 core result), 12 held-out operators, $n=256$. Cuts are relative to each method’s own cold baseline (AMG cold 106.5, GMG cold 46.7 throughout; the same checkpoint wraps either).

Seed	AMG cold	AMG+learned	Cut	GMG cold	GMG+learned	Cut
0	106.5	55.4	47.97%	46.7	43.1	7.68%
1	106.5	55.3	48.04%	46.7	43.2	7.50%
2	106.5	56.4	47.03%	46.7	43.5	6.79%

Table 2: The 3D negative result (Pillar 1 3D scope): cold AMG/GMG iterations, three independent lines of evidence.

Evidence	AMG	GMG	Ratio
Synthetic sweep (10 operators, $n=24$)	10.3	37.1	0.28×
Mendeley real RVE tensors (10 samples)	5.5	11.2	0.49×
Norne real field data (12^3 sub-block)	5	16	0.31×
<i>2D result, for reference</i>	106.5	46.7	2.28×

Per-operator variance behind the headline cut. The main text’s 48.0% figure is a mean over 12 held-out test operators; re-running the identical benchmark (same checkpoint, same 12 operators, same per-operator RHS seeds) with per-operator iteration counts saved gives a mean of 47.98% (reproducing the original to within rounding, confirming determinism) with per-operator std 1.55 percentage points, range [44.95%, 50.50%], and a 95% CI half-width of 0.98 points (Student’s t with 11 degrees of freedom, appropriate for this small a sample; a normal-approximation z multiplier would understate this by about 12%). The effect is consistent instance-by-instance, not an artifact of averaging over a heterogeneous set where some operators see a much larger cut than others.

AMG’s own stronger classical fix. Every “AMG cold” value in the original headline used PyAMG’s default classical strength measure. We also evaluated symmetric, algebraic-distance, and evolution strength on the same 12 operators. Symmetric strength is worse (136.8 iterations). Algebraic distance is also worse among the 9 completed operators (140.4 iterations); 2 operators time out and 1 raises a numerical exception, so we do not treat its incomplete mean as directly comparable. Evolution strength [Olson et al., 2010] is the decisive classical result: it reduces the mean from 106.5 to 32.6 without learning. The unchanged learned checkpoint still reduces evolution AMG from 32.6 to 26.6, a further 18.4%, with a positive reduction on every operator. Thus the learned space is complementary,

Table 3: Full detail behind the Pillar 1 robustness summary. All rows: $n=256$, $\theta \in [16, 25]^\circ$, ratio $\in [200, 5000]$, 12 held-out test operators, unless a row states otherwise.

Method	Iters	Cut vs. AMG cold	Setup+solve
AMG cold (Ruge-Stüben, default strength)	106.5	n/a	see wall-clock note below
Learned + AMG (default strength)	55.4	48.0%	1.06s [†]
Evolution-strength AMG (no learning)	32.6	69.4%	0.74s [§]
Learned + evolution-strength AMG	26.6	18.4% vs. evolution AMG	n/a [§]
Smoothed-aggregation AMG cold	82.6	22.5% vs. RS	n/a
Learned + SA-AMG	68.8	16.6% vs. SA cold	n/a
Aggregation ($k_x=4$, $k_y=8$) + AMG	99.2	6.9%	1.82s
Bilinear ($k_x=4$, $k_y=8$) + AMG	95.7	10.2%	1.75s
GenEO (2×2 subd., 8/subd., dim 32) + AMG	75.6	29.0%	9.72s
GenEO (4×4 subd., 2/subd., dim 32) + AMG	97.1	8.8%	11.48s
GenEO (4×4 subd., 4/subd., dim 64) + AMG	85.2	20.0%	7.25s
GenEO (8×8 subd., 2/subd., dim 128) + AMG	84.2	21.0%	9.89s
<i>GenEO, adaptive threshold (own mechanism, not a fixed count)[‡]:</i>			
GenEO ($\tau=20$, dim 113.7 mean) + AMG	49.2	53.8%	6.37s
GenEO ($\tau=10$, dim 208.3 mean) + AMG	37.4	64.9%	7.67s
GenEO ($\tau=5$, dim 324.5 mean) + AMG	26.5	75.1%	8.92s
<i>Resolution (AMG cold / GMG cold / ratio):</i>			
$n=128$	54.3 / 35.4	ratio 1.53×	n/a
$n=256$	106.5 / 46.7	ratio 2.28×	n/a
$n=512$	209.2 / 61.8	ratio 3.38×	n/a
<i>PyAMG cross-validation, $n=128$ (AMG / GMG, ratio):</i>			
FD stencil	53.1 / 33.4	ratio 1.59×	n/a
FE stencil	39.8 / 43.8	ratio 0.91×	n/a

but the relevant increment over a strong classical baseline is 18.4%, not the 48.0% measured against the default.

§Timing scope for evolution strength. The 0.74s total is the mean evolution-AMG setup plus solve time from the dedicated strength-measure sweep (0.36s plus 0.38s). The original follow-up combining evolution AMG with the learned correction recorded iteration counts under a per-configuration timeout, not comparable end-to-end timing. The separate paired timing in Table 4 uses a retrained operator-only checkpoint and its own right-hand-side protocol; it must not be treated as timing for this original follow-up.

Why Learned + SA-AMG ends up worse in absolute iterations than Learned + AMG. Z is trained purely against A 's own Rayleigh quotient, with no smoother in the loop, so it targets whatever near-null-space content is hardest for a generic smoother. Ruge-Stüben AMG's strength-of-connection heuristic struggles badly here (the whole Pillar-1 failure mode); SA-AMG's smoothed prolongation targets the same kind of structure, so its own hierarchy may already resolve much of what Z targets. Tested directly on the identical 12-operator set: the fraction of Z 's own energy surviving one V-cycle of each smoother's error propagator is 96.7% for Ruge-Stüben (barely touches it) but only 79.5% for SA-AMG (already removing about a fifth), consistent across every operator (0.813–0.831 ratio, no exceptions) – explaining both the smaller marginal cut and why SA-AMG's already-lower cold count leaves less absolute room for a correction overlapping with what it already resolves.

†The 1.06s figure, and why it should not be read as a stable win over cold AMG. Table 3's Learned + AMG setup+solve time (1.06s, headline architecture) is the total time recorded in the same session as the adjacent GenEO rows, reported there to support the Pillar 1 setup-cost comparisons against both GenEO's fixed configurations (7.2–11.5s, 6.8–10.8×) and its adaptive-threshold mode (6.4–8.9s, 6.0–8.4×), which reuses this same session's figure since the adaptive sweep records no separate learned-method timing of its own: the gap mostly reflects AMG's own cost advantage over GenEO's per-subdomain eigensolves, not a distinctive efficiency of the learned correction itself, since cold AMG's own total time in that same session (1.23s) is comparable to the learned method's. We flag a further nuance a reader would otherwise have to reconstruct: across the 3 dedicated timing sessions reported directly below, cold AMG's own total time is stable (1.26–1.27s), but the learned method's total time is consistently higher there (1.29–1.30s) than in this single GenEO-comparison session (1.06s), an 18–20% spread we attribute to FNO-inference/QR-overhead timing noise rather than a stable difference in the method itself. The 1.06s figure is therefore a real, single-session

measurement, valid for the GenEO-cost comparison it supports, but should not be read as evidence that Learned+AMG beats cold AMG’s own wall-clock: the canonical, multi-session answer to that separate question, using the identical architecture, is the consistent small loss reported next.

‡**GenEO’s adaptive-threshold mode, and what it actually shows.** The four fixed-configuration rows above sweep GenEO over a subdomain-count/eigenvectors-per-subdomain grid, not GenEO’s own defining selection rule: per subdomain, keep every eigenvector of the generalized eigenproblem $D_j A_{jj} D_j v = \mu A_j^{\text{Neumann}} v$ whose eigenvalue μ exceeds a threshold τ , an adaptive count rather than a fixed one (Spillane et al., 2014). We implement this directly (per subdomain, up to 28 candidate eigenvectors via shift-invert, keeping those with $\mu \geq \tau$) and sweep three τ values on the same 4×4 -subdomain partition and 12 test operators as the fixed-config rows. The result reverses the earlier comparison: every τ tested beats our own 48.0%, up to 75.1% at the most generous threshold, but only at $3.6\text{--}10.1 \times$ our own dimension (113.7–324.5 vs. our 32) and higher setup cost. The τ values themselves are an empirical sweep of the selection rule (calibrated against this benchmark’s own observed μ distribution, roughly [3.5, near-infinite for local near-nullspace modes]), not a reproduction of Spillane’s own condition-number-bound constant, which needs subdomain-diameter/overlap parameters this benchmark’s geometry does not have pinned down. Two further faithfulness caveats, checked directly against the source: our $D_j A_{jj} D_j$ right-hand side uses the partition-of-unity weighting applied to the *full* local subdomain matrix as a practical stand-in for Spillane et al.’s own right-hand bilinear form, which is restricted to the overlap zone specifically (their Definition 3.7); and our selection keeps the largest $\mu = 1/\lambda$ above τ , the reciprocal of Spillane’s own rule of keeping the smallest λ below a geometry-tied threshold (their Corollary 3.23): the same modes, on a reciprocal scale, but not the identical construction. The honest reading is a real accuracy-vs-budget trade-off against a close, not exact, implementation of GenEO’s own adaptive mechanism, not a case where either method dominates the other on both axes.

Direct comparison against the closest learned prior work’s own training objective. [Luo et al., 2024] (NeurKIIt) is, by mechanism, the closest learned coarse-space method to this paper (§1): also an FNO predicting a low-dimensional subspace conditioned via QR. Its training objective differs fundamentally, though: a projection loss supervised against each operator’s true bottom- K eigenvectors (its Eq. 10, verified against the source PDF, $l(\hat{K}, S) = \sum_i \|s_i - Q Q^\top s_i\|$), obtained via an eigensolve on every training instance, rather than this paper’s own unsupervised, eigensolve-free Rayleigh-quotient objective. We reimplement this training objective directly and train it on the identical architecture (in_channels 6, out_channels $K=32$, modes 16, width 64, 4 layers), the identical 200-operator training set and 12-operator held-out test set (same seeds as the headline result), for the same 80 epochs with Adam and a cosine schedule, differing only in the loss function, so any gap reflects the training objective, not a confound in architecture or data. Training loss converges smoothly and monotonically ($0.641 \rightarrow 0.017$ over 80 epochs), confirming a fair, non-degenerate comparison rather than an undertrained baseline. The result: 48.9% AMG cut (54.4 iterations) and 8.2% GMG cut (42.8 iterations), statistically indistinguishable from this paper’s own 47.97%/7.68% (seed 0) and well within the ± 1.55 -point per-operator std already measured above. The two objectives reach the same accuracy by a different route: the eigensolve-supervised version spends 679.14s computing bottom- K eigenvectors for the 200 training operators plus 40.88s for the 12 test operators, a substantial cost this paper’s own unsupervised objective never incurs, at training or at deployment. This is, to our knowledge, the first empirical (not only qualitative) head-to-head between this paper’s objective and the closest learned alternative’s own.

Wall-clock sensitivity to architecture size (canonical numbers for Table 3’s AMG cold / Learned + AMG rows, vs. cold AMG specifically). The end-to-end win reported in the main-text timing discussion (+9.6%) uses a smaller FNO (modes 8, width 16, 2 layers) plus a triangular-solve BNN apply. Measured with the original, larger architecture (modes 16, width 64, 4 layers, otherwise identical), the end-to-end wall-clock result is consistently negative across 3 independent timing sessions (−2.2%, −2.4%, −3.2%, i.e. a real, repeated loss, not noise scattered around zero): AMG’s higher per-iteration cost in this regime favors the learned correction in principle, but inference and QR overhead at that architecture size are large enough to erase it and then some. The smaller architecture’s +9.6% is therefore not a different measurement of the same quantity but a genuinely different, and better, architecture choice; it is the number the main text cites, and the only one that should be read as the wall-clock performance of “Learned + AMG”.

Paired end-to-end timing with evolution-strength AMG. This follow-up is separate from the original checkpoint comparison in Figure 2a. It uses a retrained operator-only FNO checkpoint, 12

held-out operators, the same 64 right-hand sides per operator, and three alternating-order repeats on each of two machines (each with a Quadro RTX 8000 and a four-thread CPU cap). Each method pays its own AMG setup once; the learned path additionally pays FNO inference, diagonal scaling, QR, and balancing-correction setup once. Both paths then solve the same right-hand sides sequentially to a relative residual of 10^{-8} . Table 4 gives the geometric mean of the 12 operator-level ratios of total cold time to total learned time. Its intervals use operators, not timing repeats, as independent units; the two machines are not pooled.

Table 4: Evolution AMG versus learned plus evolution AMG, including setup and cumulative solve time. Ratios above one favor learning; brackets give descriptive 95% operator-level intervals. This timing checkpoint differs from the main iteration-comparison checkpoint.

Right-hand sides	Machine 1 ratio [interval]	Machine 2 ratio [interval]
1	0.739 [0.670, 0.814]	0.872 [0.824, 0.922]
8	0.932 [0.898, 0.967]	0.994 [0.971, 1.017]
32	0.997 [0.975, 1.021]	1.024 [1.005, 1.043]
64	1.010 [0.991, 1.030]	1.030 [1.012, 1.048]

The first machine is slower for one solve and approximately tied after 64 solves; the second shows a small gain after many solves. Neither result establishes a single-solve or original-checkpoint speedup.

Does the learned coarse space also help GMG, not just AMG? We checked directly, since AMG is not the solver this paper’s own motivation identifies as commonly deployed. Wrapping GMG instead of AMG with the identical learned coarse space (no retraining) gives a real, consistent iteration cut across every variant already tested: 7.68% (headline seed, $46.7 \rightarrow 43.1$), 7.50%/6.79% (seeds 1/2), 6.96% (tiny architecture), 7.32% (wide band), 6.07% (tiny+wide): six for six, always positive, never a lucky outlier. On wall-clock, though, this reverses hard: -61% to -113% across the same sessions (i.e. $1.6\text{--}2.1\times$ slower than cold GMG, not faster). The mechanism is the mirror image of the AMG case: GMG’s per-iteration cost here is cheap enough that the fixed FNO-inference and QR overhead, which the AMG comparison’s much higher per-iteration cost can absorb, dominates and then some when wrapping GMG instead. In summary: Learned+GMG beats cold GMG on iterations, consistently, but loses on wall-clock, for a mechanistically clear reason, exactly the inverse of why Learned+AMG’s iteration win does translate into a wall-clock win for the smaller architecture.

The $\cot^2 \theta$ line-smoother cap, derived. For $K = R(\theta) \text{diag}(k_1, k_2) R(\theta)^\top$, the axis-aligned diagonal entries are $K_{xx} = k_1 \cos^2 \theta + k_2 \sin^2 \theta$ and $K_{yy} = k_1 \sin^2 \theta + k_2 \cos^2 \theta$. An axis-aligned line smoother (classical-controls discussion) only ever sees the ratio K_{xx}/K_{yy} , not the tensor’s own true ratio k_1/k_2 ; dividing through by k_1 and taking $k_1/k_2 \rightarrow \infty$ (the high-anisotropy regime this paper tests, ratio up to 5000) gives $K_{xx}/K_{yy} \rightarrow \cot^2 \theta$, independent of how large the true ratio actually is. For $\theta \in [16, 25]^\circ$ this caps the effective ratio a line smoother can exploit at roughly 4.6–12.2 ($\cot^2 25^\circ$ to $\cot^2 16^\circ$), regardless of whether the tensor’s true ratio is 200 or 5000, an effective regime mild enough that pointwise Jacobi already handles it about as well as a line smoother can, which is why line relaxation adds overhead without adding convergence benefit here.

Does semi-coarsening itself, applied directly to the oblique tensor, fare any better than the line smoother? Applying GMGSemiCoarsen directly to the oblique tensor (picking coarsen_axis via whichever of K_{xx}/K_{yy} is larger, the same rule the line smoother uses for its own axis) fails the same way: 35.7% worse on average ($46.67 \rightarrow 63.33$ mean iterations) on the identical 12 held-out operators, worse on all 12 of 12 individually. Semi-coarsening’s axis choice is a discrete, per-grid-line decision; once the strong direction is oblique, no single axis choice captures it, confirming the fix needs genuine rotation-awareness, not merely a different axis heuristic.

3D negative result, per-line-of-evidence detail. Synthetic sweep (10 operators, $\alpha, \beta, \gamma \sim U(0, \pi)$, ratio $\in [200, 5000]$, $n=24$): AMG 10.3 vs GMG3D 37.1 ($0.28\times$). Real Mendeley RVE tensors (10 samples, same $n=24$): AMG 5.5 vs GMG3D 11.2 ($0.49\times$). Real Norne field data (one fully-active 12^3 sub-block, origin (5, 13, 4)): AMG 5 vs GMG3D 16 ($0.31\times$).

Extended resolution sweep. The negative result above fixes $n=24$; the 2D headline’s own AMG/GMG ratio widens with resolution, so a comparable, unresolved 3D trend could in principle still cross 1.0 at larger, untested n . We swept $n \in \{32, 48, 64, 96, 128\}$ (32K to 2.1M DOF) at the hardest angle configuration the grid search found ($\theta \approx 21^\circ$, ratio 5000; contrast has no effect on the failure mode within $10^3\text{--}10^6$).

Table 5: Extended 3D resolution sweep, rotated anisotropic tensor, $\theta \approx 21^\circ$, ratio 5000.

n	AMG cold	GMG3D cold	Ratio
32	15	83	0.18×
48	22	119	0.18×
64	28	151	0.19×
96	40	204	0.20×
128	53	244	0.22×

The ratio stays far below 1.0 at every size, ruling out “just needs more resolution.” A separate finding: GMG3D’s own iteration count grows fast with resolution ($83 \rightarrow 244$) while AMG’s grows slowly ($15 \rightarrow 53$), a real, growing weakness of this paper’s own production-default 3D solver on rotated tensors. We tested the same oracle-correction mechanism that rescued the Bentheimer result directly on this weakness, at $n=64$ (cold GMG3D = 148.3, cold AMG = 30.6), sweeping $m \in \{16, 32, 64\}$: completely flat and null at every budget ($-2.5\%/-1.8\%/-1.8\%$). Unlike Bentheimer, enlarging the budget does nothing here, consistent with the line-smoother finding above that GMG3D’s weakness on rotated tensors likely comes from the smoother struggling broadly along the tilted direction, a class of error too broad for any low-rank deflation correction to patch.

A genuinely oblique 3D rotation: does richer z -axis connectivity change the ordering? Every 3D test above fixes $\beta=\gamma=0$, a purely in-plane rotation for which $K_{xz}=K_{yz}=0$ exactly, so the z -axis carries only isotropic coupling (the original headline synthetic sweep, Table 2’s first row, is not subject to this limitation: it already draws all three Euler angles independently, and none of its 10 draws land near the window found below). Sweeping $\beta \in \{0, 15, 30, 45\}^\circ$ at fixed $\alpha=21^\circ$, ratio 5000, $n=64$ shows the AMG/GMG3D ratio rises sharply once $\beta \neq 0$ engages real out-of-plane coupling ($0.19\times \rightarrow 0.63\times \rightarrow 1.00\times \rightarrow 0.72\times$), reaching exact parity at $\beta=30^\circ$. A fine local grid search around this peak ($\alpha \in \{18, 20, 21, 22, 24\}^\circ \times \beta \in \{20, 25, 30, 35, 40\}^\circ$) finds four configurations where the ordering fully reverses; the strongest, $\alpha=22^\circ, \beta=25^\circ$ (AMG 31 vs. GMG3D 28, ratio 1.107 $\times$), is robust across all 10 independent right-hand-side seeds with zero variance. An oracle correction (subspace iteration on the AMG error propagator, 200 iterations for proper convergence at this scale) exploits this window, cutting AMG’s cold 31.0 iterations to 28.0/25.0/21.4 at $m=16/32/64$ (Table 6): a genuine, growing cut of 9.7%/19.4%/31.0%. This is a real, if narrow, positive 3D result: the average 3D negative result does not mean the failure mode, or the correction, are absent everywhere in 3D, mirroring the 2D result’s own narrow $[16, 25]^\circ$ hard band.

Table 6: Oracle coarse-space correction at the 3D crossover point ($\alpha=22^\circ, \beta=25^\circ$, ratio 5000, $n=64$, AMG cold mean 31.0).

m	Oracle iters (mean)	Cut vs. AMG cold
16	28.0	9.7%
32	25.0	19.4%
64	21.4	31.0%

A real-data test for the positive result itself. Every real-data check for Pillar 1 above asks whether the classical failure mode reproduces on real tensors, never whether the correction does anything useful on one. For a genuinely SPD 3×3 real permeability tensor, every principal 2×2 submatrix (an axis-plane projection) is itself SPD (Sylvester’s criterion), giving real 2D anisotropic tensors with their own genuine (θ, ratio) , computed via eigendecomposition. Across all 629 projections from the 214 non-degenerate Mendeley tensors, the real ratio tops out at 85.5 (median 2.33), far below the training floor of 100: zero fall in either training band. Testing the 25 highest-ratio real projections plus 15 stratified across the rest (40 total, zero-shot): AMG never underperforms GMG on any real projection, including at the highest available ratio, so the failure mode simply does not manifest at any anisotropy level this real dataset reaches (dose-response correlation between real ratio and the AMG/GMG ratio: -0.21 , not the positive trend a worsening failure mode would predict). The correction still cuts iterations modestly even here (4.3% mean overall, 5.2% on the top-ratio subset). This converts “rests on synthetic data alone” into an honestly characterized real test.

Confirming the batching-shrinkage mechanism. `block_pcg` runs `nrhs` independent CG recurrences in lockstep (uncoupled math, only the matvecs/preconditioner applies are batched) and exits once every column has converged, so its returned iteration count should be exactly $\max_j(\text{column } j\text{'s own iteration count})$. Verified directly: solving the same pool of 32 random right-hand sides against one hard operator ($\theta=18.4^\circ$, ratio 1633) both via `block_pcg` and an independent per-column loop, at $\text{nrhs} \in \{2, 4, 8, 16, 32\}$, gives an *exact* match at every `nrhs` (57, 57, 57, 57, 58). Every cold baseline instead reports the *mean* of the same counts, which stays flat (107.0, 106.8, 107.5, 107.3, 107.5). As `nrhs` grows this compares a growing max (a non-decreasing order statistic) against a flat mean: a reporting-convention effect, not a cost-scaling problem in the batched correction itself.

E Pillar 2 supplementary material

Maze-family construction. Thin near-impermeable barrier planes (contrast 10^4 against a background of 1) at randomized positions, each pierced by one randomized circular gap forcing a serpentine 3D flow path; the “sieve” variant replaces the single gap with many small randomized gaps on a jittered grid (total open area per plane $\in [0.10, 0.40]$ of the plane’s area), isolating whether the mechanism needs one forced corridor or generalizes to distributed parallel channels.

Oracle vs. trained-network methodology. The oracle coarse space (Pillar 2 synthetic result) is computed by subspace iteration directly on the GMG3D error propagator for one fixed operator, not a trained/generalizable network, since none exists for this family until the training result. The trained network’s deployed coarse space is a single forward pass through the checkpoint plus QR, *not* the raw network output: the Rayleigh-quotient training loss scores $D^{-1/2}$ -rescaled vectors, and using the raw (unrescaled) network output directly as the deflation subspace Z silently deflates the wrong subspace whenever D is non-constant, which it always is here (contrast 10^4 spans 4 orders of magnitude in the diagonal). This was caught during evaluation-script development: the unscaled version produced no iteration reduction at all despite the network’s Rayleigh loss generalizing correctly to held-out instances.

Table 7: Multi-seed hardening (Pillar 2 synthetic result): 5 (training seed, evaluation seed) combinations, $n=48$, $N=110,592$, 6 held-out maze instances per combination. Train seed 1 is the only one tested against more than one evaluation stream (e7/e8/e9); seeds 2 and 3 are each tested against only the shared e7 stream.

Combination	Train seed	Eval seed	GMG3D cold	Cut
s1_e7	1	7	27.0	16.1%
s2_e7	2	7	27.0	17.3%
s3_e7	3	7	27.0	19.2%
s1_e8	1	8	23.9	22.7%
s1_e9	1	9	23.1	25.6%

Was the original 16.1–25.6% cut data-starved? We retrained the s1_e7 configuration (identical train/eval seed, architecture, epochs, protocol) with the training set enlarged from 64 to 200 instances. The cut rises from 16.1% to 24.1% ($27.0 \rightarrow 20.5$ mean corrected iterations against the same 27.0 cold baseline), close to the best of the five seed combinations above (25.6%) from a training-data change alone, with no architecture or seed change: the original checkpoint’s cut was under-trained relative to what this family supports, not a ceiling on the mechanism itself.

Does the synthetic maze family’s own oracle ceiling grow with budget, the way Benthaimer’s real-data oracle did? Table 9 only ever tests the oracle at $m=16$; the Benthaimer result finds the real-data oracle ceiling grows substantially with budget (-6.6% at $m=16$ to $+81.5\%$ at $m=64$). We swept $m \in \{16, 32, 64, 128\}$ on the same fixed representative synthetic operator ($n=48$, GMG3D cold mean 28.6) used by Table 9’s own $n=48$ row. It does, though far lower than Benthaimer’s disordered real topology: $37.1\% \rightarrow 51.1\% \rightarrow 58.1\% \rightarrow 61.6\%$ (Table 8), diminishing but still positive returns through $m=128$: even the synthetic family’s own upper bound has real, if shrinking, room to grow with budget alone.

Reading Table 10 against Table 9. The oracle table’s AMG wall-clock crossover (speedup $> 1 \times$ at 5 of 6 sizes) does not survive with the real network at any tested size. Up through $n=192$ the gap closes steadily from $n=64$ onward, reaching $0.96\times$; extending the sweep to $n=224, 256$ (16.8M

Table 8: Oracle budget sweep on the synthetic maze family’s own fixed representative operator ($n=48$, $N=110,592$, contrast 10^4 , GMG3D cold mean 28.6).

m	Oracle iters	Cut
16	18	37.1%
32	14	51.1%
64	12	58.1%
128	11	61.6%

Table 9: Scale-up sweep, full per-size detail (Pillar 2 synthetic result): 16 right-hand sides, $n=48$ to 192 (110K to 7.1M DOF). This sweep uses the oracle coarse space, not the trained network of Table 7, which is why this table’s own $n=48$ row (GMG3D cold 28.6, cut 37.1%) differs from Table 7’s $n=48$ rows (GMG3D cold 23.1–27.0, cut 16.1–25.6%): different held-out test instances (single fixed operator vs. 6 held-out draws) and a different, upper-bound coarse-space construction, not the same quantity measured twice. Iteration cut is oracle-corrected GMG3D batched vs. GMG3D cold; speedup is the batched correction’s per-right-hand-side wall-clock vs. AMG cold’s.

n	DOF	GMG3D cold iters	Iter cut	Speedup vs. AMG cold
48	110,592	28.6	37.1%	$1.31\times$
64	262,144	24.8	19.4%	$0.93\times$
96	884,736	19.9	14.5%	$1.03\times$
128	2,097,152	22.4	24.0%	$1.31\times$
160	4,096,000	22.7	11.8%	$1.22\times$
192	7,077,888	22.6	15.8%	$1.47\times$

DOF) tests directly whether that trend continues to an actual crossover. It does not: speedup reverses, falling to $0.88\times$ and $0.53\times$, further from parity than at $n=160$. The iteration-cut column explains why: unlike the oracle’s cut, which stays in an 11.8–37.1% band with no clear trend, the trained network’s cut falls toward zero through $n=192$ and then goes sharply negative (-15.8% at $n=256$, its worst result at any tested size). Since the checkpoint was fit only at $n=48$, this is consistent with a genuine resolution-generalization limitation: by $n=256$ ($5.3\times$ the training resolution) the correction has become actively harmful, not merely ineffective. The oracle-only crossover does not extend to the deployable network within a range a real 3D deployment could plausibly reach, and pushing further does not recover it.

Bentheimer segmentation methodology. The raw 75^3 -voxel, 16-bit grayscale subvolume is downsampled to the network’s trained 48^3 resolution (`scipy.ndimage.zoom`, order 1) before segmentation, so the test measures real-topology generalization only, not conflated with a resolution-transfer question. Otsu thresholding (implemented directly, no added dependency) splits the downsampled volume into pore and solid; the threshold direction (which grayscale extreme is pore vs. solid) is chosen so the resulting pore fraction falls on the same side as the dataset’s own documented core-averaged porosity (0.203–0.223), consistent in direction with X-ray attenuation physics (solid mineral attenuates more than pore space, so low intensity = pore). This confirms the threshold direction is physically sensible, not that the resulting porosity is quantitatively close to it: the segmented pore fractions for all 3 subvolumes (0.262, 0.268, 0.288) are 18–29% above the documented range’s own upper bound, a magnitude mismatch we read as further evidence for, not against, the downsampling confound below, rather than as a passed check.

The $75^3 \rightarrow 48^3$ downsampling (linear interpolation of a near-binary, 10^4 -contrast field, then Otsu segmentation) could itself distort the real topology before the network ever sees it, confounding “training family too narrow” with “resampling artifact” as the explanation for the negative-transfer result (Bentheimer result); we did not test alternative downsampling schemes (e.g. majority-vote instead of linear interpolation) to isolate the two explanations.

Retraining on a topologically closer synthetic family does not fix the negative transfer, at the original budget. To test the “training family too narrow” explanation directly, we trained a second network from scratch on a new synthetic family designed to resemble real disordered pore geometry: a random packing of overlapping low-permeability spheres, porosity randomized per training sample near Bentheimer’s own documented range (0.74–0.84 solid fraction, i.e. 0.16–0.26 porosity), identical architecture, training set size, and Rayleigh-quotient objective as every other 3D network in this paper

Table 10: The same scale-up sweep repeated with the actual trained, generalizing network, trained only at $n=48$ and evaluated at every size with no retraining, in place of the oracle, the direct counterpart to Table 9. GMG3D cold iteration counts differ substantially from that table at matched n (e.g. 39.2 vs. 28.6 at $n=48$), not merely slightly: this sweep draws its maze instance from a different random seed (the held-out stream used elsewhere for trained-network evaluation, consistent with Table 7’s convention) than Table 9’s own oracle sweep, so a harder or easier random instance at the same n , not an inconsistency in the solver or assembly, is the source of the gap. Iteration cut is trained-network-corrected GMG3D batched vs. GMG3D cold; speedup is vs. AMG cold, identical convention to Table 9.

n	DOF	GMG3D cold iters	Iter cut	Speedup vs. AMG cold
48	110,592	39.2	10.7%	$0.51\times$
64	262,144	35.6	-1.2%	$0.44\times$
96	884,736	35.1	8.7%	$0.63\times$
128	2,097,152	34.1	3.1%	$0.68\times$
160	4,096,000	35.6	1.6%	$0.72\times$
192	7,077,888	34.0	0.0%	$0.96\times$
224	11,239,424	37.4	1.2%	$0.88\times$
256	16,777,216	42.3	-15.8%	$0.53\times$

(training loss converges cleanly, $0.097 \rightarrow 0.0018$ over 250 epochs, confirming a non-degenerate fit). Evaluated on the identical 3 Bentheimer subvolumes with the identical harness, the result is unchanged: also 14.6% *worse* ($111.4 \rightarrow 127.7$ mean iterations), the same cut to within rounding as the maze-trained checkpoint, though from a different per-instance iteration distribution (134/133/116 vs. 137/133/ 113, summing to the same total by coincidence, not a duplicated run). This rules out training-family narrowness as the explanation at $m=16$: a network trained on a family deliberately closer in spirit to the real topology transfers no better than one trained on a structurally simpler single-corridor family.

A larger rank reveals mechanism headroom, but does not by itself fix learning. An oracle coarse space (subspace iteration on the GMG3D error propagator, computed directly per operator, no learning involved) removes the network entirely and asks whether the mechanism helps on real topology at any budget. Table 11 sweeps m on one representative subvolume: -6.6% at $m=16$ (also negative, like the trained network), but $+64.7\%$ at $m=32$ and $+81.5\%$ at $m=64$, with diminishing returns by $m=128$ ($+83.2\%$). Real Bentheimer topology has *more* exploitable structure for this mechanism than the synthetic training family, whose own oracle ceiling is only $\approx 37\%$.

Table 11: Oracle coarse-space dimension sweep on one representative Bentheimer subvolume (Core1_Subvol1, cold GMG3D = 119.1 iters).

m	Iters	Cut	Setup+solve
16	127	-6.6%	35.2s + 242.6s
32	42	$+64.7\%$	86.9s + 75.3s
64	22	$+81.5\%$	223.4s + 39.5s
128	20	$+83.2\%$	505.4s + 40.9s

Retraining the granular network at $m=64$ does not close this on its own: -13.4% , essentially unchanged. Increasing rank alone is not enough for cross-family generalization. A single-instance diagnostic on Core1_Subvol1, warmstarted from the $m=64$ checkpoint, is consistent with difficulty training under the unsupervised Rayleigh-quotient objective: over 5000 steps of direct per-instance training under that objective, the cut plateaus at $+17.7\%$ from step 2000 onward. Supervised distillation toward that same operator’s own oracle subspace instead (minimizing $1 - \|Q_{\text{net}}^\top Q_{\text{oracle}}\|_F^2/m$) reaches $+79.9\%$ and 99% subspace overlap with the oracle by step 5000. This shows representational capacity under per-operator supervision, but does not isolate whether the unsupervised plateau comes from its optimization landscape, objective mismatch with the wrapped GMG preconditioner, or both.

The distilled network reaches a mean $+75.5\%$ cut, 95% of the oracle’s own $+79.2\%$ ceiling, across all 3 real subvolumes: the headline Bentheimer result cited in the main-text real-topology discussion. Stated plainly: this is a win relative to GMG3D cold, not an absolute one. The distilled network’s own mean iteration count (27.3) is $2.1\times$ higher than plain, untrained AMG’s mean on the identical data (12.8), and even the oracle ceiling never reaches AMG cold; a practitioner facing this exact topology,

Table 12: Distillation toward the per-operator oracle subspace, confirmed across all 3 real Bentheimer subvolumes, $m=64$, warmstarted from the granular $m=64$ checkpoint, 2000 steps each.

Subvolume	AMG cold	GMG3D cold	Oracle cut	Distilled cut	Distilled iters
Core1_Subvol1	13.6	119.1	+81.5%	+78.2%	26
Core1_Subvol2	12.5	118.1	+78.8%	+72.0%	33
Core2_Subvol1	12.3	97.2	+77.4%	+76.3%	23
<i>Mean</i>	12.8		+79.2%	+75.5%	27.3

unconstrained to use GMG3D, does strictly better by switching to AMG. The result is genuine on its own terms (recovering GMG3D from actively harmful to usable, from an unsupervised, previously-negative starting point), not evidence this pipeline is the best available classical choice for this data. It does not, on its own, rule out the downsampling confound above, since every checkpoint here is still evaluated on the same downsampled volumes; what it establishes is that a properly-budgeted, properly-optimized coarse space captures most of the correction that topology admits.

Transient-PDE demonstration, framing caveat in full. Implicit (backward Euler) time-stepping solves the same linear system every step, $(M/\text{dt} + A)u^{n+1} = M/\text{dt} u^n + f^{n+1}$, with only the right-hand side changing. Cold GMG runs a genuine sequential simulation (each step’s right-hand side depends on the previous step’s own converged solution) and records the resulting right-hand-side trajectory; that recorded trajectory is then handed, in full, to the batched learned-coarse-space solver as a single block. This is an offline replay or re-solve of an already-known trajectory: the forcing schedule alone would not make future right-hand sides available because they depend on earlier solutions. It does not demonstrate online time stepping with each new state fed forward. Cold AMG runs its own independent sequential simulation for comparison; all solvers converge to tolerance, so the independently-chained trajectories agree with the recorded one within solver tolerance.

Table 13: Offline transient-trajectory replay, with full per-step-count detail (main-text offline replay paragraph). GMG here means GMG3D. Timing combines the learned correction and block batching and cannot isolate their effects.

Steps	GMG cold iters	Learned+GMG batched iters	Iter cut	Wall-clock vs. GMG cold
8	38.1	32	16.1%	−28.0%
16	38.6	33	14.4%	+19.2%
32	38.8	33	14.9%	+34.5%
64	39.7	36	9.2%	+30.1%

Is there an untested modification that improves either pillar? We checked two concrete candidates directly. *Pillar 1*: does the headline 48.0% cut have headroom against a genuine oracle, the way Pillar 2 and Bentheimer both did? A first attempt at this project’s usual 25-subspace-iteration convention gave an implausibly low 15.9% cut; a convergence check found the oracle still climbing steeply through 100 iterations (40.2%), plateauing only by 200 (47.7%). The full 12-operator sweep at 200 iterations gives a mean oracle cut of 48.45% (std 1.06 points), a 0.45-point gap against the trained network, well within existing seed variance: no meaningful headroom here. *Pillar 2*: does the Bentheimer distillation fix generalize to the maze family’s own gap (oracle $\approx$ 37% vs. trained network’s 20.2%)? Unlike Bentheimer’s transductive per-instance fit, we tested a genuine generalizing distillation: offline oracle targets for all 64 training instances, network trained to match each instance’s own target. Training converged cleanly (93.5% mean subspace overlap over 250 epochs), but the evaluated cut (14.9%) is not better than the existing 16.1%/20.2% result. Transductive and generalizing distillation are different optimization problems, and 64 instances’ own oracle subspaces evidently do not share enough common structure for a network to learn a transferable mapping.

F Data and code availability

- Rock permeability tensors: Xiong et al., Mendeley Data, DOI 10.17632/czfghvzkmb.1, CC BY 4.0.

- Norne field: Open Porous Media (OPM) / NTNU Norne Benchmark Case, <http://www.ipt.ntnu.no/~norne/wiki/doku.php?id=start>.
- Bentheimer sandstone micro-CT: Zenodo record 5542624, CC BY 4.0.