LARGE MATRIX MODEL: A PRETRAINED SPARSE SOLVER ACROSS EQUATION FAMILIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse linear solvers are traditionally specialized to the algebraic structure of their operators, and existing learned solvers largely inherit this specialization by training for individual matrices or within a single equation family. We ask whether sparse solving can instead admit a pretraining paradigm: can a single model learn reusable numerical priors that transfer across equation families and to previously unseen real-world operators? We introduce the Large Matrix Model (LMM), a pretrained neural–numerical solver that maps a sparse operator and its current residual to a correction subspace and is deployed with frozen weights. LMM combines residual-conditioned graph representations with node-wise gated Chebyshev directions, which overcome the rank ceiling of a shared polynomial dictionary, and couples the resulting subspace with restart-free incremental orthogonalization and an exact float64 minimal-residual projection. To evaluate cross-equation-family generalization, we construct a corpus of 240 equation families spanning five algebraic classes and 15 structured subfamilies, pretrain LMM on 80 families, and hold out the remaining 160 families by identity. A single frozen checkpoint converges on 1,590 of 1,600 systems from unseen equation families and, without adaptation, on 489 of 556 systems in a large SuiteSparse split disjoint from our smaller adaptation test. Under matched downstream data and compute, pretrained initialization improves convergence on the group-disjoint 200-system SuiteSparse test from 174 to 181 and reduces geometric-mean residual by $1.72\times$ relative to random initialization. These results provide evidence that reusable solver priors can be learned across heterogeneous equation families, moving learned sparse solvers from problem-specific training toward pretrained generalist numerical models.

1 INTRODUCTION

Large sparse linear solves dominate the cost of high-fidelity simulation across scientific computing, yet effective solver design remains strongly problem-specific. Classical Krylov methods already encode distinct algebraic contracts—CG for SPD systems, MINRES for symmetric indefinite systems, and GMRES for general nonsymmetric systems (Hestenes & Stiefel, 1952; Paige & Saunders, 1975; Saad & Schultz, 1986; Saad, 2003). Their practical performance is in turn governed by preconditioning (Benzi, 2002): AMG-preconditioned CG is highly effective for SPD elliptic operators such as Poisson-type systems (Ruge & Stüben, 1987; Falgout, 2006; Vaněk et al., 1996), whereas ILU-type preconditioners and nonsymmetric Krylov methods are often more appropriate for convection–diffusion and transport-dominated problems (Saad, 1994; van der Vorst, 1992; Feng et al., 1995). This specialization has been a strength of numerical linear algebra, but it also means that solver performance depends heavily on prior knowledge about the matrix family at hand.

The machine-learning response has so far inherited much of the same specialization. Earlier work learns multigrid components, approximate factorizations, nonlinear preconditioners, or iterative search directions inside restricted operator classes (Greenfeld et al., 2019; Luz et al., 2020; Li et al., 2023; Kaneda et al., 2023; Azulay & Treister, 2023; Häusner et al., 2024; Rudikov et al., 2024). Some recent learned preconditioners train *one network per matrix*, achieving strong performance on individual SuiteSparse instances but paying a training cost at every deployment. Others aim for broader reuse, yet still generalize only within a single PDE family, geometry distribution, or param-

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.

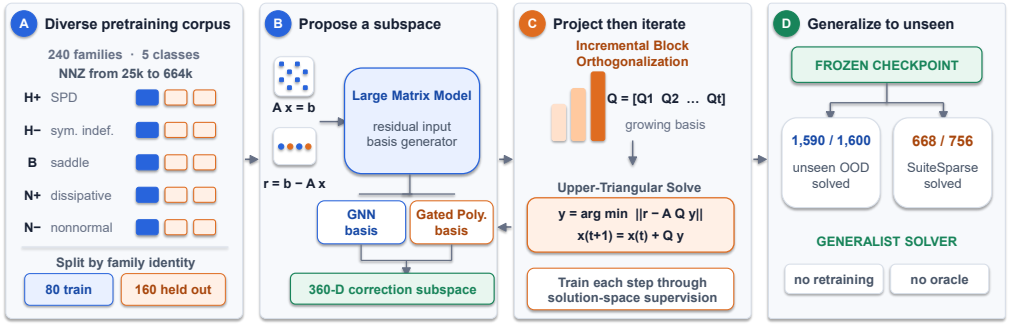

Figure 1: **Overview of the Large Matrix Model (LMM). A: Pretraining corpus.** We construct a synthetic corpus spanning five matrix classes and 240 equation families, using 80 families for pretraining and holding out 160 by family identity for evaluation. **B: Residual-conditioned subspace generation.** Given a sparse operator and the current residual, LMM predicts a correction subspace by combining graph-learned directions with node-wise gated Chebyshev directions. **C: Certified numerical update.** The proposed directions are accumulated across solver steps by Incremental Block Orthogonalization (IBO), and the iterate update is computed by an exact float64 minimal-residual projection. **D: Generalization.** A single frozen checkpoint transfers across unseen equation families and to real SuiteSparse matrices, illustrating a pretrained generalist solver rather than a class-specialized learned preconditioner.

eter sweep. In both cases, the learned solver is tied to a narrow operator distribution. What remains largely unexplored is the pretraining question: *can a single model be trained once, frozen, and then transferred across equation families and to real sparse systems it has never seen?*

This paper studies that question through the lens of *cross-equation-family generalization*. Rather than withholding only matrix instances, right-hand sides, or parameter settings, we hold out entire equation families during training and evaluate whether a frozen model can solve systems generated from families absent from pretraining. To this end, we introduce the *Large Matrix Model (LMM)*, a pretrained neural–numerical solver for sparse linear systems. LMM does not attempt to predict the solution directly or to replace numerical linear algebra wholesale. Instead, it learns to generate a correction subspace conditioned on the sparse operator and the current residual, while the actual iterate update is computed by an exact float64 minimal-residual projection. In this sense, the neural component proposes directions, and the numerical component certifies the update.

Two design choices are central to making this paradigm work in large scale matrix. First, LMM combines graph-learned directions with node-wise gated polynomial directions. This hybrid design is motivated by a structural limitation of conventional polynomial dictionaries: a shared degree-p polynomial basis remains confined to a rank-$(p+1)$ Krylov subspace, regardless of how many columns are emitted. Node-wise gating breaks this rank ceiling and substantially enlarges the effective correction space. Second, LMM uses Incremental Block Orthogonalization (IBO) to accumulate generated directions across solver steps without restart, avoiding the repeated recomputation cost of re-orthogonalizing the entire history from scratch. Together, these choices yield a learned subspace iteration that remains numerically explicit, scalable, and compatible with frozen-weight deployment.

To evaluate this setting, we construct a synthetic corpus spanning 240 equation families across five algebraic matrix classes and 15 structured subfamilies. We pretrain on 80 families and reserve the remaining 160 for family-level out-of-distribution evaluation. A single frozen LMM checkpoint converges on 1,590 of 1,600 systems from unseen equation families and transfers without adaptation to 489 of 556 systems in the large SuiteSparse split. Under matched downstream data and compute, pretrained initialization also improves convergence on the disjoint small SuiteSparse split from 174/200 to 181/200 and reduces geometric-mean final residual by $1.72\times$ relative to random initialization. These results suggest that sparse solvers can indeed acquire reusable numerical pri-

ors across heterogeneous operator families, moving learned sparse solving beyond per-instance or per-family training.

Figure 1 summarizes the core idea: pretrain once on a diverse synthetic corpus, use the frozen Large Matrix Model to generate residual-aware correction spaces, and couple it with exact numerical projection to obtain a generalist solver that transfers across unseen equation families and real sparse matrices.

Contributions.

1. **Cross-equation-family pretraining for sparse linear systems.** We formulate and study a generalization regime in which complete equation families, rather than only matrix instances or parameter settings, are withheld from training. We instantiate this setting with 240 equation families spanning five algebraic operator classes and 15 application domains, with 80 families used for pretraining and 160 reserved for Syn-OOD-L evaluation.
2. **A hybrid neural-numerical subspace solver.** We introduce the Large Matrix Model (LMM), a residual-conditioned architecture that combines graph-learned search directions with exact minimal-residual updates. We characterize the rank ceiling of conventional polynomial dictionaries, introduce node-wise gating to escape it, and develop an Incremental Block Orthogonalization (IBO) procedure that enables large, restart-free accumulated correction spaces at practical cost.
3. **Generalization across synthetic families and real operators.** A single frozen model solves 1,590/1,600 systems from equation families unseen during pretraining and 489/556 systems in the large SuiteSparse split without adaptation. A group-disjoint adaptation control on the disjoint small split further isolates the benefit of pretrained initialization over random initialization.

2 METHOD

LMM is a residual-conditioned neural–numerical iteration. The neural network does not predict the solution or an approximate inverse. Instead, at each solver action it proposes a low-dimensional correction space; a numerical least-squares step then computes the update and certifies the resulting residual. The method is built around four ingredients: residual-conditioned subspace generation, a hybrid graph–polynomial basis, restart-free accumulation through Incremental Block Orthogonalization (IBO), and cross-family pretraining. Figure 2 summarizes one solver action. Exact architecture and optimization details are given in Appendix A.

2.1 RESIDUAL-CONDITIONED NEURAL SUBSPACE ITERATION

We solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with sparse $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. At iteration t, the current residual is $\boldsymbol{r}_t = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_t$. The model generates a block of candidate correction directions

$$\boldsymbol{V}_t = g_\theta(G(\boldsymbol{A}), \boldsymbol{r}_t, \text{coords}) \in \mathbb{R}^{n\times k}, \qquad \boldsymbol{Q}_t = \text{qr}(\boldsymbol{V}_t), \tag{1}$$

where $G(\boldsymbol{A})$ is the sparse matrix graph, qr is the QR decomposition and $\boldsymbol{Q}_t$ spans the newly proposed correction block. For any correction basis $\boldsymbol{B}$, the corresponding minimal-residual update is

$$\boldsymbol{y}^\star = \arg\min_{\boldsymbol{y}} \|\boldsymbol{r}_t - \boldsymbol{A}\boldsymbol{B}\boldsymbol{y}\|_2, \qquad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{y}^\star, \qquad \boldsymbol{r}_{t+1} = \boldsymbol{r}_t - \boldsymbol{A}\boldsymbol{B}\boldsymbol{y}^\star. \tag{2}$$

Thus the network proposes directions, while the iterate update remains explicit numerical linear algebra, following the projection viewpoint underlying minimal-residual Krylov methods (Saad & Schultz, 1986; Saad, 2003). All reported residuals are recomputed from $\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}$ rather than predicted by the model. The main solver uses the accumulated basis $\boldsymbol{B}_t = [\boldsymbol{Q}_0, \ldots, \boldsymbol{Q}_t]$, maintained incrementally as described in Section 2.3.

Writing $\mathcal{W} = \text{range}(\boldsymbol{A}\boldsymbol{B})$, Eq. equation 2 is the orthogonal projection of $\boldsymbol{r}_t$ onto $\mathcal{W}$, and therefore

$$\frac{\|\boldsymbol{r}_{t+1}\|_2}{\|\boldsymbol{r}_t\|_2} = \|(\boldsymbol{I} - \boldsymbol{P}_{\mathcal{W}})\hat{\boldsymbol{r}}_t\|_2 = \sin\angle(\boldsymbol{r}_t, \mathcal{W}). \tag{3}$$

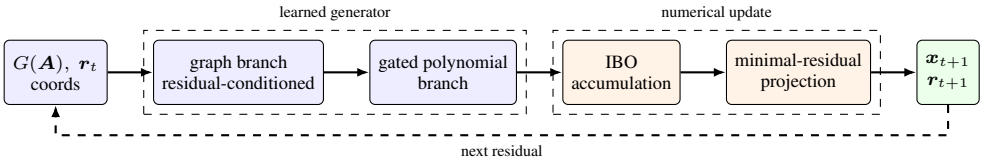

Figure 2: **One LMM solver action.** A residual-conditioned graph branch and a gated polynomial branch propose a correction space. Incremental Block Orthogonalization (IBO) incorporates the new block into the accumulated image space without restart, after which an explicit least-squares projection computes the update. The network runs in float32; the projection is evaluated in float64 in the main configuration.

The learned problem is consequently geometric: generate a compact space whose image under $\boldsymbol{A}$ aligns with the current residual. In high dimension, nominal width alone is insufficient; the emitted columns must also span many independent directions. This motivates the hybrid generator below.

2.2 Hybrid graph–polynomial correction spaces

The generator concatenates two complementary sources of directions,

$$\boldsymbol{V}_t = \left[\boldsymbol{V}_t^{\mathrm{graph}},\ \boldsymbol{V}_t^{\mathrm{poly}}\right]. \tag{4}$$

The graph branch is a residual-conditioned message-passing network over the sparsity graph of $\boldsymbol{A}$; the main implementation uses GATv2 layers (Brody et al., 2022). It can use local operator structure and node features to produce spatially heterogeneous directions that are not restricted to a Krylov sequence. The main configuration emits 300 graph-based columns. The exact GATv2 architecture, coordinate features, coarse-grid path, and readout heads are specified in Appendix A.1.

The second branch injects an explicit polynomial numerical prior. Let $\hat{\boldsymbol{A}} = \boldsymbol{A}/\max_i \sum_j |A_{ij}|$ and define Chebyshev probes $\boldsymbol{s}_\ell = T_\ell(\hat{\boldsymbol{A}})\boldsymbol{r}_t$, $\ell = 0, \dots, p$. A conventional m-column polynomial dictionary has columns $\boldsymbol{q}_j = \sum_{\ell=0}^{p} c_{j\ell}\boldsymbol{s}_\ell$. Its nominal width can greatly overstate its usable dimension:

Proposition 1 (Rank ceiling of a shared polynomial dictionary)**.** *For fixed $\hat{\boldsymbol{A}}$ and $\boldsymbol{r}_t$, any dictionary with $\boldsymbol{q}_j = \sum_{\ell=0}^{p} c_{j\ell}T_\ell(\hat{\boldsymbol{A}})\boldsymbol{r}_t$ satisfies* $\mathrm{rank}([\boldsymbol{q}_1, \dots, \boldsymbol{q}_m]) \le p + 1$*, independently of m.*

The proof is immediate from containment in $\mathcal{K}_{p+1}(\hat{\boldsymbol{A}}, \boldsymbol{r}_t)$ and is included in Appendix A.2. In our setting $p = 15$ and $m = 60$, so an ungated dictionary can contribute at most 16 independent directions even though it emits 60 columns.

We break this shared-subspace constraint by modulating each polynomial column with a node-wise gate predicted from the final graph state $\boldsymbol{h}_i$. With a learned rank-16 map ϕ and learned column embedding $\boldsymbol{u}_j$, the gate and resulting direction are

$$g_{i,j} = 1 + 0.5\tanh\left(\frac{\phi(\boldsymbol{h}_i)^\top \boldsymbol{u}_j}{\sqrt{16}}\right), \quad \boldsymbol{D}_j = \mathrm{diag}(\boldsymbol{g}_j), \qquad \boldsymbol{q}_j = \boldsymbol{D}_j \sum_{\ell=0}^{p} c_{j\ell}T_\ell(\hat{\boldsymbol{A}})\boldsymbol{r}_t. \tag{5}$$

Because $\boldsymbol{D}_j$ varies across columns, the gated directions need not lie in one degree-p Krylov space. The gates are initialized close to identity so that the branch starts from a polynomial prior while remaining free to acquire spatially varying corrections. The effective-rank measurements in Appendix A.2 confirm the intended effect: the ungated 60-column branch realizes rank 9–12, whereas gating raises it to 48–60 at the same polynomial order and width.

2.3 Restart-free accumulation with Incremental Block Orthogonalization

Using only the newest block would discard useful directions from earlier residual states. LMM therefore accumulates all generated blocks without restart. Let $\boldsymbol{B}_t = [\boldsymbol{Q}_0, \dots, \boldsymbol{Q}_t]$ denote the accumulated correction basis and $\boldsymbol{W}_t = \boldsymbol{A}\boldsymbol{B}_t \in \mathbb{R}^{n \times K_t}$ its image under the operator. Recomputing a full QR of $\boldsymbol{W}_t$ after every action costs $O(nK_t^2)$ and quickly dominates when $K_t = tk$ grows.

IBO updates the factorization only with the newly generated block, following the classical principle of stable QR updates and reorthogonalization (Björck, 1967; Daniel et al., 1976; Golub & Van Loan, 2013; Trefethen & Bau, 1997). Suppose $\boldsymbol{U}_{t-1}$ is an orthonormal basis for the previously accumulated image space and $\boldsymbol{W}_{\text{new}} = \boldsymbol{A}\boldsymbol{Q}_t$. We compute

$$\boldsymbol{P}_t = \boldsymbol{U}_{t-1}^\top \boldsymbol{W}_{\text{new}}, \qquad \boldsymbol{W}_\perp = \boldsymbol{W}_{\text{new}} - \boldsymbol{U}_{t-1}\boldsymbol{P}_t, \qquad \boldsymbol{W}_\perp = \boldsymbol{U}_{\text{new}}\boldsymbol{R}_{\text{new}}. \tag{6}$$

Only the new k columns are orthogonalized against the existing K_{t-1}-dimensional image space, reducing the leading per-action cost to $O(nK_{t-1}k)$ instead of $O(nK_t^2)$. The accumulated image basis is then $[\boldsymbol{U}_{t-1}, \boldsymbol{U}_{\text{new}}]$ and is never truncated.

A naive incremental triangular solve can become unstable when $\boldsymbol{W}_\perp$ is nearly rank deficient. We therefore separate *basis maintenance* from the *least-squares solve*: QR is used to maintain the accumulated image space, while the new orthogonal component is solved directly with a rank-revealing least-squares routine and the residual is formed from that least-squares residual. This preserves the exact minimal-residual semantics without forcing the solve through an ill-conditioned incremental factor. The full block formulas, stability analysis, and solver pseudocode are given in Appendix A.3. The measured consequence is evaluated separately in Section 4.3: relative to full QR recomputation, IBO reduces QR time by $6.25\times$ and end-to-end solve time by $1.96\times$ for the same checkpoint and stopping rule.

2.4 Cross-family pretraining

The goal of pretraining is not to memorize a fixed matrix distribution but to learn correction-space priors that remain useful after the governing equation changes. We construct a synthetic corpus of 240 equation families spanning five algebraic classes—symmetric positive definite (H+), symmetric indefinite (H-), saddle-point (B), nonsymmetric dissipative (N+), and non-normal (N-) —and 15 application domains. Pretraining uses 80 families; the remaining 160 are excluded from optimization and reserved for family-level evaluation.

Solution-space subspace supervision. Synthetic generation gives access to the exact solution $\boldsymbol{x}^\star$. At a solver state $\boldsymbol{x}_t$, let $\Delta\boldsymbol{x}_t^\star = \boldsymbol{x}^\star - \boldsymbol{x}_t$ be the exact remaining correction and let $\boldsymbol{Q}_t$ be an orthonormal basis for the generated correction space. We train the generator to capture this correction directly:

$$\boxed{\mathcal{L}_{\text{sub}}(\theta) = \frac{\left\|(\boldsymbol{I} - \boldsymbol{Q}_t\boldsymbol{Q}_t^\top)\Delta\boldsymbol{x}_t^\star\right\|_2}{\|\Delta\boldsymbol{x}_t^\star\|_2 + \varepsilon}}. \tag{7}$$

This objective is the sine of the angle between the exact correction and the generated subspace. Unlike a residual-space objective that differentiates through a least-squares problem involving $\boldsymbol{A}\boldsymbol{Q}_t$, Eq. equation 7 backpropagates through an orthogonal projector in solution space. The relation between this surrogate and residual contraction, including the condition-number bound, is given in Appendix A.4.

Training on model-generated residual states. Training only on the initial state $\boldsymbol{r}_0 = \boldsymbol{b}$ creates a mismatch because inference repeatedly feeds the model residuals produced by its own previous actions. Rather than fully unrolling the solver during every backward pass, we periodically run short solver rollouts, store visited tuples $(\boldsymbol{A}, \boldsymbol{r}_t, \boldsymbol{x}_t, \boldsymbol{x}^\star)$ in a replay buffer, and mix these states into subsequent minibatches. Each replayed state is supervised by its exact remaining correction $\Delta\boldsymbol{x}_t^\star$. This supplies model-induced state coverage at the cost of a small number of forward solver actions rather than a fully unrolled training trajectory.

The corpus is sampled across algebraic classes rather than grouped by application name. The main configuration additionally uses a shared graph readout together with coarse algebraic-class-conditioned readouts on synthetic data; real matrices without such metadata use the fixed fallback readout. This specialization is an implementation detail rather than the source of family transfer—removing the class heads leaves Syn-OOD-L convergence unchanged in our ablation—so we defer its exact construction and all optimizer settings to Appendix A.1 and Appendix A.4.

Method	Learned numerical object	Instance	Equation	Family	Syn.→real
LMM (ours)	residual-conditioned correction subspace	✓	✓	✓	✓
G-RANS (Liao et al., 2026)	residual-conditioned correction subspace	✓	✓	–	–
NeurKItt (Luo et al., 2024)	approximate invariant subspace	✓	–	–	–
GNP (Chen, 2025)	approximate inverse / nonlinear preconditioner	–	–	–	–
GMP (Zhang et al., 2026)	multilevel nonlinear preconditioner	✓	–	–	–
AutoSpec (Liu et al., 2026)	matrix-polynomial recurrence	✓	✓	–	✓

Table 1: **Comparison of representative learned sparse solvers.** The last four entries refer to the scope of reuse of a *single frozen checkpoint*. *Instance* denotes an unseen coefficient matrix from the same governing equation/problem distribution; *equation* denotes reuse across distinct governing equations or application problems within a related operator regime; and *family* denotes transfer to equation families withheld from training, including different algebraic regimes. Synthetic→real denotes zero-shot deployment of weights trained only on synthetic operators to real matrices. A new right-hand side for a fixed matrix does not count as instance transfer. A dash means that the corresponding transfer was not demonstrated in the cited work. Detailed protocol distinctions are given in Appendix D.

3 Related Work

Learning-based sparse solvers differ both in the *numerical object* they learn and in the scope over which a single set of learned weights can be reused. GNP and GMP learn nonlinear preconditioners (Chen, 2025; Zhang et al., 2026); NeurKItt predicts an approximate invariant subspace (Luo et al., 2024); G-RANS generates residual-conditioned correction subspaces (Liao et al., 2026); and AutoSpec predicts coefficients of spectrum-adaptive polynomial recurrences (Liu et al., 2026). Table 1 compares these methods under a single-checkpoint protocol, separating transfer across unseen matrix instances, across distinct equations, and across held-out equation families. We defer protocol-level details and boundary cases to Appendix D.

4 Experiments

We structure the evaluation to mirror the method. Section 4.2 first tests the central single-checkpoint claim: whether the residual-conditioned subspace solver transfers from the synthetic pretraining distribution to held-out equation families and real matrices. Section 4.3 then dissects the hybrid correction-space generator and the training ingredients that support family-level transfer. Section 4.4 isolates the numerical role of restart-free Incremental Block Orthogonalization (IBO), and Section 4.5 measures the contribution of synthetic pretraining under a matched downstream budget. Additional dataset protocol, variance measurements, class-wise baseline diagnostics, basis-width sweeps, and the full numerical screen are reported in Appendix B.

4.1 Evaluation protocol

Synthetic family holdout. The large synthetic corpus has five algebraic superclasses, 15 structured subfamilies, and 240 equation families. Each family contributes 1,000 training matrices and 10 independently seeded test matrices. LMM uses a balanced 80-family, 80,000-matrix pretraining pool. We partition the test by family identity: *Syn-ID* contains 800 new matrices from the 80 training families, while *Syn-OOD-L* contains 1,600 matrices from the 160 families withheld entirely from pretraining. The two sets are disjoint. *Syn-OOD-S* is a separate benchmark: five superclasses, four equations per superclass, and ten test matrices per equation, for 200 systems. None of its 20 equation identities occurs in the 80-family pretraining set. We treat it as secondary because its seed variance is much larger.

Real-matrix transfer. We divide the 189-entry SuiteSparse manifest into two disjoint evaluation sets. *SS-S* is the 50-matrix, 200-system adaptation test split. *SS-L* is its set complement: 139 entries and 556 systems. Each matrix has four manufactured right-hand sides.

Model	Syn-OOD-S (200)	Syn-ID (800)	Syn-OOD-L (1,600)	SS-S (200)	SS-L (556)
LMM	148 / 3.019e-3	**800 / 5.187e-4**	**1590 / 5.394e-4**	179 / 5.274e-4	489 / 1.372e-4
LMM, finetuned on SS-L	**150 / 3.025e-3**	**800** / 6.506e-4	1587 / 5.884e-4	**181 / 4.369e-4**	**493 / 1.258e-4**
G-RANS (adapted)	110 / 7.109e-3	455 / 2.749e-3	831 / 4.356e-3	95 / 6.707e-1	331 / 8.418e-3
GNP (adapted)	0 / 7.507e-2	476 / 1.998e-3	801 / 3.279e-3	46 / 2.065e-2	91 / 6.999e-2
GMP (adapted)	38 / 8.707e-2	307 / 1.439e-2	540 / 1.722e-2	42 / 2.739e-2	126 / 5.557e-2
BiCGSTAB	**150** / 3.159e-3	710 / 1.795e-3	1436 / 1.898e-3	100 / 1.286e-2	307 / 4.370e-2
GMRES	110 / 5.947e-3	750 / 1.243e-3	1443 / 1.315e-3	104 / 6.612e-3	300 / 3.072e-3
Jacobi-GMRES	112 / 7.823e-3	719 / 1.190e-3	1363 / 1.604e-3	47 / 2.933e-1	237 / 4.081e-2
MINRES (symmetric only)	74/120	410/410	840/840	79/112	214/248
CG (certified H+ only)	30/40	160/160	320/320	n/a	n/a

Table 2: **Single-checkpoint transfer across synthetic and real systems.** Entries are converged systems / geometric-mean final relative residual unless only an applicability-specific success count is shown. Each LMM row is one frozen checkpoint. The G-RANS/GNP/GMP rows are our pretraining adaptations of narrower original protocols. Classical methods are CPU-timed and reported for coverage; MINRES and CG denominators are their applicable subsets.

The entries are real, non-binary, square, have $20{,}000$–$60{,}000$ rows, and at most 2 million nonzeros. They cover 51 collection groups and symmetric, near-symmetric, and nonsymmetric regimes.

For adaptation, collection groups are disjoint across $110/29/50$ train/validation/test matrices; no source group in SS-S occurs in adaptation training. SS-L is zero-shot for synthetically pretrained models, but it contains the adaptation train and validation matrices. Consequently, after SuiteSparse adaptation, SS-S is the held-out real-matrix result and SS-L is reported only as a post-adaptation diagnostic.

Metrics and baselines. A solve is successful when the true relative residual $\mathrm{RR} = \|\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2/\|\boldsymbol{b}\|_2$ reaches 10^{-3} within 200 neural actions and a $2.5\,\mathrm{s}$ wall-clock budget; non-finite outcomes count as failures. We report successful systems, geometric-mean final residual, and, where relevant, solver actions and timing. For G-RANS, GNP and GMP we report our cross-matrix adaptations, which extend methods originally evaluated under narrower training protocols; these rows are therefore implementation-level comparison points rather than claims about the original papers. GMRES and BiCGSTAB provide general-purpose classical references (Saad & Schultz, 1986; van der Vorst, 1992), while MINRES and CG are reported only on matrices satisfying their algebraic applicability conditions (Paige & Saunders, 1975; Hestenes & Stiefel, 1952). Three independent LMM trainings give $1{,}584$–$1{,}590/1{,}600$ successes on Syn-OOD-L; the full variance analysis is in Appendix B.2.

4.2 Cross-family and synthetic-to-real transfer

Table 2 gives the main comparison. The canonical LMM checkpoint solves all 800 Syn-ID systems and $1{,}590/1{,}600$ Syn-OOD-L systems from equation families absent from pretraining. This near-saturation under family holdout is the central synthetic result: the same frozen weights remain effective when both the matrix instance and the governing equation family change. The result is stable across three independent trainings, which vary by 6 systems over the $1{,}600$-system test.

The finetuned checkpoint retains Syn-OOD-S and Syn-ID coverage and loses three Syn-OOD-L successes, while its Syn-OOD-L residual rises from $5.394\mathrm{e}{-4}$ to $5.884\mathrm{e}{-4}$. It gains two SS-S and four SS-L successes. This is mild specialization to real matrices rather than catastrophic forgetting.

The G-RANS checkpoint coverage falls to $455/800$ on Syn-ID and $831/1600$ on Syn-OOD-L, and non-finite outcomes occur on both SuiteSparse splits. The cross-matrix GNP and GMP adaptations are more strongly class-dependent: their aggregate coverage is substantially lower, with the class-wise diagnostics in Appendix B.3 showing particularly sharp degradation away from the regimes for which their preconditioning priors are best matched. These results support the paper's intended distinction between high performance inside a compatible operator regime and reuse of one learned solver across many regimes.

	converged (of 1,600)		geometric-mean RR		actions	
Variant	count	Δ (σ)	value	Δ (σ)	mean	Δ (σ)
LMM (anchor)	1588	—	5.399e-4	—	7.66	—
w/o polynomial branch	1484	-32.4	8.140e-4	$+47.2$	15.99	$+166.5$
w/o node-wise polynomial gate	1575	-4.0	5.720e-4	$+5.5$	8.26	$+12.0$
w/o replay	1580	-2.5	5.764e-4	$+6.3$	8.27	$+12.3$
w/o class-conditioned heads	1588	0.0	5.540e-4	$+2.4$	7.92	$+5.2$
w/o algebraic-class routing	1579	-3.1	5.504e-4	$+1.8$	7.63	-0.5
monomial for Chebyshev	1580	-2.8	6.853e-4	$+25.0$	8.59	$+18.6$
residual-space training loss	1581	-2.5	5.490e-4	$+1.6$	7.64	-0.4
w/o coordinate-grid coarse path	1585	-1.2	5.350e-4	-0.8	8.04	$+7.6$

Table 3: **Component ablations on held-out equation families.** Δ is measured against the LMM anchor in units of seed standard deviation ($\sigma = 3.2$ convergences and $\sigma = 5.8 \times 10^{-6}$ residual, and $\sigma = 0.050$ actions).

The classical rows give the complementary picture. Structure-specific methods remain excellent when their assumptions are known: MINRES and CG converge on every applicable Syn-ID and Syn-OOD-L system in our test. LMM is not meant to replace such specialists when their algebraic contract is available; its target is broad coverage with one learned model when the operator family is not selected in advance. Under that protocol, the same synthetic-pretrained checkpoint solves $489/556$ SS-L systems without adaptation, showing that its learned correction-space prior is not restricted to the synthetic generator.

4.3 Hybrid correction spaces and training components under family shift

We next evaluate the design choices behind Sections 2.2 and B.7. Every row in Table 3 is retrained on the same 80-family corpus and evaluated on the 1,600 Syn-OOD-L systems. Across three independent LMM trainings, the seed standard deviation is 3.2 convergences and 5.8×10^{-6} in geometric-mean residual, and 0.050 solver actions. We therefore express deltas in units of this measured standard deviation and avoid interpreting effects below roughly 2σ as decisive.

The largest architectural effect is the polynomial branch itself: removing it loses 104 convergences, raises residual by 47σ, and more than doubles the number of actions. The recurrence also matters: replacing Chebyshev directions by monomials at the same order and width raises residual by 25σ. This agrees with the structural argument in Section 2.2: at $p = 15$ an ungated 60-column polynomial dictionary is confined to a rank-16 Krylov space, whereas node-wise gating raises the measured effective rank to 48–60. Removing that gate produces a smaller but consistent drop (4.0σ in coverage and 5.5σ in residual).

Several remaining ingredients are individually modest but systematically point in the same direction. Replay improves both coverage and residual under model-induced state shift; class-conditioned readouts primarily improve residual quality; algebraic-class routing affects coverage more than residual; and the coordinate-grid coarse path is inside the final-metric variance band but reduces action count. The solution-space training objective is likewise a modest empirical improvement rather than a dominant one. We do not add these row-wise deltas—their interactions are not identified by one-at-a-time ablations—but the pattern explains why the final solver is best viewed as a system of complementary numerical and architectural choices rather than a single high-leverage trick. The detailed routed-attention replication and additional component diagnostics are retained in Appendix B.4.

The basis-width sweep gives a second view of the same mechanism. Every tested width reaches $800/800$ on Syn-ID, but Syn-OOD-L convergence and action count continue to improve as the basis grows; from the smallest to the full configuration, mean Syn-OOD-L actions fall by more than half. Capacity is therefore most valuable under distribution shift, precisely where the correction space must span directions absent from the training families (Appendix B.6).

4.4 Restart-free accumulation makes wide correction spaces practical

Section 2.3 predicts that IBO should reduce the orthogonalization cost from recomputing a factorization of all accumulated columns to updating it with only the new block. We test this numerically

QR update	restart	converged	mean actions	mean time	QR time
Incremental	**none**	**800**	**5.81**	**0.240 s**	**0.055 s**
Full recompute	none	782	5.65	0.471 s	0.344 s
Incremental	720	790	9.54	0.326 s	0.086 s
Incremental	360	766	13.97	0.407 s	0.126 s

Table 4: **Numerical effect of incremental, restart-free accumulation.** Same LMM checkpoint and 800 Syn-ID systems in every row; projection precision is float64. Full QR recomputation is $6.25\times$ more expensive in the QR component and $1.96\times$ slower end-to-end, while restart discards useful history and increases the required number of actions.

Arm	converged	gmean RR	actions	time (s)	GMRES speedup
finetuned	**181**	**4.369e-4**	**9.84**	**0.567**	$1.37\times$
from scratch	174	7.503e-4	13.86	0.817	$0.31\times$
Δ	+7	$1.72\times$	-29.0%	-30.6%	sign flips

Table 5: **Matched-budget downstream training.** Both arms train on the same 440 systems for 30 epochs. On 200 systems from 50 group-disjoint test matrices, pretraining improves coverage and residual and changes the solve-time comparison with GMRES from slower-than to faster-than.

by re-running the same frozen checkpoint on the 800 Syn-ID systems while changing only the QR update and restart rule. Table 4 reports the configurations most directly tied to the method; the full 12-configuration screen, including precision variants, is in Appendix B.5.

The measured QR ratio, $0.344/0.055 = 6.25\times$, closely matches the predicted K/k factor at the observed mean of roughly six accumulated blocks. The benefit is not only latency: under the fixed 2.5 s budget, full recomputation converts 18 otherwise successful systems into time-limit failures. Restarting is also harmful at this basis width. Truncating the accumulated space to 720 columns increases mean actions from 5.81 to 9.54, while restarting every 360 columns raises them to 13.97. Thus the large correction space used by the learned generator is operationally useful only when its history can be retained at manageable cost. Finally, float32 projection reaches the same $800/800$ coverage and is about 6% faster in this regime; we retain float64 in the main configuration as a conservative numerical default rather than as a source of the reported transfer gains.

4.5 Pretraining provides a transferable initialization

Finally, we isolate the effect of synthetic pretraining under a fixed downstream budget. Two models are trained on the same 440 SuiteSparse adaptation systems for 30 epochs with identical data, optimizer, and compute. The only difference is initialization: one starts from the frozen 80-family pretrained checkpoint and the other from random weights. Table 5 reports the group-disjoint SS-S test. Pretraining adds seven SS-S successes, reduces geometric-mean residual by $1.72\times$, and cuts mean actions from 13.86 to 9.84. The matched-tolerance comparison with GMRES changes from $0.31\times$ to $1.37\times$. This is a fixed-budget initialization result. The randomly initialized arm has not converged within 30 epochs, so the experiment does not establish that scratch training could never reach the same asymptote.

5 Conclusion

We introduced LMM, a pretrained neural–numerical sparse solver that learns residual-conditioned correction spaces rather than a direct solution or matrix-specific inverse. Across 240 synthetic equation families, a single frozen checkpoint retains near-saturated performance on families excluded from pretraining and transfers directly to unseen SuiteSparse matrices. The experiments further connect this transfer to the method design: hybrid graph–polynomial directions provide a high-rank correction space, restart-free IBO makes that wide accumulated space computationally practical, and cross-family pretraining improves real-matrix adaptation under matched downstream data and compute. Together, these results support pretraining as a viable route from family-specific learned solvers toward reusable numerical models.

REFERENCES

Yael Azulay and Eran Treister. Multigrid-augmented deep learning preconditioners for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 45(3):S127–S151, 2023. doi: 10.1137/21M1433514.

Michele Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002. doi: 10.1006/jcph.2002.7176.

Åke Björck. Solving linear least squares problems by Gram–Schmidt orthogonalization. *BIT Numerical Mathematics*, 7(1):1–21, 1967. doi: 10.1007/BF01934122.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Jie Chen. Graph neural preconditioners for iterative solutions of sparse linear systems. In *International Conference on Learning Representations*, 2025. arXiv:2406.00809.

James W. Daniel, William B. Gragg, Linda Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Mathematics of Computation*, 30(136):772–795, 1976. doi: 10.1090/S0025-5718-1976-0431641-8.

Robert D. Falgout. An introduction to algebraic multigrid. *Computing in Science & Engineering*, 8 (6):24–33, 2006. doi: 10.1109/MCSE.2006.105.

Y. T. Feng, G. J. Huang, D. R. J. Owen, and D. Perić. An evaluation of iterative methods in the solution of a convection–diffusion problem. *International Journal of Numerical Methods for Heat & Fluid Flow*, 5(3):213–223, 1995. doi: 10.1108/EUM0000000004063.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 4 edition, 2013. doi: 10.56021/9781421407944.

Markus Götz and Hartwig Anzt. Machine learning-aided numerical linear algebra: Convolutional neural networks for the efficient preconditioner generation. In *2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*, pp. 49–56. IEEE, 2018. doi: 10.1109/ScalA.2018.00010.

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. PMLR, 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. PMLR, 2024.

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.

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, pp. 72525–72624, 2024. doi: 10.52202/079017-2311.

Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

Ayano Kaneda, Osman Akar, Jingyu Chen, Victoria Alicia Trevino Kala, David Hyde, and Joseph Teran. A deep conjugate direction method for iteratively solving linear systems. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15720–15736. PMLR, 2023.

Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioners for conjugate gradient PDE solvers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19425–19439. PMLR, 2023.

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. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Weixin Liao, Mingquan Feng, Zhizhou Zhang, Youjia Wu, Yifan Fu, and Junchi Yan. G-RANS: Generalizable residual-aware neural solvers for sparse systems. In *International Conference on Machine Learning*, 2026.

Zihang Liu, Oleg Balabanov, Yaoqing Yang, and Michael W. Mahoney. Learning to discover iterative spectral algorithms. *arXiv preprint arXiv:2602.09530*, 2026.

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(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

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, pp. 128636–128667, 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*, pp. 6489–6499. PMLR, 2020.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden 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, pp. 119301–119335, 2024. doi: 10.52202/079017-3791.

Christopher C. Paige and Michael A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975. doi: 10.1137/0712047.

Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A. Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics PDEs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 104035–104064, 2024. doi: 10.52202/079017-3306.

Alexander Rudikov, Vladimir Fanaskov, Ekaterina Muravleva, Yuri M. Laevsky, and Ivan Oseledets. Neural operators meet conjugate gradients: The FCG-NO method for efficient PDE solving. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42766–42782. PMLR, 2024.

J. W. Ruge and K. Stüben. Algebraic multigrid. In Stephen F. McCormick (ed.), *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pp. 73–130. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987. doi: 10.1137/1.9781611971057.ch4.

Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986. doi: 10.1137/0907058.

Yousef Saad. ILUT: A dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications*, 1(4):387–402, 1994. doi: 10.1002/nla.1680010405.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2 edition, 2003. doi: 10.1137/1.9780898718003.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1596–1611, 2022.

Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997. doi: 10.1137/1.9780898719574.

H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2): 631–644, 1992. doi: 10.1137/0913035.

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

Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. PDEformer: Towards a foundation model for one-dimensional partial differential equations. *arXiv preprint arXiv:2402.12652*, 2024.

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 V. 2*, pp. 12952–12963, 2026.

A METHOD AND IMPLEMENTATION DETAILS

This section records the architectural and numerical details omitted from the main text. We use the same notation as Section 2 and separate implementation choices from the four method ingredients that define LMM.

A.1 ARCHITECTURE AND INPUT NORMALIZATION

Per-sample normalization. Coordinates are scaled independently by dimension, the right-hand side is normalized to unit norm, and the operator is scaled by its mean absolute diagonal,

$$\tilde{c}_{i,d} = \frac{c_{i,d}}{\max_j |c_{j,d}|}, \qquad \tilde{\boldsymbol{b}} = \frac{\boldsymbol{b}}{\|\boldsymbol{b}\|_2}, \qquad s_{\boldsymbol{A}} = \frac{1}{n}\sum_i |A_{ii}|, \qquad \tilde{\boldsymbol{A}} = \frac{\boldsymbol{A}}{s_{\boldsymbol{A}}}. \tag{8}$$

When an exact solution is available for training, it is rescaled consistently as $\tilde{\boldsymbol{x}}^\star = \boldsymbol{x}^\star s_{\boldsymbol{A}}/\|\boldsymbol{b}\|_2$.

Graph features and trunk. The sparsity pattern defines directed edges (i, j) with scalar edge attribute A_{ij}. Node inputs concatenate the normalized residual, a learned 16-dimensional positional encoding of coordinates, raw normalized coordinates, edge-geometry Fourier features, and a 6-dimensional index-periodic degree-of-freedom code. The latter is a positional encoding derived from node index modulo six; it is not assumed to encode a physical variable type. The trunk contains four GATv2 layers (Brody et al., 2022) with hidden width 128, four attention heads, LayerNorm, GELU activations, FFN expansion factor four, and dropout 0.1.

After each graph layer, a lightweight coordinate-grid path aggregates node states into an 8×8 grid on the first two normalized coordinates. Cell means are transformed by a coarse MLP, exchanged with neighboring cells, lifted back to nodes, and fused through a learned gate. This path supplies a coarse spatial context without constructing an algebraic multigrid hierarchy.

Graph readout. Each graph layer emits 38 shared and 37 algebraic-class-conditioned columns, yielding $4(38+37) = 300$ graph directions in the main configuration. The class-conditioned readout uses one of five coarse algebraic labels on synthetic data; matrices without labels use a fixed fallback head. A FiLM-style node-wise modulation driven by Fourier and index-periodic features is applied before concatenation. As shown in the ablation study, removing the class-conditioned heads leaves Syn-OOD-L convergence unchanged, so we treat this specialization as an implementation detail rather than a defining source of generalization.

Gated polynomial readout. For the polynomial branch we rescale the operator by its maximum absolute row sum, $\hat{\boldsymbol{A}} = \boldsymbol{A}/\max_i \sum_j |A_{ij}|$, build Chebyshev probes $T_0(\hat{\boldsymbol{A}})\boldsymbol{r} = \boldsymbol{r}$, $T_1(\hat{\boldsymbol{A}})\boldsymbol{r} = \hat{\boldsymbol{A}}\boldsymbol{r}$, and $T_j(\hat{\boldsymbol{A}})\boldsymbol{r} = 2\hat{\boldsymbol{A}}T_{j-1}(\hat{\boldsymbol{A}})\boldsymbol{r} - T_{j-2}(\hat{\boldsymbol{A}})\boldsymbol{r}$, and emit 60 learned combinations at order $p = 15$. The node gate for column m is

$$g_{i,m} = 1 + 0.5 \tanh\left(\frac{\phi(\boldsymbol{h}_i)^\top \boldsymbol{u}_m}{\sqrt{16}}\right), \tag{9}$$

where ϕ is a rank-16 map of the final graph hidden state. Concatenating the 300 graph directions and 60 gated polynomial directions gives the $k = 360$ basis used by the main model. The full model contains 1,740,468 trainable parameters.

Component	Setting	Component	Setting
Graph trunk	4× GATv2, width 128, 4 heads	Basis width k	$300 + 60 = 360$
Norm / activation	LayerNorm / GELU, FFN×4	Polynomial order	$p = 15$
Coarse path	8×8 coordinate grid, gated	Polynomial gate rank	16
Graph readout	shared + 5 coarse-class heads	Parameters	1,740,468
Network precision	FP32	Projection precision	float64
Restart	none	Replay capacity	4,000 states

Table 6: Implementation configuration used by the main LMM checkpoint.

A.2 Polynomial rank ceiling and effective-rank measurement

Proof of Proposition 1. Each ungated column $\boldsymbol{q}_j = \sum_{\ell=0}^{p} c_{j\ell} T_\ell(\hat{\boldsymbol{A}})\boldsymbol{r}$ lies in $\mathrm{span}\{T_0(\hat{\boldsymbol{A}})\boldsymbol{r}, \ldots, T_p(\hat{\boldsymbol{A}})\boldsymbol{r}\} = \mathcal{K}_{p+1}(\hat{\boldsymbol{A}}, \boldsymbol{r})$. Therefore any number of such columns spans a space of dimension at most $p+1$.

The gate in Eq. equation 5 replaces the common probe set by column-specific diagonally modulated probes $\boldsymbol{D}_j T_\ell(\hat{\boldsymbol{A}})\boldsymbol{r}$. Since the diagonal operators $\boldsymbol{D}_j$ vary with j, the columns are no longer confined to one degree-p Krylov space; the shared $p{+}1$ ceiling therefore no longer applies, leaving only the trivial upper bound $\min(m, n)$.

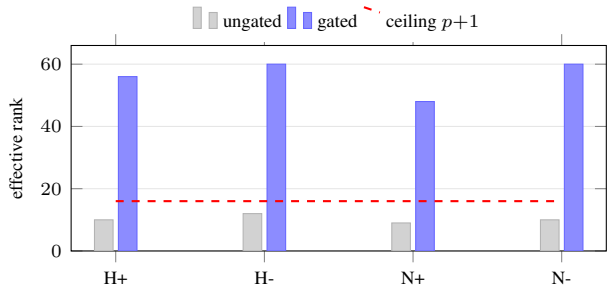

Figure 3: Effective rank of the 60-column polynomial dictionary at initialization on representative $\sim$ 30k-DOF synthetic systems, using relative singular-value threshold 10^{-4}. At order $p = 15$, the ungated dictionary remains below the theoretical ceiling of 16, whereas node-wise gating raises the realized rank to 48–60.

A.3 Incremental Block Orthogonalization and stable projection

IBO maintains an orthonormal basis of the accumulated image space $\mathrm{range}(\boldsymbol{A}\boldsymbol{B}_t)$ rather than recomputing a QR factorization of all previous columns. With the notation of Eq. equation 6, a naive block-QR implementation would use the new triangular factor directly in the least-squares solve. When $\boldsymbol{W}_\perp$ is nearly rank deficient, the resulting loss of orthogonality can be amplified according to

$$\left\|\boldsymbol{U}_{t-1}^\top \boldsymbol{U}_{\mathrm{new}}\right\| \le \left\|\boldsymbol{U}_{t-1}^\top \boldsymbol{W}_\perp\right\| \left\|\boldsymbol{R}_{\mathrm{new}}^{-1}\right\|. \tag{10}$$

In representative runs the first factor is about 1.5×10^{-3} while $\|\boldsymbol{R}_{\mathrm{new}}^{-1}\|$ can reach 2×10^2, predicting an $O(10^{-1})$ loss of orthogonality. Re-orthogonalization reduces the first factor but does not remove amplification by an ill-conditioned triangular solve.

We therefore use the QR factors only for maintaining the accumulated image basis. Let $\boldsymbol{c}_{\mathrm{old}} = \boldsymbol{U}_{t-1}^\top \boldsymbol{r}_t$ and $\boldsymbol{r}_\perp = \boldsymbol{r}_t - \boldsymbol{U}_{t-1}\boldsymbol{c}_{\mathrm{old}}$. The new coefficients are obtained from a rank-revealing least-squares solve,

$$\boldsymbol{y}_{\mathrm{new}} = \arg\min_{\boldsymbol{y}} \|\boldsymbol{W}_\perp \boldsymbol{y} - \boldsymbol{r}_\perp\|_2, \tag{11}$$

and the coefficients associated with the previous accumulated basis are recovered by

$$\boldsymbol{y}_{\mathrm{old}} = \boldsymbol{R}_{t-1}^{-1}\left(\boldsymbol{c}_{\mathrm{old}} - \boldsymbol{P}_t \boldsymbol{y}_{\mathrm{new}}\right). \tag{12}$$

The residual is formed directly as $\boldsymbol{r}_{t+1} = \boldsymbol{r}_\perp - \boldsymbol{W}_\perp \boldsymbol{y}_{\mathrm{new}}$, avoiding accumulation through the ill-conditioned new triangular block.

The numerical projection uses float64 in the main configuration. A precision screen in Section B.5 shows that float32 and float64 are indistinguishable at the current operating point, where the model typically converges in fewer than eight actions; float64 is retained as a conservative default across the full benchmark.

A.4 Training objective, replay, and optimization

A natural alternative to Eq. equation 7 is to train directly on the one-step residual,

$$\mathcal{L}_{\mathrm{res}}(\theta) = \frac{\|\boldsymbol{r} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{y}^\star\|_2^2}{\|\boldsymbol{r}\|_2^2}, \qquad \boldsymbol{y}^\star = \arg\min_{\boldsymbol{y}} \|\boldsymbol{r} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{y}\|_2. \tag{13}$$

Algorithm 1 LMM inference with restart-free Incremental Block Orthogonalization

Require: $\boldsymbol{A}$, $\boldsymbol{b}$, coordinates; tolerance τ, action cap T, wall-clock cap
$\boldsymbol{x}_0 \leftarrow \mathbf{0}$, $\boldsymbol{r}_0 \leftarrow \boldsymbol{b}$; accumulated image factors empty
for $t = 0, 1, \ldots, T-1$ **do**
  $\boldsymbol{V}_t \leftarrow g_\theta(G(\boldsymbol{A}), \boldsymbol{r}_t, \text{coords})$; $\boldsymbol{Q}_t \leftarrow \text{qr}(\boldsymbol{V}_t)$
  $\boldsymbol{W}_{\text{new}} \leftarrow \boldsymbol{A}\boldsymbol{Q}_t$
  project $\boldsymbol{W}_{\text{new}}$ against the accumulated image basis and form $\boldsymbol{W}_\perp$
  solve the orthogonal component by rank-revealing least squares
  recover coefficients for previously accumulated directions
  update $\boldsymbol{x}_{t+1}$ and form the exact least-squares residual $\boldsymbol{r}_{t+1}$
  update the accumulated image QR with $\boldsymbol{W}_\perp$
  break if $\|\boldsymbol{r}_{t+1}\|_2 / \|\boldsymbol{b}\|_2 \le \tau$ or the wall-clock budget is exhausted
end for

This objective differentiates through a least-squares problem involving $\boldsymbol{A}\boldsymbol{Q}$ and becomes increasingly poorly conditioned as the operator becomes ill conditioned. The solution-space objective in Eq. equation 7 avoids this backward path. It also controls residual contraction up to the condition number of the operator:

Proposition 2 (Relation between correction-space error and residual contraction)**.** *Let $\boldsymbol{A}$ be invertible, $\boldsymbol{Q}$ have orthonormal columns, and $\Delta\boldsymbol{x}^\star = \boldsymbol{A}^{-1}\boldsymbol{r}$. If $\rho(\boldsymbol{Q}) = \min_{\boldsymbol{y}} \|\boldsymbol{r} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{y}\|_2 / \|\boldsymbol{r}\|_2$, then*

$$\rho(\boldsymbol{Q}) \le \kappa(\boldsymbol{A}) \, \frac{\|(\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^\top)\Delta\boldsymbol{x}^\star\|_2}{\|\Delta\boldsymbol{x}^\star\|_2}. \tag{14}$$

Proof. Let $\boldsymbol{e} = (\boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^\top)\Delta\boldsymbol{x}^\star$. Evaluating the residual at $\boldsymbol{Q}\boldsymbol{Q}^\top\Delta\boldsymbol{x}^\star$ gives $\min_{\boldsymbol{y}} \|\boldsymbol{r} - \boldsymbol{A}\boldsymbol{Q}\boldsymbol{y}\|_2 \le \|\boldsymbol{A}\boldsymbol{e}\|_2 \le \|\boldsymbol{A}\|_2 \|\boldsymbol{e}\|_2$. Moreover $\|\Delta\boldsymbol{x}^\star\|_2 \le \|\boldsymbol{A}^{-1}\|_2 \|\boldsymbol{r}\|_2$. Combining the two inequalities gives the result. □

The condition number appears in this bound, but not in the backward operator of the subspace loss. We use this proposition only as motivation; the ablation in Appendix B.4 shows a modest empirical effect rather than a large one.

Each epoch mixes initial states with states from a FIFO replay buffer of capacity $4{,}000$. At epoch end, a small sample is rolled out for at most ten solver actions and the visited states are inserted into the buffer. Optimization uses AdamW with learning rate 5×10^{-4}, weight decay 10^{-5}, cosine decay, gradient clipping at 1.0, one graph per GPU, gradient accumulation of eight, FP32 network arithmetic, and gradient checkpointing. The main checkpoint is trained for 60 epochs on the 80 pretraining families.

B Extended Experimental Protocols and Results

The main paper reports the complete headline comparison and component-ablation matrices together with the numerical and pretraining results most directly tied to the method. This appendix provides the supporting evaluation record: run-to-run and seed-to-seed variability, class-wise diagnostics for cross-matrix baselines, routed-attention replication details, the full numerical projection screen, basis-width sweeps, specialist classical-solver applicability, and additional pretraining/finetuning analysis.

B.1 Dataset construction and audit

Dataset summary. Table 7 separates generated matrices from solver systems. Every synthetic system has one independently generated matrix and one manufactured right-hand side; every SuiteSparse entry has four.

The ranges report algebraic dimension and stored nonzeros, not matrix-entry clipping. Synthetic recipe parameters remain family-specific, and SuiteSparse numerical values are preserved before per-sample model normalization.

Dataset	classes	groups	equations/entries	train systems	test systems	N / NNZ range
Syn-OOD-S	5	5	20	0	200	25,392–50,001 / 141,654–232,288
Syn-ID	5	15	80	80,000	800	25,000–49,928 / 25,023–644,600
Syn-OOD-L	5	15	160	0	1,600	25,000–49,928 / 25,023–644,600
SS-S	4[†]	16	50	440[⋆]	200	20,374–60,000 / 73,249–1,801,300
SS-L	4[†]	35	139[‡]	0	556	20,000–60,000 / 15,525–1,990,382

Table 7: **Disjoint evaluation-set inventory.** Counts are matrices except where systems are stated. Syn-ID and Syn-OOD-L partition the 240-family test by training-family identity; SS-S and SS-L partition the 189-entry SuiteSparse manifest. [†]The SuiteSparse audit classes are SPD-flagged symmetric, other symmetric, near-symmetric, and nonsymmetric. [‡]SS-L has 139 manifest entries after removing the 50 SS-S entries. [⋆]The adaptation budget comprises four systems from each of 110 training matrices.

Syn-OOD-S inventory. Syn-OOD-S is a square-system subset. Each listed equation supplies $1{,}000$ training matrices and ten independently seeded test matrices.

B**:** box-constraint KKT; mixed Darcy; lid-driven-cavity Stokes; forced Stokes.

H^+**:** Dirichlet Poisson; discontinuous-coefficient Poisson; oscillatory-coefficient Poisson; reaction–diffusion.

H^-**:** low-wavenumber Helmholtz; high-wavenumber Helmholtz; shifted Laplace; buckling-like two-field system.

N^+**:** low-Péclet advection–diffusion; mid-Péclet advection–diffusion; positive-reaction ADR; rotating diffusive transport.

N^-**:** high-Péclet advection–diffusion; negative-reaction ADR; rotating high-Péclet transport; open-boundary transport.

The wider archive also contains four rectangular and four singular equations. They are excluded from Syn-OOD-S, whose evaluator links exactly the 20 square-family test files above.

Syn-ID and Syn-OOD-L taxonomy. This taxonomy has five superclasses and three structured subfamilies per superclass. Each subfamily contains the following 16 equation families.

H^+ **/ HP1, smooth coercive elliptic:** Poisson; screened Poisson; smooth variable diffusion; tensor diffusion; axis-aligned anisotropic diffusion; rotated anisotropic diffusion; reaction–diffusion; linearized Poisson–Boltzmann; weighted Laplacian; Laplace–Beltrami; clamped biharmonic; screened biharmonic; implicit heat step; implicit reaction–diffusion step; variable-metric elliptic; mapped-coordinate elliptic.

H^+ **/ HP2, multiscale and high contrast:** checkerboard diffusion; log-normal random-field diffusion; channelized permeability; horizontal layers; rotated layers; high-conductivity inclusions; low-conductivity barriers; SPE10-like permeability; fractured porous-medium diffusion; thin high-conductivity channels; high-contrast tensor diffusion; random rotated anisotropy; periodic and quasiperiodic multiscale coefficients; perforated-domain diffusion; correlated heterogeneous diffusion–reaction.

H^+ **/ HP3, vector, nullspace, and normal equations:** isotropic, anisotropic, heterogeneous, and nearly incompressible penalty elasticity; vector Laplacian; curl–curl plus mass; div–div plus mass; grounded graph Laplacian; resistor network; spring-network stiffness; projected Neumann Poisson; projected free elasticity; cotangent mesh Laplacian; regularized, weighted, and graph-regularized normal equations.

H^- **/ HM1, wave and resonance:** constant, variable-speed, variable-density, anisotropic, heterogeneous, and seismic Helmholtz; elastic Helmholtz; anisotropic elastic wave; structural dynamics; plate vibration; frequency-domain Maxwell curl–curl; waveguide Helmholtz; photonic scalar wave; Schrödinger energy shift; stationary Klein–Gordon; coupled symmetric two-field wave.

H^- **/ HM2, instability and Hessians:** Euler beam, Kirchhoff plate, shell, truss, and frame buckling; prestressed elasticity tangent; snap-through elasticity Hessian; near-instability hy-

perelastic tangent; fixed-active-set contact Hessian; phase-field fracture Hessian; spinodal Cahn–Hilliard Hessian; unstable Allen–Cahn Hessian; Ginzburg–Landau Hessian; topology-optimization Hessian; geomechanical instability tangent; constrained-energy Hessian after nullspace elimination.

H^- **/ HM3, generic symmetric indefinite:** shifted Laplacian; generalized shift; sign-changing diffusion and reaction; signed graph Laplacian; two-interval-spectrum matrix; shifted sparse Wigner; coupled positive–negative blocks; sparse symmetric arrowhead; symmetric banded indefinite; alternating-sign lattice; shifted normal matrix; difference of Gram matrices; random finite-element pencil; SPD plus negative low-rank update; mixed-sign low-rank update.

N^+ **/ NP1, mild ADR:** constant, variable, divergence-free-drift, shear-flow, rotational-flow, anisotropic, and heterogeneous convection–diffusion; positive and variable-positive-reaction ADR; linearized viscous Burgers; passive-scalar, thermal, solute, surface, and porous-medium advection–diffusion; weakly compressible drift–diffusion.

N^+ **/ NP2, M-matrix and positive stable:** monotone upwind convection–diffusion; transport–decay M-matrix; birth–death and queueing-network resolvents; PageRank; shifted directed graph Laplacian; absorbing Markov fundamental system; compartment implicit step; chemical-network M-matrix Jacobian; population and metapopulation migration–decay; stable semiconductor drift–diffusion; groundwater transport–decay; monotone anisotropic finite-volume transport; directed resistor/flow resolvent; diagonally dominant M-matrix.

N^+ **/ NP3, skew dominated with damping:** centered periodic advection plus diffusion; solid-body rotation; shear skew-advection; Coriolis damping; gyroscopic dynamics; spherical rotation–diffusion; divergence-free and Ornstein–Uhlenbeck Fokker–Planck; skew-symmetrized ADR; Hamiltonian–dissipative system; kinetic relaxation; rotating shallow water with drag; damped acoustic system; antisymmetrically coupled reaction; skew graph with damping; sparse skew plus SPD diagonal.

N^- **/ NM1, transport dominated:** high-Péclet constant convection–diffusion; boundary-layer and internal-layer transport; recirculating high-Péclet flow; discontinuous-velocity, pure-upwind, inflow/outflow, unstructured-mesh, and variable-sign-velocity transport; high-Péclet ADR; porous contaminant, heat, species, and channel transport; high-field drift–diffusion; open-boundary transport.

N^- **/ NM2, strongly nonnormal:** Grcar; sparse Jordan chain; upper-bidiagonal convection chain; nonnormal Toeplitz; defective directed cycle; sparse companion; shifted non-backtracking operator; Orr–Sommerfeld; Squire; PML and impedance Helmholtz real-block forms; damped-wave quadratic linearization; recirculating convection; upper-triangular random perturbation; prescribed departure from normality; nearly defective eigenpair family.

N^- **/ NM3, near singular or unstable:** negative and sign-changing-reaction ADR; near-singular open transport; directed Laplacian and Markov generator with tiny shift; conservative transport near a null mode; Neumann ADR near nullspace; gain–loss transport; nonsymmetric Laplacian near resonance; nonsymmetric pencil; flow Jacobian near Hopf; reaction–diffusion Jacobian near bifurcation; semiconductor Jacobian near breakdown; near-singular network transport; nearly unstable M-matrix perturbation; field-of-values-crossing-zero family.

B **/ B1, symmetric PDE saddle point:** Stokes; variable-viscosity and slip-boundary Stokes; mixed Poisson; mixed and anisotropic mixed Darcy; Brinkman; mixed and nearly incompressible elasticity; mixed biharmonic; $H(\mathrm{div})$ flux–pressure diffusion; mixed Maxwell curl–div; divergence-constrained vector Laplacian; mean-value constrained elliptic; mortar coupling; FETI/dual domain decomposition.

B **/ B2, nonsymmetric multiphysics saddle point:** Oseen; Newton and Picard Navier–Stokes; convection–Stokes; mixed convection–diffusion; flux–transport; Oseen–Brinkman; coupled Stokes–Darcy; Boussinesq linearization; linearized MHD; rotating incompressible flow; monolithic fluid–structure interaction; advective poroelasticity; reactive-transport Jacobian; electrohydrodynamic block system; saddle system with a nonnormal primal block.

B **/ B3, optimization and KKT:** equality-constrained and box-constrained active-set QP; general inequality active-set KKT; primal–dual interior-point QP; linear-program interior-point

KKT; Poisson, reaction–diffusion, and convection–diffusion control KKT; topology-optimization KKT; contact primal–dual KKT; obstacle KKT; sparse portfolio QP; network optimal-flow KKT; equality-constrained least squares; SQP Newton KKT; augmented-Lagrangian KKT.

The frozen 80-family selection is balanced at 16 families per superclass. It takes six families from the first subfamily and five from each of the other two; the other 160 family identities never enter training.

Synthetic generation and acceptance. Syn-OOD-S requests $N = 25{,}000$, with structured blocks and grids producing the actual range in Table 7. Its verifier checks exact file counts, required fields, shape/NNZ metadata, and $\|Ax - b\|_2/\|b\|_2 < 10^{-8}$ for every square sample.

For H^+ and H^-, relative symmetry error must be at most 10^{-10}. Small fixtures additionally certify positive definiteness or mixed inertia, and the repaired $N \approx 25{,}000$ files were audited before use.

The large synthetic corpus stores one canonical float64 CSR matrix, one independently seeded x, and $b = Ax$ per sample. The full release has $242{,}400$ unique sample seeds, numerical hashes, and sample hashes, with no train/test overlap.

Validation reloads every sample, checks finiteness, canonical CSR structure, dimensions, NNZ, hashes, exact family counts, and relative $Ax = b$ error at tolerance 10^{-12}. It also runs 480 stratified property checks.

All $242{,}400$ files passed a second-host content-hash sweep after transfer. Three optional ARPACK extreme- eigenvalue diagnostics did not converge; their files passed every data and hash check and were retained.

SuiteSparse construction and acceptance. We pin the SuiteSparse index snapshot dated 31 October 2023 and its SHA-256. From 2,904 entries, a mechanical filter retains real, non-binary, square matrices in the size and NNZ range of Table 7.

Each matrix is converted to canonical float64 CSR by summing duplicates, removing explicit zeros, sorting indices, and rejecting non-finite values. Geometric coordinates are unavailable and are zero-filled.

Four deterministic solution priors generate the systems: Gaussian, Rademacher, five-step graph-smoothed, and two-hop localized. Each uses a distinct seed and stores float32 x and $b = Ax$ for model input.

The audit checks source shape, canonical structure, finite nonzero probes, four distinct seeds, numerical and structural SHA-256, and every stored $Ax = b$ relation. The observed maximum relative error is 7.75×10^{-6}, below the frozen 10^{-5} threshold.

The 189 entries span 51 source groups: 88 nonsymmetric, 16 near-symmetric, 70 other symmetric, and 15 marked positive definite. Two pairs are numerically identical, so the manifest contains 187 distinct matrices.

For finetuning, whole SuiteSparse groups are assigned to train/validation/test while balancing algebraic class, size, NNZ, degree, and application kind. The split is 23/12/16 groups and 110/29/50 matrices.

SS-S is the 50-matrix adaptation test. SS-L is the other 139 entries. For synthetic-only checkpoints both are zero-shot; after adaptation, SS-S remains group-disjoint and SS-L is only a post-adaptation diagnostic.

B.2 Evaluation protocol and variability

Training configuration. The main model, **LMM**, is trained on 80 of the 240 equation families for 60 epochs at seed 43. Every component ablation is retrained on the same corpus with the same budget.

Evaluation suites. We report five disjoint sets: Syn-OOD-S (200 systems), Syn-ID (800), Syn-OOD-L (1,600), SS-S (200), and SS-L (556). Construction and audit are in Section B.1. All evaluations use a true relative residual $\mathrm{RR} = \|\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2/\|\boldsymbol{b}\|_2$, tolerance 10^{-3}, at most 200 neural iterations, and a hard 2.5 s wall-clock budget, in an exclusive quiet window. Non-finite outcomes remain in the denominator and count as failures. SS-S is group-disjoint from the 110-matrix adaptation train split. SS-L is zero-shot only for checkpoints with no SuiteSparse training.

Family-level holdout. Because training uses 80 families and the comprehensive synthetic benchmark contains all 240 families, splitting it by family identity yields 800 *in-distribution* systems (80 seen families) and 1,600 *Syn-OOD-L* systems (160 unseen families). Matrix instances are unseen in both subsets; the OOD subset additionally withholds the equation family. We report this split throughout; we do not aggregate the two subsets in the reported result tables.

Evaluation variability. We quantify variability twice, because the two sources have very different magnitudes. *Run-to-run*: every model was evaluated twice, independently, on two H200 hosts. *Seed-to-seed*: the main model was trained three times from identical configurations at seeds 43, 44, 45. Table 8 reports both.

	Syn-OOD-S (200)	Syn-ID (800)	Syn-OOD-L (1,600)	SS-S (200)	SS-L (556)
Run-to-run: maximum absolute convergence-count difference					
LMM and its ablations	2	0	2	1	4
Seed-to-seed					
converged (43 / 44 / 45)	148 / 132 / 131	800 / 800 / 800	1590 / 1590 / 1584	179 / 179 / 180	489 / 491 / 490
2σ	**19.1**	0.0	6.9	1.2	2.0
geometric-mean RR, rel. σ	**11.6%**	1.2%	1.1%	—	—

Table 8: Variance floors under the disjoint reporting protocol. Seed variation is negligible on Syn-ID, Syn-OOD-L, and both SuiteSparse splits, and *large* on Syn-OOD-S. The original G-RANS checkpoint has one formal evaluation and is therefore excluded from the repeated-run rows.

Seed sensitivity of Syn-OOD-S. Three identical trainings of the main model score 148, 132, and 131 on Syn-OOD-S, a 2σ band of 19 convergences and an 11.6% relative spread in residual. Every ablation in this paper was trained at seed 43 and compared against the seed-43 anchor, so a difference on this benchmark smaller than ≈ 19 cannot be separated from the seed. We therefore treat Syn-OOD-S as a secondary evaluation: we report it for completeness, we mark effects that fall inside the band, and we rest the ablation argument on Syn-OOD-L (1,600 systems, $\sigma = 3.2$) and SS-L (556 systems, $\sigma = 0.58$). We note that a 200-system synthetic benchmark is a common size in this literature; our measurement suggests it is too small for component-level conclusions.

B.3 Additional analysis of cross-family and real-matrix transfer

Table 2 in the main paper gives the aggregate comparison. Here we provide the class-wise and protocol-level diagnostics that explain those aggregate results. The second repetition reproduces every conclusion below. Section B.7 provides additional detail for the matched-budget pretraining comparison.

Cross-matrix GNP adaptation. Cross-matrix GNP (Chen, 2025) — our multi-matrix adaptation of the originally per-matrix method — converges on $476/800$ Syn-ID, $801/1600$ Syn-OOD-L, $46/200$ SS-S, and $91/556$ SS-L systems. The corresponding LMM counts are 800, 1590, 179, and 489. GNP exhausts the iteration cap on every Syn-OOD-S system. The aggregate is not competitive, but the disaggregation is the informative part:

Syn-ID ∪ Syn-OOD-L class	B	H+	H−	N+	N−
converged (of 480)	10	**480**	167	378	242
mean iterations	196.5	**13.0**	170.8	56.2	127.7

A 10,001-parameter network asked to represent an approximate inverse converges on *every* symmetric positive definite system in 13 iterations on average, while solving 10 of 480 saddle-point

systems. The failure is therefore not a failure of neural preconditioning but a class-coverage failure: the object being learned — an approximate inverse — is easy to represent where the spectrum is benign and hard where it is not. This is the same boundary that separates the classical specialists in Table 2, and it is the gap a pretrained subspace generator is meant to close.

Two further observations. Under family shift the adaptation degrades gracefully in coverage ($476/800$ Syn-ID versus $801/1600$ Syn-OOD-L) but loses 64% in geometric-mean residual, so its transfer is weaker than its convergence count suggests. And training longer does not help transfer: between epochs 20 and 60 the training loss continues to fall and combined synthetic convergence rises by 39, while combined SuiteSparse convergence *drops* from 153 to 137 and both residuals worsen. We read this as objective mismatch rather than undertraining, and it is one reason we supervise a subspace rather than an inverse.

Cross-matrix GMP adaptation. Cross-matrix GMP — our multi-matrix adaptation of the Graph Neural Multilevel Preconditioner (Zhang et al., 2026), which trains a two-level ResGCN hierarchy with learned restriction, interpolation, and coarse solve — converges on $307/800$ Syn-ID, $540/1600$ Syn-OOD-L, $42/200$ SS-S, and $126/556$ SS-L systems. Its class breakdown follows the same SPD-favouring structure:

Syn-ID ∪ Syn-OOD-L class	B	H+	H−	N+	N−
converged (of 480)	10	**443**	44	320	30
mean iterations	196.0	**24.1**	187.0	71.3	190.5

The multilevel prior is highly effective on H+ ($443/480$, 24 iterations) and useful on N+ ($320/480$), but weak on B, H−, and especially N− ($30/480$). This is consistent with an AMG-like hierarchy being most reliable for positive-definite operators and much less robust for indefinite or strongly nonnormal systems. The upstream implementation is a per-matrix method: it binds the network and the AMG hierarchy to one static matrix and trains a separate model per test matrix; we adapted it to share parameters across matrices by replacing only the hierarchy data at each call. Because this modification changes the training scope from per-matrix to cross-matrix, we report these results as our cross-matrix GMP adaptation rather than attributing them to the original GMP protocol. The $45{,}506$-parameter model trained for 60 epochs at seed 43 on the same 80-family corpus.

General-purpose versus structure-specific classical solvers. Against the best classical method applicable to every square system in the benchmark, LMM adds 50 Syn-ID, 147 Syn-OOD-L, 75 SS-S, and 189 SS-L convergences. Two observations cut the other way and we state them plainly. BiCGSTAB solves 150 Syn-OOD-S systems against LMM's 148 — a tie inside the seed band, on the track where the learned solver is weakest. And MINRES and CG converge on *every* applicable large-synthetic system ($1250/1250$ and $480/480$): where a matrix satisfies the algebraic contract of a specialized method, that method attains complete coverage on its applicable subset. LMM instead provides a single set of weights when the class is unknown — MINRES covers $410/800$ Syn-ID and $840/1600$ Syn-OOD-L systems, while CG covers 160 and 320; selecting between them requires a classification the benchmark does not supply for real matrices. We report CG as not applicable on SuiteSparse rather than running it under an unverified SPD assumption. We also note that Jacobi preconditioning *degrades* GMRES on both real-matrix tracks (SS-S: $104 \to 47$; SS-L: $300 \to 237$), which is why the unpreconditioned variant is the anchor we report.

B.4 Additional component analysis under family shift

Evaluation split. We evaluate every ablation on Syn-OOD-L: $1{,}600$ systems from 160 equation families absent from training, single host, single quiet window. This is the most stable measurement available to us — seed-to-seed relative spread is 1.08% in residual against 3.39% on SS-L and 6.96% on Syn-OOD-S — and it is a genuine generalization test rather than a fit measurement, since the in-distribution split is saturated at $800/800$ for all but two models. We report differences as multiples of the seed standard deviation σ measured on three independent trainings of the main model, and we do not claim effects below 2σ.

Table 3 in the main paper reports the complete one-at-a-time ablation matrix. We expand the interpretation of those rows below and retain the routed-attention negative replication because it distinguishes transfer quality from measured latency.

Polynomial branch and recurrence. Removing the branch is by far the largest effect in the table: -32σ in convergence, $+47\sigma$ in residual, and mean iterations more than double from 7.66 to 15.99. Replacing Chebyshev with monomials of equal order and equal column count is milder but unambiguous, at $+25\sigma$ in residual and 8.59 iterations. Since these two variants differ only in the polynomial recurrence, the specific basis — not merely the presence of a gated dictionary — is doing the work. The polynomial branch is therefore the only component with a large effect across all three reported metrics.

Node-wise gating, replay, and class-conditioned readouts. Removing the gate costs 4.0σ in convergence and 5.5σ in residual; removing replay costs 2.5σ and 6.3σ. The class-conditioned heads are the weakest of the three: convergence is identical to the anchor and residual moves 2.4σ, so we describe their contribution as a residual-quality effect only. Removing algebraic-class routing costs 3.1σ in convergence, but its residual effect (1.8σ) is inside the band.

Training loss and coordinate-grid path. Training with the residual objective equation 13 in place of equation 7 costs 2.5σ in convergence, with a residual effect inside the band. Taken with Proposition 2 this is consistent with, but weaker than, the mechanism the proposition describes; we do not claim a large effect. Disabling the coordinate-grid coarse path is inside the variance band on both final metrics. It does raise mean iterations from 7.68 to 8.04, suggesting that any contribution is primarily to iteration efficiency rather than final accuracy.

Routed-attention variant. An earlier evaluation on a smaller corpus suggested a 15.5% end-to-end speedup for the routed-attention variant. That effect does not reproduce on the current benchmark. On the 800 Syn-ID systems the routed-attention model takes 0.24620 s against 0.24584 s, a 0.15% *slowdown*, with a family-clustered 95% bootstrap interval of $[-1.83\%, +1.53\%]$; no track yields an interval excluding zero in the favourable direction, and the effect is already absent at epoch 60. The mean time of a complete solver action is likewise unchanged (42.30 vs 42.26 ms), and restricting to systems where both models use the same number of iterations leaves a $+0.30\%$ difference. The earlier figure is traceable to the dense baseline measuring 0.370 s in that session against 0.311 s here: it was an evaluation-session artifact, not an architectural speedup.

Two mechanisms explain the absence. The implementation selects the expert with a host-side `.item()` call and a Python module lookup, forcing a device synchronization at each of the four layers; and the routed attention operator is only one term in an action that also contains input projection, normalization and feed-forward blocks, the coarse-grid path, basis readout, polynomial generation, orthogonalization, sparse products, and the projection solve, none of which shrink with expert width. Parameter counts are a poor latency proxy for sparse graph operators. What the variant does deliver is better transfer: 19.5% and 16.1% lower residual on SS-S and SS-L, at a cost of 15 Syn-OOD-L convergences. We therefore treat it as a capacity/transfer variant rather than an efficiency mechanism and retain the negative replication for completeness.

Comparison with G-RANS. We use the original G-RANS configuration: four stages, five emitted directions per stage ($k = 20$), but trained on cross-family corpus. This anchor solves $110/200$ Syn-OOD-S, $455/800$ Syn-ID, $831/1600$ Syn-OOD-L, $95/200$ SS-S, and $331/556$ SS-L systems. It is substantially weaker than LMM on every transfer split and produces 59 and 80 non-finite outcomes on SS-S and SS-L, respectively.

B.5 Numerical projection and orthogonalization study

Section 2.3 predicts that maintaining the factorization incrementally should reduce the orthogonalization term by a factor of K/k, the ratio of accumulated to per-action columns. We test this directly by re-running the same checkpoint on the 800 Syn-ID systems under a 12-configuration comparison over the QR update rule, restart threshold, and projection precision (Table 9). Step profiling is enabled in every arm, so absolute times carry identical synchronization overhead and are comparable within the table only.

QR update	Restart	Prec.	Conv.	gmean RR	iters	mean time	**QR time**
Incremental	none	f64	**800**	5.188e-4	5.81	0.240 s	**0.055 s**
Incremental	none	f32	**800**	5.192e-4	6.02	**0.225 s**	0.050 s
Full recompute	none	f64	782	5.245e-4	5.65	0.471 s	0.344 s
Full recompute	none	f32	786	5.240e-4	5.66	0.427 s	0.304 s
Incremental	720	f64	790	5.410e-4	9.54	0.326 s	0.086 s
Incremental	360	f64	766	5.493e-4	13.97	0.407 s	0.126 s
Full recompute	720	f64	790	5.409e-4	9.55	0.331 s	0.131 s
Full recompute	360	f64	766	5.499e-4	13.96	0.408 s	0.125 s

Table 9: Projection ablation on the 800 Syn-ID systems, same checkpoint throughout. Four of the twelve configurations are omitted for space; they follow the same pattern.

	Syn-ID (800)		Syn-OOD-L (1,600)		SS-S (200)		SS-L (556)	
k	conv.	gmean RR	conv.	gmean RR	conv.	gmean RR	conv.	gmean RR
20	800	6.835e-4	1548	9.460e-4	141	1.454e-3	371	1.075e-3
40	800	5.868e-4	1572	6.084e-4	143	1.246e-3	405	5.345e-4
80	800	5.188e-4	1567	5.734e-4	169	7.688e-4	474	3.439e-4
160	800	5.817e-4	1573	5.902e-4	177	7.777e-4	486	2.577e-4
360 (LMM)	**800**	**5.187e-4**	**1590**	**5.394e-4**	**179**	**5.274e-4**	**489**	**1.372e-4**

Table 10: Basis-count sweep, 60 epochs on the same corpus and seed. Mean iterations at $k = 20$ versus $k = 160$ fall from 18.31 to 8.63 on Syn-OOD-L and from roughly 52.5 to 15.5 on both real-matrix splits.

Three results. First, the predicted ratio is confirmed: at a mean of 5.81 iterations per solve the theory predicts a $K/k = 5.81\times$ reduction in orthogonalization cost, and the measurement gives $0.344/0.055 = 6.25\times$, with $1.96\times$ lower end-to-end time. Recomputation is not merely slower — it converts 18 systems into time-limit failures under the fixed budget.

Second, restarting is actively harmful at this basis width. Truncating to 720 columns costs 10 convergences and raises mean iterations from 5.81 to 9.54; truncating to 360 costs 34 and raises them to 13.97. This is the quantitative form of the claim in Section 2.3 that restart-free accumulation is what makes a wide basis usable. Consistently, the two QR rules become equivalent at restart 360, where every action begins a fresh factorization and there is no history to amortize.

Third, float32 is sufficient at this operating point: it converges $800/800$ at 5.192×10^{-4} against float64's $800/800$ at 5.188×10^{-4}, while being 6% faster. We retain float64 as the default for safety across the corpus, but the precision requirement we previously reported belongs to a regime with roughly ten times as many accumulation steps.

B.6 Basis-width scaling under distribution shift

We trained three compressed models at $k \in \{20, 40, 80, 160\}$ on the same 80-family corpus, seed, and 60-epoch budget as LMM, changing only the basis allocation across the graph and polynomial branches. Table 10 reports the sweep against the uncompressed $k = 360$ anchor.

Three observations. *All* of the value of a wider basis appears out of distribution. Every basis count solves $800/800$ Syn-ID systems, so that track cannot separate them; on the 1,600 Syn-OOD-L systems, mean iterations fall 52.9% (18.31 → 8.63) and convergence rises from 96.8% to 98.3% between $k = 20$ and $k = 160$. The class breakdown localizes this: H+ and N+ are saturated in convergence at $k = 20$, while H− and especially N− consume nearly all the extra iteration budget (N−: 30.0 → 13.1 mean iterations, and the only class where the anchor still converges everything, 480 against 464 at $k = 160$).

Second, the compression is real but not free. $k = 160$ reaches 98.9% versus the anchor's 99.4% coverage on Syn-OOD-L and comes within one point on both SuiteSparse splits while using 44% of its columns; $k = 80$ is the first practical low-rank point. Fewer columns do not, however, make

inference faster end-to-end: the smaller projection is overwhelmed by the extra neural actions it requires. This is the same lesson as the projection ablation: at this operating point the network forward, not the linear algebra, sets the wall-clock.

Third, the sweep is not pointwise monotonic, so we read basis count as a capacity/compute tradeoff rather than a guaranteed improvement per checkpoint. We find no evidence for $k = 20$ or $k = 40$ as defaults: their cheaper per-action projection is more than offset by additional actions, timeouts, and non-finite outcomes. We retain $k = 360$ for the main results, where it has the best convergence and residual across the reported disjoint tracks and its largest margin on N$-$ and shifted-distribution sparse matrices.

Classical solver applicability. Table 2 includes five classical anchors. Two observations are unfavourable to us and we state them plainly. BiCGSTAB solves 150 Syn-OOD-S systems against LMM's 148 — a tie inside the seed band, on the track where the neural model is weakest. And MINRES and CG converge on *every* applicable large-synthetic system ($1250/1250$ and $480/480$): where a matrix satisfies the algebraic contract of a specialized method, that method attains complete coverage on its applicable subset.

The claim we do make is coverage without knowing the class. MINRES applies to $410/800$ Syn-ID and $840/1600$ Syn-OOD-L systems, while CG applies to 160 and 320; selecting among them requires a classification the benchmark does not supply for real matrices, which is why we report CG as not applicable on SuiteSparse rather than running it under an unverified SPD assumption. Against the best classical method that applies to every square system, LMM adds 50 Syn-ID, 147 Syn-OOD-L, 75 SS-S, and 189 SS-L convergences. We also note that Jacobi preconditioning *degrades* GMRES on both real-matrix tracks ($104 \to 47$ and $300 \to 237$), which is why the unpreconditioned variant is the anchor we report.

B.7 Additional analysis of matched-budget synthetic pretraining

The main SS-L result uses a model frozen after synthetic pretraining. To test initialization at a fixed downstream budget, we train two models on 440 systems from 110 SuiteSparse matrices for 30 epochs.

Data, optimizer, and compute are identical. One arm starts from the 80-family checkpoint and the other from random weights. Table 5 evaluates both on 50 group-disjoint SS-S matrices.

The pretrained arm gains seven convergences and improves residual by $1.72\times$ under identical downstream training. It also reduces actions by 29.0% and mean time by 30.6%.

At matched tolerance, scratch adaptation is slower than right-Jacobi GMRES ($0.31\times$), whereas the pretrained arm is faster ($1.37\times$). Pretraining therefore flips the sign of the measured speedup.

C Broader Context for Learned Solvers and Scientific Pretraining

A broader line of work learns numerical ingredients rather than full solution operators. Learned multigrid methods predict prolongation operators for diffusion-type systems (Greenfeld et al., 2019; Luz et al., 2020); learned preconditioners include matrix-structure prediction (Götz & Anzt, 2018), GNN-based approximate factorizations (Li et al., 2023; Häusner et al., 2024), equation-specific neural–multigrid hybrids (Azulay & Treister, 2023), and neural-operator preconditioning inside flexible CG (Rudikov et al., 2024). DCDM instead learns improved conjugate directions for SPD Poisson systems (Kaneda et al., 2023). These works motivate learning inside an iterative numerical algorithm, but their demonstrated reuse remains tied to narrower algebraic or PDE regimes than the family holdout studied here.

Neural operators and PDE foundation models address a complementary form of transfer by learning solution or evolution operators directly. Representative examples include DeepONet and FNO (Lu et al., 2021; Li et al., 2021), message-passing PDE solvers (Brandstetter et al., 2022), and recent multi-physics pretraining approaches such as MPP, DPOT, CoDA-NO, Poseidon, and PDEformer (McCabe et al., 2024; Hao et al., 2024; Rahman et al., 2024; Herde et al., 2024; Ye et al., 2024).

LMM shares their pretraining-and-transfer motivation but targets a different object: a correction space for an algebraic sparse system, coupled to an explicit residual-certified numerical projection rather than an end-to-end surrogate trajectory.

Most prior methods establish only part of this transfer hierarchy. GNP's large-scale SuiteSparse evaluation is based on matrix-specific training. NeurKItt and GMP demonstrate reuse within a fixed PDE setting, including size or resolution extrapolation; in GMP, the shared-weight experiment is confined to anisotropic diffusion. G-RANS extends one solver across several related second-order PDEs. AutoSpec transfers from synthetic spectra to large real SPD operators, while explicitly defining its test-time generalization within structurally matched spectral classes. LMM instead targets a single pretrained solver that is reused across unseen instances, distinct equations, entirely held-out equation families, and real sparse matrices.

The comparison in Table 1 is intentionally narrow: it contains the learned sparse solvers for which the reuse scope of a single checkpoint can be compared directly. This section records adjacent lines of work that motivate LMM but are not placed in that table because they learn a different numerical object or are evaluated under a different problem contract.

Learned multigrid and preconditioning. Machine learning has long been used to replace hand-designed components of classical iterative methods. Greenfeld et al. (Greenfeld et al., 2019) learn prolongation for a class of two-dimensional diffusion problems, while Luz et al. (Luz et al., 2020) learn AMG prolongation with a graph neural network for sparse symmetric positive (semi-)definite systems. Götz and Anzt (Götz & Anzt, 2018) use a CNN to infer block structure for block-Jacobi preconditioning. More recent methods learn the preconditioner itself: Li et al. (Li et al., 2023) learn a GNN-based approximate factorization for PCG; Häusner et al. (Häusner et al., 2024) learn incomplete factorizations; Azulay and Treister (Azulay & Treister, 2023) combine CNNs with multigrid ingredients for high-frequency Helmholtz systems; and Rudikov et al. (Rudikov et al., 2024) use a neural operator as a nonlinear preconditioner inside flexible CG. These methods preserve a classical outer iteration, as LMM does, but are demonstrated within more restricted matrix classes or equation families.

Learned search directions. DCDM (Kaneda et al., 2023) is particularly close in spirit because the network improves iterative search directions rather than predicting the final solution. Its experiments focus on large SPD systems from Poisson equations. LMM differs in using a block correction subspace, an exact minimal-residual projection, and a pretraining protocol whose primary axis is transfer across held-out equation families and algebraic regimes.

Neural PDE solvers and operator learning. DeepONet (Lu et al., 2021), FNO (Li et al., 2021), and message-passing PDE solvers (Brandstetter et al., 2022) learn solution or evolution operators and have established important forms of resolution, geometry, parameter, and trajectory generalization. PDEBench (Takamoto et al., 2022) helped standardize evaluation across multiple PDEs. These models solve a different task from an algebraic iterative solver: their prediction is itself the approximate physical solution or trajectory, whereas LMM emits a numerical search space and retains an explicit residual-based update.

Multi-physics pretraining and foundation models. Recent work has made cross-system pretraining an explicit objective. MPP (McCabe et al., 2024) pretrains a single transformer over heterogeneous physical systems; DPOT (Hao et al., 2024) scales autoregressive denoising pretraining over more than ten PDE datasets; CoDA-NO (Rahman et al., 2024) pretrains a neural operator across multiphysics systems; Poseidon (Herde et al., 2024) studies downstream transfer to new physics; and PDEformer (Ye et al., 2024) represents PDEs symbolically and numerically to support multi-equation inference. These works are the closest precedent for the *pretraining* motivation of LMM. The distinction is numerical: they learn mappings between physical fields, while LMM learns reusable priors for constructing correction spaces of sparse linear systems and leaves the final update to explicit linear algebra. Consequently, the relevant OOD unit for LMM is not only a new PDE trajectory or parameter setting, but a new equation family that may induce a different sparse algebraic regime.

D GENERALIZATION PROTOCOLS FOR RELATED WORK

Table 1 records only transfer demonstrated with the same learned weights; it does not infer capability from architecture alone. We use a deliberately strict hierarchy: changing only the right-hand side for a fixed operator is below *instance* transfer; *instance* requires an unseen matrix from the same governing problem distribution; *equation* requires the same checkpoint to be reused across distinct equations or application problems; and *family* requires an equation family to be withheld from training.

GNP. GNP learns a graph neural approximation to the inverse and uses it as a nonlinear preconditioner (Chen, 2025). Its principal SuiteSparse study evaluates many matrix types, but the learned model is optimized separately for the matrix being solved; breadth of the benchmark therefore does not constitute cross-matrix checkpoint transfer.

NeurKItt. NeurKItt predicts an approximate invariant subspace and uses it to warm-start Krylov iteration (Luo et al., 2024). Models are trained for individual PDE datasets. The paper demonstrates transfer across matrix sizes within the same Darcy-flow problem, so we count instance-level reuse but not cross-equation reuse.

GMP. GMP learns a multilevel nonlinear preconditioner built on an AMG hierarchy (Zhang et al., 2026). Its main 867-matrix SuiteSparse study uses per-matrix training. The paper separately demonstrates cross-matrix generalization by training one model on anisotropic-diffusion matrices and applying the same weights to unseen anisotropy parameters and larger grids. Because this experiment remains within the same governing PDE, we count instance transfer but not equation transfer.

G-RANS. G-RANS learns residual-conditioned correction subspaces and reports cross-equation transfer among Poisson, advection–diffusion, reaction–diffusion, and Helmholtz FEM systems (Liao et al., 2026). We therefore count both instance and equation transfer, while reserving *family* transfer for a stricter holdout of heterogeneous equation families spanning distinct algebraic regimes.

AutoSpec. AutoSpec learns spectral-probe-conditioned coefficients for executable matrix-polynomial recurrences (Liu et al., 2026). Both backbone and embedding training use synthetic spectra, followed by frozen-model evaluation on large real SPD operators. The authors characterize this as generalization across operator instances within a structurally defined spectral class and note that performance can degrade when test spectra contain features absent from training. We therefore count synthetic-to-real and cross-equation/application transfer within the demonstrated SPD regime, but not family-level transfer across distinct algebraic regimes.

E LIMITATIONS

The current evidence is deliberately broad but not universal. The pretraining taxonomy is synthetic and cannot cover every industrial operator; the real-matrix evaluation contains 189 SuiteSparse matrices and uses a fixed hardware-dependent wall-clock protocol; and structure-specific classical solvers remain preferable when their algebraic assumptions are known and satisfied. Our matched-budget experiment also establishes the benefit of pretrained initialization at the tested downstream budget, not an asymptotic impossibility result for training from scratch. Extending the family taxonomy, testing substantially larger and more heterogeneous operators, and learning when to invoke specialized numerical tools are natural next steps toward a genuinely general-purpose solver.

AI USE STATEMENT

ICLR 2027 requires authors to disclose how generative AI tools were used in the preparation of a submission, distinguishing tasks that *must* be disclosed from those whose disclosure is recommended. In this work, we used generative AI tools for the following tasks with required disclosure:

- **Implementing methods.** Portions of the training, evaluation, and analysis code for the solver, the pretraining pipeline, and the evaluation tooling were written with AI assistance.

- **Research methodology and experiment design.** Feedback on experiment design, evaluation protocols, and the choice of controls and baselines.
- **Interpreting results.** Assistance in interpreting ablations and cross-corpus transfer results.
- **Formulating mathematical claims and proving ingredients.** Assistance in organizing the convergence argument for subspace-projection updates and the numerical-stability analysis of the incremental QR factorization.

The following tasks fall under recommended disclosure: drafting and editing sections of this manuscript, creating and analyzing software code, assistance in summarizing and finding literature, formatting references, and suggesting paper structure and titles.

We did not use generative AI tools for the following: synthetic data sets, theoretical models or conceptual frameworks, dataset cleaning and reformatting, or qualitative analysis. The AI tools did not generate the numerical results reported in this paper; all reported measurements were produced by code executed on our own hardware and are traceable to frozen artifacts. All content produced with AI assistance was reviewed, verified, and edited by the authors, who take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe here the artifacts and procedures needed to reproduce our results; additional detail appears in the appendix.

Data. The synthetic pretraining corpus is generated deterministically from a taxonomy of 240 equation families. Generation fixes `PYTHONHASHSEED`, `CUBLAS_WORKSPACE_CONFIG`, and per-sample RNG streams, and the generator additionally seeds Python, NumPy, SciPy, and Torch; a cross-host check regenerated all 720 smoke-test files byte-for-byte identically. The manifest records per-sample SHA-256 hashes of the canonical `(pattern, numeric, sample)` triple, and all 242,400 samples were content-verified from a second host after transfer. The SuiteSparse benchmark is built from a pinned collection index with a frozen selection manifest; the selection, the group-disjoint adaptation split, and the four right-hand-side priors with the resulting $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|$ validation errors are all recorded in that manifest.

Code. Training and evaluation entrypoints, configuration files, and experiment registries are specified by path in the appendix. Every experiment is launched from a version-controlled configuration with the referenced commit recorded.

Evaluation protocol. All reported comparisons use a true relative residual $\mathrm{RR} = \|\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2/\|\boldsymbol{b}\|_2$ with tolerance 10^{-3}, and wall-clock measurements were made in exclusive windows with no concurrent training or heavy I/O. We state the hardware for each timing comparison and do not compare absolute convergence across different hosts.

ETHICS STATEMENT

This work develops numerical algorithms for solving sparse linear systems and uses only publicly available benchmark matrices and synthetically generated problems. No human subjects, personal data, or sensitive information are involved. We foresee no direct negative societal impacts specific to this work. As with any learned method that may be deployed inside a numerical pipeline, we emphasize that our solver reports a certified residual for every solve, that non-convergent and non-finite outcomes are preserved and counted rather than discarded, and that our reported gains are stated as coverage improvements under a fixed wall-clock budget rather than as unconditional speedups.