

A Neural Hierarchical-Matrix Preconditioner for Real-Time GPU Solves

Carl Osborne
MIT CSAIL
Cambridge, USA
osbo@mit.edu

Minghao Guo
MIT CSAIL
Cambridge, USA
guomh2014@gmail.com

Crystal Owens
MIT CSAIL
Cambridge, USA
crystal@mit.edu

Wojciech Matusik
MIT CSAIL
Cambridge, USA
wojciech@csail.mit.edu

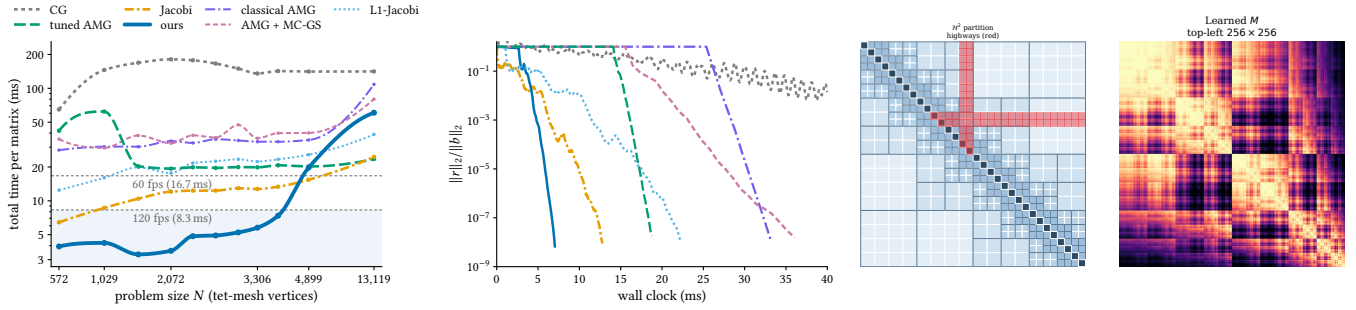


Figure 1: Left to right: median wall clock per matrix on the barrier-heat ladder, where ours alone fits 120 fps up to $\sim 3.6k$; residual against wall clock at $N = 3,647$, same legend; the nested $\mathcal{H}^2$ partition (highways in red); and a learned factor on a 2D multiphase Poisson frame.

Abstract

Interactive simulation solves $Ax = b$ for a sparse SPD A that changes every frame, inside an 8–16 ms budget. At a few thousand unknowns, algebraic multigrid’s setup alone exceeds that budget, while Jacobi and other local preconditioners have no setup but cannot move error across the domain. We learn a preconditioner for this gap: a graph-and-attention network predicts an SPD approximate inverse in $\mathcal{H}^2$ -matrix format. On a spatially ordered 3D mesh, blocks of the true inverse lose rank as the clusters they couple move apart; the format’s nested bases follow that decay, so inference and apply are dominated by leaf-block work linear in N , where a dense inverse costs N^2 . Our main finding concerns training. Probe losses reach M only through a product with A , so their gradient vanishes on the near-null modes that set the conjugate-gradient iteration count. A truncated Kaporin condition number has no such factor; changing only the objective cuts iterations on a held-out frame from 116 to 33. On a ladder of stiff tetrahedral diffusion problems ours alone fits an 8.3 ms (120 fps) frame from $N = 572$ to 3,647.

CCS Concepts

• **Computing methodologies** → **Physical simulation**; **Neural networks**.

Keywords

preconditioning, iterative solvers, hierarchical matrices, neural operators, real-time simulation

ACM Reference Format:

Carl Osborne, Minghao Guo, Crystal Owens, and Wojciech Matusik. 2026. A Neural Hierarchical-Matrix Preconditioner for Real-Time GPU Solves. In *SIGGRAPH Asia 2026 Posters (SA Posters '26)*, December 01–04, 2026, Kuala Lumpur, Malaysia. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/3829333.3847989>

1 Setting

Interactive simulations of pressure, implicit diffusion and elasticity solve $Ax = b$ with A sparse, SPD and different every frame, and at 60–120 fps the whole solve gets 8–16 ms on the GPU. The systems we target are small by solver standards and sit inside larger pipelines: the coarse levels of a multigrid cycle, the per-subdomain solves of a Schwarz preconditioner [Wu et al. 2022], and the systems that follow a change of contact set.

Two families of preconditioner meet this regime from opposite sides. Algebraic multigrid [Naumov et al. 2015] moves error between scales, so its iteration count barely grows with N ; but on an unstructured mesh it rebuilds its hierarchy for every new matrix, and that setup alone costs 12–16 ms at our sizes. Jacobi, and learned sparse factors and approximate inverses [Li et al. 2023; Yang et al. 2025], have negligible setup but act locally, so they stall when error has to cross long, high-contrast seams. We want the global transport of multigrid at the setup cost of a local method: setup is one feed-forward inference (1.8–2.5 ms). Hierarchical matrices give the operator its shape; networks built on that structure have so far learned solution maps with the hierarchy fixed in their weights [Fan et al. 2019], or compressed a learned Green’s function into an



This work is licensed under a Creative Commons Attribution 4.0 International License. *SA Posters '26, Kuala Lumpur, Malaysia*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2839-6/2026/12
<https://doi.org/10.1145/3829333.3847989>

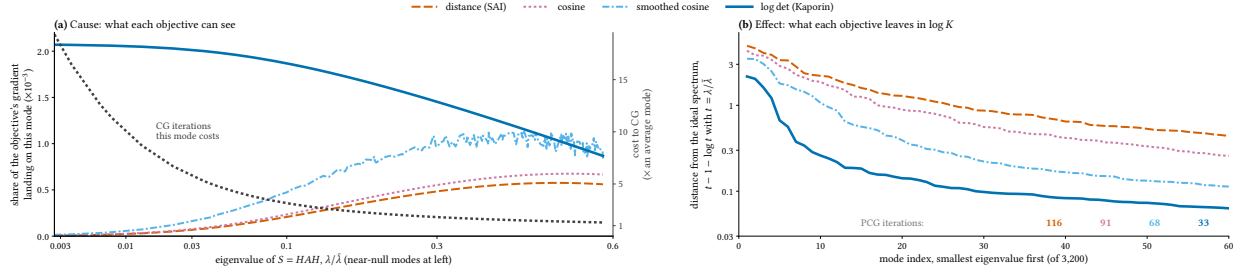


Figure 2: (a) What each objective can see on the spectrum of $S = HAH$, held-out $N = 3,306$ frame. A probe loss reaches M only through MAz , so its gradient share on a mode (left axis) carries the eigenvalue and vanishes at the near-null end, where CG spends its iterations (right axis). (b) What each objective leaves in $\log K$: each mode's term after training, with its PCG count.

$\mathcal{H}$ -matrix per operator [Xu et al. 2025]. We predict the factor for each new matrix and use it inside conjugate gradients.

2 Preconditioner

Structure and apply. Unknowns are ordered along a space-filling curve and cut into K leaves of $L = 128$ nodes. This induces a weak-admissibility $\mathcal{H}$ -matrix partition (Fig. 1, third panel): K dense diagonal blocks and $O(K)$ tiles that grow away from the diagonal.

The partition follows the inverse. For elliptic operators, even with rough, high-contrast coefficients like our barriers, a block of A^{-1} coupling two separated clusters is numerically low-rank [Bebendorf and Hackbusch 2003], and for neighbouring clusters, which weak admissibility also compresses, the rank grows with the area of their shared interface rather than their volume. We subdivide each tile by this interface-area rule: a tile spanning S leaves splits into sub-blocks of about $S^{1/3}$ leaves, so its rank grows as $S^{2/3}$, and every sub-block touching a leaf reuses that leaf's basis. These nested bases make an $\mathcal{H}$ -matrix an $\mathcal{H}^2$ -matrix [Hackbusch et al. 2000]. Storage and apply are dominated by the NL entries of the leaf blocks, linear in N at fixed L , against a dense inverse's N^2 . Applying M is a batch of dense matrix products; with no host allocation or data-dependent control flow, each PCG iteration is one CUDA graph.

Network. A width-128 encoder (an MLP and two A -weighted graph convolutions) feeds two transformer stacks over the block layout. A diagonal stack emits each dense leaf block as $F_k = U_k U_k^T$; an off-diagonal stack emits low-rank factor pairs in the shared bases from strip embeddings pooled to 32 tokens per tile of any size. Scatter-added row, column and global buffers (“highways”) route information between blocks at $O(Nd)$: a communication scaffold, not a coarse solve. A learned per-node Jacobi gate adds $\text{diag}(\lambda)D^{-1/2}$ to H , helping early in training. Inference is $\Theta(N)$ in the leaf products plus two attention launches over $K = N/L$ tokens.

SPD by construction. Mirroring the off-diagonal blocks makes H symmetric, so $M = H^2$ is SPD whenever H is nonsingular; we monitor $\lambda_{\min}(H)$ and would fall back to Jacobi, and no held-out frame has needed it. Learning $M \approx A^{-1}$ directly gave indefinite M on unseen frames, which is what motivated the squared form.

3 Training objective

What a probe loss can see. Let $S = HAH$, similar to MA . Conjugate gradients spends its iterations on the smallest eigenvalues

of S (Fig. 2a, right axis), the modes M must amplify most: on an eigenmode $Aw = \lambda w$ the ideal M acts as $Mw = w/\lambda$. Approximate inverses are trained self-supervised on probes: draw z , form Az , and compare MAz with z , whether by the distance $\|MAz - z\|_2^2$ with A scaled by $\|A\|_\infty$ (the sparse-approximate-inverse loss of Yang et al. [2025] up to norm and operand order), or by an angular loss on a block Z of $n_z = 256$ probes, $\mathcal{L}_{\cos} = 1 - \langle MAZ, Z \rangle_F / (\|MAZ\|_F \|Z\|_F)$. The distance asks the spectrum of MA to cluster at the particular value $\|A\|_\infty$; the angular loss is blind to the scale of M , so it rewards clustering wherever the spectrum can gather most tightly.

Every one of them touches M only through its product with A . With $y = MAz$, $\partial \mathcal{L} / \partial M = (\partial \mathcal{L} / \partial y)(Az)^T$, which on a mode $Aw = \lambda w$ is $O(\lambda)$; the AMz ordering picks up the same factor through A^T . A annihilates the mode before M can learn from it, and the signal that would teach M its required action is suppressed by the very factor that makes the mode expensive (Fig. 2a). The symptom is known: Chen [2025] reweights probes toward the bottom of the spectrum, and Xu et al. [2025] report a flat loss surface near singular modes. Reweighting moves probe energy, not the factor. Pre-smoothing, $Z \leftarrow (I - \omega D^{-1/2} A)Z$ twice with $\omega = 0.6$ and $D = \text{diag}(A)$, the relaxation bootstrap AMG uses to expose algebraically smooth error [Brandt et al. 2011], shifts probe energy toward the low end and cuts iterations from 91 to 68: a tilt, not a fix.

A term that never passes a probe through A . One quantity escapes this. The gradient of $-\log \det MA$ with respect to M is $-M^{-T}$: no probe, no product with A , and the same weight on every mode in $\log \lambda$, the near-null modes included.

Kaporin's condition number makes that term a training target:

$$K(S) = \frac{(\frac{1}{N} \text{tr } S)^N}{\det S}, \quad (1)$$

the ratio of the arithmetic to the geometric mean of the eigenvalues of S , raised to the N th power. It is 1 exactly when every eigenvalue is equal, it does not change under $M \rightarrow sM$, and it bounds PCG directly: $\lceil \log_2 K + \log_2(1/\epsilon) \rceil$ iterations reduce the M -norm residual by ϵ [Kaporin 1994]. Cali et al. [2023] train on K with triangular factors, whose determinant is cheap. Ours has no such shortcut, and a stochastic Lanczos estimate of $\log \det$ is too noisy to descend on.

So we truncate. Only the $p = 8$ smallest eigenvalues $\lambda_1 \leq \dots \leq \lambda_p$ of S are treated exactly; everything above them is summarized

by its mean $\bar{m}$. We minimize

$$\widehat{\log K} = N \log \left(\frac{1}{N} \text{tr } S^2 \right) - 2 \left[\sum_{i \leq p} \log \lambda_i + (N - p) \log \bar{m} \right], \quad (2)$$

an estimate of $\log K(S^2)$ whose traces come from Hutchinson probes. The tail is exact: the p smallest eigenpairs come from Chebyshev-filtered subspace iteration and are detached, so gradient reaches them through their Rayleigh quotients, and there the estimator has the log-determinant's λ -free gradient exactly. The bulk is approximate: the mean stands in for its log-determinant, so by Jensen's inequality the estimate is a lower bound, and the returning λ factor turns the term into pressure to cluster the bulk.

What changes. With the same architecture and data, PCG iterations on the held-out $N = 3,306$ frame fall from 116 (distance) to 91 (cosine), 68 (smoothed probes) and 33 (truncated Kaporin), and $\kappa(S)$ from 611 to 42. Even with the suppression gone from training, the low end remains the hardest part of the spectrum for this format to learn (Fig. 2b): after Kaporin training, the eight smallest modes still hold 26% of the $\log K$ that remains.

4 Results

Benchmark. Heat diffusion on TetWild tetrahedralizations [Hu et al. 2018] of a Thingi10K solid [Zhou and Jacobson 2016] from $N = 572$ to 13,119, with a random barrier field of contrast 10^4 resampled every frame. One network is trained per size on 60–100 frames, in about ten minutes on one GPU, and evaluated on unseen frames of the same mesh. *Baselines.* Everything stays on the GPU: at these sizes a host round-trip costs more than the solve. We compare unpreconditioned CG, GPU Jacobi, and tuned AMG, the best of an AmgX [Naumov et al. 2015] sweep over aggregation and classical hierarchies with L1-Jacobi, multicolour Gauss–Seidel and DILU smoothers. At $N \leq 1k$ the hierarchy collapses to a one-iteration direct coarse solve, and that dense solve is what AMG pays for there. *Protocol.* Every solve runs to a true relative residual of 10^{-8} .

Table 1: Per new matrix on one L40S: medians over unseen test frames; each of our columns is its own median.

N	Baselines, ms (iters)			Ours			
	CG	Jacobi	AMG	inf	solve	it (min–max)	tot
572	64.9 (1938)	6.4 (192)	42.0 (1)	1.9	1.6	24 (18–39)	3.9
1,029	145.6 (4269)	8.7 (252)	62.2 (1)	1.8	2.3	34 (33–42)	4.2
1,479	168.9 (5000)	10.4 (306)	20.3 (28)	1.9	1.5	21 (21–45)	3.3
2,072	181.1 (5000)	12.1 (332)	19.4 (26)	1.8	1.8	24 (24–27)	3.6
2,695	166.0 (4530)	12.3 (336)	19.6 (23)	1.9	3.0	39 (33–45)	4.9
3,306	135.5 (3636)	12.7 (342)	19.9 (23)	2.4	3.3	44 (36–51)	5.8
3,647	142.3 (3738)	13.3 (348)	20.7 (24)	2.5	4.8	39 (33–48)	7.4
4,899	140.9 (3598)	15.4 (394)	20.2 (24)	4.3	15.5	192 (177–195)	19.8
13,119	141.4 (3460)	24.6 (582)	23.5 (32)	9.8	50.9	306 (300–327)	60.7

Through $\sim 3.6k$ the solve alone (1.5–4.8 ms) is faster than Jacobi or AMG, and with inference the total stays inside 120 fps; beyond that, Jacobi and then AMG overtake us. Residual traces (Fig. 1, second panel) separate early, so looser tolerances widen the band.

5 The learning gap and generalization

Leaf diagonals are learned first, and the off-diagonal cores degrade first as N grows. This is a learning gap rather than a missing level

of the hierarchy: on smaller meshes, fitting the same format to $A^{-1/2}$ directly by per-tile SVD gives an oracle that converges in 3–17 iterations, so the format is not the binding constraint.

Generalization is narrow in our trained example, whose setup is tiny: one mesh, 60–100 frames and minutes of training per size. Held-out barrier fields at a trained size stay in band, but a network trained at $N = 2,072$ needs 314 iterations at 3,647, one applied below it hits the cap, and a chamber geometry at $N = 3,306$ costs 81 iterations against 44 in distribution (supplementary).

Limitations. Quality lags an exact fit of the format and collapses past $\sim 5k$, where that fit degrades mildly. The format and objective can beat local methods; training a network that does so robustly across sizes, geometries and PDEs is open.

References

- Mario Bebendorf and Wolfgang Hackbusch. 2003. Existence of $\mathcal{H}$ -Matrix Approximations to the Inverse FE-Matrix of Elliptic Operators with L^∞ -Coefficients. *Numer. Math.* 95, 1 (2003), 1–28. doi:10.1007/s00211-002-0445-6
- Achi Brandt, James Brannick, Karsten Kahl, and Ira Livshits. 2011. Bootstrap AMG. *SIAM Journal on Scientific Computing* 33, 2 (2011), 612–632. doi:10.1137/090752973
- Salvatore Cali, Daniel C. Hackett, Yin Lin, Phiala E. Shanahan, and Brian Xiao. 2023. Neural-network preconditioners for solving the Dirac equation in lattice gauge theory. *Physical Review D* 107, 3 (2023), 034508. doi:10.1103/PhysRevD.107.034508
- Jie Chen. 2025. Graph Neural Preconditioners for Iterative Solutions of Sparse Linear Systems. In *International Conference on Learning Representations*.
- Yuwei Fan, Jordi Feliu-Fabà, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núñez. 2019. A Multiscale Neural Network Based on Hierarchical Nested Bases. *Research in the Mathematical Sciences* 6, 2, Article 21 (2019). doi:10.1007/s40687-019-0183-3
- Wolfgang Hackbusch, Boris N. Khoromskij, and Stefan A. Sauter. 2000. On $\mathcal{H}^2$ -Matrices. In *Lectures on Applied Mathematics*, Hans-Joachim Bungartz, Ronald H. W. Hoppe, and Christoph Zenger (Eds.). Springer, Berlin, 9–29. doi:10.1007/978-3-642-59709-1_2
- Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panofsky. 2018. Tetrahedral Meshing in the Wild. *ACM Transactions on Graphics* 37, 4 (2018), 60:1–60:14. doi:10.1145/3197517.3201353
- Igor E. Kaporin. 1994. New Convergence Results and Preconditioning Strategies for the Conjugate Gradient Method. *Numerical Linear Algebra with Applications* 1, 2 (1994), 179–210. doi:10.1002/nla.1680010208
- Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. 2023. Learning Preconditioners for Conjugate Gradient PDE Solvers. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 19425–19439.
- M. Naumov, M. Arsaev, P. Castonguay, J. Cohen, J. Demouth, J. Eaton, S. Layton, N. Markovskiy, I. Reguly, N. Sakharnykh, V. Sellappan, and R. Strzodka. 2015. AmgX: A Library for GPU Accelerated Algebraic Multigrid and Preconditioned Iterative Methods. *SIAM Journal on Scientific Computing* 37, 5 (2015), S602–S626. doi:10.1137/140980260
- Botao Wu, Zhendong Wang, and Huamin Wang. 2022. A GPU-Based Multilevel Additive Schwarz Preconditioner for Cloth and Deformable Body Simulation. *ACM Transactions on Graphics* 41, 4 (2022), 63:1–63:14. doi:10.1145/3528223.3530085
- Tianshi Xu, Rui Peng Li, and Yuanzhe Xi. 2025. Neural Approximate Inverse Preconditioners. *arXiv preprint arXiv:2510.13034* (2025).
- Zherui Yang, Zhehao Li, Kangbo Lyu, Yixuan Li, Tao Du, and Ligang Liu. 2025. Learning Sparse Approximate Inverse Preconditioners for Conjugate Gradient Solvers on GPUs. In *Advances in Neural Information Processing Systems*.
- Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).