

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

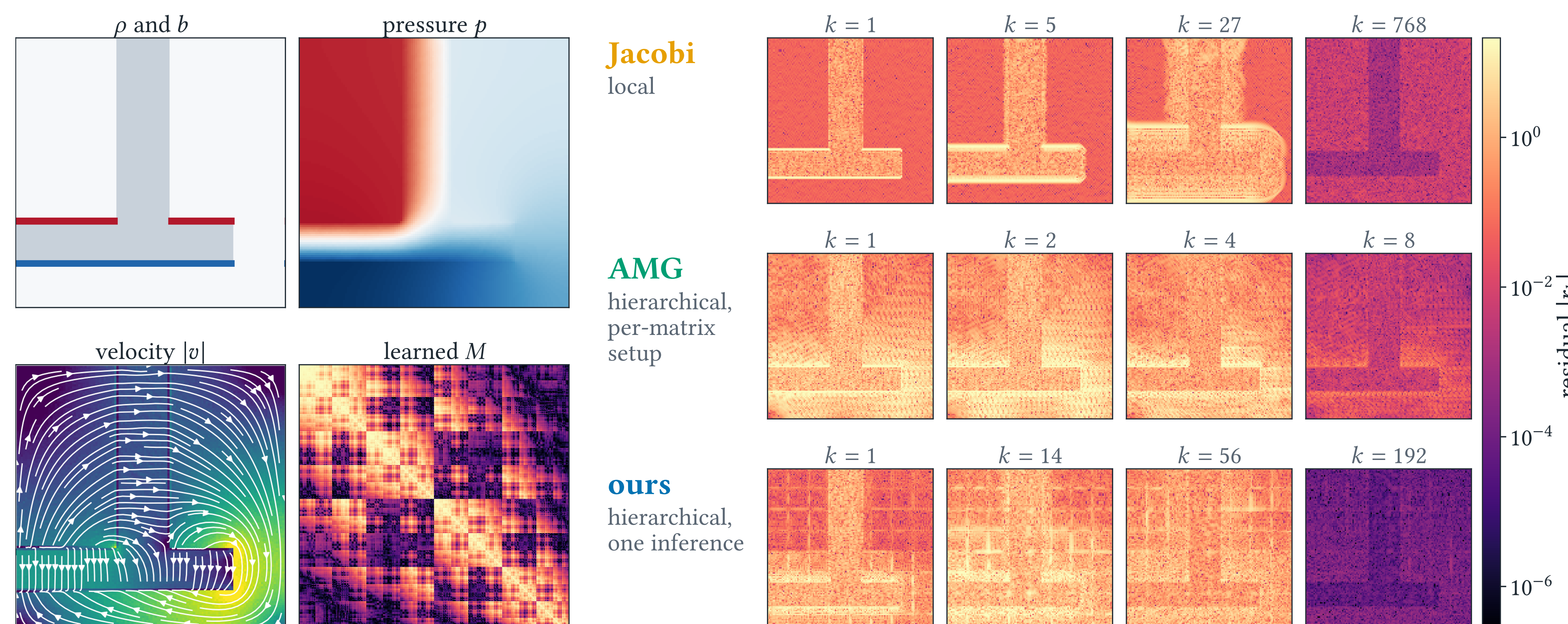
Carl Osborne Minghao Guo Crystal Owens Wojciech Matusik

MIT CSAIL · osbo@mit.edu · doi:10.1145/3829333.3847989



abstract +
supplement

SIGGRAPH Asia 2026 Posters
Kuala Lumpur, 2–4 December



The gap: setup versus reach

Interactive simulation solves a new sparse SPD system $Ax = b$ every frame, inside an 8–16 ms GPU budget.

Multigrid: moves error between scales, so its iterations barely grow with N ; but on an unstructured mesh it rebuilds its hierarchy for every new matrix, and that setup alone costs 12–16 ms.

Local methods: Jacobi and learned sparse inverses have almost no setup, but they stall when error must cross long, high-contrast seams (left).

Ours: multigrid’s global transport at a local method’s setup cost: one 1.8–2.5 ms inference predicts an SPD approximate inverse in $\mathcal{H}^2$ -matrix format.

Left: a 2D multiphase Poisson frame (density ρ with forcing b , pressure p , velocity v), the top-left 256×256 of our learned M , and the residual $|r_k|$ during PCG, each method sampled at iterations k on its own schedule (supplement, S3).

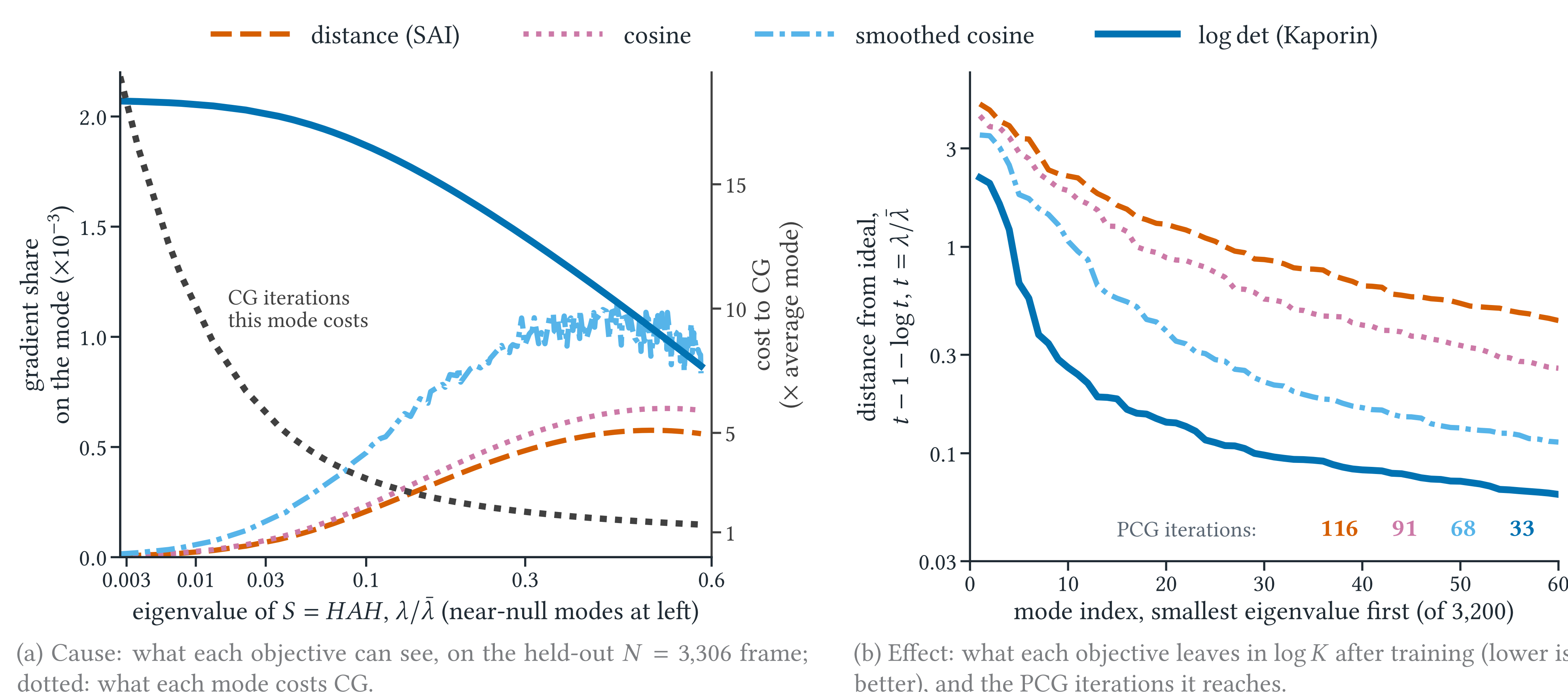
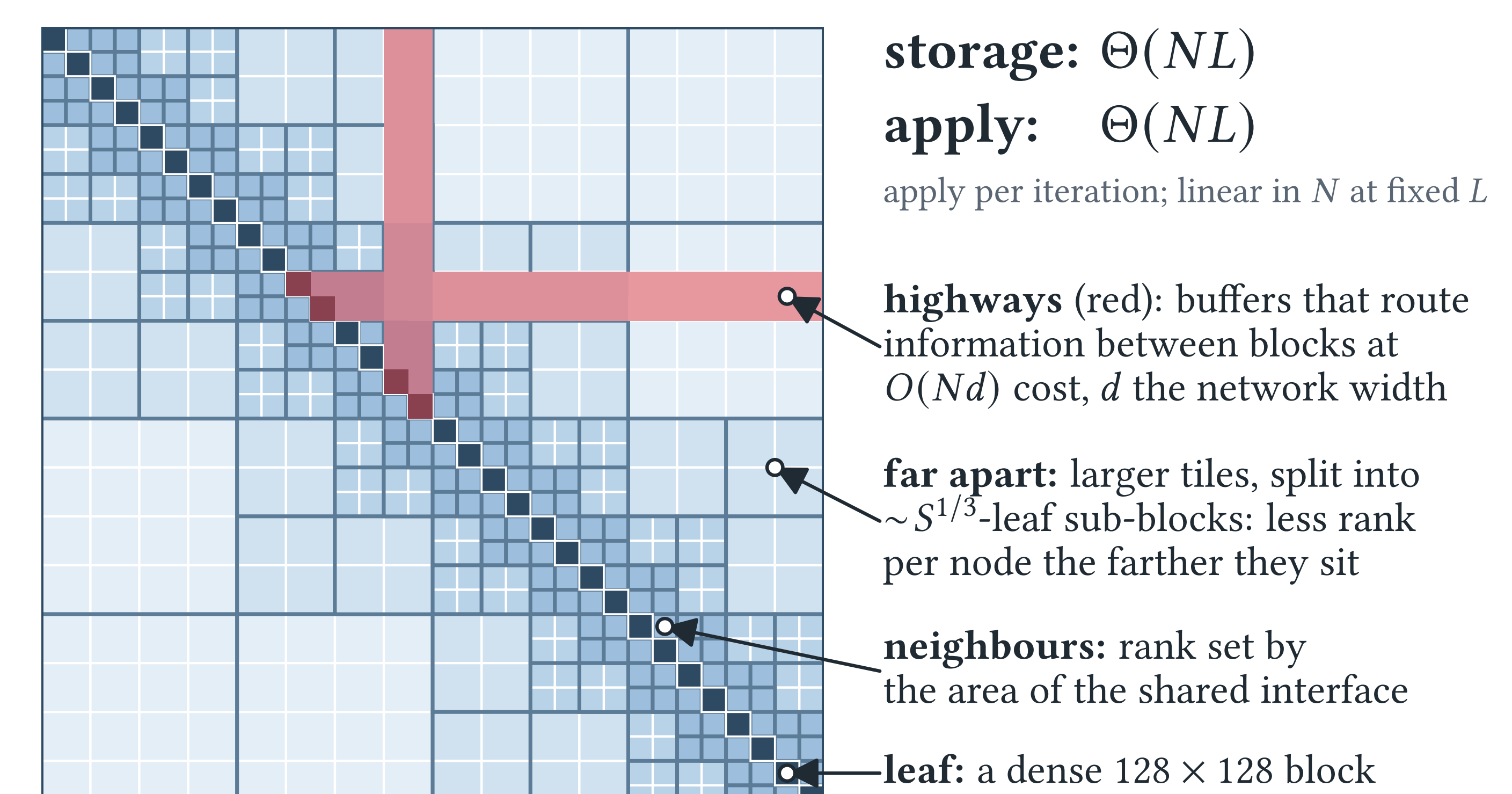
An $\mathcal{H}^2$ format that follows the inverse

Unknowns are ordered along a space-filling curve and cut into K leaves of $L = 128$ nodes: K dense diagonal blocks and $O(K)$ tiles that grow away from the diagonal (right).

Follows the inverse: on a spatially ordered 3D mesh, a block of A^{-1} that couples more distant groups of nodes, farther from the diagonal, has lower numerical rank, and neighbouring groups couple at a rank set by their shared interface [Bebendorf & Hackbusch 2003]. Tiles are split and sized to match, and nested bases make H an $\mathcal{H}^2$ -matrix [Hackbusch et al. 2000].

Network: a graph encoder and two transformer stacks over the block layout emit each leaf block as $U_k U_k^T$ and each tile in the shared bases. No shape depends on a frame’s specific matrix, so there is no dynamic allocation and each PCG iteration is one CUDA graph.

SPD by construction: mirrored tiles make H symmetric, so $M = H^2$ is SPD whenever H is nonsingular. Learning M directly gave indefinite M on unseen frames.



Probe losses
 $y = MAz$, probe $z = w$
on a mode $Aw = \lambda w$

$$\frac{\partial \mathcal{L}}{\partial M} = \frac{\partial \mathcal{L}}{\partial y} (Aw)^T = \lambda \frac{\partial \mathcal{L}}{\partial y} w^T \Rightarrow \text{carries } \lambda: \text{ blind to the near-null modes that cost CG its iterations}$$

Log-determinant
no probe, no Aw term

$$\frac{\partial (-\log \det MA)}{\partial M} = -M^{-T} \Rightarrow \text{no } \lambda: \text{ every mode weighs alike in } \log \lambda, \text{ the near-null modes included}$$

Probe losses cannot see the modes that cost CG its iterations

CG spends its iterations on the smallest eigenvalues of $S = HAH$ (similar to MA). On a mode $Aw = \lambda w$ the ideal M acts as w/λ , so these near-null modes need the most from M .

Probe losses: compare MAz with z by a distance (the SAI loss [Yang et al. 2025]) or an angle. Each touches M only through A , so its gradient vanishes on exactly those modes (a). Reweighting [Chen 2025] or smoothing [Brandt et al. 2011] the probes tilts the loss, but its gradient on each mode still scales with that mode’s eigenvalue, so the near-null modes stay faint.

Truncated Kaporin: $K(S) = (\frac{1}{N} \text{tr } S)^N / \det S$ is 1 when every eigenvalue is equal, ignores the scale of M , and bounds PCG iterations [Kaporin 1994]. We keep the 8 smallest eigenvalues exact, the rest by their mean [cf. Cali et al. 2023].

Result: same network and data, held-out $N = 3,306$ frame: $116 \rightarrow 91 \rightarrow 68 \rightarrow 33$ PCG iterations, $\kappa(S)$ from 611 to 42 (b).

Different paths: probe losses are not weaker versions of Kaporin: trained alone each yields a working preconditioner, yet a Kaporin run drives them up.

Hardest for the format: near-null modes span the whole domain, so a tiled matrix must assemble them from many low-rank pieces. Even after Kaporin training, the eight smallest modes hold 26% of the $\log K$ that remains.

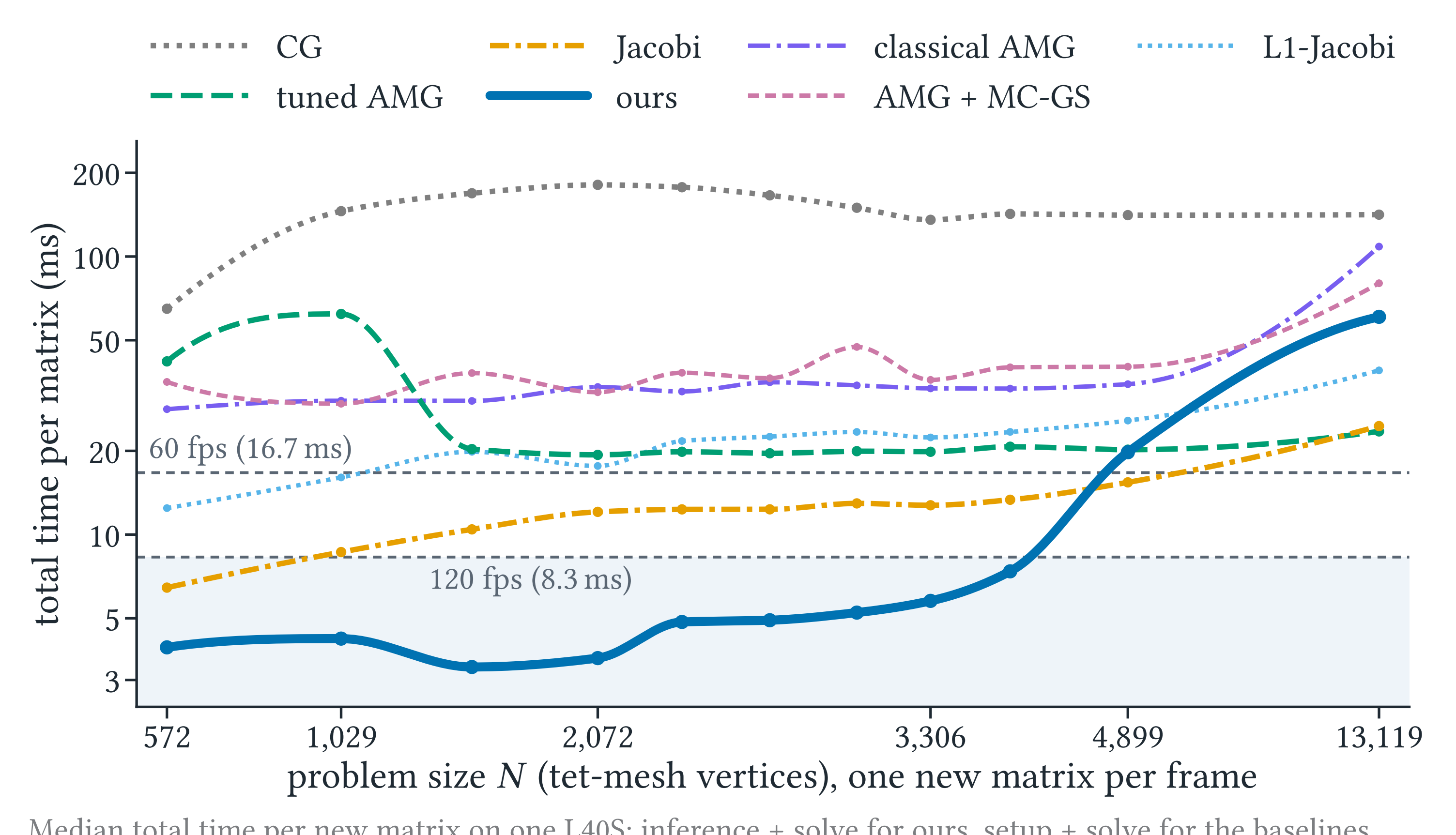
Multiscale error transport over 120 fps

Ours is the only method inside a 120 fps frame from $N = 572$ to 3,647 (right).

Benchmark: heat diffusion on TetWild meshes of a Thingi10K solid, barrier contrast 10^4 resampled every frame. One network per size, trained on 60–100 frames in about ten minutes on one GPU, tested on unseen frames. Every solve stays on the GPU and runs to a true relative residual of 10^{-8} ; AMG is the best of an AmgX sweep.

Wall clock: every time counts our inference plus solve per new matrix: 3.3–7.4 ms through $N = 3,647$, below the fastest baseline (Jacobi, 6.4–13.3 ms) at every size. Beyond that, Jacobi and then AMG overtake us.

Open: each network sees one mesh, 60–100 frames and minutes of training, so it holds up on new barrier fields but transfers poorly to other sizes and geometries. The format and the objective are per-matrix and size-agnostic; training a network that produces them robustly across sizes, geometries and PDEs is the open problem.



References · Bebendorf & Hackbusch 2003, Numer. Math. · Hackbusch, Khoromskij & Sauter 2000, Lect. Appl. Math. · Kaporin 1994, Numer. Linear Algebra Appl. · Cali et al. 2023, Phys. Rev. D · Yang et al. 2025, NeurIPS · Chen 2025, ICLR · Brandt et al. 2011, SISC · Naumov et al. 2015, SISC (AmgX) · Hu et al. 2018, ACM TOG (TetWild) · Zhou & Jacobson 2016 (Thingi10K)