A Transformer-Derived Iterative Preconditioner

Patrick Lutz* **Themistoklis Haris** **Aditya Gangrade** **Venkatesh Saligrama**
Departments of Computer Science and Electrical and Computer Engineering
Boston University
{plutz,tharis,gangrade,srv}@bu.edu

Abstract

Transformers trained on in-context prediction problems can learn known numerical algorithms, e.g., gradient descent. We go further, showing that their internal computations can reveal *new* algorithms with provable guarantees. A softmax-attention transformer trained to solve linear systems $Xw = y$ implicitly improves matrix conditioning. Tracing $XX^\top$ reveals an iterative preconditioning primitive, which we develop into RAPID, a standalone algorithm requiring no learned components. RAPID is an anytime preconditioner for symmetric positive definite (SPD) matrices, using randomized mixing and sparse local corrections to improve conditioning. Proof-of-concept experiments on synthetic and real-world problems consistently improve conditioning and reduce conjugate gradient iterations needed to reach a target accuracy. For a Haar-mixing idealization, we prove convergence to any fixed target condition number regardless of spectral distribution, with only logarithmic dependence on initial conditioning. Together, these results show how a transformer can discover a broadly applicable numerical algorithm.

1 Introduction

Transformers can learn from examples in their input at test time without updating their parameters, a capability known as in-context learning [Brown et al., 2020, Garg et al., 2022]. Studies of in-context linear regression have shown that transformers can implement familiar numerical algorithms, including gradient descent, preconditioned gradient descent, and Newton-like methods [Akyürek et al., 2023, Von Oswald et al., 2023, Ahn et al., 2023, Fu et al., 2024]. We ask:

Can studying learned transformer computations lead to new, general-purpose numerical algorithms that work independently of the transformer and beyond its training distribution?

We answer affirmatively by uncovering a new preconditioning primitive in a trained transformer's computations and developing it into a standalone numerical algorithm.

The resulting algorithm: RAPID. Preconditioning is often crucial for efficiently solving symmetric positive definite (SPD) systems $Ax = b$. Conjugate gradient (CG) avoids direct factorization, but its convergence depends strongly on the spectrum of A [Hestenes and Stiefel, 1952, Saad, 2003]. Preconditioners improve convergence by transforming the system into an equivalent, better-conditioned one [Benzi, 2002, Wathen, 2015]. Classical approaches often construct approximate factorizations, inverses, or surrogate operators [Meijerink and van der Vorst, 1981, Benzi et al., 1996].

We introduce RAPID (Randomized Anytime Preconditioning by Iterative Dynamics), which improves global conditioning through simple local corrections. Its key idea is to repeatedly randomize the coordinate system, exposing different matrix interactions that can be corrected locally. Specifically, RAPID alternates two steps. First, random orthogonal mixing redistributes matrix interactions without changing the spectrum. Second, each row selects its smallest off-diagonal entry and applies a sparse

*corresponding author plutz@bu.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

correction that acts to weaken the selected coupling. Repeating these steps allows local corrections to progressively improve global conditioning without explicitly computing eigenvectors or matrix inverses.

RAPID accumulates these transformations into a preconditioning factor P_t, yielding $A_t = P_t A P_t^\top$. Each iteration produces a valid preconditioner, making RAPID an anytime algorithm. Fast structured mixing and sparse corrections enable efficient, highly parallel dense implementations. The underlying update is derived from the trained transformer's internal computation, as described next.

Algorithm discovery. We train a softmax-attention transformer on in-context linear systems $Xw = y$, where X need not be SPD. Layerwise orthogonal symmetry interventions [Lutz et al., 2026] allow us to trace the evolution of the Gram matrix $G_\ell = X_\ell X_\ell^\top$ during prediction. We observe that its conditioning improves and identify a simple learned congruence update responsible for this effect. Crucially, this update depends only on G_ℓ, allowing RAPID to operate directly on arbitrary SPD matrices without the cost of forming a Gram matrix.

The connection to RAPID follows naturally: the symmetry intervention introduces random rotations, while learned attention applies local corrections that tend to weaken strong matrix interactions. The residual update applies the same row transformation to examples and labels, inducing the congruence update. To obtain an efficient algorithm, RAPID takes the low-temperature limit of softmax to select sparse off-diagonal row-min corrections and replaces dense rotations with fast structured mixing [Yu et al., 2016]. The resulting algorithm requires no learned components.

Results and guarantees. We evaluate RAPID on four synthetic spectral families and 31 SuiteSparse SPD matrices, well beyond the transformer's training distribution. RAPID substantially improves conditioning across all four families and, remarkably, on every SuiteSparse matrix at every tested step count. In contrast, randomized Nyström preconditioning is more sensitive to spectral structure. These improvements also translate into substantially fewer conjugate gradient (CG) iterations to reach a fixed accuracy. Progressive gains with additional RAPID steps illustrate its anytime nature.

Theoretically, we prove that an idealized version of RAPID, using Haar mixing and block-restricted row-min corrections, reaches any fixed target condition number in $O(d \log d \log \kappa(A))$ expected iterations. This holds for any SPD matrix A, regardless of its eigenvalue distribution. At this iteration scale, RAPID's per-step cost of $O(d^2 \log d)$ corresponds to near-cubic setup complexity, comparable to classical dense factorization. Unlike direct factorization, however, RAPID can stop early to balance setup cost against preconditioner quality. Although the iteration-complexity guarantee remains limited to an idealized variant, experiments show that its assumptions do not accelerate conditioning on tested spectra, suggesting they facilitate the proof rather than the algorithm's effectiveness.

Contributions. We make three core contributions:

- We discover a new preconditioning primitive in a trained transformer: a solution-preserving update to the linear system $Xw = y$ that improves the conditioning of the Gram matrix $XX^\top$.
- We develop this mechanism into RAPID, a standalone anytime preconditioner for arbitrary SPD matrices, requiring no learned components and using efficient, parallelizable iterations.
- We establish theoretical and empirical evidence of RAPID's effectiveness: an idealized variant provably reaches any fixed target condition number for arbitrary SPD matrices, while dense experiments demonstrate broad conditioning gains and substantially faster CG convergence.

2 Related Work

Transformers as algorithm learners. For linear prediction, trained transformers approximate least-squares predictors and learn familiar methods resembling gradient descent, ridge regression, and second-order methods [Garg et al., 2022, Akyürek et al., 2023, Von Oswald et al., 2023, Ahn et al., 2023, Zhang et al., 2024, Fu et al., 2024]. In other settings, trained transformers also learn temporal-difference updates and graph-connectivity algorithms [Wang et al., 2025, Ye et al., 2026]. Beyond known methods, transformers have revealed novel optimization strategies for noisy linear regression [Vladymyrov et al., 2024] and iterative methods for masked matrix completion [Lutz et al., 2025]. We extend this direction by extracting a new conditioning primitive and developing a standalone preconditioner for arbitrary SPD matrices, with theoretical guarantees, broad conditioning gains and improved CG convergence.

Preconditioning for SPD systems. Conjugate gradient (CG) solves SPD systems $Ax = b$ without direct factorization, but its convergence depends strongly on the spectrum of A [Hestenes and Stiefel, 1952, Saad, 2003]. Classical preconditioners seek to improve this spectrum through incomplete Cholesky factorizations [Meijerink and van der Vorst, 1981], sparse approximate inverses [Benzi et al., 1996], or algebraic multigrid hierarchies [Vaněk et al., 1996]. Rather than constructing such approximations, RAPID iteratively improves conditioning through randomized mixing and sparse local corrections, accumulating a factor P_t with $A_t = P_t A P_t^\top$. Each iteration yields a valid SPD preconditioned system, allowing RAPID to stop at any point without requiring a complete factorization or inverse approximation.

Randomized preconditioning and structured transforms. Randomized preconditioners often construct approximations of the original problem. For least squares, sketching methods use subsampled orthogonal transforms or Gaussian projections to build smaller systems for factorization [Avron et al., 2010, Meng et al., 2014]; for SPD systems, randomized Nyström methods construct low-rank approximations for preconditioned CG [Frangella et al., 2023]. More broadly, fast randomized Hadamard transforms provide efficient alternatives to dense random projections [Ailon and Chazelle, 2009, Le et al., 2013, Yu et al., 2016]. RAPID uses fast randomized Hadamard mixing, but applies it repeatedly to expose matrix interactions for sparse local corrections rather than to construct a one-shot approximation.

Learned preconditioners and neural solvers. Recent work has used neural networks to accelerate iterative linear solvers by learning algebraic multigrid components, approximate factorizations, or preconditioners for conjugate-gradient methods [Luz et al., 2020, Li et al., 2023, Trifonov et al., 2026]. These approaches evaluate a trained network to construct preconditioners for new systems. In contrast, we use the transformer only to discover a numerical algorithm. Once extracted, RAPID operates directly on arbitrary SPD matrices through explicit randomized updates, without learned components or neural inference at solve time.

Symmetry and equivariance in neural networks. When task symmetries are known, equivariant architectures provide a way to encode them as inductive biases, restricting the learned function class and improving generalization [Cohen and Welling, 2016, Kondor and Trivedi, 2018, Finzi et al., 2021]. Layerwise symmetry interventions instead use random, task-preserving coordinate changes to reveal learned transformer computations [Lutz et al., 2026]. This technique was originally developed for in-context classification using permutation symmetries; we adapt their identification method to the orthogonal symmetries of linear systems, revealing the conditioning primitive underlying RAPID.

3 Algorithm discovery through in-context learning

In this section, we use a transformer trained on in-context linear systems to discover a numerical preconditioning primitive. We train an attention-only model to predict from examples satisfying $Xw = y$, using a layerwise symmetry intervention to make its learned computation identifiable. We extract a two-scalar update that preserves the linear system and evolves the Gram matrix $XX^\top$ by congruence, substantially improving its conditioning. This primitive motivates RAPID, developed in Section 4 as a standalone SPD preconditioner with no learned components.

3.1 Transformer setup and symmetry intervention

In-context linear systems. Following Garg et al. [2022], we train transformers on synthetic linear systems $Xw = y$, where $X \in \mathbb{R}^{d\times d}$ and $w \in \mathbb{R}^d$, $d = 15$. Each task samples X and a query $x_q \in \mathbb{R}^d$ with independent standard Gaussian entries, and $w \sim \mathcal{N}(0, I_d/d)$, then sets $y = Xw$. The model observes (X, y, x_q), but not w, and is trained to predict $x_q^\top w$. Since w is resampled for every task, prediction requires solving a new instance rather than memorizing a fixed solution.

Transformer architecture. We encode each task as

$$Z_0 = \begin{bmatrix} X & y \\ x_q^\top & 0 \end{bmatrix},$$

with the query prediction read from the bottom-right entry of Z_L (with a negative sign) after processing through L transformer layers. The attention-only transformer applies

$$Z_{\ell+1} = Z_\ell + \mathrm{Attn}_\ell(Z_\ell), \qquad \mathrm{Attn}_\ell(Z) = \sigma\big(ZW_{Q,\ell}W_{K,\ell}^\top Z^\top + M\big)\, ZW_{V,\ell}W_{P,\ell}, \tag{1}$$

where σ is row-wise softmax and M prevents attention to the query token as a key. The model is trained by squared prediction loss, without any supervision on conditioning. Training details are given in Appendix A.1.

Layerwise symmetry intervention. The linear system has an orthogonal symmetry: for any $U, V \in O(d)$, the group of $d \times d$ orthogonal matrices, the transformation

$$\widetilde{X} = UXV, \qquad \widetilde{y} = Uy, \qquad \widetilde{x}_q = V^\top x_q$$

preserves the problem, since $\widetilde{X}(V^\top w) = \widetilde{y}$ and $\widetilde{x}_q^\top (V^\top w) = x_q^\top w$. Thus the same prediction problem can be represented in different example and feature coordinates.

To make the learned computation identifiable, we adapt the layerwise symmetry intervention of Lutz et al. [2026]. At each layer, we sample independent $U_\ell, V_\ell \sim \mathrm{Haar}(O(d))$, extend them to fix the query and label coordinates as

$$\bar{U}_\ell = \begin{bmatrix} U_\ell & 0 \\ 0 & 1 \end{bmatrix}, \qquad \bar{V}_\ell = \begin{bmatrix} V_\ell & 0 \\ 0 & 1 \end{bmatrix}.$$

and evaluate attention in the transformed coordinates before mapping the residual update back:

$$Z_{\ell+1} = Z_\ell + \bar{U}_\ell^\top \, \mathrm{Attn}_\ell\big(\bar{U}_\ell Z_\ell \bar{V}_\ell\big) \, \bar{V}_\ell^\top. \tag{2}$$

This is the transformer architecture we train and analyze throughout this section. The intervention discourages reliance on arbitrary coordinate choices, making the learned computation easier to identify, without imposing any conditioning objective or preconditioning algorithm.

3.2 Extracting a conditioning primitive

The transformer achieves increasingly accurate predictions with depth, following an error profile distinct from GD and CG at matched iteration budgets (Appendix A.2). This suggests that the model learns a distinct, nontrivial procedure rather than simply unrolling either classical solver. We therefore examine its trained layers to identify the underlying computation.

Identifying the learned update. The attention layers depend on the weights only through the products $W_{QK,\ell} = W_{Q,\ell} W_{K,\ell}^\top$ and $W_{VP,\ell} = W_{V,\ell} W_{P,\ell}$. These products exhibit a simple dominant structure: $W_{QK,\ell}$ is concentrated on the feature-feature identity block, while $W_{VP,\ell}$ is approximately a scalar identity (Figure 1, left). We therefore approximate each layer by

$$W_{QK,\ell} \approx \gamma_\ell \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix}, \qquad W_{VP,\ell} \approx \alpha_\ell I_{d+1}, \tag{3}$$

where γ_ℓ and α_ℓ are the mean diagonal entries of the respective identity blocks.

Substituting these approximated weight products into the intervened transformer gives an explicit iteration. Writing X_ℓ, y_ℓ for the context state and $G_\ell = X_\ell X_\ell^\top$ for its Gram matrix, each layer computes

$$K_\ell = \sigma\big(\gamma_\ell U_\ell G_\ell U_\ell^\top\big), \qquad C_\ell = I + \alpha_\ell U_\ell^\top K_\ell U_\ell, \tag{4}$$

and updates examples and labels by the same row transformation:

$$X_{\ell+1} = C_\ell X_\ell, \qquad y_{\ell+1} = C_\ell y_\ell. \tag{5}$$

To assess the approximation, we replace the learned weight products in every layer by their two-scalar forms in Eq. (3) and compare predictions against the trained transformer on the same held-out tasks, using identical random rotations. The predictions agree closely ($R^2 > 0.98$), with a test MSE change of only 6.4×10^{-3} (Appendix A.3). This confirms that the two-scalar update faithfully reproduces the trained transformer's predictive behavior.

A solution-preserving congruence update. The key property of Eq. (5) is that it preserves the linear-system relation: if $y_\ell = X_\ell w$, then $y_{\ell+1} = C_\ell X_\ell w = X_{\ell+1} w$. When C_ℓ is invertible, the transformed system has the same solution. Moreover, the Gram matrix evolves by congruence:

$$G_{\ell+1} = C_\ell G_\ell C_\ell^\top. \tag{6}$$

Crucially, C_ℓ depends only on G_ℓ and the random rotation U_ℓ, not on the labels or query. Thus the extracted iteration defines a self-contained conditioning mechanism on SPD matrices, independent of the original prediction task.

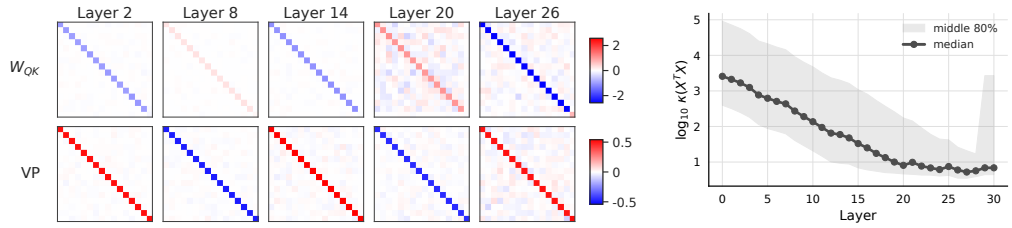

Figure 1: **Extracting a conditioning primitive.** *Left:* Representative learned weight products $W_{QK,\ell}$ and $W_{VP,\ell}$, showing the dominant two-scalar structure in Eq. (3). *Right:* Median condition number of the context Gram matrix across layers of the trained transformer, decreasing from roughly 3,000 to below 10 at the best layer.

Empirical conditioning and the learned sign rule. The trained dynamics substantially improve conditioning: the median condition number of G_ℓ decreases from roughly 3,000 to below 10 at the best layer (Figure 1, right). The two-scalar approximation exhibits similar behavior.

Which layers improve conditioning? We find that layers satisfying

$$\operatorname{sign}(\alpha_\ell) \neq \operatorname{sign}(\gamma_\ell), \qquad |\alpha_\ell| < 1 \tag{7}$$

typically reduce the condition number. Across runs, 80% of layers satisfy this rule, and in the median such layer, 95.5% of test instances experience a condition-number decrease (Appendix A.4).

Together, these results identify a Gram-dependent conditioning primitive within the transformer's learned prediction procedure. The symmetry intervention provides random coordinate mixing, while the learned attention determines the row corrections. The next section develops this primitive into RAPID by replacing dense rotations and soft attention with efficient structured mixing and sparse updates, yielding a standalone preconditioner without learned components at solve time.

4 RAPID: Randomized Anytime Preconditioning by Iterative Dynamics

Section 3 identified a Gram-dependent congruence update learned by a transformer trained only for prediction. We now develop this primitive into RAPID, a standalone preconditioner for explicit SPD systems $Ax = b$. RAPID retains random coordinate mixing and row-wise corrections, but replaces dense rotations and soft attention with efficient structured transforms and sparse updates. It requires no trained model or learned parameters at solve time.

4.1 Constructing the RAPID update

Randomized mixing. Starting from $A_0 = A$, RAPID repeatedly applies an orthogonal change of coordinates,

$$\widetilde{A}_t = Q_t A_t Q_t^\top .$$

This preserves the eigenvalues, but exposes different off-diagonal interactions that can be corrected locally. To obtain this mixing efficiently, RAPID replaces dense random rotations with products of randomized Walsh–Hadamard transforms:

$$Q_t = HS_{t,k} \cdots HS_{t,2} HS_{t,1}, \tag{8}$$

where H is the normalized Walsh–Hadamard matrix and $S_{t,j}$ are independent diagonal matrices with random ± 1 entries. Such transforms spread information across coordinates similarly to dense random rotations while preserving orthogonality [Ailon and Chazelle, 2009, Le et al., 2013, Yu et al., 2016]. For fixed k, they cost $O(d \log d)$ per vector, rather than $O(d^2)$ for a dense rotation.

From learned attention to sparse row corrections. The learned conditioning rule (7) identifies opposite signs for α_ℓ and γ_ℓ, with $|\alpha_\ell| < 1$. RAPID fixes the convention of positive residual steps $\alpha_\ell > 0$, corresponding to negative attention scales $\gamma_\ell < 0$ in the learned model; the opposite convention analogously motivates negative steps with row-maximum selection.

With a negative attention scale, softmax favors smaller entries of the mixed Gram matrix. For computational efficiency, RAPID replaces dense soft attention with selection of the smallest off-diagonal entry in each row:

$$\tau_i(B) := \arg\min_{j \neq i} B_{ij}, \qquad \mathcal{R}(B)_{ij} := \mathbf{1}\{j = \tau_i(B)\}. \tag{9}$$

We exclude the diagonal to target interactions between different rows rather than simply rescaling individual rows. The resulting one-sparse-per-row operator reduces the cost of the congruence update from $O(d^3)$ to $O(d^2)$. This sparsification is a design choice motivated by the learned attention.

Writing $R_t = \mathcal{R}(\widetilde{A}_t)$, RAPID applies

$$C_t = I + \alpha_t R_t, \qquad A_{t+1} = C_t \widetilde{A}_t C_t^\top. \tag{10}$$

This retains the congruence structure of the learned iteration, now with an explicit sparse row operator.

Why does this improve conditioning? The intuition is clearest for a matrix with a single exceptional eigenvalue. Consider $\widetilde{A} = I + (\lambda - 1)vv^\top$, where $\lambda > 0$, $\lambda \neq 1$, and $\|v\| = 1$, so all eigenvalues equal 1 except λ. Adding a small positive fraction α of row j to row i, together with the corresponding column operation, changes the exceptional eigenvalue to first order by

$$\lambda(\alpha) = \lambda + \frac{2\alpha\lambda}{\lambda - 1}\widetilde{A}_{ij} + O(\alpha^2). \tag{11}$$

Thus, for sufficiently small $\alpha > 0$, a negative $\widetilde{A}_{ij}$ moves λ toward 1: decreasing it when $\lambda > 1$ and increasing it when $\lambda < 1$. Random mixing can expose negative off-diagonal entries even when none are present initially, and row-minimum selection favors such entries when available. For general matrices, individual corrections need not improve conditioning; Section 6 establishes a global guarantee for an idealized variant. The first-order calculation is detailed in Appendix C.9.

Step-size selection. Each row of R_t selects one column, but multiple rows may select the same column. Let

$$m_t = \max_j \sum_i (R_t)_{ij}$$

denote the maximum column load. Since $\|R_t\|_{\mathrm{op}} = \sqrt{m_t}$, we use

$$\alpha_t = \frac{s_\alpha}{\sqrt{m_t d \log d}}, \tag{12}$$

where $s_\alpha > 0$ is a tunable scale. This keeps the overall update magnitude $\alpha_t \|R_t\|_{\mathrm{op}} = s_\alpha / \sqrt{d \log d}$ independent of the column load, controlling the effect of simultaneous row corrections. Section 6 provides a complementary motivation based on the expected improvement in conditioning.

4.2 RAPID as an anytime preconditioner

Accumulating the preconditioner. RAPID maintains a factor P_t satisfying

$$P_0 = I_d, \qquad P_{t+1} = C_t Q_t P_t, \qquad A_t = P_t A P_t^\top. \tag{13}$$

Thus the random basis changes are accumulated rather than reversed after each correction.

To ensure a valid preconditioner at every iteration, we choose s_α so that $\alpha_t < 1$. Indeed, R_t is row-stochastic, so its eigenvalues have magnitude at most 1. Hence $C_t = I + \alpha_t R_t$ is invertible whenever $0 < \alpha_t < 1$. Each congruence update therefore preserves positive definiteness.

After any T iterations, RAPID produces the equivalent SPD system

$$A_T z = P_T b, \qquad x = P_T^\top z,$$

which can be solved using conjugate gradients. Since every iteration yields a valid factor P_t, RAPID is an anytime preconditioner: the iteration budget trades setup cost against the conditioning achieved. The factor can also be reused for multiple right-hand sides.

Complete algorithm. Algorithm 1 summarizes the practical RAPID iteration, using structured orthogonal mixing and row-minimum selection.

Computational cost and theoretical scope. Row-minimum selection and the sparse congruence update each cost $O(d^2)$. Applying a fixed number of Walsh–Hadamard transforms to a dense matrix costs $O(d^2 \log d)$, which dominates the per-iteration complexity. These operations are highly parallelizable and require no matrix factorization. For dimensions that are not powers of two, the implementation uses padding (Appendix B).

Section 5 evaluates the practical algorithm, while Section 6 analyzes a related variant using Haar mixing and block-restricted row selection.

Algorithm 1 RAPID: Practical preconditioning algorithm

Input: $A \succ 0$, iterations T, mixer depth k, scale $0 < s_\alpha < \sqrt{d \log d}$
$A_0 \leftarrow A, \quad P_0 \leftarrow I_d$
for $t = 0, \ldots, T-1$ **do**
  Draw Q_t via (8); $\widetilde{A}_t \leftarrow Q_t A_t Q_t^\top$ ▷ random mixing
  $R_t \leftarrow \mathcal{R}(\widetilde{A}_t)$ defined as in (9) ▷ row selection
  $m_t \leftarrow \max_j \sum_i (R_t)_{ij}$; $\alpha_t \leftarrow s_\alpha / \sqrt{m_t d \log d}$ ▷ compute step-size
  $C_t \leftarrow I + \alpha_t R_t$ ▷ sparse correction
  $A_{t+1} \leftarrow C_t \widetilde{A}_t C_t^\top$; $P_{t+1} \leftarrow C_t Q_t P_t$ ▷ update operator
end for
return P_T, A_T ▷ verifies $A_T = P_T A P_T^\top$

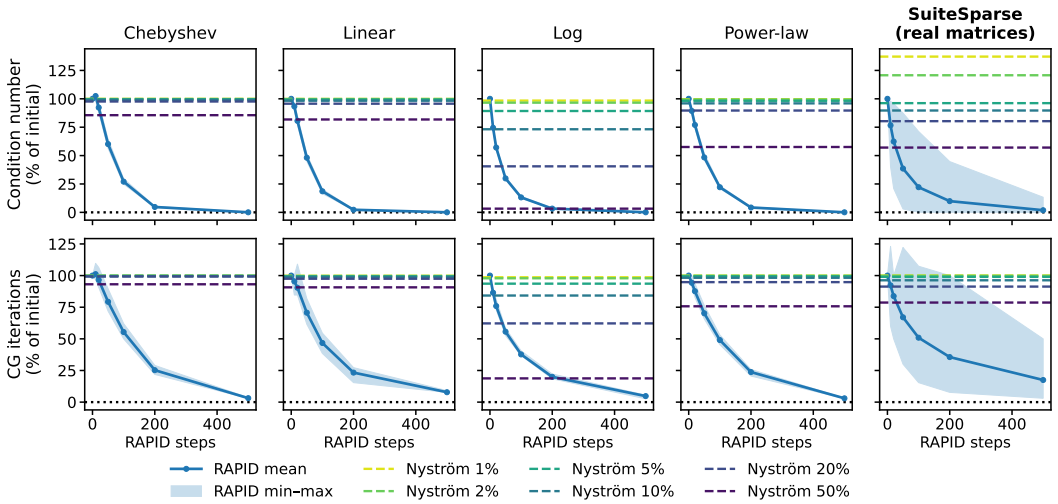

Figure 2: **Conditioning and CG convergence across synthetic and real-world SPD matrices.** Top: condition number relative to the Jacobi-scaled system. Bottom: CG iterations required to reach the common residual tolerance, relative to Jacobi-preconditioned CG. Lower is better in both rows. Blue curves show RAPID means and shaded bands the minimum and maximum; dashed lines show mean Nyström results at different sketch sizes. Each synthetic panel contains 100 independent matrices; SuiteSparse contains 31. The corresponding absolute values are reported in Appendix B.

5 Experimental Evaluation

In this section, we show that RAPID improves conditioning across SPD matrices well beyond the transformer's training distribution, and that these improvements translate into faster CG convergence. We use the practical algorithm of Section 4, with Walsh–Hadamard mixing and global row-minimum selection. No trained model is used in these experiments.

Experimental protocol. We generate synthetic matrices $A = U \Lambda U^\top$, with Haar-random U, dimension $d = 1024$, and condition number $\kappa = 10^4$. We consider four distinct spectral distributions: Chebyshev, linear, logarithmic, and power-law, with 100 independent matrices per family. We also evaluate 31 SPD matrices from SuiteSparse [Kolodziej et al., 2019], selected independently of RAPID's performance by dimension ($100 \leq d \leq 6000$) and post-Jacobi condition number (10^3–10^6). All matrices are treated as dense operators. Details appear in Appendix B.

After Jacobi scaling, we compare RAPID ($s_\alpha = 10$) with randomized Nyström preconditioning [Frangella et al., 2023], which uses a one-shot low-rank approximation rather than RAPID's iterative congruence updates (sketch sizes 1%–50% of d). We report condition numbers relative to the Jacobi-scaled matrix A_0 and CG iterations relative to Jacobi-preconditioned CG. All methods use the same right-hand side and stopping criterion $\|A_0 \widehat{x} - b_0\|_2 / \|b_0\|_2 \leq 0.002$, evaluated in the original Jacobi-scaled coordinates, ensuring a common accuracy target.

Broad improvements in conditioning. Figure 2 (top) shows that RAPID substantially improves conditioning across all four spectral families. In contrast, Nyström is strongly dependent on the eigenvalue distribution: it is particularly effective on the logarithmic spectrum, but achieves much smaller improvements on Chebyshev and linear spectra, even with large sketches. Remarkably, at every tested step count T, RAPID reduces the condition number of *all 31* SuiteSparse matrices, not just the average. These results demonstrate that RAPID generalizes well beyond the Gaussian Gram matrices used during transformer training.

Fewer CG iterations at fixed accuracy. Figure 2 (bottom) shows that RAPID's conditioning gains also translate into substantially fewer CG iterations to reach the same accuracy, across all four synthetic families and, on average, across SuiteSparse matrices. This is significant because CG convergence depends on the full spectrum, not just the condition number. Thus, RAPID not only improves conditioning broadly, but also makes these systems easier to solve iteratively. All solvers reached the target accuracy, except for Nyström preconditioning with a 1% sketch which failed on a single SuiteSparse instance.

Practical considerations. The progressive improvements with increasing T illustrate RAPID's anytime nature, allowing a trade-off between setup cost and preconditioner quality. A RAPID step costs $O(d^2 \log d)$, matching a dense CG iteration up to logarithmic factors. In practice, RAPID can be more expensive per step, as its operations do not benefit from the same highly optimized GPU implementations available for CG's matrix–vector products. The setup cost may nevertheless pay off through fewer CG iterations, especially when amortized across multiple right-hand sides. A definitive runtime comparison requires optimized implementations of both methods and is left for future work.

6 Analyzing the Extracted Preconditioner

We analyze an idealized version of RAPID using two simplifications relative to Algorithm 1: (i) Haar-random mixing, which enables an exact calculation of a certain expected 'row-min drift' term, and (ii) block-restricted row selection, which bounds the maximum column load of R_t. These assumptions enable our proof, but are not required by the practical algorithm. We first establish the conditioning guarantee, then examine the empirical effect of both simplifications.

6.1 Conditioning guarantee under simplifying assumptions

Our proof tracks a smooth approximation to the log-condition number and shows that it decreases in expectation until the matrix becomes well-conditioned. The key is to balance two effects: the favorable first-order drift induced by row-min selection, and a second-order error that depends on how many rows select the same column. Haar mixing allows us to characterize the drift, while blocking controls the error.

Specifically, we take $Q_t \sim \mathrm{Haar}(O(d))$ and replace the global row-min operator $\mathcal{R}$ with a block-restricted version. Partition the indices into blocks of size at most b, and let $B(i)$ denote the block containing i. Define

$$\tau_i^b(A) := \arg\min_{j \in B(i)\setminus\{i\}} A_{ij}, \qquad \mathcal{R}_b(A)_{ij} := \mathbf{1}\{j = \tau_i^b(A)\}. \tag{14}$$

Here $3 \le b \le d$ is the block size; setting $b = d$ recovers the original global row-min operator $\mathcal{R}$. By restricting selection to each row's block, we ensure that no column is selected more than $b-1$ times. We now establish the two bounds and combine them to obtain the conditioning guarantee. Full proofs appear in Appendix C.

Potential Function & First-order analysis. To measure progress toward a well-conditioned matrix, we construct the following potential function:

$$\Phi_\beta(A) := \frac{1}{\beta} \log \frac{\mathrm{tr}(A^\beta)}{d} + \frac{1}{\beta} \log \frac{\mathrm{tr}(A^{-\beta})}{d},$$

where $\beta \ge 1$. This scale and basis invariant potential essentially corresponds to the log of the ratio of a softmax and softmin of the spectrum (with temperature β) and satisfies

$$0 \le \Phi_\beta(A) \le \log \kappa(A) \le \Phi_\beta(A) + \frac{2 \log d}{\beta}.$$

We will set $\beta = \Theta(\log(d))$ below, making the approximation error constant. The key lemma capturing the decay of this potential arises from a Taylor expansion, where the curvature term is controlled by submultiplicativity of Schatten p-norms.

Lemma 1. *Let $m_t := \|R_t\|_1 = \max_j \sum_i (R_t)_{ij}$ be the maximal column load of the matrix R_t. If $\alpha\sqrt{m_t} \le 1/2$, then*

$$\Phi(A_{t+1}) \le \Phi(A_t) + 2\alpha f(R_t, \widetilde{A}_t) + 12\alpha^2\beta m_t$$

where

$$f(R_t, \widetilde{A}_t) := \left(\frac{\mathrm{tr}(R_t \widetilde{A}_t^{\beta})}{\mathrm{tr}(\widetilde{A}_t^{\beta})} - \frac{\mathrm{tr}(R_t \widetilde{A}_t^{-\beta})}{\mathrm{tr}(\widetilde{A}_t^{-\beta})} \right).$$

Row-min drift. The key question is whether the first-order term $f(R_t, \widetilde{A}_t)$ is negative in expectation, so that it outweighs the second-order growth. Haar mixing allows us to characterize this term exactly.

Lemma 2. *Suppose $Q_t \sim \mathrm{Haar}(O(d))$ independently of the history, and that $R_t = \mathcal{R}_b(\widetilde{A}_t)$. Then*

$$\mathbb{E}\left[f(R_t, \widetilde{A}_t) \middle| A_t \right] = \frac{-\nu_{b,d}}{\sqrt{d-1}} S_\beta(A_t), \tag{15}$$

where $\nu_{b,d} \ge \sqrt{\log(b)}$, and S_β depends on the spectral properties of A_t and d. In particular, if $\beta \ge \log(8d)/\log(8/7)$ and $\kappa(A_t) \ge 2$, then $S_\beta(A_t) \ge 1/4$.

$S_\beta(A_t)$ is a measure of the correlation of spectrum of A with its βth power. The lemma guarantees a negative first-order drift of magnitude $\alpha \cdot \Omega(\sqrt{\log(b)/d})$ when $\kappa(A_t) \gg 1$. This bound is conservative: $S_\beta(A)$ can be as large as $\sqrt{d}$ when A or A^{-1} has a highly spiked spectrum (Appendix C.8).

Main Result. With control on both the drift and the second order growth, as long as $\kappa(A_t) \ge 2$, and the column-load m_t is not too large, we can observe a decay in the potential Φ_β. Notice that this column load is directly controlled by using $\mathcal{R}_b$, since due to the blocking structure, each column can have at most $b-1$ ones. Exploiting this yields a concrete bound on $\mathbb{E}[\Phi_\beta(A_t) - \Phi_\beta(A_0)]$, which allows us to control the expected time to conditioning.

Theorem 3. *Suppose $Q_t \sim \mathrm{Haar}(O(d))$, and $R_t = \mathcal{R}_b(\widetilde{A}_t)$, with $b = O(\log(d))$. Let $\beta_* := \log(d)/\log(8/7)$, and set $\alpha = \nu_{b,d}/\beta_*(b-1)\sqrt{d-1}$. Let $\tau_* := \inf\{t : \kappa(A_t) \le 2\}$. Then*

$$\mathbb{E}[\tau_*] = O\left(d \cdot \mathrm{polylog}(d) \cdot \log(\kappa(A_0))\right).$$

Thus, the idealized RAPID variant achieves conditioning in $\widetilde{O}(d)$ expected iterations, with logarithmic dependence on the initial condition number (see Appendix C). The proof uses the pessimistic bound $S_\beta \ge 1/4$; more favorable spectra can yield faster conditioning and permit larger step sizes.

Practical step size without blocking. Balancing the negative drift of Lemma 2 against the second-order term of Lemma 1 suggests that we should choose α to $\alpha_* = \frac{\nu_{b,d}}{12\beta\cdot\sqrt{d-1}} \cdot \frac{S_\beta}{m_t}$ to maximize the expected gain. While m_t is easily computed, S_β depends on the spectrum and is difficult to estimate. The proof therefore controls m_t through blocking and uses a worst-case bound on S_β. Empirically, however, we observe $S_\beta/\sqrt{m_t} = \Omega(1)$ even with global row selection, across diverse spectra and dimensions (Appendix C.10). This suggests the practical step-size rule $\alpha'_* = \Theta(1/\sqrt{\beta m_t d})$, which we use in our experiments.

Beyond Haar mixing. The Taylor bound of Lemma 1 and the spectral bound on S_β do not depend on the mixing distribution. Haar mixing is used only to establish the exact expected-drift identity (15), exploiting its rotational symmetry to characterize the effect of row-min selection. Randomized Walsh–Hadamard transforms also spread information across coordinates, but lack full rotational symmetry, making the drift harder to characterize. Extending the guarantee to randomized Walsh–Hadamard mixing would therefore require a corresponding drift bound, which remains open (see Appendix C.7).

6.2 Empirical role of the simplifying assumptions

Theorem 3 assumes Haar mixing to compute the expected drift and block-restricted row minima to bound the column load. We test whether these assumptions also improve empirical conditioning. Both comparisons use synthetic matrices with $d = 2048$ and initial $\kappa = 10^4$.

Table 1: **Haar versus Walsh–Hadamard mixing.** Final condition numbers after $T = 100$ steps (mean $\pm$ standard deviation).

Spectrum	Haar	Walsh–Hadamard
Chebyshev	5726 ± 83	5699 ± 90
Linear	4397 ± 77	4397 ± 82
Log	2848 ± 27	2844 ± 25
Power-law	4555 ± 58	4564 ± 62

Table 2: **Global versus block row minima.** Mean condition-number reduction (%).

	$T = 100$		$T = 500$	
Spectrum	Global	Block	Global	Block
Chebyshev	43.1	1.0	98.8	26.6
Linear	56.2	6.5	99.6	37.8
Log	71.5	16.2	98.7	55.0
Power-law	54.5	8.2	98.8	38.2

Fast mixing. We compare Haar rotations with randomized Walsh–Hadamard transforms, using identical initial matrices, global row selection, and step-size rules. Across 100 trials per spectral family, Table 1 shows that the final condition numbers after $T = 100$ steps differ by at most 0.5%. Thus, fast structured mixing achieves essentially the same conditioning as Haar mixing.

Global versus block row selection. We compare the practical global selector (9) with block-restricted selection (14) (block size 8), using identical Walsh–Hadamard mixing sequences. The global variant uses $\alpha_t = s_\alpha / \sqrt{m_t d \log d}$, while the blocked variant uses $\alpha_t = s_\alpha / \sqrt{d \log d}$. Table 2 shows that both improve conditioning, but the practical configuration achieves substantially larger reductions. Results are averaged over 20 trials; standard deviations are below 1.1 percentage points everywhere.

Together, these results show that the simplifying assumptions used in our proof are not needed for effective conditioning in practice: fast mixing matches Haar, while the practical global configuration substantially outperforms blocking. Establishing a theoretical guarantee for the practical algorithm remains open.

7 Conclusion

We introduced RAPID, an anytime SPD preconditioner derived from a conditioning mechanism discovered in a transformer trained for in-context learning. The extracted update preserves the linear-system relation and evolves its Gram matrix by congruence. RAPID turns this mechanism into a standalone algorithm combining randomized mixing and sparse row corrections, accumulating a preconditioner $A_t = P_t A P_t^\top$. The transformer provides only the algorithmic primitive; RAPID requires neither model inference nor learned parameters at solve time. For an idealized Haar-mixing, block-restricted variant, we prove convergence to any fixed condition-number target for arbitrary SPD matrices, with near-linear dependence on dimension and logarithmic dependence on initial conditioning. Experiments with practical RAPID show broad conditioning improvements and reduced CG iteration counts on synthetic and SuiteSparse matrices.

Several challenges remain. Extending the convergence guarantee to practical Walsh–Hadamard mixing and global row-minimum selection would close the gap between theory and implementation. Establishing competitive end-to-end runtime requires optimized implementations and careful accounting of setup costs, particularly when amortized across multiple right-hand sides. Finally, extending RAPID to sparse and matrix-free systems will require avoiding explicitly formed dense mixed matrices. More broadly, RAPID illustrates how studying learned transformer computations can lead to independent numerical algorithms that apply well beyond the original learning task.

Acknowledgments

This research was supported by the National Science Foundation grant CPS-2317079.

References

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 45614–45650. Curran Associates,

Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8ed3d610ea4b68e7afb30ea7d01422c6-Paper-Conference.pdf.

Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009. doi: 10.1137/060673096. URL https://doi.org/10.1137/060673096.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I.

Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging LAPACK's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010. doi: 10.1137/090767911.

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.

Michele Benzi, Carl D Meyer, and Miroslav Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

Tri Dao. Fast hadamard transform in cuda, with a pytorch interface, 2024. URL https://github.com/Dao-AILab/fast-hadamard-transform.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International conference on machine learning*, pages 3318–3328. PMLR, 2021.

Zachary Frangella, Joel A. Tropp, and Madeleine Udell. Randomized Nyström preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752, 2023. doi: 10.1137/21M1466244.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn to achieve second-order convergence rates for in-context linear regression. *Advances in Neural Information Processing Systems*, 37:98675–98716, 2024.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30583–30598. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c529dba08a146ea8d6cf715ae8930cbe-Paper-Conference.pdf.

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.

Scott P Kolodziej, Mohsen Aznaveh, Matthew Bullock, Jarrett David, Timothy A Davis, Matthew Henderson, Yifan Hu, and Read Sandstrom. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International conference on machine learning*, pages 2747–2755. PMLR, 2018.

Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioners for conjugate gradient pde solvers. In *International Conference on Machine Learning*, pages 19425–19439. PMLR, 2023.

Patrick Lutz, Aditya Gangrade, Hadi Daneshmand, and Venkatesh Saligrama. Linear transformers implicitly discover unified numerical algorithms. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=XxR70zr9Sf.

Patrick Lutz, Themistoklis Haris, Arjun Chandra, Aditya Gangrade, and Venkatesh Saligrama. Layerwise dynamics for in-context classification in transformers. *arXiv preprint arXiv:2604.11613*, 2026.

Ilay Luz, Meirav Galun, Haggai Maron, Ronen Basri, and Irad Yavneh. Learning algebraic multigrid using graph neural networks. In *International Conference on Machine Learning*, pages 6489–6499. PMLR, 2020.

J Meijerink and Henk A van der Vorst. Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *Journal of computational physics*, 44(1):134–155, 1981.

Xiangrui Meng, Michael A. Saunders, and Michael W. Mahoney. LSRN: A parallel iterative solver for strongly over- or underdetermined systems. *SIAM Journal on Scientific Computing*, 36(2): C95–C118, 2014. doi: 10.1137/120866580.

Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

Vladislav Trifonov, Alexander Rudikov, Oleg Iliev, Yuri M Laevsky, Ivan Oseledets, and Ekaterina Muravleva. Learning from linear algebra: A graph neural network approach to preconditioner design for conjugate gradient solvers. *Computational Methods in Applied Mathematics*, 2026.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Max Vladymyrov, Johannes von Oswald, Mark Sandler, and Rong Ge. Linear transformers are versatile in-context learners. In *Advances in Neural Information Processing Systems*, 2024.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35151–35174. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/von-oswald23a.html.

Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers can learn temporal difference methods for in-context reinforcement learning. In *International Conference on Learning Representations*, volume 2025, 2025.

Andrew J Wathen. Preconditioning. *Acta Numerica*, 24:329–376, 2015.

Qilin Ye, Deqing Fu, Robin Jia, and Vatsal Sharan. Transformers provably learn algorithmic solutions for graph connectivity, but only with the right data. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. *Advances in neural information processing systems*, 29, 2016.

Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024. URL http://jmlr.org/papers/v25/23-1042.html.

A Supplementary material for Section 3

A.1 Transformer architecture and training details

This appendix specifies the training setup used for the in-context regression experiments in Section 3.1. We train attention-only transformers on noiseless synthetic linear regression instances. Unless stated otherwise, the experiments use $d = 15$ covariates and $d + 1 = 16$ tokens: d context examples and one query example.

Data distribution. For each batch element, we sample

$$x_1, \dots, x_d, x_q \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d), \qquad w \sim \mathcal{N}(0, I_d),$$

and set

$$y_i = x_i^\top w/\sqrt{d}, \qquad y_q = x_q^\top w/\sqrt{d}.$$

The factor $1/\sqrt{d}$ is a normalization convention; equivalently, the effective regression coefficient has covariance I_d/d. The model input is

$$Z_0 = \begin{bmatrix} x_1^\top & y_1 \\ \vdots & \vdots \\ x_d^\top & y_d \\ x_q^\top & 0 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)},$$

and the loss is the squared error on the query label only,

$$\mathcal{L} = \big((Z_L)_{d+1,d+1} + y_q\big)^2.$$

The implementation reads the prediction y_q from the bottom-right entry of the final token matrix with the same negative-output convention as Von Oswald et al. [2023]: $\widehat{y}_q = -(Z_L)_{d+1,d+1}$. Fresh inputs and fresh regression coefficients are resampled at every training iteration.

Architecture. We use a residual attention-only transformer with no feed-forward blocks. Each layer has one attention head and embedding dimension $d + 1 = 16$. For $Z \in \mathbb{R}^{(d+1)\times(d+1)}$, layer ℓ computes

$$Z_{\ell+1} = Z_\ell + A_\ell(Z_\ell) Z_\ell W_{V,\ell} W_{P,\ell},$$

where

$$A_\ell(Z) = \operatorname{softmax}\left(Z W_{Q,\ell} W_{K,\ell}^\top Z^\top + M\right).$$

All linear projections are bias-free. The matrices $W_{Q,\ell}, W_{K,\ell}, W_{V,\ell}$ are initialized with Xavier uniform initialization, while the output projection $W_{P,\ell}$ is initialized at zero. Thus the initial network is the identity map through the residual connection. For notational simplicity, we absorb the $1/\sqrt{d+1}$ factor commonly present in the attention architecture [Vaswani et al., 2017] into the attention weights.

The attention mask removes the query token as a key/value:

$$M = \begin{bmatrix} \mathbf{0}_{(d+1)\times d} & -\infty\, \mathbf{1}_{d+1} \end{bmatrix}.$$

Thus every token may attend to the d labeled context examples, but no token may attend to the query token. The query row is therefore used only as the prediction site.

Attention-logit clipping. For numerical stability and to avoid nearly one-hot attention early in training, we clip attention logits after centering each attention row. Let

$$s_{ij} = \frac{q_i^\top k_j}{\sqrt{d+1}}$$

denote the unmasked attention score, and let $\mathcal{A}_i$ be the set of allowed (non-masked) keys under the attention mask. Before the softmax, we replace each allowed score by

$$\widetilde{s}_{ij} = \max\left\{ s_{ij} - \max_{j' \in \mathcal{A}_i} s_{ij'}, -6 \right\}, \qquad j \in \mathcal{A}_i,$$

and keep masked scores equal to $-\infty$. Hence the largest allowed logit in each row is zero, and all allowed logits are lower bounded by -6. This bounds the maximum probability ratio between two allowed tokens by e^6. We found this clipping useful for numerical stability when training deep models.

Symmetry augmentation. The reported training runs use the symmetry-augmented implementation. At each layer and for each forward pass, we draw independent Haar orthogonal matrices $U \in \mathbb{R}^{d\times d}$ and $V \in \mathbb{R}^{d\times d}$, extend them by a fixed final row/column, and apply the attention block in the transformed coordinates:

$$\widetilde{Z}_\ell = \mathrm{diag}(U, 1) Z_\ell \,\mathrm{diag}(V, 1).$$

The attention update is then rotated back before being added residually to Z_ℓ. This augmentation preserves the distinguished query row and label coordinate while randomizing the basis of the context examples and features.

Optimization. We train each model with AdamW for $30{,}001$ steps using batch size 8192, learning rate 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.99)$, no weight decay, and global gradient clipping at norm 1. The learning rate is linearly warmed up for the first 500 steps and then held fixed. We resample both the covariates and regression coefficient at every step. We train ten independent runs for each depth

$$L \in \{1, 2, 3, 5, 7, 10, 15, 20, 30\},$$

and use the depth-30 models for the extraction analysis unless otherwise specified.

Training curves. Figure 3 reports the training loss for the depth-30 models used in the extraction analysis. Thin curves show individual runs; the thick curve shows the mean across runs, with the shaded band denoting the central 80% interval.

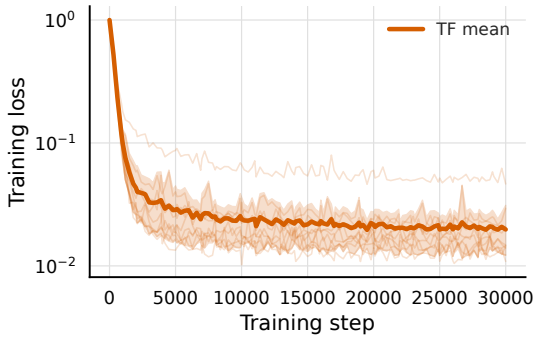

Figure 3: **Training curves for depth-30 transformers.** The loss is the squared error on the query prediction. Inputs and regression coefficients are resampled at every optimization step.

A.2 Behavioral diagnostics

We evaluate the trained symmetry-intervened transformers in the square $d = 15$ linear-system setting. For each depth $L \le 30$, GD and CG are run for L iterations from zero. GD uses a fixed stepsize selected separately for each iteration budget from 60 logarithmically spaced values in $[10^{-4}, 3]$, minimizing query-prediction MSE on 4,096 validation tasks. Results are evaluated on 4,096 independent test tasks.

Prediction error. Figure 4 (left) shows that test MSE decreases with model depth. The error profile differs from GD and CG at matched iteration budgets, suggesting that the transformer does not simply reproduce either classical solver.

Spectral inverse-kernel diagnostic. For each fixed context X, we probe the endpoint with random queries and fit the best linear query map

$$\widehat{y}_q^{\mathrm{TF}} \approx x_q^\top \widehat{\beta}_{\mathrm{TF}}(X).$$

We then express $\widehat{\beta}_{\mathrm{TF}}$ in the eigenbasis of $H = X^\top X/d = U\,\mathrm{diag}(\lambda_j)U^\top$. Since exact OLS satisfies

$$\beta_{\star,e,j} = \lambda_j^{-1} b_{e,j}, \qquad b_e = U^\top X^\top y/d,$$

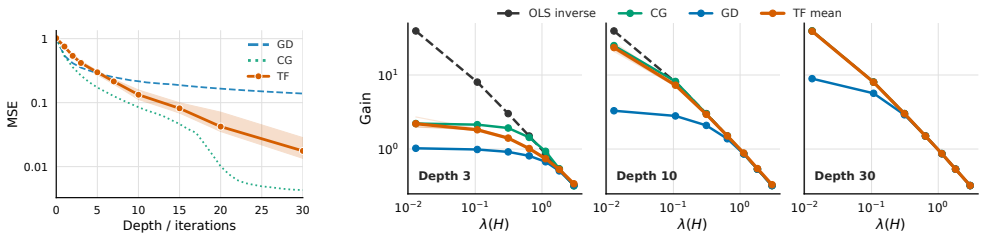

Figure 4: **Prediction behavior across transformer depths.** *Left:* Test MSE decreases with model depth, following a profile distinct from GD and CG at matched iteration budgets. *Right:* The fitted endpoint spectral gain approaches the OLS response $1/\lambda$ across eigendirections of $X^\top X/d$, including small eigenvalues.

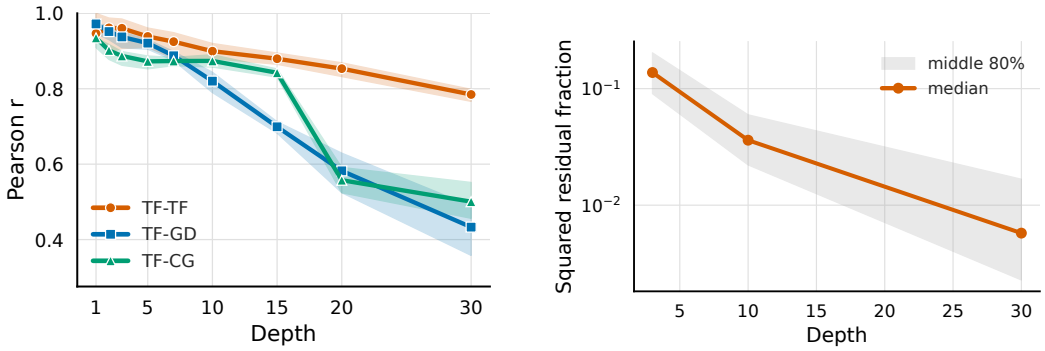

Figure 5: **Additional behavioral diagnostics.** *Left:* Pearson correlations between centered residual-error vectors on shared test tasks. Transformer runs remain more aligned with each other than with GD or CG. *Right:* Squared residual fraction of the best linear query-to-prediction map for each context. The small residual at depth 30 supports the spectral-filter interpretation.

the ideal gain in eigendirection j is $1/\lambda_j$. The plotted gain is the zero-intercept slope mapping $b_{e,j}$ to $\widehat{\beta}_{e,j}$, after pooling eigendirections across contexts and binning by λ_j:

$$g_B = \frac{\sum_{(c,j)\in B} \widehat{\beta}_{e,c,j} b_{e,c,j}}{\sum_{(c,j)\in B} b_{e,c,j}^2}.$$

Figure 4 (right) shows that the fitted gain approaches the OLS response $1/\lambda$ as depth increases, including at small eigenvalues. Thus the transformer increasingly captures the full inverse spectral response rather than only the well-conditioned directions.

Linearity of the query map. This spectral diagnostic is meaningful only if, for fixed context, the endpoint is well approximated by a linear function of the query. We check this by plotting the squared residual fraction

$$\frac{\sum_q \left(\widehat{y}_q^{\mathrm{TF}} - x_q^\top \widehat{\beta}_{\mathrm{TF}}\right)^2}{\sum_q \left(\widehat{y}_q^{\mathrm{TF}}\right)^2}.$$

As shown in Figure 5 (right), this residual fraction decreases with depth and is below 10^{-2} at depth 30. Thus the fitted linear query map explains more than 99% of the transformer's prediction energy on the probe queries, validating the spectral-filter interpretation for the depth-30 models.

Residual-error alignment. To compare failure modes, we compute the centered residual-error vector

$$e_c = (\widehat{y} - y_q) - \frac{1}{N}\mathbf{1}\mathbf{1}^\top(\widehat{y} - y_q)$$

on a shared test set, and report Pearson correlations between such vectors. Transformer-Transformer (TF-TF) alignment is computed by splitting trained transformer runs into two groups and correlating

the group-averaged residuals; TF-GD and TF-CG compare the same transformer averages to GD and CG. As shown in Figure 5 (left), TF-TF alignment remains high at depth 30 ($r \approx 0.8$), while TF-GD and TF-CG fall below 0.5. Thus independently trained transformers share a stable residual structure that is not well explained by either classical baseline.

A.3 Extracted iteration and fidelity diagnostics

We describe the two-scalar approximation used in Section 3.2. For each trained layer, let

$$W_{QK,\ell} = W_{Q,\ell} W_{K,\ell}^\top, \qquad W_{VP,\ell} = W_{V,\ell} W_{P,\ell}.$$

We approximate these products as

$$\widehat{W}_{QK,\ell} = \gamma_\ell \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix}, \qquad \widehat{W}_{VP,\ell} = \alpha_\ell I_{d+1}.$$

Here γ_ℓ is the mean diagonal entry of the feature-feature block of $W_{QK,\ell}$, and α_ℓ is the mean diagonal entry of the feature-feature block of $W_{VP,\ell}$. In the projected value map, the label coordinate is tied to the same scalar α_ℓ, giving the two-scalar template used in the main text. All off-diagonal entries and all QK entries involving the label coordinate are discarded.

Write the full state as

$$Z_\ell = \begin{bmatrix} X_\ell & y_\ell \\ x_{q,\ell}^\top & \hat{y}_\ell \end{bmatrix}.$$

Under the symmetry intervention, the layer applies attention in a randomly rotated context basis and then rotates the update back. With the projected weights, the context attention is

$$A_\ell = \sigma\left(\gamma_\ell U_\ell X_\ell X_\ell^\top U_\ell^\top\right),$$

and the corresponding context update is

$$X_{\ell+1} = C_\ell X_\ell, \qquad y_{\ell+1} = C_\ell y_\ell, \qquad C_\ell = I + \alpha_\ell U_\ell^\top A_\ell U_\ell.$$

Thus the projection gives the row-side iteration used in the main text. The same projected layer also updates the query row and readout. If

$$a_\ell^\top = \sigma\left(\gamma_\ell x_{q,\ell}^\top X_\ell^\top U_\ell^\top\right)$$

is the query-to-context attention row, then

$$x_{q,\ell+1}^\top = x_{q,\ell}^\top + \alpha_\ell a_\ell^\top U_\ell X_\ell, \qquad \hat{y}_{\ell+1} = \hat{y}_\ell + \alpha_\ell a_\ell^\top U_\ell y_\ell.$$

Fidelity of the projection. We evaluate the projection by comparing the original depth-30 transformer and the projected iteration on the same held-out tasks, using the same random rotations in both forward passes. Predictions are pooled across evaluation tasks and trained runs. The reported R^2 is the squared Pearson correlation between the original transformer predictions and the projected-iteration predictions:

$$R^2 = \mathrm{Corr}\left(\widehat{y}^{\mathrm{TF}}, \widehat{y}^{\mathrm{proj}}\right)^2.$$

The MSE change is computed on the same pooled predictions as

$$\Delta\mathrm{MSE} = \frac{1}{RN} \sum_{r=1}^{R} \sum_{i=1}^{N} \left[\left(\widehat{y}_{r,i}^{\mathrm{proj}} - y_i\right)^2 - \left(\widehat{y}_{r,i}^{\mathrm{TF}} - y_i\right)^2 \right].$$

The projection preserves the trained model closely: $R^2 > 0.98$, and the test MSE changes by only 6.4×10^{-3}. Figure 6 shows the corresponding prediction-level agreement.

A.4 Conditioning behavior and the layerwise sign rule

This appendix gives the empirical evidence behind the preconditioning interpretation in Section 3.2. The main questions are: (i) does the extracted iteration actually improve conditioning layer by layer, and (ii) when does a layer act as a conditioning step?

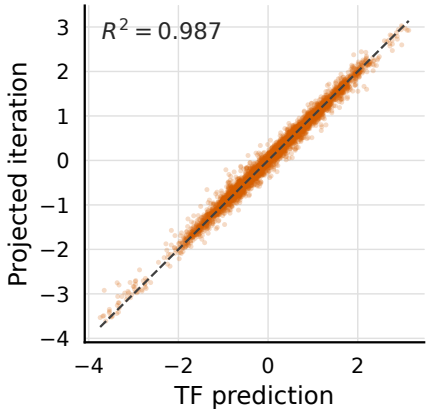

Figure 6: **Fidelity of the projected iteration.** Each point compares the original depth-30 transformer prediction with the prediction from the two-scalar projected iteration on the same held-out task. The high R^2 shows that the low-dimensional projection captures the dominant predictive behavior of the trained model.

Conditioning under the extracted and projected iterations. Figure 7 compares the conditioning trajectories of the trained depth-30 transformer and its two-scalar approximation. In both cases, the median condition number of $G_\ell = X_\ell X_\ell^\top$ decreases from above 3×10^3 to below 10 at the best layer. Thus the low-dimensional approximation captures the conditioning behavior of the trained model.

The right panel shows that most learned step sizes α_ℓ are close to 1. Since the softmax matrix K_ℓ is row-stochastic, $|\alpha_\ell| < 1$ guarantees that $I + \alpha_\ell K_\ell$ is invertible, preserving positive definiteness under the congruence update. The learned values suggest that the model prefers large residual steps within this invertibility range.

Both trajectories exhibit occasional increases in condition number, particularly near the final layers. This is consistent with the transformer being trained for best final prediction at fixed depth, rather than for monotone conditioning at every intermediate layer.

Which layers condition? Empirically, layers tend to reduce the condition number when their parameters satisfy

$$\operatorname{sign}(\alpha_\ell) \neq \operatorname{sign}(\gamma_\ell), \qquad |\alpha_\ell| < 1. \tag{16}$$

For example, when $\alpha_\ell > 0$ and $\gamma_\ell < 0$, attention favors smaller entries of the mixed Gram matrix, which receive positive weight in the residual update. Figure 8 compares this rule with the observed layerwise conditioning trajectories.

Across runs, 80% of layers satisfy the rule, and in the median such layer, 95.5% of test instances decrease their condition number. Thus the learned parameters provide a useful empirical indicator of conditioning steps, although the rule does not guarantee a decrease on every instance.

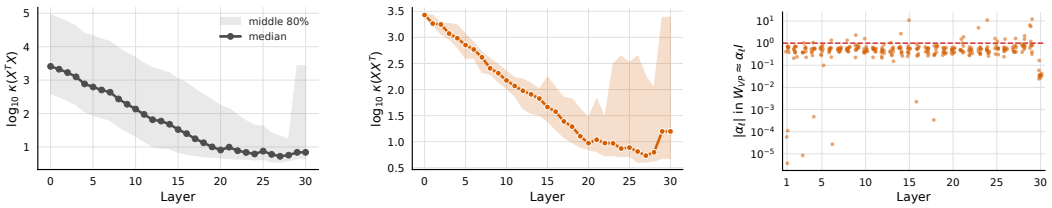

Figure 7: **Conditioning overview.** *Left:* Condition number of $G_\ell = X_\ell X_\ell^\top$ across layers of the trained transformer. *Center:* The corresponding trajectory under the two-scalar approximation. Both reduce the median condition number from above 3×10^3 to below 10 at the best layer. *Right:* Distribution of learned step sizes α_ℓ, most of which lie close to the sufficient invertibility threshold $\alpha_\ell = 1$.

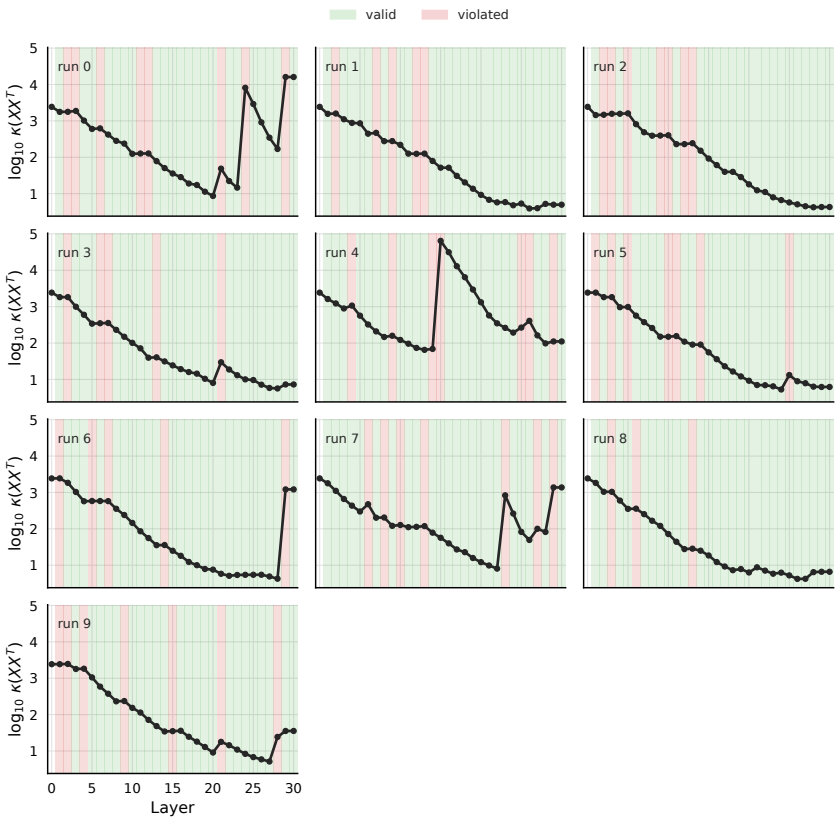

Figure 8: **Layerwise sign rule and conditioning.** Layerwise condition numbers compared with the empirical criterion $\mathrm{sign}(\alpha_\ell) \neq \mathrm{sign}(\gamma_\ell)$ and $|\alpha_\ell| < 1$. Layers satisfying this rule typically decrease the condition number.

B Additional Experimental Details

Synthetic matrices. We generate $A = U\Lambda U^\top$, where U is Haar-random orthogonal and Λ has one of four spectra, defined for $i = 0, \dots, d-1$:

$$\begin{aligned}
\text{Chebyshev:} \quad & \lambda_i = \frac{\kappa+1}{2} - \frac{\kappa-1}{2}\cos\frac{(2i+1)\pi}{2d}, \\
\text{Linear:} \quad & \lambda_i = 1 + (\kappa-1)\frac{i}{d-1}, \\
\text{Logarithmic:} \quad & \lambda_i = \kappa^{i/(d-1)}, \\
\text{Power-law:} \quad & \lambda_i = 1 + (\kappa-1)\left(\frac{i}{d-1}\right)^2.
\end{aligned}$$

We set the spectral endpoints to 1 and $\kappa = 10^4$ and use $d = 1024$, with 100 independent draws per family.

SuiteSparse matrices. We use 31 SPD matrices from the SuiteSparse Matrix Collection [Kolodziej et al., 2019], selected by dimension ($100 \le d \le 6000$) and post-Jacobi condition number (10^3–10^6), independently of RAPID's performance. The selected instances, grouped by collection, are:

`Boeing`: msc00726, msc01440;
`Cannizzo`: sts4098;
`Cylshell`: s1rmq4m1, s1rmt3m1;
`HB`: 1138_bus, 494_bus, 662_bus, 685_bus, bcsstk03, bcsstk04, bcsstk05, bcsstk06, bcsstk07, bcsstk08, bcsstk09, bcsstk10, bcsstk13, bcsstk14, bcsstk15, bcsstk21, bcsstk22, bcsstk26, bcsstk27, bcsstm12, lund_a, nos3, nos5;
`Nasa`: nasa1824, nasa2910;
`TKK`: plbuckle.

All matrices are converted to dense form. We normalize by an estimate of the largest eigenvalue (20 power iterations) and apply Jacobi scaling. For each matrix, we sample an independent Gaussian right-hand side before applying the same coordinate scaling.

RAPID implementation. We implement RAPID on the GPU using PyTorch/Triton and fast Walsh–Hadamard transforms [Dao, 2024]. We use one randomized Hadamard mixing stage ($k = 1$), global row-minimum selection, and $s_\alpha = 10$. A single mixing stage suffices in practice, as repeated RAPID iterations progressively mix the coordinates. The row-selection kernel finds the smallest off-diagonal entry in each row and computes the maximum column multiplicity m_t. A parallel Triton kernel then applies the congruence update entrywise:

$$(A_{t+1})_{ij} = \widetilde{A}_{ij} + \alpha_t \widetilde{A}_{\tau(i),j} + \alpha_t \widetilde{A}_{i,\tau(j)} + \alpha_t^2 \widetilde{A}_{\tau(i),\tau(j)},$$

where $\alpha_t = s_\alpha / \sqrt{m_t d \log d}$. We accumulate the transformation P_t as in Section 4 to recover solutions in the original coordinates. Non-power-of-two dimensions are padded to the next power of two with an identity block; the right-hand side is padded with zeros. We evaluate RAPID after $T \in \{10, 20, 50, 100, 200, 500\}$ steps.

Nyström preconditioning. For each sketch fraction $q \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, we use a Gaussian sketch of rank $r = \lceil qd \rceil$, orthonormalized by QR. We form $Y = A_0\Omega$ and $B = \Omega^\top Y$, then construct a stabilized Nyström approximation using Cholesky and eigendecomposition. Its approximate eigenvectors U and positive eigenvalue estimates $\widehat{\lambda}_i$ define the preconditioner

$$M = I + U\,\mathrm{diag}\left(\frac{\theta}{\widehat{\lambda}_i} - 1\right)U^\top, \qquad \theta = \min_i \widehat{\lambda}_i.$$

We apply M within preconditioned CG.

CG and evaluation. All solves use a common GPU float32 CG implementation, initialized at zero, with at most 10^4 iterations. The true relative residual is checked every 10 iterations, after recovering the solution in the Jacobi-scaled reference system $A_0 x = b_0$. Convergence requires

$$\frac{\|A_0\widehat{x} - b_0\|_2}{\|b_0\|_2} \le 2 \times 10^{-3}.$$

Table 3: **Absolute condition numbers and CG iterations.** Mean values across 100 independent matrices per synthetic family and 31 SuiteSparse matrices. All RAPID runs reached the target accuracy ($100/100$ per synthetic family and $31/31$ for SuiteSparse) at every tested step count. The only non-convergent run was Nyström with a 1% sketch on the SuiteSparse matrix `Cylshell/s1rmt3m1`. These averages of absolute quantities differ from the averages of per-instance ratios shown in Figure 2.

		RAPID steps T					
Dataset	Jacobi baseline	10	20	50	100	200	500
Condition number							
Chebyshev	10,155.43	10,402.86	9,371.71	6,106.86	2,746.62	480.23	4.89
Linear	10,007.98	9,342.59	8,066.53	4,816.50	1,857.98	228.63	4.82
Logarithmic	9,978.65	7,429.91	5,697.62	2,983.64	1,316.00	330.48	6.07
Power-law	10,010.99	8,940.11	7,707.07	4,845.05	2,216.73	425.81	4.94
SuiteSparse	94,992.34	82,687.66	73,149.07	53,286.53	35,442.72	18,391.60	4,600.00
CG iterations to target accuracy							
Chebyshev	319.10	322.60	308.00	252.90	176.60	80.50	10.00
Linear	127.90	122.00	115.50	90.30	59.70	29.80	10.00
Logarithmic	373.30	322.70	283.40	207.80	140.70	74.50	17.80
Power-law	332.80	313.80	291.60	233.60	163.20	79.00	10.00
SuiteSparse	266.45	250.65	230.32	189.35	145.48	99.35	40.65

Condition numbers are computed from the extreme eigenvalues of the transformed matrices. Figure 2 reports condition numbers and CG iteration counts relative to the Jacobi baseline, with means and minimum-maximum ranges for RAPID and means for Nyström. All comparisons use the same matrices and right-hand sides.

Algorithm 2 Block row-min preconditioning

Input: $A_0 \succ 0$, step size α, iterations T, block size b, mixer distribution $\mathcal{Q}$.
for $t = 0, \dots, T-1$ **do**
  Draw $Q_t \sim \mathcal{Q}$ and a block partition $\mathcal{B}_t$ of $[d]$ with block size b.
  $\widetilde{A}_t \leftarrow Q_t^\top A_t Q_t$.
  $R_t \leftarrow \mathrm{BRM}_{\mathcal{B}_t}(\widetilde{A}_t)$.
  $A_{t+1} \leftarrow (I + \alpha R_t)^\top \widetilde{A}_t (I + \alpha R_t)$.
end for
return A_T.

C Proof of the Haar Block-Row-Min Guarantee

This appendix proves Theorem 3. Let us define the block row selector:

Block row-min selector. For a partition $\mathcal{B}$ of $[d]$ into blocks of size b, Define

$$\mathrm{BRM}_{\mathcal{B}}(M)_{ij} = \mathbf{1}\left\{ j = \operatorname*{arg\,min}_{\ell \in B(i)\setminus\{i\}} M_{i\ell} \right\}, \tag{17}$$

where $B(i) \in \mathcal{B}$ is the block containing i. Ties are broken deterministically. The theory uses $\boldsymbol{Q}_t \sim \mathrm{Haar}(O(d))$; experiments also use block-local choices based on product of Hadamard matrices of $\boldsymbol{Q}_t$.

Let us state the theorem in full generality here. We analyze the Gram recursion extracted in Section 3:

$$\widetilde{A}_t = Q_t^\top A_t Q_t, \qquad A_{t+1} = (I + \alpha R_t)^\top \widetilde{A}_t (I + \alpha R_t), \qquad R_t = \mathrm{BRM}_{\mathcal{B}_t}(\widetilde{A}_t), \tag{18}$$

started from an arbitrary $A_0 \succ 0$. The theorem is proved for fresh uniform random orthogonal mixing, $Q_t \sim \mathrm{Haar}(O(d))$. This distributional assumption is used only in the local row-min drift calculation; the Taylor expansion, trace identities, block-load bound, spectral lower bound, and stopping-time argument hold for any orthogonal mixer.

For the block selector defined in (17), each row selects one off-diagonal entry from its block and each column can be selected by at most $b-1$ rows. Thus

$$\|R_t\|_{\mathrm{op}}^2 \le b - 1. \tag{19}$$

This deterministic load bound is the only property of the block mask used in the Taylor remainder.

Theorem 4 (Uniform-random-rotation block row-min preconditioning)**.** *Let $d \ge 3$, $3 \le b \le d$, and $b \mid d$. Fix a target condition number $\kappa_\star > 1$, and define*

$$\rho_{\kappa_\star} = \max\left\{ 1 - \frac{1 - \kappa_\star^{-1}}{4}, \left(1 + \frac{\kappa_\star - 1}{4}\right)^{-1} \right\} < 1. \tag{20}$$

Choose

$$\beta \ge \frac{\log(8d)}{-\log \rho_{\kappa_\star}}. \tag{21}$$

Let

$$\nu_{b,d} = -\sqrt{d-1}\,\mathbb{E}\left[\min_{1 \le r \le b-1} U_r\right], \qquad U \sim \mathrm{Unif}(S^{d-2}). \tag{22}$$

There are universal constants $C_{\mathrm{step}}, C_{\mathrm{hit}} > 0$ such that, for

$$\alpha = \frac{\nu_{b,d}}{C_{\mathrm{step}}\,\beta(b-1)\sqrt{d-1}}, \tag{23}$$

the iteration (18) *with fresh $Q_t \sim \mathrm{Haar}(O(d))$ satisfies*

$$\mathbb{E}[\tau_{\kappa_\star}] \le C_{\mathrm{hit}} \frac{\beta(b-1)(d-1)}{\nu_{b,d}^2} \log \kappa(A_0), \qquad \tau_{\kappa_\star} = \inf\{t : \kappa(A_t) \le \kappa_\star\}. \tag{24}$$

For each fixed $b \ge 3$, $\nu_{b,d}^2 = \Theta_b(1)$ uniformly in d. Hence, for fixed b and fixed $\kappa_\star$,

$$\mathbb{E}[\tau_{\kappa_\star}] = O_{\kappa_\star, b}\left(d \log d\, \log \kappa(A_0)\right). \tag{25}$$

Theorem 4 is a refinement of Theorem 3 from the main text. The rest of the analysis will focus on proving this result. In the process, we will show improved versions of Lemma 2 and Lemma 1.

C.1 Setup and potential

Let $A_0 \succ 0$ and let

$$A_{t+1} = (I + \alpha R_t)^\top \widetilde{A}_t (I + \alpha R_t), \qquad \widetilde{A}_t = Q_t^\top A_t Q_t, \qquad Q_t \sim \mathrm{Haar}(O(d)).$$

Fix a block partition $[d] = B_1 \sqcup \cdots \sqcup B_{d/b}$, $|B_r| = b$. For $i \in B_r$, define

$$\tau_i = \operatorname*{arg\,min}_{j \in B_r,\, j \neq i} (\widetilde{A}_t)_{ij}, \qquad (R_t)_{ij} = \mathbf{1}\{j = \tau_i\},$$

with deterministic tie-breaking. A fresh independent random partition may also be used, but the Haar proof only needs the candidate sets to be fixed independently of Q_t.

For $\beta \geq 1$, define

$$\Phi_\beta(A) = \frac{1}{\beta} \log \frac{\mathrm{tr}(A^\beta)}{d} + \frac{1}{\beta} \log \frac{\mathrm{tr}(A^{-\beta})}{d}.$$

This potential is scale-invariant and satisfies

$$0 \leq \Phi_\beta(A) \leq \log \kappa(A) \leq \Phi_\beta(A) + \frac{2 \log d}{\beta}.$$

Let $\lambda = (\lambda_1, \ldots, \lambda_d)$ be the eigenvalues of A, and let

$$p \sim \mathrm{Dir}(1/2, \ldots, 1/2).$$

Define

$$\mathrm{Var}_p(\lambda) = \sum_i p_i \lambda_i^2 - \left(\sum_i p_i \lambda_i \right)^2,$$

$$\mathrm{Cov}_p(\lambda, \lambda^\gamma) = \sum_i p_i \lambda_i^{\gamma+1} - \left(\sum_i p_i \lambda_i \right) \left(\sum_i p_i \lambda_i^\gamma \right).$$

Ratios below are defined to be zero when $\mathrm{Var}_p(\lambda) = 0$. Set

$$s_\beta(A) = \mathbb{E}_p \left[\frac{\mathrm{Cov}_p(\lambda, \lambda^\beta)}{\sqrt{\mathrm{Var}_p(\lambda)}} \right], \qquad s_{-\beta}(A) = -\mathbb{E}_p \left[\frac{\mathrm{Cov}_p(\lambda, \lambda^{-\beta})}{\sqrt{\mathrm{Var}_p(\lambda)}} \right],$$

and

$$S_\beta(A) = \frac{d\, s_\beta(A)}{\mathrm{tr}(A^\beta)} + \frac{d\, s_{-\beta}(A)}{\mathrm{tr}(A^{-\beta})}. \tag{26}$$

C.2 Block masking controls the load

For each row i, exactly one entry of R_t is nonzero, so $\|R_t\|_\infty = 1$. Fix a column $j \in B_r$. Only rows in $B_r \setminus \{j\}$ can select column j, so the column has at most $b - 1$ nonzeros. Therefore

$$\|R_t\|_1 \leq b - 1, \qquad \|R_t\|_{\mathrm{op}} \leq \sqrt{\|R_t\|_1 \|R_t\|_\infty} \leq \sqrt{b - 1}.$$

C.3 Taylor drift estimate

Lemma 5 (Taylor estimate)**.** *Let $\widetilde{A} \succ 0$, $R \in \mathbb{R}^{d \times d}$, and*

$$A(s) = (I + sR)^\top \widetilde{A} (I + sR), \qquad f(s) = \Phi_\beta(A(s)).$$

If $\beta \geq 1$ and $\alpha \|R\|_{\mathrm{op}} \leq 1/2$, then

$$f'(0) = 2 \left(\frac{\mathrm{tr}(R\widetilde{A}^\beta)}{\mathrm{tr}(\widetilde{A}^\beta)} - \frac{\mathrm{tr}(R\widetilde{A}^{-\beta})}{\mathrm{tr}(\widetilde{A}^{-\beta})} \right),$$

and there is a universal constant C_{Tay} such that, for every $s \in [0, \alpha]$,

$$f''(s) \leq C_{\mathrm{Tay}} \beta \|R\|_{\mathrm{op}}^2.$$

One may take $C_{\mathrm{Tay}} = 40$.

Proof. Since

$$A'(0) = R^\top \widetilde{A} + \widetilde{A} R,$$

the trace-power chain rule gives

$$\frac{d}{ds}\Big|_{s=0} \frac{1}{\beta} \log \operatorname{tr}(A(s)^\beta) = \frac{\operatorname{tr}(\widetilde{A}^{\beta-1}(R^\top \widetilde{A} + \widetilde{A} R))}{\operatorname{tr}(\widetilde{A}^\beta)}.$$

By cyclicity of trace,

$$\operatorname{tr}(\widetilde{A}^{\beta-1}(R^\top \widetilde{A} + \widetilde{A} R)) = 2 \operatorname{tr}(R \widetilde{A}^\beta).$$

The negative-power term similarly gives

$$\frac{d}{ds}\Big|_{s=0} \frac{1}{\beta} \log \operatorname{tr}(A(s)^{-\beta}) = -2 \frac{\operatorname{tr}(R \widetilde{A}^{-\beta})}{\operatorname{tr}(\widetilde{A}^{-\beta})}.$$

This proves the formula for $f'(0)$.

For the curvature bound, put

$$P_s = I + sR, \qquad A_s = P_s^\top \widetilde{A} P_s, \qquad C_s = P_s^{-1} R.$$

For $s \in [0, \alpha]$, the assumption $\alpha \|R\|_{\mathrm{op}} \le 1/2$ implies

$$\|P_s^{-1}\|_{\mathrm{op}} \le 2, \qquad \|C_s\|_{\mathrm{op}} \le 2\|R\|_{\mathrm{op}}.$$

Moreover,

$$A(s+t) = (I + tC_s)^\top A_s (I + tC_s).$$

Let $p = 2\beta \ge 2$. For the Schatten p-norm, the standard 2-smoothness estimate for $N(X) = \|X\|_p^2$ gives

$$DN(X)[Y] \le 2\|X\|_p \|Y\|_p, \qquad D^2 N(X)[Y,Y] \le 2(p-1)\|Y\|_p^2.$$

Consequently, for a twice differentiable path $X(t)$,

$$\frac{d^2}{dt^2} \log \|X(t)\|_p^2 \le 2(p-1) \frac{\|X'(t)\|_p^2}{\|X(t)\|_p^2} + 2 \frac{\|X''(t)\|_p}{\|X(t)\|_p}.$$

For the positive-power term, write

$$\frac{1}{\beta} \log \operatorname{tr}(A(s+t)^\beta) = \log \|A_s^{1/2}(I + tC_s)\|_{2\beta}^2.$$

With $X_+(t) = A_s^{1/2}(I + tC_s)$, we have $X_+''(0) = 0$, and Schatten submultiplicativity gives

$$\frac{d^2}{dt^2}\Big|_{t=0} \frac{1}{\beta} \log \operatorname{tr}(A(s+t)^\beta) \le (4\beta - 2)\|C_s\|_{\mathrm{op}}^2.$$

For the negative-power term,

$$A(s+t)^{-1} = (I + tC_s)^{-1} A_s^{-1} (I + tC_s)^{-\top},$$

so

$$\frac{1}{\beta} \log \operatorname{tr}(A(s+t)^{-\beta}) = \log \|(I + tC_s)^{-1} A_s^{-1/2}\|_{2\beta}^2.$$

With $X_-(t) = (I + tC_s)^{-1} A_s^{-1/2}$,

$$X_-'(0) = -C_s A_s^{-1/2}, \qquad X_-''(0) = 2C_s^2 A_s^{-1/2},$$

and hence

$$\frac{d^2}{dt^2}\Big|_{t=0} \frac{1}{\beta} \log \operatorname{tr}(A(s+t)^{-\beta}) \le 6\beta \|C_s\|_{\mathrm{op}}^2.$$

Combining the two terms and using $\|C_s\|_{\mathrm{op}} \le 2\|R\|_{\mathrm{op}}$ yields

$$f''(s) \le 40\beta \|R\|_{\mathrm{op}}^2.$$

□

Applying Lemma 5 to one update and using $\|R\|_{\mathrm{op}}^2 \le b - 1$ gives

$$\Phi_\beta(A^+) \le \Phi_\beta(A) + 2\alpha \left(\frac{\operatorname{tr}(R\widetilde{A}^\beta)}{\operatorname{tr}(A^\beta)} - \frac{\operatorname{tr}(R\widetilde{A}^{-\beta})}{\operatorname{tr}(A^{-\beta})} \right) + C_{\mathrm{Tay}} \alpha^2 \beta (b-1),$$

where $\operatorname{tr}(\widetilde{A}^{\pm\beta}) = \operatorname{tr}(A^{\pm\beta})$ by orthogonal invariance.

C.4 Block-local row-min drift

Lemma 6 (Haar row-min drift)**.** *For fixed $A \succ 0$, let $Q \sim \mathrm{Haar}(O(d))$, $\widetilde{A} = Q^\top AQ$, and let R be the block row-min selector. Then*

$$\mathbb{E}\left[\frac{\mathrm{tr}(R\widetilde{A}^\beta)}{\mathrm{tr}(A^\beta)} - \frac{\mathrm{tr}(R\widetilde{A}^{-\beta})}{\mathrm{tr}(A^{-\beta})}\,\middle|\, A\right] = -\frac{\nu_{b,d}}{\sqrt{d-1}} S_\beta(A).$$

Proof. For any real γ, symmetry of $\widetilde{A}^\gamma$ gives

$$(R\widetilde{A}^\gamma)_{ii} = \sum_j R_{ij}(\widetilde{A}^\gamma)_{ji} = (\widetilde{A}^\gamma)_{\tau_i i} = (\widetilde{A}^\gamma)_{i\tau_i}.$$

Thus

$$\mathrm{tr}(R\widetilde{A}^\gamma) = \sum_{i=1}^d (\widetilde{A}^\gamma)_{i\tau_i}.$$

It suffices to compute the one-row expectation. For each row, the candidate set has size $b-1$ and is independent of Q. By exchangeability of Haar coordinates, the one-row law depends only on this size. We analyze row 1 with a fixed candidate set $J \subset [d] \setminus \{1\}$, $|J| = b-1$.

Let

$$x_j = (\widetilde{A})_{1j}, \qquad (y_\gamma)_j = (\widetilde{A}^\gamma)_{1j}, \qquad j \neq 1,$$

and

$$\tau = \arg\min_{j \in J} x_j.$$

Then the row contribution is $(y_\gamma)_\tau$.

Write $A = V\Lambda V^\top$ and set $U = Q^\top V$. Then $U \sim \mathrm{Haar}(O(d))$. Conditional on the first row $u \in S^{d-1}$ of U, choose an orthonormal basis $B \in \mathbb{R}^{d\times(d-1)}$ for $u^\perp$. The remaining rows of U form a Haar frame in $u^\perp$, so there is $W \sim \mathrm{Haar}(O(d-1))$ such that

$$x = aW^\top, \qquad y_\gamma = c_\gamma W^\top, \qquad a = u\Lambda B, \qquad c_\gamma = u\Lambda^\gamma B.$$

The vectors a, c_γ are deterministic after conditioning on u.

Assume first that $\|a\| > 0$. Decompose

$$c_\gamma = \frac{\langle a, c_\gamma\rangle}{\|a\|^2} a + c_\gamma^\perp, \qquad c_\gamma^\perp \perp a.$$

Then

$$y_\gamma = \frac{\langle a, c_\gamma\rangle}{\|a\|^2} x + c_\gamma^\perp W^\top.$$

The index τ is a function only of $x = aW^\top$. Conditional on x, the law of W is invariant under rotations fixing a. These rotations leave x, and hence τ, unchanged, while they average the $a^\perp$-component to zero. Therefore

$$\mathbb{E}[(c_\gamma^\perp W^\top)_\tau \mid a, c_\gamma, J] = 0,$$

and

$$\mathbb{E}[(y_\gamma)_\tau \mid a, c_\gamma, J] = \frac{\langle a, c_\gamma\rangle}{\|a\|^2}\mathbb{E}[x_\tau \mid a, J].$$

If $\|a\| = 0$, both sides are interpreted using the zero-ratio convention.

Since $x = aW^\top \overset{d}{=} \|a\|U$, with $U \sim \mathrm{Unif}(S^{d-2})$,

$$\mathbb{E}[x_\tau \mid a, J] = -\frac{\nu_{b,d}}{\sqrt{d-1}}\|a\|.$$

Thus

$$\mathbb{E}[(y_\gamma)_\tau \mid a, c_\gamma, J] = -\frac{\nu_{b,d}}{\sqrt{d-1}}\frac{\langle a, c_\gamma\rangle}{\|a\|}.$$

Now average over u. If $p_i = u_i^2$, then $p \sim \mathrm{Dir}(1/2, \dots, 1/2)$. Since $BB^\top = I - u^\top u$,

$$\|a\|^2 = u_{\mathrm{var}} := \sum_i p_i \lambda_i^2 - \left(\sum_i p_i \lambda_i \right)^2 = \mathrm{Var}_p(\lambda),$$

$$\langle a, c_\beta \rangle = \mathrm{Cov}_p(\lambda, \lambda^\beta), \qquad \langle a, c_{-\beta} \rangle = \mathrm{Cov}_p(\lambda, \lambda^{-\beta}).$$

Therefore

$$\mathbb{E}[(y_\beta)_\tau \mid A] = -\frac{\nu_{b,d}}{\sqrt{d-1}} s_\beta(A),$$

and

$$\mathbb{E}[(y_{-\beta})_\tau \mid A] = \frac{\nu_{b,d}}{\sqrt{d-1}} s_{-\beta}(A).$$

Summing over the d rows gives

$$\mathbb{E}[\mathrm{tr}(R\widetilde{A}^\beta) \mid A] = -\frac{\nu_{b,d}}{\sqrt{d-1}} d\, s_\beta(A),$$

$$\mathbb{E}[\mathrm{tr}(R\widetilde{A}^{-\beta}) \mid A] = \frac{\nu_{b,d}}{\sqrt{d-1}} d\, s_{-\beta}(A).$$

Dividing by the deterministic traces yields the claimed identity. □

C.5 Lower bound on the spectral factor

Lemma 7 (Spectral lower bound)**.** *Let $\kappa_\star > 1$, and define*

$$\rho_{\kappa_\star} = \max \left\{ 1 - \frac{1 - \kappa_\star^{-1}}{4}, \left(1 + \frac{\kappa_\star - 1}{4} \right)^{-1} \right\}.$$

If $\kappa(A) \geq \kappa_\star$ and

$$\beta \geq \frac{\log(8d)}{-\log \rho_{\kappa_\star}},$$

then

$$S_\beta(A) \geq \frac{1}{4}.$$

Proof. By scale invariance, set $\lambda_{\min} = 1$, $\lambda_{\max} = \kappa \geq \kappa_\star$, and $\Delta = \kappa - 1$. Define

$$\mu_+ = \frac{\sum_i \lambda_i^{\beta+1}}{\sum_i \lambda_i^\beta}, \qquad \mu_- = \frac{\sum_i \lambda_i^{1-\beta}}{\sum_i \lambda_i^{-\beta}}, \qquad \bar{\lambda} = \frac{1}{d} \sum_i \lambda_i.$$

Since $\lambda_i \in [1, \kappa]$,

$$\mathrm{Var}_p(\lambda) \leq \frac{\Delta^2}{4}.$$

Monotonicity gives $\mathrm{Cov}_p(\lambda, \lambda^\beta) \geq 0$ and $-\mathrm{Cov}_p(\lambda, \lambda^{-\beta}) \geq 0$. Hence

$$s_\beta \geq \frac{2}{\Delta} \mathbb{E}_p[\mathrm{Cov}_p(\lambda, \lambda^\beta)], \qquad s_{-\beta} \geq \frac{2}{\Delta} \mathbb{E}_p[-\mathrm{Cov}_p(\lambda, \lambda^{-\beta})].$$

For $p \sim \mathrm{Dir}(1/2, \dots, 1/2)$,

$$\mathbb{E}_p[\mathrm{Cov}_p(u, v)] = \frac{d \sum_i u_i v_i - (\sum_i u_i)(\sum_i v_i)}{d(d+2)}.$$

Applying this identity gives

$$\mathbb{E}_p[\mathrm{Cov}_p(\lambda, \lambda^\beta)] = \frac{\mathrm{tr}(A^\beta)}{d+2} (\mu_+ - \bar{\lambda}),$$

$$\mathbb{E}_p[-\mathrm{Cov}_p(\lambda, \lambda^{-\beta})] = \frac{\mathrm{tr}(A^{-\beta})}{d+2} (\bar{\lambda} - \mu_-).$$

Therefore

$$S_\beta(A) \geq \frac{2d}{\Delta(d+2)}(\mu_+ - \mu_-) \geq \frac{\mu_+ - \mu_-}{\Delta}.$$

It remains to show $\mu_+ - \mu_- \geq \Delta/4$. Let

$$r = 1 + \frac{\Delta}{4}, \qquad s = \kappa - \frac{\Delta}{4}.$$

Then $s - r = \Delta/2$. Let

$$W_+ = \frac{\sum_{\lambda_i < s} \lambda_i^\beta}{\sum_i \lambda_i^\beta}.$$

Since at least one eigenvalue equals κ,

$$W_+ \leq d\left(\frac{s}{\kappa}\right)^\beta \leq d\rho_{\kappa_\star}^\beta \leq \frac{1}{8}.$$

Therefore

$$\mu_+ \geq s - (s-1)W_+ \geq s - \frac{3\Delta}{32}.$$

Similarly, with

$$W_- = \frac{\sum_{\lambda_i > r} \lambda_i^{-\beta}}{\sum_i \lambda_i^{-\beta}},$$

one has

$$W_- \leq dr^{-\beta} \leq d\rho_{\kappa_\star}^\beta \leq \frac{1}{8},$$

and hence

$$\mu_- \leq r + (\kappa - r)W_- \leq r + \frac{3\Delta}{32}.$$

Thus

$$\mu_+ - \mu_- \geq (s - r) - \frac{6\Delta}{32} = \frac{5\Delta}{16} \geq \frac{\Delta}{4}.$$

The lemma follows. □

C.6 Hitting-time bound

For $t < \tau_{\kappa_\star}$, Lemma 7 gives $S_\beta(A_t) \geq 1/4$. Combining Lemma 6 with the Taylor estimate gives

$$\mathbb{E}[\Phi_\beta(A_{t+1}) - \Phi_\beta(A_t) \mid A_t] \leq -\frac{\alpha\nu_{b,d}}{2\sqrt{d-1}} + C_{\mathrm{Tay}}\alpha^2\beta(b-1).$$

Choose

$$\alpha = \frac{\nu_{b,d}}{8C_{\mathrm{Tay}}\beta(b-1)\sqrt{d-1}},$$

and enlarge the universal constant in Theorem 4, if necessary, to ensure $\alpha\|R_t\|_{\mathrm{op}} \leq 1/2$. Then

$$\mathbb{E}[\Phi_\beta(A_{t+1}) - \Phi_\beta(A_t) \mid A_t] \leq -\delta, \qquad \delta = \frac{3\nu_{b,d}^2}{64C_{\mathrm{Tay}}\beta(b-1)(d-1)}.$$

Therefore

$$\Phi_\beta(A_{t\wedge\tau_{\kappa_\star}}) + \delta(t \wedge \tau_{\kappa_\star})$$

is a supermartingale. Since $\Phi_\beta \geq 0$, optional stopping gives

$$\mathbb{E}[t \wedge \tau_{\kappa_\star}] \leq \frac{\Phi_\beta(A_0)}{\delta}.$$

Letting $t \to \infty$ and using $\Phi_\beta(A_0) \leq \log\kappa(A_0)$ yields

$$\mathbb{E}[\tau_{\kappa_\star}] \leq C\frac{\beta(b-1)(d-1)}{\nu_{b,d}^2}\log\kappa(A_0).$$

It remains to record the fixed-block behavior of $\nu_{b,d}$. Since $b-1 \geq 2$,

$$-\mathbb{E} \min_{1\leq r\leq b-1} U_r \geq -\mathbb{E} \min(U_1, U_2) = \frac{1}{2}\mathbb{E}|U_1 - U_2|.$$

By rotational invariance, $U_1 - U_2 \overset{d}{=} \sqrt{2}\, U_1$, so

$$\nu_{b,d} \geq \sqrt{\frac{d-1}{2}}\, \mathbb{E}|U_1|.$$

For $U \sim \mathrm{Unif}(S^{d-2})$,

$$\mathbb{E}|U_1| = \frac{\Gamma((d-1)/2)}{\sqrt{\pi}\,\Gamma(d/2)},$$

which gives a positive lower bound uniformly in d. Conversely,

$$-\min_{1\leq r\leq b-1} U_r \leq \max_{1\leq r\leq b-1} |U_r| \leq \left(\sum_{r=1}^{b-1} U_r^2\right)^{1/2}.$$

Thus, by Jensen,

$$\nu_{b,d} \leq \sqrt{d-1}\left(\mathbb{E}\sum_{r=1}^{b-1} U_r^2\right)^{1/2} = \sqrt{b-1}.$$

Hence $\nu_{b,d}^2 = \Theta_b(1)$ for fixed b. Since $\beta = O_{\kappa_\star}(\log d)$, Theorem 4 follows.

C.7 Fast structured mixing and the required transfer condition

Dense Haar mixing is the clean object for analysis, but it is not the intended implementation for large matrices. Sampling and applying a dense Haar matrix costs cubic time at the scale of a dense $d \times d$ Gram matrix. A much faster alternative is to use a product of normalized Walsh-Hadamard transforms and random sign diagonals. Let W_d denote the normalized Walsh-Hadamard matrix, and let $D_1, \ldots, D_k$ be independent diagonal matrices with Rademacher entries. A typical fast mixer is

$$Q_{\mathrm{FHT}}^\top = \Pi\, \mathsf{W}_d D_k \mathsf{W}_d D_{k-1} \cdots \mathsf{W}_d D_1,$$

where Π is an independent random permutation. Each factor is orthogonal, and multiplication by W_d can be performed with the fast Hadamard transform. Thus forming $Q_{\mathrm{FHT}}^\top A Q_{\mathrm{FHT}}$ for dense A costs $O(kd^2 \log d)$, rather than the dense-rotation cost. The sparse row-min update then costs $O(d^2)$, and the block row-min search costs $O(db)$ once the relevant entries of $\widetilde{A}$ are available.

The theorem above does not automatically apply to Q_{FHT}. Haar mixing supplies exact conditional rotational invariance, which is used in the row-min drift lemma. A finite product of Hadamard/sign factors is a structured discrete distribution and need not have that exact invariance. A single Hadamard-sign layer is certainly insufficient in the worst case: if $Q = \mathsf{W}_d D$ and $A = \mathsf{W}_d \Lambda \mathsf{W}_d^\top$, then $Q^\top A Q = D\Lambda D = \Lambda$, so the mixed matrix is diagonal and the row-min signal vanishes. Multiple layers and a random permutation are used to avoid such alignment effects. This worst-case observation does not contradict the empirical effectiveness of a single mixing stage per RAPID iteration ($k = 1$), since successive iterations introduce fresh mixing interleaved with sparse corrections, allowing the effects of mixing to accumulate over time.

The rest of the proof, however, is insensitive to the source of orthogonal mixing. The Taylor expansion, trace invariance, load bound, spectral lower bound, and stopping-time argument all hold for any orthogonal distribution. The only missing ingredient is a local row-min comparison estimate. We state it explicitly.

For an orthogonal mixing distribution $\mathcal{Q}$, define, for a fixed row i, block candidate set J_i, and exponent $\gamma \in \{\beta, -\beta\}$,

$$x_j = (Q^\top A Q)_{ij}, \qquad y_j^{(\gamma)} = ((Q^\top A Q)^\gamma)_{ij}, \qquad j \in J_i,$$

and

$$\tau_i = \arg\min_{j\in J_i} x_j.$$

Let

$$F_{i,J_i}^{(\gamma)}(Q,A) := y_{\tau_i}^{(\gamma)}.$$

Define the local Haar-comparison errors

$$\delta_{\mathcal{Q},\gamma}(A) := \max_i \left| \mathbb{E}_{Q\sim\mathcal{Q}} F_{i,J_i}^{(\gamma)}(Q,A) - \mathbb{E}_{Q\sim\mathrm{Haar}} F_{i,J_i}^{(\gamma)}(Q,A) \right|,$$

and the normalized error

$$\varepsilon_{\mathcal{Q},\beta}(A) := \frac{d\,\delta_{\mathcal{Q},\beta}(A)}{\mathrm{tr}(A^{\beta})} + \frac{d\,\delta_{\mathcal{Q},-\beta}(A)}{\mathrm{tr}(A^{-\beta})}.$$

Proposition 8 (Transfer from Haar to a structured orthogonal mixer)**.** *Let $\mathcal{Q}$ be any distribution on $O(d)$. Suppose that for every $A \succ 0$ with $\kappa(A) \geq \kappa_\star$,*

$$\varepsilon_{\mathcal{Q},\beta}(A) \leq \frac{\nu_{b,d}}{8\sqrt{d-1}}.$$

Then the block row-min iteration with $Q_t \sim \mathcal{Q}$ satisfies the same hitting-time guarantee as Theorem 4, up to universal constant factors:

$$\mathbb{E}[\tau_{\kappa_\star}] \leq C \frac{\beta(b-1)(d-1)}{\nu_{b,d}^2} \log \kappa(A_0).$$

Proof. The Haar drift identity gives

$$\mathcal{D}_{\mathrm{Haar}}(A) := \mathbb{E}_{\mathrm{Haar}} \left[\frac{\mathrm{tr}(R\widetilde{A}^{\beta})}{\mathrm{tr}(A^{\beta})} - \frac{\mathrm{tr}(R\widetilde{A}^{-\beta})}{\mathrm{tr}(A^{-\beta})} \,\middle|\, A \right] = -\frac{\nu_{b,d}}{\sqrt{d-1}} S_\beta(A).$$

By the definition of $\varepsilon_{\mathcal{Q},\beta}$, the corresponding drift under $Q \sim \mathcal{Q}$ satisfies

$$\mathcal{D}_{\mathcal{Q}}(A) \leq \mathcal{D}_{\mathrm{Haar}}(A) + \varepsilon_{\mathcal{Q},\beta}(A).$$

If $\kappa(A) \geq \kappa_\star$, then $S_\beta(A) \geq 1/4$, and the assumed comparison bound gives

$$\mathcal{D}_{\mathcal{Q}}(A) \leq -\frac{\nu_{b,d}}{4\sqrt{d-1}} + \frac{\nu_{b,d}}{8\sqrt{d-1}} = -\frac{\nu_{b,d}}{8\sqrt{d-1}}.$$

Substituting this into the same Taylor estimate used in the Haar proof yields

$$\mathbb{E}[\Phi_\beta(A_{t+1}) - \Phi_\beta(A_t) \mid A_t] \leq -\frac{\alpha\nu_{b,d}}{4\sqrt{d-1}} + C_{\mathrm{Tay}}\alpha^2\beta(b-1)$$

until $\tau_{\kappa_\star}$. Choosing $\alpha \asymp \nu_{b,d}/(\beta(b-1)\sqrt{d-1})$ and applying the same optional-stopping argument proves the claim. □

Proposition 8 is stated as a local comparison condition, not as a new convergence theorem for fast Hadamard products. Proving the condition for Q_{FHT} requires showing that the hard row-min statistic $F_{i,J_i}^{(\gamma)}$ has nearly the same expectation under the structured mixer as under Haar. This is the precise theoretical gap left by replacing continuous Haar rotations with fast structured orthogonal transforms. Existing incoherence and fast Johnson-Lindenstrauss results for randomized Hadamard transforms suggest why such a comparison should be plausible after several independent sign/Hadamard layers, but they do not by themselves supply the conditional rotational invariance used in the Haar proof. Empirically, we therefore use the fast Hadamard mixer for scalability, while the theorem above gives the exact guarantee for Haar mixing and identifies the comparison estimate needed to transfer the guarantee to the fast implementation.

C.8 Spiky spectra give a $\sqrt{d}$ drift gain

In the worst-case argument above we only used the uniform lower bound

$$S_\beta(A) \geq c_{\kappa_\star} > 0 \qquad \text{whenever} \qquad \kappa(A) \geq \kappa_\star.$$

For spiky spectra, $S_\beta(A)$ can be much larger. We record the simplest case, where the spectrum consists of one large spike and a flat bulk.

Lemma 9 (Rank-one spike)**.** *Assume, after rescaling, that*

$$\lambda_1 = \kappa, \qquad \lambda_2 = \cdots = \lambda_d = 1, \qquad \kappa > 1.$$

Let

$$M = \kappa^\beta.$$

Then

$$S_\beta(A) = dA_d \left[\frac{M-1}{M+d-1} + \frac{1-M^{-1}}{M^{-1}+d-1} \right],$$

where

$$A_d = \mathbb{E}\sqrt{q(1-q)}, \qquad q \sim \mathrm{Beta}\left(\frac{1}{2}, \frac{d-1}{2}\right).$$

Moreover,

$$A_d = \frac{2}{d} \frac{\Gamma(d/2)}{\sqrt{\pi}\,\Gamma((d-1)/2)} = \Theta(d^{-1/2}).$$

Consequently, if

$$\kappa^\beta \geq 2d,$$

then

$$S_\beta(A) \geq c\sqrt{d}$$

for a universal numerical constant $c > 0$. In fact, in this regime,

$$S_\beta(A) = \Theta(\sqrt{d}).$$

Proof. Let

$$p \sim \mathrm{Dir}\left(\frac{1}{2}, \ldots, \frac{1}{2}\right), \qquad q = p_1.$$

For the rank-one spike,

$$\lambda_i = 1 + (\kappa - 1)\mathbf{1}_{\{i=1\}}.$$

Set

$$\Delta = \kappa - 1.$$

Then

$$\lambda_i^\beta = 1 + (M-1)\mathbf{1}_{\{i=1\}}, \qquad \lambda_i^{-\beta} = 1 + (M^{-1}-1)\mathbf{1}_{\{i=1\}}.$$

A direct calculation gives

$$\mathrm{Var}_p(\lambda) = \Delta^2 q(1-q),$$
$$\mathrm{Cov}_p(\lambda, \lambda^\beta) = \Delta(M-1)q(1-q),$$
$$-\mathrm{Cov}_p(\lambda, \lambda^{-\beta}) = \Delta(1-M^{-1})q(1-q).$$

Therefore, using the zero-ratio convention when $\mathrm{Var}_p(\lambda) = 0$,

$$s_\beta(A) = (M-1)\mathbb{E}\sqrt{q(1-q)},$$
$$s_{-\beta}(A) = (1-M^{-1})\mathbb{E}\sqrt{q(1-q)}.$$

Writing

$$A_d := \mathbb{E}\sqrt{q(1-q)},$$

we have

$$s_\beta(A) = (M-1)A_d, \qquad s_{-\beta}(A) = (1-M^{-1})A_d.$$

Also,

$$\mathrm{tr}(A^\beta) = M + d - 1, \qquad \mathrm{tr}(A^{-\beta}) = M^{-1} + d - 1.$$

Hence

$$S_\beta(A) = \frac{d\, s_\beta(A)}{\mathrm{tr}(A^\beta)} + \frac{d\, s_{-\beta}(A)}{\mathrm{tr}(A^{-\beta})}$$

equals

$$dA_d \left[\frac{M-1}{M+d-1} + \frac{1-M^{-1}}{M^{-1}+d-1} \right].$$

It remains to estimate A_d. Since

$$q \sim \mathrm{Beta}\left(\frac{1}{2}, \frac{d-1}{2}\right),$$

we get

$$A_d = \frac{B\left(1, d/2\right)}{B\left(1/2, (d-1)/2\right)}.$$

Using

$$B\left(1, d/2\right) = \frac{2}{d}$$

and

$$B\left(1/2, (d-1)/2\right) = \frac{\sqrt{\pi}\,\Gamma((d-1)/2)}{\Gamma(d/2)},$$

we obtain

$$A_d = \frac{2}{d}\frac{\Gamma(d/2)}{\sqrt{\pi}\,\Gamma((d-1)/2)}.$$

By the standard gamma-ratio estimate,

$$\frac{\Gamma(d/2)}{\Gamma((d-1)/2)} = \Theta(\sqrt{d}),$$

and hence

$$A_d = \Theta(d^{-1/2}).$$

Thus

$$dA_d = \Theta(\sqrt{d}).$$

Finally, if $M = \kappa^\beta \geq 2d$, then

$$\frac{M-1}{M+d-1} \geq c_0$$

for a universal constant $c_0 > 0$. The second term is nonnegative, so

$$S_\beta(A) \geq c_0 dA_d \geq c\sqrt{d}.$$

The upper bound $S_\beta(A) \leq C\sqrt{d}$ follows from the same formula and the fact that both bracketed terms are at most 1. □

Remark 1 (Interpretation)**.** *The uniform theorem only uses $S_\beta(A) \gtrsim 1$. For a rank-one spike with $\kappa^\beta \gtrsim d$, the useful spectral drift is larger:*

$$S_\beta(A) \asymp \sqrt{d}.$$

This happens because a Haar row assigns mass

$$q = p_1 \sim \mathrm{Beta}\left(\frac{1}{2}, \frac{d-1}{2}\right)$$

to the spiked eigendirection, and

$$\mathbb{E}\sqrt{q(1-q)} \asymp d^{-1/2}.$$

The definition of S_β contains an additional factor of d, so the net scale is $\sqrt{d}$.

C.8.1 Consequence for the drift rate in the spiky regime

We now spell out the resulting improvement in the one-step drift. Recall the general drift estimate

$$\mathbb{E}[\Phi_\beta(A_{t+1}) - \Phi_\beta(A_t) \mid A_t] \leq -\frac{2\alpha\nu_{b,d}}{\sqrt{d-1}} S_\beta(A_t) + C_{\mathrm{Tay}}\alpha^2\beta(b-1).$$

Suppose that, on some portion of the trajectory before reaching the target condition number, the iterates satisfy the stronger spiky lower bound

$$S_\beta(A_t) \geq \sigma\sqrt{d}$$

for some $\sigma > 0$. Using the same globally safe step size

$$\alpha = \frac{\nu_{b,d}}{8C_{\mathrm{Tay}}\beta(b-1)\sqrt{d-1}},$$

we obtain

$$\mathbb{E}[\Phi_\beta(A_{t+1}) - \Phi_\beta(A_t) \mid A_t] \le -c\frac{\sigma\nu_{b,d}^2}{C_{\mathrm{Tay}}\beta(b-1)\sqrt{d}}$$

for a universal numerical constant $c > 0$, after absorbing the Taylor remainder into the leading negative term.

Consequently, if

$$S_\beta(A_t) \ge \sigma\sqrt{d}$$

holds until the hitting time

$$\tau_{\kappa_\star} = \inf\{t : \kappa(A_t) \le \kappa_\star\},$$

then the stopped-supermartingale argument gives

$$\mathbb{E}[\tau_{\kappa_\star}] \le C\frac{\beta(b-1)\sqrt{d}}{\sigma\nu_{b,d}^2}\Phi_\beta(A_0).$$

Since

$$\Phi_\beta(A_0) \le \log\kappa(A_0),$$

we get

$$\mathbb{E}[\tau_{\kappa_\star}] \le C\frac{\beta(b-1)\sqrt{d}}{\sigma\nu_{b,d}^2}\log\kappa(A_0).$$

For fixed block size b, $\nu_{b,d}^2 = \Theta_b(1)$. Thus, in the spiky regime,

$$\mathbb{E}[\tau_{\kappa_\star}] = O_{\kappa_\star,b,\sigma}\left(\beta\sqrt{d}\,\log\kappa(A_0)\right).$$

With the usual choice $\beta = O_{\kappa_\star}(\log d)$, this becomes

$$\mathbb{E}[\tau_{\kappa_\star}] = O_{\kappa_\star,b,\sigma}\left(\sqrt{d}\,\log d\,\log\kappa(A_0)\right).$$

This is a genuine $\sqrt{d}$ improvement over the worst-case $O(d\log d\,\log\kappa(A_0))$ bound.

C.9 Establishing intuition: First-order effect of a row correction

We now justify the illustrative eigenvalue calculation in Section 4; this example is not used in the proof of Theorem 4. Consider $\widetilde{A} = I + (\lambda-1)vv^\top$, with $\lambda > 0$, $\lambda \ne 1$, and $\|v\| = 1$. The eigenvalue λ is simple, with eigenvector v, while all remaining eigenvalues equal 1.

For distinct coordinates i, j, the congruence update

$$\widetilde{A}(\alpha) = (I + \alpha e_i e_j^\top)A(I + \alpha e_i e_j^\top)^\top.$$

Expanding around $\alpha = 0$ gives

$$\widetilde{A}(\alpha) = \widetilde{A} + \alpha(e_i e_j^\top\widetilde{A} + \widetilde{A}e_j e_i^\top) + O(\alpha^2).$$

By first-order eigenvalue perturbation, the eigenvalue $\lambda(\alpha)$ originating from λ satisfies

$$\begin{aligned}\lambda'(0) &= v^\top(e_i e_j^\top\widetilde{A} + \widetilde{A}e_j e_i^\top)v \\ &= 2\lambda v^\top e_i e_j^\top v = 2\lambda v_i v_j.\end{aligned}$$

Since $\widetilde{A}_{ij} = (\lambda-1)v_i v_j$ for $i \ne j$,

$$\lambda(\alpha) = \lambda + \frac{2\alpha\lambda}{\lambda-1}\widetilde{A}_{ij} + O(\alpha^2).$$

For $\widetilde{A}_{ij} < 0$ and sufficiently small $\alpha > 0$, the exceptional eigenvalue therefore moves toward 1, whether $\lambda > 1$ or $\lambda < 1$.

This calculation tracks only the exceptional eigenvalue under a single row correction. Other eigenvalues may also change, so it does not establish a decrease in the condition number of the full matrix.

C.10 Empirical evidence for the unblocked load bound

The proof in Section 6 uses block selection to control the load of the row-min update. We test whether the same type of control holds for the unblocked selector used in the practical algorithm. For a mixed matrix $A = Q\Lambda Q^\top$, define

$$\tau_i(A) = \arg\min_{j \neq i} A_{ij}, \qquad m(A) = \|\mathcal{R}(A)\|_1 = \max_j |\{i : \tau_i(A) = j\}|.$$

Thus $m(A)$ is the maximum number of rows whose minimum off-diagonal entry selects the same column. The step-size scaling used in our simulations is supported if

$$\frac{S_\beta(A)}{\sqrt{m(A)}} = \Omega(1),$$

where S_β is the spectral drift used in Lemma 2 and defined in (26). We therefore measure this ratio directly after Haar mixing with global argmin selection, with no blocking.

Throughout this experiment we set $\beta = \log d$. We generate centered log-spectra $x \in \mathbb{R}^d$, with $\lambda_i = \exp(x_i)$, and retain only spectra with $\Phi_\beta(\Lambda) \geq 1$. To search for possible failures of the bound, we sweep several adversarial spectral families: top spikes (k large eigenvalues), bottom holes (k small eigenvalues), two-block spectra (a fraction f at $+h$ and the rest at $-h$), power-law spectra ($x_i = h\,\mathrm{sign}(t_i)|t_i|^p$), and lognormal spectra ($x_i = \sigma g_i$, $g_i \sim N(0,1)$). Concretely, we vary

$$k \in \{1, 2, 4, \lfloor\sqrt{d}\rfloor, d/8, d/2\}, \quad f \in \{0.05, 0.10, 0.25, 0.50, 0.75, 0.90\},$$

$$h \in \{1, \log d, 2\log d, \sqrt{d}\}, \quad p \in \{0.25, 0.5, 1, 2, 4\}, \quad \sigma \in \{0.5, 1, 2, 4\}.$$

These families target concentrated outliers, low-eigenvalue holes, balanced two-cluster spectra that may create collisions, smooth spectral decay, and irregular heavy-tailed spectra.

For each retained spectrum and dimension, we draw 256 independent Haar matrices Q, form $A = Q\Lambda Q^\top$, compute the exact global row-min selector $\tau_i(A)$, and record $m(A)$. We estimate $S_\beta(\Lambda)$ independently using 3000 Dirichlet$(1/2, \ldots, 1/2)$ samples, matching the squared coordinates of a Haar row. Figure 9 summarizes the results, and Table 4 reports the main aggregate statistics by dimension.

Across the tested dimensions and spectral families, $S_\beta(\Lambda)/\sqrt{m(A)}$ remains bounded away from zero. The smallest values occur for the most symmetric high-collision cases, especially half-mass two-block spectra and the matching top-spike and bottom-hole constructions. Even there, the ratio stays well above zero, supporting the empirical relation

$$\frac{S_\beta(\Lambda)}{\sqrt{m(A)}} = \Omega(1).$$

Substituting this into the one-step drift bound gives

$$\alpha'_* = \Theta\left(\frac{1}{\sqrt{\beta\, m(A)\, d}}\right),$$

matching the blocked rate up to replacing the block load by the actual unblocked load.

D Reproducibility details

D.1 Compute resources

All experiments were run on a machine with an Intel Xeon Gold 6438M CPU, 125GiB system memory, and four NVIDIA RTX 6000 Ada Generation GPUs. Reported experiments used a single GPU. The software stack used Python 3.12.4, PyTorch 2.4.0+cu121, CUDA 12.8 tooling, and Triton 3.0.0.

The transformer diagnostics used 9 depths and 10 independent runs per depth, for 90 total training runs requiring approximately 90 cumulative GPU-hours. The RAPID numerical experiments were run on the same hardware using PyTorch/Triton kernels, explicit CUDA synchronization for timing, and GPU fast Walsh–Hadamard transforms [Dao, 2024].

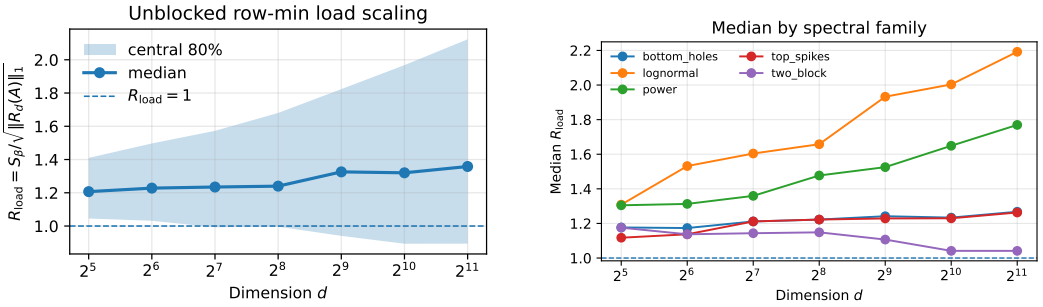

Figure 9: **Empirical load scaling for the unblocked selector.** *Left:* distribution of $S_\beta/\sqrt{m(A)}$ versus dimension, aggregated over all retained spectra and Haar trials. *Right:* median $S_\beta/\sqrt{m(A)}$ by spectral family. In all tested cases the ratio remains bounded away from zero.

d	# spectra	min $\frac{S_\beta}{\sqrt{m}}$	10th perc. $\frac{S_\beta}{\sqrt{m}}$	median $\frac{S_\beta}{\sqrt{m}}$	median m	90th perc. m	max m
32	67	0.814	1.045	1.207	5	15	24
64	67	0.804	1.031	1.228	7	22	42
128	75	0.810	0.993	1.235	8	37	81
256	75	0.753	0.996	1.240	9	117	152
512	76	0.706	0.941	1.326	9	121	289
1024	76	0.706	0.893	1.320	9	265	561
2048	76	0.707	0.894	1.358	9	430	1085

Table 4: **Aggregate statistics for the unblocked load experiment.** Each row aggregates over all retained spectra and Haar trials at the given dimension.

D.2 Existing assets and licenses

The experiments use synthetic matrices generated by our scripts and public SuiteSparse Matrix Collection instances. SuiteSparse matrices were accessed through `ssgetpy`; the SuiteSparse Matrix Collection is credited in the main text, and `ssgetpy` is distributed under the MIT/X11 license.

Our implementation uses Python 3.12.4, PyTorch 2.4.0+cu121, Triton 3.0.0, CUDA tooling, NumPy, SciPy, and `fast-hadamard-transform`. PyTorch is distributed under a BSD-style license, Triton under the MIT license, NumPy and SciPy under BSD-style licenses, `fast-hadamard-transform` under the BSD-3-Clause license, and the CUDA Toolkit is governed by the NVIDIA CUDA Toolkit EULA. We use these packages only as software dependencies and do not redistribute modified versions of them.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's contributions and scope: extracting a transformer-derived conditioning primitive, turning it into RAPID for SPD systems, proving a Haar/block-row-min idealization, and evaluating the practical dense-matrix implementation. The claims are scoped to the tested dense SPD settings and to the stated theoretical idealization.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in the conclusion and Appendix B, including the focus on explicit dense matrices, the open problem of proving the same rate for the practical fast-mixer implementation without blocking, and precision/optimization limits of the prototype GPU implementation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The main theoretical statements in Section 6 state the relevant assumptions, including Haar mixing, block row-min selection, step-size conditions, and stopping-time targets. Complete proofs and supporting lemmas are provided in Appendix C, with the main text giving the proof structure.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The algorithm is specified in Algorithm 1, and Appendix A.1 and Appendix B describe the training distribution, architecture, optimization, matrix families, baselines, and implementation details needed to reproduce the reported experiments in substance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide an anonymized supplemental notebook with installation instructions and a self-contained GPU implementation of RAPID.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the synthetic regression setup, model architecture, optimizer, training schedule, and evaluation diagnostics in Appendix A.1. Appendix B further specifies the SPD matrix families, right-hand-side generation, RAPID implementation, baseline preconditioners, and CG setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Transformer-training diagnostics report aggregation across runs using central 80% intervals. The RAPID experiments benchmark numerical solvers on fixed matrix instances; the reported conditioning, and CG quantities are direct numerical measurements rather than estimates over random train/test splits or sampled datasets.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D.1 reports the hardware, software stack, and cumulative transformer-training compute time.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses synthetic regression tasks and public numerical matrix benchmarks, and does not involve human-subject data, private or sensitive data. We are not aware of any aspect of the research that deviates from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational numerical linear algebra research on synthetic and public SPD systems; we do not identify a direct societal-impact pathway beyond generic improvements in computational efficiency. Accordingly, no separate positive/negative societal-impact analysis is included.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work does not release high-risk assets such as pretrained generative models, scraped web datasets, or human-subject data. The proposed method is a numerical preconditioning algorithm, so misuse safeguards of this type are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Appendix D.2 lists the external datasets and software dependencies used in the experiments, credits their creators or maintainers, and states the relevant licenses or terms of use. The paper uses these assets only as public datasets or software dependencies and does not redistribute modified versions of them.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The current manuscript does not release a new dataset, benchmark, or pre-trained model asset. The proposed algorithm and implementation are described for reproducibility, but no separately documented new asset is introduced in the paper.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper uses synthetic data and public numerical matrix collections only. It does not involve crowdsourcing, user studies, or other research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects. Therefore participant-risk disclosure and IRB or equivalent approval are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No LLM is used as an important, original, or non-standard component of the method. The trained model studied in the paper is a small task-specific attention-only transformer for synthetic linear regression, not an LLM.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.