Faster Solvers for Sparse Systems with Large Spectral Outliers

Yang P. Liu* Hoai-An Nguyen† Richard Peng‡ Junzhao Yang§

Computer Science Department, Carnegie Mellon University

Abstract

We study high-accuracy algorithms for solving normal equations $A^\top Ax = b$ when $A \in \mathbb{R}^{n\times d}$ has at most s nonzero entries per row. We assume that A has k large spectral outliers, meaning that all but its k largest singular values lie within a constant factor of its smallest singular value. Our randomized algorithm returns an ϵ-accurate solution with high probability in bit complexity

$$\widetilde{O}\Big((ns + \sqrt{d/k}\, k^2 + k^{\omega+\eta})n^{o(1)}\Big)$$

for every fixed $\eta > 0$, where $\widetilde{O}(\cdot)$ suppresses only polylogarithmic factors.

The main ingredient is a ridge leverage score sampling scheme that samples actual input rows at each ridge level while controlling the number of sampled rows. This gives a sparse preconditioning chain compatible with active-coordinate endpoint solvers and avoids the dense sampling bottleneck in the recursive preconditioning framework of Dereziński and Sidford (SODA 2026).

*yangpliu@cmu.edu

†hnnguyen@andrew.cmu.edu

‡yangp@andrew.cmu.edu

§junzhaoy@andrew.cmu.edu

Contents

1 Introduction **1**
1.1 Roadmap of the paper 4

2 Preliminaries **4**
2.1 Notations 4
2.2 Finite-Precision Solver Primitives 5

3 Overview **6**

4 Ridge Scores and Sparse Sampling **8**
4.1 Ridge Schedule 8
4.2 Scale Normalization 9
4.3 Ridge Score Mass Bounds 10
4.4 Score Estimation 11
4.5 Ridge Leverage Score Sampling 12

5 Constructing the Ridge Chain **13**
5.1 Top-Down Construction 13

6 Active Endpoints and Recursive Solving **16**
6.1 Active Endpoint Primitive 16
6.2 Query-Lazy Recursion 18
6.3 Final Proof 23

A Comparison Accounting **26**

1 Introduction

Linear systems and least-squares problems are basic computational primitives in numerical optimization, statistics, and machine learning. Given a design matrix A and observations y, the least-squares minimizer satisfies the normal equation $A^\top Ax = A^\top y$; closely related Gram systems arise in regression and empirical-risk minimization [Woo14, AKK+20]. Regularized variants are central to ridge regression, kernel methods, and Gaussian-process regression [AM15, DMY25]. These problems are often solved repeatedly, so reducing the cost of each high-accuracy solve can directly lower the cost of a larger learning or optimization pipeline.

Large instances frequently exhibit two useful structures at once. First, the data matrix is sparse: a sample may involve only a small subset of the available features. Second, the spectrum may have a few dominant directions above a comparatively flat bulk, as in a low-dimensional signal superimposed on an approximately isotropic noise floor. Regularization can create a similar spectral profile. Recent work shows that this flat-tail structure can substantially accelerate dense linear systems and regression [DY24, DMY25, DS25]. The sparse and spectral structures, however, do not combine automatically: a dense sketch, factorization, or preconditioner can erase the advantage of a sparse input.

This paper asks whether one can exploit both structures simultaneously. Let $A \in \mathbb{R}^{n\times d}$ have full column rank, $n \geq d$, and at most s nonzeros in every row. We solve

$$A^\top Ax = b$$

to high Euclidean accuracy when, for an input parameter k, the singular values after the kth are clustered:

$$\frac{\sigma_{k+1}(A)}{\sigma_d(A)} = O(1).$$

Thus only the top k directions may be much larger than the smallest singular value. The model includes the regression normal equations by taking $b = A^\top y$, but we allow an arbitrary right-hand side b. Since the matrix is accessed sparsely, the natural leading cost is $N = \mathrm{nnz}(A) \leq ns$. The central obstacle is to exploit the spectral tail without introducing an ambient d^2 computation or repeatedly scanning dense intermediate matrices.

Our main idea is a sparsity-aware multi-level ridge preconditioning chain. At ridge α, the algorithm samples actual input rows according to overestimates of

$$\tau_r(\alpha) = a_r^\top (A^\top A + \alpha I)^{-1} a_r,$$

the ridge leverage score of row $a_r^\top$. The ordinary ridge-score mass controls how many rows are sampled, while the weighted mass $\sum_r \mathrm{nnz}(a_r)\tau_r(\alpha)$ controls how many nonzeros survive in the sampled matrix. This second accounting is what keeps the entire recursive chain sparse. Together with active-coordinate endpoint solves, it prevents the low-rank machinery from reintroducing a dense ambient-dimensional cost.

Theorem 1.1 (Main theorem). *Let $A \in \mathbb{R}^{n\times d}$ be full column rank with $n \geq d$, with at most s nonzeros per row, and let $b \in \mathbb{R}^d$. Assume that A and b have $O(\log n)$-bit rational entries, polynomially bounded norms, and $\kappa(A) \leq \mathrm{poly}(n)$. Let $k \in \{1, \ldots, d-1\}$ be an input parameter such that*

$$\sigma_{k+1}(A) \leq C\, \sigma_d(A)$$

for an absolute constant C. For every fixed $\eta > 0$ and $\epsilon \in (0,1)$, there is a randomized algorithm which, with high probability, outputs $\widetilde{x}$ satisfying

$$\left\|\widetilde{x} - (A^\top A)^{-1} b\right\|_2 \le \epsilon$$

using

$$n^{o(1)} \widetilde{O}\left(ns + \sqrt{d/k}\, k^2 + k^{\omega+\eta}\right)$$

bit operations. Here $\widetilde{O}(\cdot)$ hides only polylogarithmic factors in the input size, $1/\epsilon$, condition parameters, and inverse failure probability.

The theorem is stated in bit complexity. Endpoint systems are restricted to active coordinate sets of size $\widetilde{O}(ks)$, while the dense Woodbury algebra inside the endpoint routine has dimension $\widetilde{O}(k)$. By Theorem 2.5, stable fast linear algebra gives the $k^{\omega+\eta}$ term for every fixed $\eta > 0$. We use the real-RAM primitives of Dereziński and Sidford [DS25] through the finite-precision versions in Theorems 2.3 and 2.4.

Comparison. The closest predecessor is the low-rank-tail line of work on dense solvers [DS25, DY24, DMY25]. Our proof uses the recursive preconditioning and endpoint primitives of [DS25]; the main new constraint is that the entire chain must remain a sequence of sparse row submatrices.

Plain CG or Lanczos gives a useful sparse baseline. One multiplication by $A^\top A$ costs $O(\mathrm{nnz}(A))$, and under the same spectral clustering assumption the usual outlier-cluster analysis gives $k + \mathrm{polylog}(1/\epsilon)$ iterations, hence $\widetilde{O}(nks)$ sparse work. Table 1 compares this baseline, the prior recursive framework, and our result.

Method	Running time	Comment
CG/Lanczos sparse matvec baseline	$\widetilde{O}(nks)$	Sparse matvec baseline.
Prior recursive framework [DS25]	$\widetilde{O}\left(ns + \sqrt{d/k}\left(k^2 + \min\{ns, \sqrt{nskd}\}\right) + k^\omega\right)$	Sparse specialization; see Section A.
This work	$n^{o(1)} \widetilde{O}\left(ns + \sqrt{d/k}\, k^2 + k^{\omega+\eta}\right)$	Bit-complexity bound.

Table 1: Comparison under the sparse normal-equation model with row sparsity s, $N = \mathrm{nnz}(A) \le ns$, and bounded spectral tail after the input parameter k. The prior-framework row is the sparse accounting derived in Section A.

The prior-framework accounting has an additional

$$\sqrt{d/k}\, \min\{ns, \sqrt{nskd}\}$$

term for sparse sampled levels. Ridge sampling charges their nonzeros directly to the input sparsity, leaving the endpoint contribution $\sqrt{d/k}\, k^2$, up to the $n^{o(1)}$ recursion factor.

When is the bound near input sparsity? Set $n = d$, write $s = d^x$ and $k = d^y$, and assume $0 \le x, y \le 1$. Then $\mathrm{nnz}(A) = ns = d^{1+x}$. The CG/Lanczos baseline is near input sparsity only for $y = 0$, up to polylogarithmic k, because the nks term is d^{1+x+y}. The prior-framework accounting

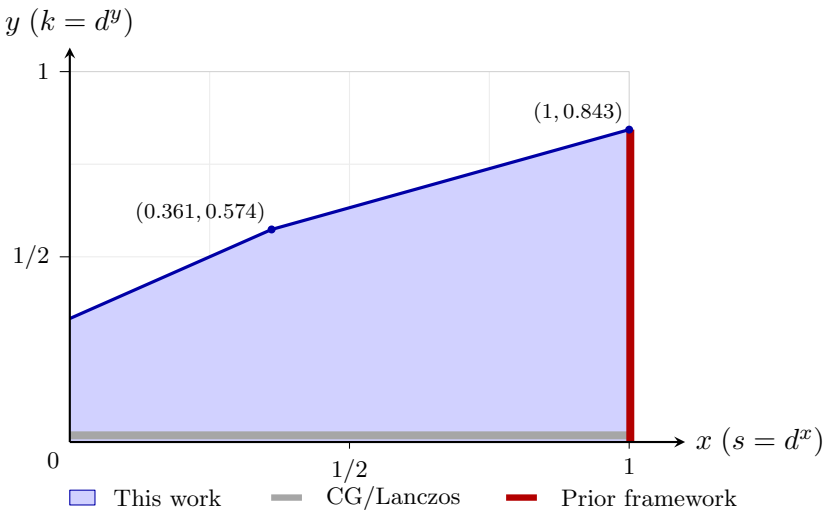

Figure 1: Exponent-region schematic for $n = d$, $s = d^x$, and $k = d^y$. The analytic condition for the blue region is $y \le \min\{(2x+1)/3, (1+x)/(\omega+\eta)\}$. The displayed curve uses the feasible bound $\omega < 2.371339$ [ADW+25], treating the arbitrarily small fixed η as negligible for visualization. Throughout the shaded region the non-input terms in Theorem 1.1 are at most $d^{1+x} = \mathrm{nnz}(A)$, giving $n^{o(1)}\widetilde{O}(\mathrm{nnz}(A))$ time. The CG/Lanczos baseline and the sparse accounting of [DS25] are drawn as gray and red line segments to make their horizontal and vertical character visible.

is near input sparsity only on the dense-row boundary $x = 1$, apart from the matrix-multiplication cap on k. For Theorem 1.1, the non-input terms are bounded by d^{1+x} when

$$\frac{1}{2} + \frac{3}{2}y \le 1 + x \qquad \text{and} \qquad (\omega+\eta)y \le 1 + x,$$

or equivalently

$$y \le \min\left\{\frac{2x+1}{3}, \frac{1+x}{\omega+\eta}\right\}.$$

In this region, Theorem 1.1 gives $n^{o(1)}\widetilde{O}(\mathrm{nnz}(A))$ time.

Related work. Randomized numerical linear algebra reduces large regression and approximation problems by first compressing the input. Sparse subspace embeddings made it possible to obtain leading terms proportional to $\mathrm{nnz}(A)$ for least squares, leverage-score approximation, and low-rank approximation [CW13, NN12, Woo14]. This sketching viewpoint is especially effective when the reduced problem can be handled without forming a new ambient dense matrix.

Sampling rows or columns of the input provides a complementary, data-aware route to sparsity preservation. Iterative row sampling and leverage-score methods have led to faster sparse regression and empirical-risk minimization [LMP13, AKK+20]. Ridge leverage scores incorporate a regularization scale and control an effective dimension rather than the full rank. They support input-sparsity low-rank approximation, column subset selection, and streaming algorithms [CMM17], and are closely connected to Nyström approximations for kernel matrices [AM15, MM17]. Our use of ridge scores differs in that sampling is repeated across a solver hierarchy and must control the number of retained nonzeros, not only the number of sampled rows.

For structured linear systems, classical Krylov methods already benefit from spectral clustering, while recent randomized algorithms explicitly exploit a flat spectral tail. Dereziński and Yang developed faster algorithms parameterized by the number of large singular values [DY24]. Multi-level sketched preconditioning subsequently gave improved tradeoffs and effective-dimension bounds for regularized systems [DMY25]. Dereziński and Sidford obtained near-optimal dense-system and regression bounds using recursive preconditioning [DS25]. The present work focuses on the complementary sparse-row regime: under the stronger row-sparsity and flat-tail assumptions, the goal is to retain the spectral gains without paying an ambient d^2 term.

1.1 Roadmap of the paper

Section 2 fixes notation, the sparse access model, and the finite-precision solver primitives, while Section 3 gives the proof outline. The technical proof begins in Section 4, which specifies the ridge schedule, gives the scale-normalization reduction, and then establishes the ridge-score mass bounds, score estimation, and ridge leverage score sampling. Section 5 constructs the sampled ridge preconditioning chain from high ridge levels down to low ones. Section 6 proves the active-coordinate endpoint primitive, the query-lazy recursive solver, the shifted solves used during construction, and the final running-time combination. Section A contains the comparison accounting behind Table 1.

2 Preliminaries

2.1 Notations

For an integer m, let $[m] = \{1, \dots, m\}$; I denotes an identity matrix of the dimension clear from context, $\|\cdot\|$ denotes the Euclidean norm for vectors and the spectral norm for matrices, and $\|x\|_M = (x^\top M x)^{1/2}$ for positive definite M. We write $\mathrm{tr}(M)$ for the trace, $\mathrm{nnz}(M)$ for the number of nonzero entries, $\#\,\mathrm{rows}(M)$ for the number of rows, and A_S for the restriction of A to columns indexed by S. For symmetric matrices, $X \preceq Y$ is the Loewner order, and, for $c \geq 1$, $X \approx_c Y$ means $c^{-1}Y \preceq X \preceq cY$.

Throughout, $A \in \mathbb{R}^{n \times d}$ has full column rank, $H = A^\top A$, $a_r^\top$ is row r of A, $w_r = \mathrm{nnz}(a_r)$, and $N = \mathrm{nnz}(A) = \sum_r w_r$; the row-sparsity assumption is $w_r \leq s$, and full column rank implies $N \geq d$. We order the singular values as $\sigma_1 \geq \cdots \geq \sigma_d > 0$ and write $\lambda_j = \sigma_j^2$ for the eigenvalues of H; under the normalization $\lambda_d = 1$, the hypothesis $\sigma_{k+1}/\sigma_d = O(1)$ is equivalent to $\lambda_{k+1} = O(1)$. The notation $\widetilde{O}(\cdot)$ hides factors polylogarithmic in n/ϵ, the promised condition parameters, and inverse failure probabilities. We say that an event holds with high probability if it occurs with probability at least $1 - (n/\epsilon)^{-c}$ for a sufficiently large absolute constant $c > 0$.

Definition 2.1 (Ridge matrices and ridge scores)**.** *For $\alpha \geq 0$, write $M(\alpha) = H + \alpha I$. For $\alpha > 0$, define the ridge score of row r by $\tau_r(\alpha) = a_r^\top M(\alpha)^{-1} a_r$. The identity*

$$\sum_{r=1}^{n} \tau_r(\alpha) = \mathrm{tr}\big(H(H + \alpha I)^{-1}\big)$$

is used repeatedly in the sampling analysis.

Sparse access model. The matrix A is stored by its nonzeros. For an explicit dense vector $v \in \mathbb{R}^d$, computing $Hv = A^\top(Av)$ costs $\widetilde{O}(N)$ bit operations; coordinate access and dense-vector arithmetic cost $O(d) \leq O(N)$.

2.2 Finite-Precision Solver Primitives

The following lemmas state the finite-precision variants of the two real-RAM solver primitives used in the proof.

Definition 2.2 (Relative solver)**.** *Let G be positive definite, let $\vartheta \in (0,1)$, and let $\delta \in [0,1)$. A (ϑ,δ)-solver for G is a randomized routine which, on input y, returns $\widehat{x}$ satisfying*

$$\left\|\widehat{x} - G^{-1}y\right\|_G \le \vartheta \left\|G^{-1}y\right\|_G$$

with failure probability at most δ. If the routine is deterministic after a successful randomized preprocessing step, the parameter δ refers to that preprocessing failure probability.

Lemma 2.3 (Lemma 7 of [DS25], finite-precision version)**.** *Let $M, N \in \mathbb{R}^{q\times q}$ be positive definite with $M \preceq N \preceq \kappa M$, where $\kappa \ge 1$, and let f be a deterministic $((10\kappa)^{-1},0)$-solver for N with bit cost T_f. Suppose a product with M costs T_M bit operations, the explicit input scalars have $\mathrm{polylog}(n/\epsilon)$-bit representations and polynomial magnitude, $\kappa \le (n/\epsilon)^{O(1)}$, and $\log(1/\vartheta) = \mathrm{polylog}(n/\epsilon)$. There is a finite-precision $(\vartheta,0)$-solver for M that uses at most*

$$\left\lceil 4\sqrt{\kappa}\log(2/\vartheta)\right\rceil$$

calls to f, the same number of products with M, and $O(q\sqrt{\kappa}\log(1/\vartheta))$ additional vector operations. Its bit complexity is

$$O\big(\sqrt{\kappa}\log(2/\vartheta)(T_f + T_M)\big) + \widetilde{O}\big(q\sqrt{\kappa}\log(2/\vartheta)\big)\,.$$

Finite-precision justification. Run the real-RAM recurrence from the cited lemma with target $\vartheta/2$. Its total number Q of scalar and oracle operations is polynomial in n/ϵ. Using $p = \mathrm{polylog}(n/\epsilon)$ bits makes the energy-norm error of each product, preconditioner call, and vector operation at most $\vartheta/(100(Q+1))$ times the relevant input norm. A direct induction on the fixed recurrence bounds the accumulated numerical error by $\vartheta/2$; arithmetic on p-bit words contributes only the displayed polylogarithmic overhead.

Lemma 2.4 (Lemma 15 of [DS25], finite-precision version)**.** *Let $C \in \mathbb{R}^{m\times q}$, $\nu > 0$, and $\vartheta,\delta \in (0,1)$. Suppose the entries of C and ν have $\mathrm{polylog}(n/\epsilon)$-bit representations and polynomial magnitude, $\kappa = 1 + \|C\|_2^2/\nu \le (n/\epsilon)^{O(1)}$, and $\log(1/\vartheta)+\log(1/\delta) = \mathrm{polylog}(n/\epsilon)$. A finite-precision randomized algorithm with preprocessing bit complexity*

$$\widetilde{O}\big(\mathrm{nnz}(C) + (m+\log(1/\delta))^{\omega+\eta}\big)$$

returns, with probability at least $1-\delta$, a (ϑ,δ)-solver for $C^\top C + \nu I_q$ whose application time is

$$\widetilde{O}\big((\mathrm{nnz}(C) + m^2)\log(\kappa/\vartheta)\big)$$

bit operations, for every fixed $\eta > 0$.

Finite-precision justification. Run the real-RAM construction from the cited lemma with constant slack and target $\vartheta/4$. Sparse embedding entries and row-rescaling factors can be rounded to $p = \mathrm{polylog}(n/\epsilon)$ bits so that the resulting Gram-matrix perturbation has relative norm at most $1/100$; this only changes the embedding constants by absolute factors. The dense factorization is covered by Theorem 2.5, and the subsequent Woodbury reduction and preconditioned iteration

are covered by the preceding finite-precision version of Lemma 7. Summing the per-operation errors absorbs them into the remaining $3\vartheta/4$ slack, while splitting the failure budget between the embedding and real-RAM success events gives overall failure probability at most δ.

Finally, we use stable fast dense linear algebra [DDH07].

Lemma 2.5 (Stable fast dense linear algebra)**.** *For every fixed $\eta > 0$, the polynomially conditioned dense $r \times r$ inversions, factorizations, and associated solves can be performed stably in finite precision using $\widetilde{O}(r^{\omega+\eta})$ bit operations.*

The following standard Chebyshev statement records the fixed linear approximate inverse used when the preconditioner itself is a positive definite linear operator.

Lemma 2.6 (Finite-precision Chebyshev approximate inverse)**.** *Let $G \in \mathbb{R}^{q\times q}$ be positive definite and let Z be a positive definite linear operator on $\mathbb{R}^q$. Suppose the spectrum of $Z^{1/2}GZ^{1/2}$ is contained in $[\alpha, \beta]$, and let $\kappa = \beta/\alpha$. For every $\vartheta \in (0, 1/2)$, there is a fixed linear operator $\mathcal{C}_\vartheta(G, Z)$ such that*

$$\left\|\mathcal{C}_\vartheta(G,Z)y - G^{-1}y\right\|_G \le \vartheta \left\|G^{-1}y\right\|_G \quad \text{for all } y,$$

and

$$(1-\vartheta)G^{-1} \preceq \mathcal{C}_\vartheta(G,Z) \preceq (1+\vartheta)G^{-1}.$$

It can be applied in finite precision using $O(\sqrt{\kappa}\log(2/\vartheta))$ applications of Z, the same number of products with G, and $\widetilde{O}(q\sqrt{\kappa}\log(2/\vartheta))$ additional bit operations, provided the inputs have polylogarithmic bit length and polynomial magnitude, $\kappa \le (n/\epsilon)^{O(1)}$, and $\log(1/\vartheta) = \mathrm{polylog}(n/\epsilon)$.

Proof. The Chebyshev minimax polynomial p_t for t^{-1} on $[\alpha, \beta]$ satisfies

$$\max_{\lambda\in[\alpha,\beta]} |1 - \lambda p_t(\lambda)| \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t.$$

Taking $t = O(\sqrt{\kappa}\log(2/\vartheta))$ makes the right-hand side at most $\vartheta/2$. Define

$$\mathcal{C}_\vartheta(G,Z) = Z^{1/2}p_t(Z^{1/2}GZ^{1/2})Z^{1/2}.$$

Functional calculus gives the Loewner bounds and the G-norm error in exact arithmetic. The three-term Chebyshev recurrence applies this operator using only products with G and applications of Z. Running that fixed recurrence with $p = \mathrm{polylog}(n/\epsilon)$ working bits keeps the accumulated rounding error below the remaining $\vartheta/2$ slack and adds only the displayed polylogarithmic bit overhead. □

3 Overview

Normalization and ridge scores. The proof starts from the normal equations

$$Hx = b, \qquad H = A^\top A, \qquad N = \mathrm{nnz}(A),$$

and uses the bounded spectral tail to build a sparse preconditioning chain for the ridge matrices $M(\alpha) = H + \alpha I$. A scale-grid reduction in Theorem 4.2 lets the analysis assume $\lambda_d(H) = 1$. Under this normalization the tail condition gives $\lambda_{k+1}(H) = O(1)$, and a sufficiently accurate residual solve for $Hx = b$ is already a Euclidean-accurate approximation to $H^{-1}b$.

The main sampling quantity is the ridge leverage score

$$\tau_r(\alpha) = a_r^\top (H + \alpha I)^{-1} a_r$$

of row $a_r^\top$. The identity

$$\sum_r \tau_r(\alpha) = \mathrm{tr}\big(H(H + \alpha I)^{-1}\big)$$

and the bounded-tail assumption give

$$\sum_r \tau_r(\alpha) \le k + O(d/\alpha), \qquad \sum_r \mathrm{nnz}(a_r)\tau_r(\alpha) \le sk + O(N/\alpha).$$

These bounds are proved in Theorem 4.3. The second controls the number of sampled nonzeros and keeps the ridge chain sparse.

Ridge scales and chain construction. The algorithm constructs a preconditioning chain for the ridge systems $M(\nu_i) = H + \nu_i I$ for $i \in [0, T]$. We use a geometric sequence of ridge scales so that adjacent systems differ by a uniform factor while the chain reaches polynomially large ridge parameters in a logarithmic number of levels. The important scales are

$$\nu_0 = 0, \qquad \nu_1 = \gamma, \qquad \nu_L = \rho := \max\{d/k, 2\}, \qquad \nu_T = \rho^D \ge \lambda_1(H).$$

Here $\nu_0 = 0$ is the unregularized target and ν_1 is the first sampled level. We set $L = \lceil \sqrt{\log n} \rceil$ and $\gamma = \rho^{1/L}$. At the intermediate level L, the ridge-score mass bound becomes $k + O(d/\nu_L) = O(k)$, which marks the start of the endpoint regime. The sequence continues until ν_T dominates the spectrum of H, so $M(\nu_T)$ has constant condition number and can be solved directly. The precise schedule is given in Theorem 4.1.

The corresponding positive-ridge preconditioners are constructed top down. For $i < T$, the already constructed matrix P_{i+1} approximates $M(\nu_{i+1})$, and

$$M(\nu_i) \preceq M(\nu_{i+1}) \preceq \gamma M(\nu_i).$$

Thus P_{i+1} is an $O(\gamma)$-conditioned preconditioner for $M(\nu_i)$. The resulting approximate solves in $M(\nu_i)$ allow the Johnson–Lindenstrauss estimator to compute overestimates $u_{i,r} \ge \tau_r(\nu_i)$. Sampling with these overestimates produces a row matrix B_i such that

$$P_i := B_i^\top B_i + \nu_i I \approx_c H + \nu_i I, \qquad \#\,\mathrm{rows}(B_i) = \widetilde{O}\left(k + \frac{d}{\nu_i}\right), \qquad \mathrm{nnz}(B_i) = \widetilde{O}\left(sk + \frac{N}{\nu_i}\right).$$

The zero-ridge matrix is not sampled. Note we set $P_0 = H$.

Active-coordinate endpoints. Endpoint levels are the levels $i \ge L$, where $\nu_i \ge \nu_L = \rho \ge d/k$. The preceding bounds give $\#\,\mathrm{rows}(B_i) = \widetilde{O}(k)$ and $\mathrm{nnz}(B_i) = \widetilde{O}(ks)$. Let

$$T_i = \{q \in [d] : \text{ column } q \text{ of } B_i \text{ contains a nonzero}\}, \qquad d_i = |T_i|.$$

Then $d_i \le \mathrm{nnz}(B_i) = \widetilde{O}(ks)$. The endpoint primitive applies Theorem 2.4 to the restricted matrix B_{i,T_i}, not to an ambient d-dimensional matrix. After permuting coordinates,

$$B_i^\top B_i + \nu_i I_d = \begin{pmatrix} B_{i,T_i}^\top B_{i,T_i} + \nu_i I_{d_i} & 0 \\ 0 & \nu_i I_{T_i^c} \end{pmatrix},$$

so the inverse action outside T_i is just diagonal scaling by ν_i^{-1}. The restricted matrix B_{i,T_i} has only $\widetilde{O}(k)$ rows and $\widetilde{O}(ks)$ nonzeros, yielding endpoint preprocessing $\widetilde{O}(ks + k^{\omega+\eta})$ in bit complexity and endpoint application time

$$T_{\text{end}} = \widetilde{O}(ks + k^2)$$

on the lazy active-coordinate inputs used by the recursion.

The lazy-vector interface is essential only for recursive boundary calls. A right-hand side passed to the endpoint solver is represented as a lazy scalar multiple of an inherited vector plus an explicit correction on active coordinates. The endpoint forms the restriction to T_i, solves the active block, and returns

$$P_i^{-1} y = \nu_i^{-1} y + \Delta_{T_i}$$

with Δ_{T_i} stored explicitly. Dense right-hand sides are materialized only at the root and in the score-estimation calls, where their cost is charged to the sparse root access bound $\widetilde{O}(N)$. This is why the endpoint cost used repeatedly in the recursion is T_{end} rather than an ambient d-dimensional vector cost.

Recursive solving. After constructing $B_1, \ldots, B_L$ and the corresponding preconditioners $P_1, \ldots, P_L$, the algorithm performs the final solve recursively from the endpoint P_L down to $P_0 = H$. Adjacent matrices satisfy a constant-factor version of

$$P_i \preceq O(1) P_{i+1} \preceq O(\gamma) P_i,$$

so the robust adjacent-preconditioning primitive in Theorem 2.3 turns a sufficiently accurate solver for P_{i+1} into a solver for P_i using $O(\sqrt{\gamma})$ preconditioner calls, up to logarithmic factors. The recurrence depth is $L = \lceil \sqrt{\log n} \rceil$, so the accumulated endpoint multiplier is

$$(\sqrt{\gamma}\ \operatorname{polylog} n)^L = n^{o(1)} \sqrt{\rho}.$$

Running time. Combining the chain construction, endpoint preprocessing, and the final first-block solve gives

$$n^{o(1)} \widetilde{O}\big(N + n + \sqrt{\rho}\,(ks + k^2) + k^{\omega+\eta}\big).$$

Since $\rho = \max\{d/k, 2\}$, $N \le ns$, $n \ge d$, and $\sqrt{d/k}\, ks \le ns$, this is

$$n^{o(1)} \widetilde{O}\Big(ns + \sqrt{d/k}\, k^2 + k^{\omega+\eta}\Big).$$

4 Ridge Scores and Sparse Sampling

This section first fixes the ridge schedule and reduces the input to the normalized scale used throughout the analysis. It then proves the row-sampling primitives that keep the resulting preconditioning chain sparse: ridge-score mass bounds, score estimation from approximate solves, and sampling that preserves each ridge matrix.

4.1 Ridge Schedule

Definition 4.1 (Ridge schedule)**.** *The algorithm uses*

$$\rho = \max\{d/k, 2\}, \qquad L = \lceil \sqrt{\log n} \rceil, \qquad \gamma = \rho^{1/L}.$$

Choose $D = O(\log n)$ large enough that ρ^D dominates the promised polynomial upper bound on $\lambda_1(H)$ in the normalized scale, set $T = DL$, and define

$$\nu_0 = 0, \qquad \nu_i = \gamma^i \quad (1 \le i \le T).$$

Thus $\nu_L = \rho$ and $M(\nu_T)$ has constant condition number. Failure budgets are assigned so that their sum is at most $(n/\epsilon)^{-c}$ for a sufficiently large absolute constant c.

The schedule is analyzed in the normalized scale $\lambda_d(H) = 1$. The next reduction justifies this assumption without changing the target solution.

4.2 Scale Normalization

Lemma 4.2 (Normalization by a scale grid)**.** *Suppose there is an algorithm that, whenever a scaled instance satisfies*

$$\frac{1}{2} \le \lambda_d(A^\top A) \le 2 \qquad \text{and} \qquad \lambda_{k+1}(A^\top A) \le C_1$$

for an absolute constant C_1, returns a ξ-accurate Euclidean solution in $T_{\mathrm{norm}}(\xi)$ bit operations. Then the original unscaled instance can be solved using $O(\log n)$ calls to this algorithm and an additional $\widetilde{O}(N)$ bit operations per call for residual checks.

Proof. Let $H = A^\top A$. The promises $\|A\|_2 \le \mathrm{poly}(n)$ and $\kappa(A) \le \mathrm{poly}(n)$ imply that, for some fixed exponent C_* determined by the promise class,

$$K^{-1} \le \lambda_d(H) \le \lambda_1(H) \le K, \qquad \|b\|_2 \le K, \qquad K = n^{C_*}.$$

The algorithm may use this fixed polynomial bound; increasing C_* only changes polylogarithmic factors.

The detailed proof below is written with the convenient normalization $\lambda_d(H) = 1$. The same arguments give the robust form in the lemma statement when $\lambda_d(H) \in [1/2, 2]$: Theorem 4.3 changes only by absolute constants, the adjacent-conditioning comparisons still have condition number $O_c(\gamma)$, and the final conversion from H-norm error to Euclidean error uses $H \succeq (1/2)I$ instead of $H \succeq I$.

For a scale $\theta > 0$, define $A_\theta = \theta^{-1/2} A$ and $b_\theta = \theta^{-1} b$. Then

$$A_\theta^\top A_\theta x = b_\theta, \qquad A_\theta^\top A_\theta = \theta^{-1} H,$$

and the exact solution remains $H^{-1}b$. Use the geometric grid

$$\Theta = \{2^j : K^{-1}/2 \le 2^j \le 2K\}.$$

This grid has $O(\log n)$ elements and contains a scale θ_* satisfying

$$\frac{1}{2}\lambda_d(H) \le \theta_* \le 2\lambda_d(H).$$

For every $\theta \in \Theta$, the scaled right-hand side obeys $\|b_\theta\|_2 \le 2K^2$, so it remains in the polynomial norm regime used by the normalized solver. For this scale,

$$\frac{1}{2} \le \lambda_d(A_{\theta_*}^\top A_{\theta_*}) \le 2, \qquad \lambda_{k+1}(A_{\theta_*}^\top A_{\theta_*}) = \frac{\lambda_{k+1}(H)}{\theta_*} \le O(1),$$

using the input tail assumption.

Set $\xi = \epsilon/(16K^2)$. For every $\theta \in \Theta$, run this algorithm on (A_θ, b_θ) with target ξ and failure probability $\delta/(4|\Theta|)$. Let x_θ be the returned candidate. Compute a residual estimate $\widehat{r}_\theta$ for $r_\theta = Hx_\theta - b$ to additive Euclidean accuracy $\epsilon/(16K)$, using the sparse product $Hx_\theta = A^\top(Ax_\theta)$ under the sparse access model. Output a candidate minimizing $\|\widehat{r}_\theta\|_2$.

On the successful scale θ_*, the candidate satisfies

$$\left\|x_{\theta_*} - H^{-1}b\right\|_2 \le \xi,$$

and hence

$$\|r_{\theta_*}\|_2 \le \lambda_1(H)\xi \le \frac{\epsilon}{16K}.$$

The residual-estimation tolerance and the minimum-residual selection rule imply that the selected candidate $\widetilde{x}$ has

$$\|H\widetilde{x} - b\|_2 \le \frac{\epsilon}{4K}.$$

Since $\lambda_d(H) \ge K^{-1}$,

$$\left\|\widetilde{x} - H^{-1}b\right\|_2 \le \left\|H^{-1}\right\|_2 \|H\widetilde{x} - b\|_2 \le \epsilon.$$

The factor K^2 in ξ changes only the polylog(n/ϵ) precision factors. The union bound over the grid gives the claimed high-probability guarantee, and the residual computations cost $\widetilde{O}(N)$ bit operations per scale. □

With the normalized scale and geometric schedule now fixed, we turn to the score bounds that control the size and sparsity of every sampled level.

4.3 Ridge Score Mass Bounds

The mass bounds below are the only point where the bounded spectral tail and row sparsity enter the sampling analysis directly. The unweighted bound controls the number of sampled rows, and the weighted bound controls the number of sampled nonzeros.

Lemma 4.3 (Ridge tail and weighted mass bounds)**.** *Assume $\lambda_d(H) = 1$ and $\lambda_{k+1}(H) \le C_0$ for an absolute constant C_0. For every $\alpha > 0$,*

$$\sum_{r=1}^{n} \tau_r(\alpha) \le k + \frac{C_0 d}{\alpha}$$

and

$$\sum_{r=1}^{n} w_r \tau_r(\alpha) \le sk + \frac{C_0 N}{\alpha}.$$

Proof. Let $u_1, \dots, u_d$ be an orthonormal eigenbasis for H, with $Hu_j = \lambda_j u_j$. Since $H = \sum_r a_r a_r^\top$,

$$\sum_{r=1}^{n} \tau_r(\alpha) = \operatorname{tr}(H(H + \alpha I)^{-1}) = \sum_{j=1}^{d} \frac{\lambda_j}{\lambda_j + \alpha}.$$

The first k summands are at most 1, and for $j > k$ we have $\lambda_j \le C_0$, so the first bound follows.

For the weighted estimate, let $P_{\le k} = \sum_{j \le k} u_j u_j^\top$ and $P_{>k} = I - P_{\le k}$. For the top part,

$$\sum_r w_r \, a_r^\top P_{\le k} (H + \alpha I)^{-1} P_{\le k} a_r \le s \sum_{j \le k} \frac{\lambda_j}{\lambda_j + \alpha} \le sk.$$

For the tail part, $(H + \alpha I)^{-1} \preceq \alpha^{-1} I$ on the tail subspace, so

$$\sum_r w_r \, a_r^\top P_{>k} (H + \alpha I)^{-1} P_{>k} a_r \le \frac{1}{\alpha} \sum_r w_r \, \|P_{>k} a_r\|_2^2 .$$

The matrix $AP_{>k}$ has spectral norm at most $\sqrt{\lambda_{k+1}} \le \sqrt{C_0}$, hence every row satisfies $\|P_{>k} a_r\|_2^2 \le C_0$. The last display is therefore at most $C_0 N/\alpha$. □

4.4 Score Estimation

This subsection converts absolute solves for $M(\alpha)$ into row-wise overestimates of $\tau_r(\alpha)$. The estimates have a small additive floor so that approximate solves and sampling concentration can be union bounded over all rows and levels.

Definition 4.4 (Absolute solve call)**.** *Let G be positive definite. An absolute G-solve call with tolerance $\mu > 0$ and right-hand side g returns a vector $\widehat{x}$ satisfying*

$$\left\| \widehat{x} - G^{-1} g \right\|_G \le \mu.$$

Lemma 4.5 (JL ridge-score estimator)**.** JLRidgeScoreEstimate *satisfies its stated guarantee and uses $q = O(\log(n/\delta))$ absolute solve calls, plus $\widetilde{O}(qN + qn)$ bit operations.*

Proof. Let

$$C_\alpha = \begin{pmatrix} A \\ \sqrt{\alpha} I_d \end{pmatrix}, \qquad M_\alpha = C_\alpha^\top C_\alpha.$$

For row r, define $v_r = C_\alpha M_\alpha^{-1} a_r$. Then

$$\|v_r\|_2^2 = a_r^\top M_\alpha^{-1} C_\alpha^\top C_\alpha M_\alpha^{-1} a_r = \tau_r(\alpha).$$

The finite-set Rademacher JL guarantee gives, with probability at least $1 - \delta$,

$$\frac{3}{4} \tau_r(\alpha) \le \|S v_r\|_2^2 \le \frac{5}{4} \tau_r(\alpha) \qquad \text{for all } r.$$

Condition on this event. If $x_j = M_\alpha^{-1} g_j$, then the jth coordinate of $S v_r$ equals $a_r^\top x_j$. Let $e_j = \widehat{x}_j - x_j$. By the absolute solve guarantee and Cauchy–Schwarz in the M_α inner product,

$$|a_r^\top e_j| \le \sqrt{a_r^\top M_\alpha^{-1} a_r} \, \|e_j\|_{M_\alpha} \le \sqrt{\tau_r(\alpha)} \, \mu_{\mathrm{JL}}.$$

Thus the perturbation vector in the q sketched coordinates has norm at most $c_{\mathrm{JL}} \sqrt{\tau_r(\alpha)}$. Choosing c_{JL} sufficiently small changes the squared norm by only a constant factor. Therefore the output is at least $\tau_r(\alpha)$ and at most a constant times $\tau_r(\alpha) + \zeta$. The work bound is the cost of forming q right-hand sides, making q solve calls, multiplying by A, and evaluating the row inner products. □

JLRidgeScoreEstimate: Ridge-score overestimates from absolute solves.
Input: A, a ridge parameter $\alpha > 0$, a floor $\zeta > 0$, a failure probability $\delta \in (0, 1/2)$, and an oracle supporting absolute $M(\alpha)$-solve calls.
Output: overestimates $u_r(\alpha)$ for $r \in [n]$.
Guarantee: with probability at least $1 - \delta$,

$$\tau_r(\alpha) \le u_r(\alpha) \le C_{\mathrm{JL}}(\tau_r(\alpha) + \zeta)$$

for every row r, where C_{JL} is an absolute constant.
Description:

1. Set $q = O(\log(n/\delta))$ and $\mu_{\mathrm{JL}} = (c_{\mathrm{JL}}/\sqrt{q}) \min\{1, \sqrt{\zeta}\}$. Draw a Johnson–Lindenstrauss matrix $S = q^{-1/2}R$, where $R \in \{-1, +1\}^{q \times (n+d)}$ has independent Rademacher entries.

2. For each $j \in [q]$, form

$$g_j = \begin{pmatrix} A \\ \sqrt{\alpha} I_d \end{pmatrix}^\top S_{j,:}^\top$$

   and compute an absolute $M(\alpha)$-solve $\widehat{x}_j$ for g_j with tolerance μ_{JL}.

3. Output

$$u_r(\alpha) = 2 \left(\sum_{j=1}^{q} (a_r^\top \widehat{x}_j)^2 + \zeta \right)$$

   for each row r.

RidgeScoreSample: Ridge leverage score row sampling.
Input: A, a ridge $\alpha > 0$, row weights $w_r = \mathrm{nnz}(a_r)$, overestimates $u_r \ge \tau_r(\alpha)$, and failure probability $\delta \in (0, 1/2)$.
Output: a reweighted sampled-row matrix B.
Guarantee: with probability at least $1 - \delta$,

$$B^\top B + \alpha I \approx_c H + \alpha I,$$

$$\#\,\mathrm{rows}(B) = \widetilde{O}(U + 1), \qquad \mathrm{nnz}(B) = \widetilde{O}(W + s),$$

where $U = \sum_r u_r$ and $W = \sum_r w_r u_r$.
Description:

1. For every $r \in [n]$, set $p_r = \min\{1, C_s u_r \log(d/\delta)\}$, where C_s is a sufficiently large absolute constant.

2. For every $r \in [n]$, independently sample row r with probability p_r. If it is sampled, append $a_r^\top / \sqrt{p_r}$ to B.

4.5 Ridge Leverage Score Sampling

Given row-wise overestimates, the sampler below draws actual input rows and rescales them. Its spectral guarantee is the standard ridge leverage sampling matrix Bernstein argument, while scalar

concentration gives the row and nonzero counts needed later.

Lemma 4.6 (Ridge leverage score sampling)**.** RidgeScoreSample *satisfies its stated guarantee and runs in* $\widetilde{O}(n+N+\mathrm{nnz}(B))$ *bit operations once the overestimates and row sparsities are available.*

Proof. Let $M = H + \alpha I$. The sampled matrix satisfies $\mathbb{E}[B^\top B] = H$. For each row, define

$$X_r = \left(\frac{\xi_r}{p_r} - 1\right) M^{-1/2} a_r a_r^\top M^{-1/2},$$

with $X_r = 0$ when $p_r = 1$. Since

$$\left\| M^{-1/2} a_r a_r^\top M^{-1/2} \right\|_2 = \tau_r(\alpha) \le u_r,$$

the choice of p_r gives $\|X_r\|_2 \le O(1/\log(d/\delta))$, and the variance proxy is bounded by the same logarithmic factor times

$$\sum_r M^{-1/2} a_r a_r^\top M^{-1/2} = M^{-1/2} H M^{-1/2} \preceq I.$$

Matrix Bernstein, with C_s large enough, gives

$$\left\| M^{-1/2}(B^\top B - H) M^{-1/2} \right\|_2 \le \frac{1}{2}$$

with probability at least $1 - \delta/2$, which implies $B^\top B + \alpha I \approx_c M$.

The expected row count is at most $O(U \log(d/\delta))$, and scalar Chernoff gives $\#\,\mathrm{rows}(B) = \widetilde{O}(U+1)$. Similarly,

$$\mathbb{E}[\mathrm{nnz}(B)] = \sum_r w_r p_r \le O(W \log(d/\delta)),$$

and scalar Bernstein with increments bounded by s gives $\mathrm{nnz}(B) = \widetilde{O}(W+s)$. The running time is the scan, sampling, and sampled-row writing time. □

5 Constructing the Ridge Chain

This section builds the sampled positive-ridge preconditioners $P_i = B_i^\top B_i + \nu_i I$ top down. Each step estimates scores at the current ridge using solvers for larger ridges, samples a row matrix, and, at endpoint levels, preprocesses the active-coordinate endpoint routine used by the recursion in Section 6.

5.1 Top-Down Construction

At level i, every score-estimation solve uses either the direct top-level solver or already constructed matrices with indices larger than i. The shifted recursive solver needed for low levels is proved later, but its contract is stated here as a dependency of the top-down routine.

Lemma 5.1 (Top-down chain construction)**.** *Under the normalized hypotheses* $\lambda_d(H) = 1$ *and* $\lambda_{k+1}(H) \le C_0$*, there is a top-down randomized construction which, with probability at least* $1-\delta$*, outputs sampled row matrices* B_i *for all* $1 \le i \le T$ *such that*

$$P_i := B_i^\top B_i + \nu_i I \approx_c H + \nu_i I$$

TopDownRidgeChain: Sampled ridge preconditioning chain.
Input: A, k, s, N, the ridge sequence $\nu_1, \dots, \nu_T$, the score-mass constant C_0, and failure probability $\delta \in (0, 1/2)$.
Output: sampled matrices B_i for $1 \le i \le T$ and active endpoint data structures for $i \ge L$.
Guarantee: with probability at least $1 - \delta$, for every $1 \le i \le T$,

$$B_i^\top B_i + \nu_i I \approx_c H + \nu_i I,$$

and the sampled sizes satisfy

	$\#\,\mathrm{rows}(B_i)$	$\mathrm{nnz}(B_i)$
$i < L$	$\widetilde{O}(k + d/\nu_i)$	$\widetilde{O}(sk + N/\nu_i)$
$i \ge L$	$\widetilde{O}(k)$	$\widetilde{O}(ks)$

For every $i \ge L$, the active set T_i constructed in Step 4 satisfies $|T_i| = \widetilde{O}(ks)$.
Description:

1. Build a solver for the constant-conditioned system $M(\nu_T)$ using sparse products by A and $A^\top$.
2. For $i = T, T-1, \dots, 1$, set
$$\zeta_i = \frac{k + C_0 d/\nu_i}{100ns}, \qquad \delta_i = \frac{\delta}{6T}.$$
Invoke JLRidgeScoreEstimate with ridge $\alpha = \nu_i$, floor ζ_i, failure probability δ_i, and tolerance $\mu_i = (c_{\mathrm{JL}}/\sqrt{q_i}) \min\{1, \sqrt{\zeta_i}\}$, where $q_i = O(\log(nT/\delta))$. Use the direct $M(\nu_T)$ solver for these absolute solve calls if $i = T$, the one-step endpoint-level routine from Theorem 6.4 with a scalar multiple of P_{i+1} if $L \le i < T$, and the shifted first-block routine from Theorem 6.7 with root $M(\nu_i)$ and the already constructed suffix $P_{i+1}, \dots, P_L$ if $i < L$.
3. Run RidgeScoreSample with ridge $\alpha = \nu_i$, row weights w_r, the overestimates $u_{i,r}$ from the previous step, and failure probability δ_i, producing B_i and $P_i = B_i^\top B_i + \nu_i I$.
4. If $i \ge L$, form
$$T_i = \{q \in [d] : \text{ column } q \text{ of } B_i \text{ contains a nonzero}\}$$
and preprocess the active endpoint solver from Theorem 6.1 for $B_{i,T_i}^\top B_{i,T_i} + \nu_i I$ with preprocessing failure probability δ_i. Store the resulting endpoint routine as target-flexible; each later invocation supplies its actual solver target and per-call failure budget.

for an absolute constant c and

$$\#\,\mathrm{rows}(B_i) = \widetilde{O}\left(k + \frac{d}{\nu_i}\right), \qquad \mathrm{nnz}(B_i) = \widetilde{O}\left(sk + \frac{N}{\nu_i}\right).$$

For every $i \ge L$,

$$\#\,\mathrm{rows}(B_i) = \widetilde{O}(k), \qquad \mathrm{nnz}(B_i) = \widetilde{O}(ks), \qquad |T_i| = \widetilde{O}(ks).$$

Moreover, endpoint levels $i \ge L$ admit the active-coordinate preprocessing and application guarantees of Theorem 6.1; over all such levels the endpoint preprocessing cost is $\widetilde{O}(k^{\omega+\eta})$ plus near-linear terms in the sampled sizes. The total construction time, including the JL score-estimation solve

calls, is

$$n^{o(1)}\widetilde{O}\big(N+n+\sqrt{\rho}\,(ks+k^2)+k^{\omega+\eta}\big)\,.$$

Proof. Consecutive ridge matrices satisfy

$$M(\nu_i)\preceq M(\nu_{i+1})\preceq \gamma M(\nu_i)\qquad (1\le i<T),$$

and $M(\nu_T)$ has constant condition number. Thus the direct solver handles level T. For $L\le i<T$, the already constructed matrix P_{i+1} satisfies $P_{i+1}\approx_c M(\nu_{i+1})$, so a constant scalar multiple of P_{i+1} lies between $M(\nu_i)$ and $O_c(\gamma)M(\nu_i)$; Theorem 6.4 gives an $M(\nu_i)$-solver with one adjacent preconditioning step and endpoint-level applications of P_{i+1}. For $i<L$, all suffix matrices $P_{i+1},\dots,P_L$ and the level-L endpoint data structure have already been constructed by the top-down loop, so the hypotheses of Theorem 6.7 hold with root matrix $M(\nu_i)$.

The calls made inside JLRidgeScoreEstimate request absolute $M(\nu_i)$-norm error μ_i. For a sketch vector $s=S_{j,:}^\top$, write

$$C_i=\begin{pmatrix}A\\ \sqrt{\nu_i}I_d\end{pmatrix}.$$

Then $g=C_i^\top s$ and

$$\big\|M(\nu_i)^{-1}g\big\|_{M(\nu_i)}=\Big\|M(\nu_i)^{-1/2}C_i^\top s\Big\|_2\le\|s\|_2\,.$$

Since the entries of S are $\pm q_i^{-1/2}$, every sketch vector has $\|s\|_2=\sqrt{(n+d)/q_i}$. Thus all these norms are at most $R_{\mathrm{JL}}=\sqrt{(n+d)/q_i}$. The maintained relative solvers are therefore invoked with target μ_i/R_{JL} and failure budgets included in δ_i, giving the absolute solve calls required by Theorem 4.5.

At level i, choose the estimator floor $\zeta_i=(k+C_0d/\nu_i)/(100ns)$. By Theorem 4.5, with the assigned failure budget the produced estimates satisfy

$$\tau_r(\nu_i)\le u_{i,r}\le C_{\mathrm{JL}}(\tau_r(\nu_i)+\zeta_i)\qquad\text{for all } r.$$

Using Theorem 4.3,

$$\sum_r u_{i,r}\le C_{\mathrm{JL}}\sum_r\tau_r(\nu_i)+C_{\mathrm{JL}}n\zeta_i=O\!\left(k+\frac{d}{\nu_i}\right),$$

and

$$\sum_r w_r u_{i,r}\le C_{\mathrm{JL}}\sum_r w_r\tau_r(\nu_i)+C_{\mathrm{JL}}N\zeta_i=O\!\left(sk+\frac{N}{\nu_i}\right),$$

where $N\le ns$ and $N\ge d$ absorb the floor contribution. Applying Theorem 4.6 at ridge ν_i gives the stated spectral approximation and sampled row and nonzero bounds. The additive $+1$ and $+s$ terms in the sampling lemma are absorbed because $k\ge 1$ and $s\le sk$.

If $i\ge L$, then $\nu_i\ge\nu_L=\rho\ge d/k$, so $k+d/\nu_i\le 2k$. Therefore $\#\,\mathrm{rows}(B_i)=\widetilde{O}(k)$. Since every sampled row is a rescaled input row with at most s nonzeros, $\mathrm{nnz}(B_i)=\widetilde{O}(ks)$, and the active coordinate set satisfies $|T_i|\le\mathrm{nnz}(B_i)=\widetilde{O}(ks)$.

For each $i\ge L$, Theorem 6.1 applied to B_{i,T_i} gives preprocessing $\widetilde{O}(ks+k^{\omega+\eta})$ and endpoint application time $\widetilde{O}(ks+k^2)$ for admissible lazy active-vector right-hand sides. The number of endpoint levels is $O(\log^{3/2}n)$, so the $k^{\omega+\eta}$ terms remain within $\widetilde{O}(k^{\omega+\eta})$ and the near-linear sampled-size terms are polylogarithmic sums of the displayed sampled sizes.

It remains to account for the score-estimation solve calls. Each level uses only $O(\log(nT/\delta))$ absolute solve calls and $\widetilde{O}(N+n)$ direct sparse work. For levels $i < L$, Theorem 6.7, invoked with root $M(\nu_i)$, tolerance μ_i, and bound R_{JL}, gives solve-call cost

$$n^{o(1)}\widetilde{O}\left(N + \sqrt{\frac{\rho}{\nu_i}}\,(ks+k^2)\right),$$

because the endpoint is at $\nu_L = \rho$. Summing over $i < L$ and using $\nu_i = \gamma^i$ gives

$$\sum_{i=1}^{L-1}\sqrt{\frac{\rho}{\nu_i}} \le L\sqrt{\rho},$$

which is absorbed by $n^{o(1)}\sqrt{\rho}$. For levels $i \ge L$, Theorem 6.8 gives the one-step endpoint-level bound $n^{o(1)}\widetilde{O}(N+ks+k^2)$ per JL batch; the $O(T)$ batches are again only polylogarithmically many. Adding the direct row-scan work from the JL and sampling primitives gives total construction time

$$n^{o(1)}\widetilde{O}\big(N+n+\sqrt{\rho}\,(ks+k^2)+k^{\omega+\eta}\big).$$

A union bound over JL estimation, sampling, endpoint preprocessing, and any randomized endpoint calls gives total failure probability at most δ. The construction never samples at $\nu_0 = 0$. □

6 Active Endpoints and Recursive Solving

This section applies the endpoint and adjacent-solve primitives from Section 2.2 through a query-lazy vector interface, then gives the recursive first-block solver, the shifted solves used by the chain construction, and the final proof of Theorem 1.1. The active endpoint keeps dense work confined to the sampled row dimension, and the query-lazy interface prevents recursive boundary calls from materializing ambient d-vectors.

6.1 Active Endpoint Primitive

The endpoint primitive applies Theorem 2.4 only after restricting to the active coordinate set of the sampled matrix. This subsection gives the resulting active-coordinate wrapper and application routine.

Lemma 6.1 (Active-coordinate endpoint)**.** *Let $B \in \mathbb{R}^{m\times d}$ have at most s nonzero entries per row, let $\nu > 0$, and let*

$$S = \{q \in [d] : \text{ column } q \text{ of } B \text{ contains a nonzero}\}.$$

Suppose $m = \widetilde{O}(k)$. With probability at least $1-\delta$, one can preprocess the restricted matrix $B_S \in \mathbb{R}^{m\times|S|}$ in

$$\widetilde{O}(\mathrm{nnz}(B_S)+|S|) + \widetilde{O}((m+\log(1/\delta))^{\omega+\eta}) = \widetilde{O}(ks+k^{\omega+\eta})$$

bit operations and obtain the following target-flexible endpoint apply routine. Given any requested target accuracy $\vartheta \in (0,1/2)$ and any representation of a vector y from which y_S can be formed in time R_S, the routine returns a lazy representation of a vector $\widehat{z}$ satisfying the (ϑ,δ)-solver guarantee of Theorem 2.2 for

$$P = B^\top B + \nu I_d,$$

ActiveEndpointApply: Active endpoint inverse application.
Input: the active set S, $\nu > 0$, the preprocessed active-coordinate solver $\mathcal{S}_S$ for $Q = B_S^\top B_S + \nu I_{|S|}$, a representation of $y \in \mathbb{R}^d$ that supports restriction to S, and target relative accuracy ϑ.
Output: a lazy representation of an approximate solution to $Pz = y$, where $P = B^\top B + \nu I_d$.
Guarantee: if $\mathcal{S}_S$ is a ϑ-solver for Q, the returned vector is a ϑ-solver output for P in the P-norm.
Description:

1. Query the representation of y on S and form y_S in the compact coordinate order.
2. Compute $\widehat{z}_S = \mathcal{S}_S(y_S)$.
3. Return the lazy vector
$$\widehat{z} = \nu^{-1} y + \Delta_S, \qquad \Delta_S = \widehat{z}_S - \nu^{-1} y_S,$$
where Δ_S is stored explicitly on S and $\nu^{-1} y$ is kept as a lazy diagonal scaling.

and its application time is

$$R_S + \widetilde{O}(\mathrm{nnz}(B_S) + m^2) = R_S + \widetilde{O}(ks + k^2),$$

up to factors polylogarithmic in the condition parameter and target solver accuracy.

Proof. Because S contains every nonzero column of B, after permuting coordinates one has

$$P = \begin{pmatrix} B_S^\top B_S + \nu I_{|S|} & 0 \\ 0 & \nu I_{S^c} \end{pmatrix}.$$

Also $|S| \le \mathrm{nnz}(B_S) \le ms = \widetilde{O}(ks)$.

Apply Theorem 2.4 with $C = B_S$. In the construction underlying that lemma, the preprocessed object does not depend on the later right-hand side, and the requested accuracy ϑ affects only the logarithmic number of solver iterations in the application time. Thus the same data structure can be used as a target-flexible routine.

In the present invocation, sampled-row entries have $p = \mathrm{polylog}(n/\epsilon)$-bit representations, and the row-rescaling factors are polynomially bounded because the score-estimator sampling floor is at least inverse-polynomial. Moreover,

$$\kappa(B_S^\top B_S + \nu I_{|S|}) \le 1 + \frac{\|B_S\|_2^2}{\nu} \le (n/\epsilon)^{O(1)}$$

on the spectral-approximation event used by the chain construction. Thus the hypotheses of Theorem 2.4 hold, and its bit-complexity bounds give the claimed preprocessing and application costs. The dense inverse or factorization inside the endpoint construction is counted as $\widetilde{O}(m^{\omega+\eta})$, and $m = \widetilde{O}(k)$.

It remains to verify the full-space application routine. The exact solution has $z_S^\star = Q^{-1} y_S$ and $z_{S^c}^\star = \nu^{-1} y_{S^c}$. Therefore ActiveEndpointApply is exact outside S and has the required relative error on the active block:

$$\|\widehat{z} - z^\star\|_P = \left\|\widehat{z}_S - Q^{-1} y_S\right\|_Q \le \vartheta \left\|Q^{-1} y_S\right\|_Q \le \vartheta \left\|P^{-1} y\right\|_P.$$

The running time follows from one restriction query, one active-block solve, and storage of a correction supported on S. In endpoint levels of the chain, $S = T_i$; outside T_i, the inverse action is only the lazy scaling by ν_i^{-1}. The recursive interface never materializes an arbitrary dense d-vector at endpoint levels, so no ambient $\Omega(d)$ endpoint application term is introduced. □

6.2 Query-Lazy Recursion

The recursion solves $P_0 = H$ by moving through adjacent preconditioners until it reaches the endpoint P_L. Query-lazy vector representations let internal calls restrict vectors to active coordinate sets without scanning all d coordinates.

Definition 6.2 (Query-lazy vector)**.** *A query-lazy vector representation is a directed acyclic expression graph whose leaves are dense root vectors or sparse coordinate dictionaries and whose internal nodes represent scalar multiplication or vector addition. For a finite coordinate set S, the operation* $\mathrm{Restrict}(v, S)$ *returns v_S by querying dense leaves only on S and by dictionary lookup for sparse corrections. The representation memoizes restrictions requested during one solver call.*

For the level-L endpoint, a query-lazy vector is endpoint-admissible if it is presented as $y = g + z$, where g is an inherited query-lazy base object whose restriction to T_L can be formed within the recursive application bound, and z is an explicit sparse correction supported on $T_{L-1} \cup T_L$. The endpoint support condition refers to this explicit correction, not to the full support of the inherited base term.

Lemma 6.3 (Adjacent conditioning)**.** *Assume $P_i \approx_c H + \nu_i I$ for $1 \le i \le L$ and set $P_0 = H$. There is a constant C_c, depending only on c, such that for every $0 \le i < L$,*

$$(C_c\gamma)^{-1} P_{i+1} \preceq P_i \preceq C_c P_{i+1}.$$

Consequently the generalized condition number of the pair (P_i, P_{i+1}) is $O_c(\gamma)$.

Proof. For $i \ge 1$, use

$$M(\nu_i) \preceq M(\nu_{i+1}) \preceq \gamma M(\nu_i)$$

and the two spectral approximations. For $i = 0$, $\lambda_d(H) = 1$ implies $H \succeq I$, hence

$$H \preceq H + \gamma I \preceq (1+\gamma)H \preceq 2\gamma H.$$

Combining this with $P_1 \approx_c H + \gamma I$ gives the displayed comparison after enlarging C_c. □

For a fixed positive definite preconditioner, the linear approximate inverse and its Loewner guarantee are supplied by Theorem 2.6. The recursive calls below are instead inexact and randomized, so we use the robust preconditioned iteration from Theorem 2.3.

Lemma 6.4 (Inexact adjacent solve)**.** *Let G, N be positive definite matrices satisfying $G \preceq N \preceq \kappa G$. Suppose products with G are available and $\mathcal{F}_N$ is a (ξ, δ_f)-solver for N with $\xi \le (20\kappa)^{-1}$. Given a right-hand side y, target $\vartheta \in (0, 1/2)$, and failure budget δ, the fixed-iteration preconditioned accelerated-gradient method returns a ϑ-approximate solution to $Gx = y$ in the G-norm with failure probability at most*

$$\delta + O\big(\sqrt{\kappa}\log(2/\vartheta)\,\delta_f\big).$$

It uses $O(\sqrt{\kappa}\log(2/\vartheta))$ products with G, the same number of calls to $\mathcal{F}_N$, and lower-order vector operations.

Proof. Apply Theorem 2.3 with $M = G$. The requirement $\xi \le (20\kappa)^{-1}$ is stronger than the $(10\kappa)^{-1}$ threshold in that lemma, so every call to $\mathcal{F}_N$ is accurate enough for the robust preconditioned iteration. The failure probability is the union bound over the preconditioner calls and the solver's own failure event.

In every invocation in this proof, $\kappa \le O_c(\gamma)$ or $\kappa \le O_c(1)$ and hence $\kappa \le (n/\epsilon)^{O(1)}$, the product oracle is either $v \mapsto A^\top(Av) + \alpha v$ or $v \mapsto B_i^\top(B_i v) + \nu_i v$ with polynomially bounded rounded entries, and the recursive preconditioner target is at least inverse polynomial in n/ϵ up to hidden polylogarithmic factors. Thus the finite-precision hypotheses of Theorem 2.3 hold, and that lemma directly gives the corresponding bit-operation bound. □

The endpoint inverse is block diagonal outside its active set. This is the formal reason lazy endpoint calls avoid dense output.

Lemma 6.5 (Endpoint block action)**.** *Let y be a query-lazy vector for which y_{T_L} can be formed without scanning all d coordinates. The exact inverse satisfies*

$$P_L^{-1} y = \nu_L^{-1} y + \Delta_{T_L},$$

where

$$\Delta_{T_L} = \left(B_{L,T_L}^\top B_{L,T_L} + \nu_L I_{T_L}\right)^{-1} y_{T_L} - \nu_L^{-1} y_{T_L}$$

with Δ_{T_L} supported on T_L. Moreover, for every target $\vartheta \in (0, 1/2)$, Theorem 6.1 returns a lazy vector $\widehat{z} = \nu_L^{-1} y + \widehat{\Delta}_{T_L}$ satisfying the (ϑ, δ)-solver guarantee for P_L. If y is endpoint-admissible in the sense of Theorem 6.2, then both the exact solution and the returned approximation admit endpoint-admissible representations, and the approximate application cost is the cost of forming y_{T_L} plus $\widetilde{O}(ks + k^2)$.

Proof. After permuting coordinates so that T_L comes first,

$$P_L = \begin{pmatrix} B_{L,T_L}^\top B_{L,T_L} + \nu_L I_{T_L} & 0 \\ 0 & \nu_L I_{T_L^c} \end{pmatrix}.$$

Solving the first block and multiplying by ν_L^{-1} on the second block gives the displayed expression. Multiplying the inherited base by ν_L^{-1} preserves it as a lazy base term, and the new explicit correction remains supported on $T_{L-1} \cup T_L$.

For the approximate statement, apply Theorem 6.1 with $B = B_L$ and $S = T_L$. The active solver is run on

$$B_{L,T_L}^\top B_{L,T_L} + \nu_L I_{T_L}$$

with target ϑ, and the output is embedded into the full vector as $\nu_L^{-1} y + \widehat{\Delta}_{T_L}$. The proof of Theorem 6.1 shows that the full P_L-norm error is exactly the active-block error, because the inactive block is multiplied by ν_L^{-1} exactly. The support and running-time claims follow from the same block form and the endpoint-admissible representation. □

Lemma 6.6 (Recursive first-block solver)**.** *Assume $\lambda_d(H) = 1$. Let $P_i = B_i^\top B_i + \nu_i I$ for $1 \le i \le L$ satisfy $P_i \approx_c H + \nu_i I$, set $P_0 = H$, and assume*

$$\mathrm{nnz}(B_i) = \widetilde{O}\left(sk + \frac{N}{\nu_i}\right) \quad (1 \le i < L), \qquad \mathrm{nnz}(B_L) = \widetilde{O}(ks).$$

ApplyFirstBlockOperator: Recursive first-block inverse application.
Input: a level $i \in \{0, \dots, L\}$, a right-hand side y given as an explicit dense vector if $i = 0$ and as a query-lazy vector if $i \geq 1$, a target $\vartheta_i \in (0, 1/2)$, and a failure budget δ_i. The routine also receives the matrices $P_i, \dots, P_L$, the active sets $T_1, \dots, T_L$, the root sparse access oracle, and the target-flexible endpoint routine for P_L from Theorem 6.1.
Output: a dense vector if $i = 0$ and a query-lazy vector if $i \geq 1$.
Guarantee: with failure probability at most δ_i, the output is a ϑ_i-approximate solution in the P_i-norm, with $P_0 = H$.
Description: For every $i < L$, let N_i be the constant rescaling of P_{i+1} given by Theorem 6.3. Set $\kappa_i = C_c\gamma$, $\xi_i = (20\kappa_i)^{-1}$, and $q_i = \lceil C_{\mathrm{pc}}\sqrt{\kappa_i}\log(2/\vartheta_i)\rceil$.

1. If $i = L$, apply ActiveEndpointApply to y with target ϑ_i and failure budget δ_i, using the block representation in Theorem 6.5, and return the resulting query-lazy vector.

2. If $1 \leq i < L$, apply the solver of Theorem 6.4 to $P_i x = y$ with target ϑ_i, failure budget $\delta_i/2$, and preconditioner N_i. Implement products with P_i as

$$P_i v = B_i^\top (B_i v) + \nu_i v,$$

   storing the regularization term lazily and the sampled product as a correction supported on T_i. Implement preconditioner calls by recursively invoking ApplyFirstBlockOperator at level $i + 1$ with target ξ_i and failure budget $\delta_i/(2q_i + 2)$, followed by the scalar rescaling for N_i^{-1}. Return the final iterate as a query-lazy vector.

3. If $i = 0$, apply the solver of Theorem 6.4 to $Hx = y$ with target ϑ_i, failure budget $\delta_i/2$, the root sparse product oracle for $P_0 = H$, and preconditioner N_0. Implement preconditioner calls by recursively invoking ApplyFirstBlockOperator at level 1 with target ξ_0 and failure budget $\delta_0/(2q_0 + 2)$, followed by the scalar rescaling for N_0^{-1}. Materialize the root iterate as a dense vector after every vector update, charging $O(d)$ to the root $\widetilde{O}(N)$ term, and return the final dense vector.

Assume the level-L endpoint data structure supports, for every $\vartheta_L, \delta_L \in (0, 1/2)$, a (ϑ_L, δ_L)-solver for P_L on endpoint-admissible lazy vectors in time

$$T_{\mathrm{end}} = \widetilde{O}(ks + k^2).$$

Here the $\widetilde{O}(\cdot)$ notation includes the logarithmic dependence on ϑ_L^{-1} and δ_L^{-1}. Under the root sparse access model, after preprocessing the matrices P_i and the endpoint data structure, there is a randomized algorithm which, for every b with $\|b\|_2 \leq \mathrm{poly}(n)$ and every requested failure probability $\delta \in (0, 1/2)$, returns $\widetilde{x}$ satisfying

$$\left\|\widetilde{x} - H^{-1}b\right\|_2 \leq \epsilon$$

with probability at least $1 - \delta$ in

$$n^{o(1)}\widetilde{O}(N + \sqrt{\rho}\, T_{\mathrm{end}})$$

additional bit operations.

Proof. Increase the constant C_c in Theorem 6.3, if necessary, so that each adjacent pair can be passed to Theorem 6.4 with generalized condition bound

$$\kappa_* := C_c\gamma.$$

Set the internal preconditioner target

$$\xi_* = (20\kappa_*)^{-1}.$$

For each level i, define a recursive solver $\mathcal{S}_i(\vartheta_i, \delta_i)$ for P_i as in ApplyFirstBlockOperator: at $i = L$ it calls the endpoint (ϑ_i, δ_i)-solver. At $i < L$, Theorem 6.3 gives

$$P_i \preceq C_c P_{i+1} \qquad \text{and} \qquad C_c P_{i+1} \preceq C_c^2 \gamma P_i.$$

After increasing C_c, set the preconditioner matrix for Theorem 6.4 to

$$N_i = C_c P_{i+1},$$

so that $P_i \preceq N_i \preceq \kappa_* P_i$. Preconditioner calls to N_i are implemented by calling $\mathcal{S}_{i+1}(\xi_*, \delta_i/(2q_i+2))$ on the same right-hand side and multiplying the returned vector by C_c^{-1}; this scalar rescaling preserves the relative energy-norm solver guarantee. The target is ϑ_i, and

$$q_i = \lceil C_{\mathrm{pc}} \sqrt{\kappa_*} \log(2/\vartheta_i) \rceil.$$

We prove by downward induction that $\mathcal{S}_i(\vartheta_i, \delta_i)$ is a (ϑ_i, δ_i)-solver for P_i. The base case is exactly the endpoint hypothesis and Theorem 6.5. For $i < L$, the induction hypothesis supplies the required $(\xi_*, \delta_i/(2q_i + 2))$-solver for the preconditioner matrix. By Theorem 6.4, the adjacent solve has error at most ϑ_i in the P_i-norm. Its failure probability is at most

$$\frac{\delta_i}{2} + q_i \frac{\delta_i}{2q_i + 2} \le \delta_i,$$

where the first term is the failure budget assigned to the robust preconditioned iteration itself and the second term is the union bound over recursive preconditioner calls.

The lazy endpoint invariant follows by induction on the call tree. A product with P_i for $1 \le i < L$ scans B_i, queries the input vector only on T_i, stores $B_i^\top (B_i v)$ as an explicit correction on T_i, and stores $\nu_i v$ as a scalar lazy node. Thus inside a level $L-1$ call, every endpoint right-hand side has explicit correction support contained in $T_{L-1} \cup T_L$. By Theorem 6.5, the endpoint forms only y_{T_L} and returns $\nu_L^{-1} y + \Delta_{T_L}$ lazily. Dense materialization occurs only at level 0.

Let

$$q_* = C\sqrt{\gamma} \log(40 C_c \gamma)$$

for a sufficiently large constant C. This bounds the number of lower-level calls in every non-root internal solve, since those calls use target ξ_*. Let R_i be the worst-case time to apply $\mathcal{S}_i(\xi_*, \cdot)$ to a query-lazy right-hand side for $i \ge 1$, including restriction queries and endpoint calls. The endpoint gives

$$R_L = \widetilde{O}(T_{\mathrm{end}}).$$

For $1 \le i < L$, the sampled product representation and memoized restriction queries give

$$R_i \le q_* \left(\widetilde{O}(a_i) + R_{i+1} \right), \qquad a_i = \mathrm{nnz}(B_i) = \widetilde{O}\left(sk + \frac{N}{\nu_i} \right).$$

For the root solve with target ϑ,

$$R_0(\vartheta) \le C\sqrt{\gamma} \log(2/\vartheta) \left(\widetilde{O}(N) + R_1 \right),$$

because root products and dense vector updates cost $\widetilde{O}(N)$.

Unrolling the recurrence and using $\nu_i = \gamma^i$ gives

$$R_0(\vartheta) \le \widetilde{O}(\log(2/\vartheta)) \left[q_* N + \sum_{i=1}^{L-1} q_*^{i+1} \left(sk + \frac{N}{\gamma^i} \right) + q_*^L T_{\text{end}} \right].$$

Since $L = \lceil \sqrt{\log n} \rceil$ and $\rho \le d \le N \le \text{poly}(n)$,

$$q_*^L = (C\sqrt{\gamma}\log(40C_c\gamma))^L = n^{o(1)}\sqrt{\rho}.$$

The N terms are at most $n^{o(1)}N$, because the extra powers of $C\log(40C_c\gamma)$ over L levels are $n^{o(1)}$ and $\gamma^{(i+1)/2-i} \le \sqrt{\gamma} = n^{o(1)}$. The sk terms are at most

$$n^{o(1)}\sqrt{\rho}\, ks \le n^{o(1)}\sqrt{\rho}\, T_{\text{end}},$$

as T_{end} includes the $\widetilde{O}(ks)$ active-coordinate extraction and correction cost. Therefore

$$R_0(\vartheta) = n^{o(1)}\widetilde{O}(N + \sqrt{\rho}\, T_{\text{end}}),$$

with the $\log(2/\vartheta)$ factor hidden in $\widetilde{O}(\cdot)$.

If $b = 0$, return 0. Otherwise let $B_0 = \max\{1, \|b\|_2\}$ and set

$$\vartheta = \min\left\{ \frac{1}{4}, \frac{\epsilon}{B_0} \right\}.$$

The promise $\|b\|_2 \le \text{poly}(n)$ gives $\log(2/\vartheta) = O(\log(n/\epsilon))$. Since $\lambda_d(H) = 1$, $H \succeq I$ and $H^{-1} \preceq I$, so

$$\left\|\widehat{x} - H^{-1}b\right\|_2 \le \left\|\widehat{x} - H^{-1}b\right\|_H \le \vartheta \left\|H^{-1}b\right\|_H \le \vartheta \|b\|_2 \le \epsilon.$$

Run the root call as $\mathcal{S}_0(\vartheta, \delta)$. The failure-probability induction above gives success probability at least $1 - \delta$. Every matrix in this recursion has condition number at most $\text{poly}(n)$: $H \succeq I$, $H \preceq \text{poly}(n)I$, and each P_i is a constant-factor approximation to $H + \nu_i I$ with polynomially bounded ν_i. The requested internal targets have logarithms bounded by $\text{polylog}(n/\epsilon)$. Hence Theorems 2.3 and 2.4 apply to the adjacent and endpoint calls. The constant-conditioned direct solves and the sparse root products are ordinary fixed-point operations at the same $\text{polylog}(n/\epsilon)$ working precision. Therefore the displayed operation bound is a bit-operation bound up to the $\widetilde{O}(\cdot)$ factors already recorded. □

The chain construction needs solvers with root matrix $H + \nu_j I$ before P_j has been sampled. The next two lemmas separate the low-ridge recursive case from the endpoint-level one-step case.

Lemma 6.7 (Low-ridge shifted recursive solver)**.** *Assume $\lambda_d(H) = 1$, fix $1 \le j < L$, and set $G_j = M(\nu_j) = H + \nu_j I$. Suppose the already constructed suffix $P_{j+1}, \ldots, P_L$ and the level-L endpoint data structure satisfy the hypotheses of Theorem 6.6 restricted to these levels, with $T_{\text{end}} = \widetilde{O}(ks + k^2)$. Suppose products with G_j cost $\widetilde{O}(N)$.*

Given g, $\mu > 0$, $\delta \in (0, 1/2)$, and $R_g \ge \left\|G_j^{-1}g\right\|_{G_j}$, a randomized routine returns $\widehat{x}$ satisfying $\left\|\widehat{x} - G_j^{-1}g\right\|_{G_j} \le \mu$ with failure probability at most δ in

$$n^{o(1)}\widetilde{O}\left(N + \sqrt{\frac{\rho}{\nu_j}}\, T_{\text{end}} \right)$$

bit operations, up to a logarithmic factor in R_g/μ.

Proof. Use only the suffix $P_{j+1}, \dots, P_L$ and replace the root matrix $P_0 = H$ in the proof of Theorem 6.6 by $G_j = H + \nu_j I$. This uses no matrix P_j. Since $M(\nu_j) \preceq M(\nu_{j+1}) \preceq \gamma M(\nu_j)$ and $P_{j+1} \approx_c M(\nu_{j+1})$, there is a constant $a_c > 0$, depending only on c, such that

$$G_j \preceq a_c P_{j+1} \preceq O_c(\gamma) G_j.$$

Thus the first preconditioner call uses the suffix solver for P_{j+1}, followed by scaling its output by a_c^{-1}. Subsequent comparisons are those of Theorem 6.3. Products with G_j cost $\widetilde{O}(N)$ because $G_j v = Hv + \nu_j v$.

Run the recursion with relative target $\vartheta_j = \min\{1/4, \mu/R_g\}$. Its relative G_j-norm guarantee gives the desired absolute error. The recurrence has $L - j$ levels and endpoint ratio $\nu_L/\nu_j = \rho/\nu_j$, which gives the stated time bound. □

Lemma 6.8 (Endpoint-level shifted solver)**.** *Assume $\lambda_d(H) = 1$, fix $L \le j < T$, and set $G_j = M(\nu_j) = H + \nu_j I$. Suppose $P_{j+1} \approx_c M(\nu_{j+1})$ and its active endpoint data structure supports the guarantee of Theorem 6.1 with application time $T_{\mathrm{end}} = \widetilde{O}(ks + k^2)$. Suppose products with G_j cost $\widetilde{O}(N)$.*

Given g, $\mu > 0$, $\delta \in (0, 1/2)$, and $R_g \ge \left\| G_j^{-1} g \right\|_{G_j}$, a randomized routine returns $\widehat{x}$ satisfying $\left\| \widehat{x} - G_j^{-1} g \right\|_{G_j} \le \mu$ with failure probability at most δ in

$$n^{o(1)} \widetilde{O}(N + \sqrt{\gamma}\, T_{\mathrm{end}})$$

bit operations, up to a logarithmic factor in R_g/μ.

Proof. The matrix P_{j+1} is already available in the top-down order. The adjacent ridge comparison and $P_{j+1} \approx_c M(\nu_{j+1})$ give a constant $a_c > 0$, depending only on c, such that

$$G_j \preceq a_c P_{j+1} \preceq O_c(\gamma) G_j.$$

Applying the P_{j+1}-solver and scaling its output by a_c^{-1} gives a solver for $a_c P_{j+1}$ with the same relative energy-norm error. Apply Theorem 6.4 once with $G = G_j$, $N = a_c P_{j+1}$, target $\vartheta_j = \min\{1/4, \mu/R_g\}$, and failure budget δ. The adjacent condition number is $O_c(\gamma)$, which gives the stated time bound and absolute-error guarantee. □

6.3 Final Proof

This subsection combines normalization, sampling, active endpoints, and the recursive solver to prove Theorem 1.1.

Proof of Theorem 1.1. By Theorem 4.2, it is enough to analyze the normalized case $\lambda_d(H) = 1$ and then run a logarithmic scale search. In that normalization, the tail hypothesis gives $\lambda_{k+1}(H) \le C_0$ for an absolute constant C_0, so Theorem 4.3 supplies the ridge-score mass estimates required by Theorem 5.1. Applying Theorem 5.1 with failure probability at most $(n/\epsilon)^{-c}$ constructs the sampled chain $P_i = B_i^\top B_i + \nu_i I$ and, at level L, gives $\mathrm{nnz}(B_L) = \widetilde{O}(ks)$ and an active set of size $\widetilde{O}(ks)$. By Theorem 6.1, the level-L endpoint solver has preprocessing time $\widetilde{O}(ks + k^{\omega+\eta})$ and endpoint application time

$$T_{\mathrm{end}} = \widetilde{O}(ks + k^2)$$

for endpoint-admissible lazy vectors.

Theorem 6.6 then solves the normalized system to Euclidean error ϵ in additional time

$$n^{o(1)}\widetilde{O}\big(N + \sqrt{\rho}\,(ks + k^2)\big)\,.$$

By Theorem 5.1, the chain construction and endpoint preprocessing contribute only

$$n^{o(1)}\widetilde{O}\big(N + n + \sqrt{\rho}\,(ks + k^2) + k^{\omega+\eta}\big)\,.$$

Combining this with the solve time above preserves the same asymptotic bound. Since $\rho = \max\{d/k, 2\}$ and $1 \le k < d$, one has $\sqrt{\rho} = O(\sqrt{d/k})$, so this is

$$n^{o(1)}\widetilde{O}\left(N + n + \sqrt{\frac{d}{k}}\,(ks + k^2) + k^{\omega+\eta}\right).$$

Using $N \le ns$, $s \ge 1$, $n \ge d$, and $s\sqrt{dk} \le sd \le sn$, the first variable terms satisfy

$$N + n + \sqrt{\frac{d}{k}}\,ks \le 2ns + s\sqrt{dk} \le 3ns,$$

while

$$\sqrt{\frac{d}{k}}\,k^2 = d^{1/2}k^{3/2}.$$

Thus the total bit complexity is

$$n^{o(1)}\widetilde{O}\Big(ns + \sqrt{d/k}\,k^2 + k^{\omega+\eta}\Big)\,.$$

Finally, Theorem 4.2 transfers the normalized guarantee back to the original scale with only logarithmic overhead and preserves the target vector $H^{-1}b = (A^\top A)^{-1}b$. The union bound over the chain construction, endpoint preprocessing, recursive solves, and scale grid gives the claimed high-probability success guarantee. □

Acknowledgements

A significant amount of the algebraic details was generated and verified using ChatGPT 5.5 and 5.6. The algorithms and proof outlines are entirely due to the authors, who assume responsibility for all content.

References

[ADW+25] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2025. 3

[AKK+20] Naman Agarwal, Sham Kakade, Rahul Kidambi, Yin Tat Lee, Praneeth Netrapalli, and Aaron Sidford. Leverage score sampling for faster accelerated regression and ERM. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pages 22–47. PMLR, 2020. 1, 3

[AM15] Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*, 2015. 1, 3

[CMM17] Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017. 3

[CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2013. 3

[DDH07] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007. 6

[DMY25] Michał Dereziński, Christopher Musco, and Jiaming Yang. Faster linear systems and matrix norm approximation via multi-level sketched preconditioning. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2025. 1, 2, 4

[DS25] Michał Dereziński and Aaron Sidford. Approaching optimality for solving dense linear systems with low-rank structure, 2025. 1, 2, 3, 4, 5, 26

[DY24] Michał Dereziński and Jiaming Yang. Solving dense linear systems faster than via preconditioning. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, 2024. 1, 2, 4

[LMP13] Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 127–136, 2013. 3

[MM17] Cameron Musco and Christopher Musco. Recursive sampling for the Nyström method. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 2017. 3

[NN12] Jelani Nelson and Huy L. Nguyen. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings, 2012. 3

[Woo14] David P. Woodruff. Sketching as a tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014. 1, 3

A Comparison Accounting

This appendix derives the running-time bounds used in Table 1.

CG/Lanczos. A multiplication by $H = A^\top A$ costs one multiplication by A and one by $A^\top$, hence $O(N) \le O(ns)$ arithmetic operations. Under the bounded-tail assumption, all but k eigenvalues of H lie in a constant factor interval above $\lambda_d(H)$. The standard outlier-cluster interpretation of CG or Lanczos therefore gives $k + \mathrm{polylog}(1/\epsilon)$ iterations, so the sparse matvec baseline is

$$\widetilde{O}(kN) \subseteq \widetilde{O}(nks).$$

Prior-framework sparse accounting. The recursive framework of [DS25] uses endpoint systems with a $\sqrt{d/k}$ recursive multiplier and a k^2-type endpoint application term. If the sampled matrix at ridge level i is charged only through the available sparse dimensions of that level, write $m_i = d/\gamma^i$ for its sampled-row bound, where $k \le m_i \le d$. Its sparsity is bounded by

$$\widetilde{O}(\min\{ns, dm_i\}).$$

After factoring out the overall recursive multiplier $\sqrt{d/k}$, the relative weight at this level is $\sqrt{k/m_i}$. Its sparse application contribution is therefore bounded by

$$\sqrt{\frac{d}{k}}\sqrt{\frac{k}{m_i}}\min\{ns, dm_i\} \le \sqrt{\frac{d}{k}}\min\{ns, \sqrt{nskd}\}.$$

Here the first inner bound uses $m_i \ge k$, while the second follows from $\min\{a, b\} \le \sqrt{ab}$. Summing over the geometrically many levels only contributes polylogarithmic factors. Together with the input read, the k^2 endpoint-application cost, and dense endpoint preprocessing, this yields the sparse-accounting row

$$\widetilde{O}\Big(ns + \sqrt{d/k}\left(k^2 + \min\{ns, \sqrt{nskd}\}\right) + k^\omega\Big).$$

This is the sparse specialization reported in Table 1.