

Exact Error-Compensated Pruning in Matrix Multiplication Time

Xiaoyu Li

University of New South Wales

xiaoyu.li2@unsw.edu.au

14 September 2026

Abstract

We characterize the arithmetic complexity of reproducing a prescribed error-compensated pruning procedure: select weights for removal, set them to zero, and propagate each removal error to later columns. With its interaction factor supplied, an $m \times d$ layer with $m = \Theta(d^\alpha)$ admits a recursive execution in $O_\varepsilon(d^{\omega(\alpha,1,1)+\varepsilon} + C)$ operations, where C is the decision cost. The execution preserves every corrected group snapshot and every subsequent decision, including unequal groups and stateful column policies. For every fixed row-wise r -of- q rule, the exponent is exactly $\omega(\alpha, 1, 1)$; one row already has quadratic complexity. In square dimensions, matching lower bounds also hold for per-column half selection, whole-group half selection under every contiguous partition, and the reference implementation’s inclusive threshold rule with its actual removal count. Under the inverse-Cholesky kernel, all these square lower bounds survive raw calibration and every prescribed nonnegative damping. The square reductions encode an arbitrary product with quadratic overhead, weights bounded by four, covariance condition number below two, and separated score cuts. Their mechanism is a source-to-target factor cancellation, with separate snapshot bounds and isolated guards enforcing the different selectors. A local coefficient-extraction argument validates the exponent comparison in the presence of comparisons and square roots. The results concern exact numerical execution in a nonuniform scalar model; raw rectangular preprocessing and the direct inverse-matrix kernel are treated separately.

1 Introduction

Error compensation gives a pruning decision consequences beyond the weight that is removed. After a column is pruned, its error is distributed to the columns that remain; those corrected columns then determine subsequent decisions. This feedback is central to SparseGPT [5]. It also creates an apparent obstacle to fast execution: the update vectors cannot be computed before the decisions that produce them.

The obstacle concerns information, however, whereas the cost comes from moving that information through a dense matrix. These are different parts of the computation. A column must wait for all earlier errors that affect it. It need not receive those errors one at a time.

Li, Liang, Shi, and Song [8] analyze one level of lazy updates and obtain the square-layer bound

$$O\left(d^\omega + d^{2+a+o(1)} + d^{1+\omega(1,1,a)-a}\right), \quad 0 \leq a \leq 1. \quad (1)$$

Their chosen parameters give an exponent below 2.53. Here $\omega = \omega(1, 1, 1)$ is the square matrix-multiplication exponent, and $\omega(1, 1, a)$ is the exponent for multiplying an $n \times n$ matrix by an $n \times \lceil n^a \rceil$ matrix in the arithmetic model; see [1, 10] for square and rectangular bounds.

As of 14 September 2026, the best known square bound is $\omega < 2.371177$, due to Dupont et al. [4], improving the earlier bound $\omega < 2.371339$ of Alman et al. [1]. The exact value of ω remains unknown. Rectangular bounds depend on the shape: for example, $\omega(1, 1, 1/2) \leq 2.042776$ [1], while $\omega(1, 1, a) = 2$ for $0 \leq a \leq 0.321334$ [10]. These statements use the standard permutation symmetry of the three dimension exponents; an exponent of 2 means an $O_\varepsilon(n^{2+\varepsilon})$ bound for every $\varepsilon > 0$.

The two update terms in (1) balance the cost of frequent local corrections against that of occasional global corrections. We remove this balance by scheduling corrections at every scale.

The resulting upper bound raises a second question. Is matrix multiplication merely a convenient implementation tool, or does it describe the intrinsic arithmetic difficulty of the pruning trajectory? We show that it describes both, down to the rectangular shape of the layer. A single well-conditioned pruning instance can encode an arbitrary product under several of the algorithm’s own score-based selection rules. For the inverse-Cholesky kernel, the square reductions also survive every prescribed nonnegative damping.

Notation. All matrices are real. Matrix indices start at zero, and $[a, b) = \{a, a + 1, \dots, b - 1\}$ for integer endpoints. For ordered row and column index lists I, J , the submatrix $A_{I,J}$ retains their order; a colon denotes all indices, so $A_{:,j}$ is column j . For the column-indexed arrays W, V, Q, E , we abbreviate $Z_I = Z_{:,I}$; likewise, $U_{IJ} = U_{I,J}$. The symbol I also denotes an identity matrix when used as a matrix, with its order determined by context. The product $x \odot y$ is entrywise.

We use $\|x\|_2$ for the Euclidean norm of a vector and $\|A\|_2$ for the induced operator norm of a matrix. The Frobenius and entrywise maximum norms are $\|A\|_F = (\sum_{i,j} A_{ij}^2)^{1/2}$ and $\|A\|_{\max} = \max_{i,j} |A_{ij}|$. For symmetric H , $\text{spec}(H)$ is its eigenvalue set, and $\lambda_{\min}(H), \lambda_{\max}(H)$ are its extreme eigenvalues. We write $H \succ 0$ for positive definiteness and $A \preceq B$ when $B - A$ is positive semidefinite. For $H \succ 0$, its spectral condition number is $\kappa_2(H) = \lambda_{\max}(H)/\lambda_{\min}(H)$.

1.1 The problem and the main result

The computational task is to reproduce a prescribed pruning procedure. The input is a dense weight matrix, a matrix specifying how a removed weight affects later columns, and a fixed rule for choosing which positions to remove. The required output is the numerical weight matrix after all these decisions and corrections. The selected positions depend on the evolving weights, so determining them is part of the computation.

Here *structured* means row-wise r -of- q selection: in every consecutive group of q columns, exactly r positions in each row are designated for removal. The parameters satisfy $1 \leq r < q$. A designated position is set to zero; a retained position may also have value zero. Thus the rule specifies the mask cardinality, not the number of nonzero output entries.

Definition 1.1 (Exact structured pruning). Fix integers $1 \leq r < q$. The inputs are $W \in \mathbb{R}^{m \times d}$ and an upper-triangular interaction factor $U \in \mathbb{R}^{d \times d}$ with positive diagonal, where q divides d . The weight matrix represents a linear layer with d input coordinates and m output coordinates. Rows are pruned separately, using the same U . Initialize a mutable matrix $V = W$ and process the groups $G_\ell = [\ell q, (\ell + 1)q)$ in increasing order, for $0 \leq \ell < d/q$.

- (1) *Choose the group’s mask.* Record $S_G = V_{:,G}$, the group’s current values before any of its columns is processed. We call this the *group-entry snapshot*. In each row i , rank the scores $(S_G)_{ij}^2 / U_{jj}^2$ for $j \in G$, keeping the original column labels. Set $M_{ij} = 0$ at the r smallest scores and $M_{ij} = 1$ at the others, with ties resolved by a fixed deterministic convention. This mask stays fixed while the group is processed.

(2) *Prune and compensate.* For each $j \in G$ in increasing order, form

$$v_j = V_{:,j}, \quad q_j = M_{:,j} \odot v_j, \quad e_j = \frac{v_j - q_j}{U_{jj}}.$$

Store q_j as output column j . Then update every later column, including those outside G , by

$$V_{:,k} \leftarrow V_{:,k} - e_j U_{jk} \quad (j < k < d).$$

Thus $v_j - q_j$ is the removed part of the current column, and U_{jk}/U_{jj} specifies its compensation in column k .

The required output is $Q = [q_0 \cdots q_{d-1}] \in \mathbb{R}^{m \times d}$; its binary mask M may also be returned. An exact algorithm must produce this same Q in real arithmetic for every valid input. The steps specify the output, and need not be evaluated one update at a time.

For example, take one row, $r = 1$, $q = 2$, and

$$W = (1, 4 \mid 2, 3), \quad U = \begin{pmatrix} 1 & 0 & 6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The first group removes 1 and retains 4. Its compensation changes the third entry from 2 to $2 - 6 = -4$. The second group therefore sees $(-4, 3)$, removes 3, and retains -4 . The exact output is $Q = (0, 4 \mid -4, 0)$ and the mask is $M = (0, 1 \mid 1, 0)$. The example also shows why later masks must use corrected values.

Write $E = [e_0 \cdots e_{d-1}] \in \mathbb{R}^{m \times d}$ for the matrix of compensation vectors. The update rule is equivalently

$$v_j = W_{:,j} - \sum_{k < j} e_k U_{kj}, \quad e_j = \frac{v_j - q_j}{U_{jj}}. \quad (2)$$

We call the sequence of group-entry snapshots, masks, corrected columns v_j , outputs q_j , and compensation vectors e_j the *pruning trajectory*. Exact execution refers to these real-arithmetic values, rather than a tolerance or a particular floating-point evaluation order.

In the *supplied-factor* problem, the input is (W, U) as in Definition 1.1. In the *raw-calibration* problem, it is (W, X) together with a prescribed damping parameter $\lambda \geq 0$. The columns of $X \in \mathbb{R}^{d \times h}$ are h example input vectors for the layer; these are the *calibration vectors*. Damping adds λ to the diagonal of $XX^\top$. Valid inputs satisfy the positive-definiteness condition below. We must first form

$$H = XX^\top + \lambda I \succ 0, \quad U^\top U = H^{-1},$$

with U upper triangular and positive on the diagonal, then compute the same pruning output. This preprocessing is included in the raw-calibration cost. The term *inverse-Cholesky kernel* refers to this choice of U ; the direct inverse-matrix variant is distinguished in Section 8.

We count scalar arithmetic operations, comparisons and nonnegative square roots in nonuniform computation over $\mathbb{R}$; constants may depend on dimension. For fixed $a, b, c \geq 0$, $\omega(a, b, c)$ is the infimum of exponents θ for nonuniform real arithmetic circuits multiplying a $\lceil n^a \rceil \times \lceil n^b \rceil$ matrix by a $\lceil n^b \rceil \times \lceil n^c \rceil$ matrix in $O(n^\theta)$ operations. A square exponent $\tau > 2$ is *admissible* if an $O(n^\tau)$ square-multiplication family exists. Section 6 justifies the comparison of the pruning model with this arithmetic-circuit exponent. In $O_\varepsilon(\cdot)$, the hidden

constant may depend on the fixed $\varepsilon > 0$; it may also depend on parameters declared fixed, but not on growing dimensions or input entries. An infimum exponent need not be attained.

The main result identifies the cost of this specified task, including the cost of obtaining U from raw calibration.

Theorem 1.2 (Exact structured pruning). *Fix integers $1 \leq r < q$ and a damping parameter $\lambda \geq 0$. The layer width d is divisible by q .*

- (i) *For square weights and calibration $X \in \mathbb{R}^{d \times h}$ with $h = O(d)$ and $XX^\top + \lambda I \succ 0$, exact r -of- q pruning can be executed in $O_\varepsilon(d^{\omega+\varepsilon})$ operations for every $\varepsilon > 0$.*
- (ii) *Given $A, B \in \mathbb{R}^{n \times n}$, one can construct, in $O(n^2)$ operations, square weights and square calibration of dimension $d = 2qn$ such that one pruning output determines AB in another $O(n^2)$ operations. The construction satisfies*

$$\max_{i,j} |W_{ij}| \leq 4, \quad \kappa_2(XX^\top + \lambda I) \leq \frac{81}{49} < 2. \quad (3)$$

At every mask selection, the smallest retained score exceeds the largest removed score by at least $225(1 + \lambda)/64$.

- (iii) *The infimal exponent for this exact pruning problem is ω . This remains true on the promised inputs in (3).*

The same exponent holds when the upper factor U is supplied. In that representation the lower bound already holds for $U = I + T$ with $T^2 = 0$ and $\|T\|_2 \leq 1/8$.

More concretely, every admissible square-multiplication bound $O(d^\tau)$ with $\tau > 2$ gives an $O(d^\tau)$ execution bound for the fixed- r -of- q problem in part (i), by Theorem 3.1 and Appendix A. The current matrix-multiplication bound [4] therefore gives $O(d^{2.371177})$ arithmetic operations. The equality in part (iii) concerns the infimum of admissible exponents; it does not assert that the endpoint $O(d^\omega)$ is attained.

The dimensions in part (ii) are constant multiples of the matrix-product dimension because r, q are fixed. In particular, the theorem applies to 2-of-4 pruning. The positive damping is part of the reduction, not preprocessing that is supplied for free.

The upper bound holds for a substantially larger class of decisions than those in Theorem 1.2. A group may select its mask from its entire corrected snapshot; different groups may have different lengths. Its later column decisions may be nonlinear and may depend on the history already produced. No masks are moved to an earlier selection time. This permits an exact application to adaptive pruning and to fixed-order quantization.

The layer's aspect ratio yields a sharper characterization. Write $\rho(\alpha) = \omega(\alpha, 1, 1)$ for the exponent of multiplying a $\lceil n^\alpha \rceil \times n$ matrix by an $n \times n$ matrix. When $m = \Theta(d^\alpha)$ and the factor is supplied, the general causal execution costs

$$O_\varepsilon(d^{\rho(\alpha)+\varepsilon} + C), \quad (4)$$

where C is the total decision cost. For every fixed r -of- q rule, its infimal exponent is exactly $\rho(\alpha)$ (Theorem 7.1). Thus a single row has exponent 2, while square weights have exponent ω . An explicit implementation using only a square exponent $2 < \tau \leq 3$ costs $O(d^2 m^{\tau-2} + C)$ when $m \leq d$, and $O(md^{\tau-1} + C)$ when $m \geq d$. Raw calibration adds its own preprocessing cost; it is not implicit in the supplied-factor characterization.

The square lower bound also extends beyond row-wise structure (Theorem 5.1). Removing half the entries in each column, or half the entries across each prescribed group, already carries a full matrix product. The group result holds for every contiguous partition, even a single group containing the entire layer. The reference

implementation requires one more distinction: at sparsity parameter $1/2$, its inclusive threshold can remove half plus one entry. An isolated guard construction implements precisely that predicate, for every fixed block width at least two. These three inverse-Cholesky reductions retain the boundedness, conditioning, and positive-damping guarantees above. For the direct inverse-matrix kernel, we prove the corresponding supplied-input and raw zero-damping results separately.

1.2 Why the two directions work

The upper bound uses a prefix invariant. On entering an interval, all errors from before the interval have already been incorporated. Solve the left part, use one matrix product to send its errors to the right part, and continue. The decisions remain sequential; only the transport is batched. A mask group requires an extra step: its entire snapshot must be current before the mask is selected. Choosing the group that contains the midpoint gives a recursion that respects this requirement even for very unequal group lengths. To retain rectangular complexity, a node of width s divides its fixed m rows into blocks of $\lceil s^\alpha \rceil$ and uses an α -shaped product on each block. The resulting costs decrease geometrically across scales.

The lower bound separates signal coordinates, which carry the encoded matrix entries, from sentinel coordinates, whose fixed weights force the desired selections. A source group contains one encoded entry of A , small sentinels that are removed, and large sentinels that are retained. A target group contains one coordinate that will record a matrix-product entry. The interaction matrix carries the entries of B from sources to targets. The score gaps force exactly the intended decisions.

For zero damping, this can be arranged with a square-zero perturbation of the identity. Damping changes the Cholesky factor and introduces interactions among source coordinates, so the same calculation no longer suffices. The useful invariant is then a block identity. If R is the source block of the inverse-Cholesky factor, the source errors contain R^{-1} while their transport to targets contains R . The factors cancel. A norm bound on the hypothetical all-pruned source trajectory, followed by induction through the actual selectors, makes this cancellation legitimate. Section 4.2 supplies the full argument.

For whole-group selection, that induction must control every score at the group’s earlier snapshot, across different column diagonals. Bounding the entire remaining source vector provides this control. For the reference threshold, a removed positive guard contributes an additional error; placing it on an isolated coordinate prevents that error from corrupting the product. The selection proofs thereby separate two jobs: forcing the desired mask and preserving the signal transported by its errors.

1.3 Relation to earlier work

Fast matrix multiplication has long been used to accelerate triangular factorization and inversion [2]. Demmel, Dumitriu, and Holtz [3] also give recursive reductions and stability results for standard linear algebra; their matrix-multiplication-from-inversion construction is an antecedent of algebraic encodings such as ours. Relaxed online algorithms can also preserve an inherently sequential interface while obtaining matrix-multiplication complexity; Storjohann and Yang [9] establish this for a different problem in which successive leading linear systems must be solved online. Our recursive update primitive belongs to this established family of ideas. We do not claim a new general principle of divide and conquer.

The contribution here is its exact interface with adaptive compression, together with matching reductions under the compression rules’ own constraints. SparseGPT supplies adaptive masks and a shared compensation factor [5]; GPTQ supplies a closely related quantization recurrence [6]. The present analysis keeps their decision order fixed. It improves the arithmetic upper bound of [8], identifies the supplied-factor exponent for

rectangular layers, and proves square lower bounds across row, column, and block selection. The inverse-Cholesky reductions survive positive damping and separated score cuts; the exponent comparison also accounts for comparisons and square roots.

The theorems concern exact execution of a specified compression procedure. They give neither an approximation guarantee for the best sparse matrix nor a wall-clock speedup for a floating-point implementation. These distinctions matter: the lower-bound family has easy, predetermined mask outcomes, while its retained numerical values encode the hard computation.

2 A causal interface for error compensation

Definition 1.1 fixes a particular mask rule. We now allow other deterministic selection and quantization rules while preserving the same order of information: a decision may use its current values and completed history, but cannot inspect mutable future columns outside the interface specified below. This is what we mean by *causal*.

Let $\mathcal{G} = (G_1, \dots, G_s)$ be a prescribed partition of $[0, d)$ into s nonempty consecutive intervals; $|G|$ is the number of columns in a group G . The entries below the diagonal of the supplied interaction matrix are irrelevant, so we take U to be upper triangular and assume $U_{jj} \neq 0$.

For a group $G = [a, b)$, define its entry snapshot by

$$S_G = W_G - E_{[0,a)} U_{[0,a),G}. \quad (5)$$

A *group oracle* is a charged decision procedure called once at group entry. It receives S_G , fixed metadata (immutable parameters of the rule), and the observable history produced before a . It installs policies for the columns in G . A *column policy* is the rule for producing a column's output; these rules are evaluated in column order. Policy j receives its current corrected column v_j and the previous history, produces q_j , and may update its state. The coefficients are given by (2).

The history can include emitted columns, coefficients, earlier snapshots, and policy state. It is separate from the mutable work array used by an implementation. The restrictions are that a group oracle cannot inspect mutable columns outside its current group, and a column policy cannot inspect any other unprocessed column. The fixed matrix U and immutable metadata are available throughout. The partition and the column order are fixed before execution.

These restrictions describe the information interface, not the mathematical form of a decision. A policy may compare entries, select a mask, quantize a value, or perform any other deterministic computation. We charge all such work to C , including group-oracle work and leaf-policy work. Its state space is also charged to the policy, rather than assumed to be constant.

Example 2.1 (Snapshot masks). A group oracle returns a binary mask $M_G \in \{0, 1\}^{m \times |G|}$ with 1 denoting retention. Keeping the original column labels, write $M_{:,j}$ for its column indexed by $j \in G$. Its leaf policy is $q_j = M_{:,j} \odot v_j$. Selecting scores across the whole group, separately in each row, or separately in each column all fit this interface. Sorting costs at most $O(m|G| \log(2+md))$, so across a partition $C = O(md \log(2+md))$. For fixed r -of- q masks, $C = O(md)$.

Example 2.2 (Quantization). An oracle may install a finite quantization grid from fixed metadata or from the group's entry snapshot. Each leaf then quantizes its corrected column. More generally, a causal grid update is allowed if its inputs satisfy the interface and its work is charged to C . Future-dependent recalibration is not implicitly included.

All bounds below hold for every fixed admissible $\tau > 2$; hidden constants may depend on τ . Thus choosing $\tau = \omega + \varepsilon$ gives the usual exponent statement. We never require an algorithm to attain the infimum ω itself, or assume that $\omega > 2$.

3 Recursive transport preserves the trajectory

Theorem 3.1 (Causal simulation). *Every grouped causal procedure in Section 2 can be simulated exactly in*

$$O(d^\tau + md^{\tau-1} + C)$$

operations, for every admissible $\tau > 2$. The simulation preserves every group-entry snapshot, every policy call and state transition, every emitted column, and every compensation coefficient.

The proof has two components. First, once a group's oracle has run, a binary recursion executes its column policies. Second, a recursion over whole groups establishes the snapshot required by each oracle. Neither recursion predicts a future policy decision.

3.1 Executing installed policies

Maintain a mutable residual matrix V , initially W , and separate output matrices Q, E . For a consecutive interval $I = [a, b)$, the entry invariant is

$$V_I = W_I - E_{[0,a)}U_{[0,a),I}. \quad (6)$$

Suppose all policies within I have already been installed by its group oracle. If $I = \{j\}$, evaluate the policy at $V_{:,j}$ and store q_j and $e_j = (V_{:,j} - q_j)/U_{jj}$. Otherwise split I into consecutive halves L, R . Execute L , perform

$$V_R \leftarrow V_R - E_L U_{LR}, \quad (7)$$

and execute R .

Lemma 3.2 (Local invariant). *Given (6), this binary recursion produces the same Q_I, E_I and policy history as the sequential recurrence on I .*

Proof. At a leaf, the invariant is exactly the formula for v_j . For an internal node, the left call inherits the invariant and, by induction, produces its exact coefficients. Before (7), V_R includes the contribution of columns preceding I . The product adds precisely the missing contribution of L , establishing the invariant for R . The right call is therefore exact as well. Leaves are visited in the original column order, so each stateful policy receives the same earlier history. No processed column is queried through the mutable residual; its emitted value and coefficient are available from Q, E . $\square$

3.2 Preparing complete mask snapshots

Applying the local recursion blindly across a mask boundary would be incorrect. An oracle that examines a whole group needs every column of that group to be current before it selects a mask. We enforce this by making each group an indivisible unit in an outer recursion.

Let I be a consecutive union of groups. Choose the group G containing a middle column of I , and let L and R be the unions of groups before and after G . Each of L, R has at most $|I|/2$ columns. Process L

Algorithm 1 Process a consecutive union I of complete groups

Require: Entry invariant (6) on I

- 1: **if** I is empty **then**
 - 2: **return**
 - 3: **end if**
 - 4: Let G contain a middle column of I ; write $I = L G R$
 - 5: **PROCESSGROUPS**(L)
 - 6: $V_{G \cup R} \leftarrow V_{G \cup R} - E_L U_{L, G \cup R}$
 - 7: Install the policies of G using the snapshot V_G
 - 8: **PROCESSINSTALLEDPOLICIES**(G) ▷ Lemma 3.2
 - 9: $V_R \leftarrow V_R - E_G U_{GR}$
 - 10: **PROCESSGROUPS**(R)
-

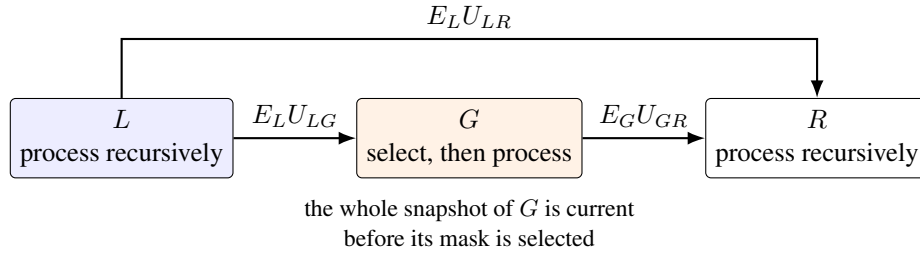


Figure 1: Data flow at one midpoint-group node. The left interval is completed first. Its two outgoing corrections are computed together as $E_L U_{L, G \cup R}$. Only then is the oracle for G called. After processing G , the correction $E_G U_{GR}$ completes the prefix needed by R . The boxes represent consecutive column intervals; their widths are schematic.

recursively, send its errors to $G \cup R$, call the oracle for G , process G using Lemma 3.2, send its errors to R , and finally process R recursively. Figure 1 shows the cross-interval products.

Lemma 3.3 (Group invariant). *Algorithm 1 invokes every oracle on (5) and preserves the complete sequential trajectory.*

Proof. Induct on the number of groups in $I = [a, b)$. After the call on L , its coefficients are exact. The first update changes the prefix represented in $V_{G \cup R}$ from $[0, a)$ to all columns preceding G . Consequently $V_G = S_G$ before the oracle is called. Lemma 3.2 gives the exact coefficients of G . The second update incorporates these coefficients into V_R , establishing the entry invariant for the final call. The empty intervals contribute no work. Both group oracles and leaf policies occur in their original order, which also proves the claim about the observable history. $\square$

3.3 Summing the sequential work

The analysis counts every product in its actual execution order. Products at a common depth are not assumed to run simultaneously.

Lemma 3.4 (A padded product bound). *If $a, b \leq s$, an $m \times a$ matrix times an $a \times b$ matrix can be computed in*

$$O(ms^{\tau-1} + s^\tau)$$

operations. Adding the result to an $m \times b$ residual has the same upper bound.

Proof. Split the first matrix into at most $\lceil m/s \rceil$ row blocks and pad each resulting product to size $s \times s$. The multiplication work is $O(\lceil m/s \rceil s^\tau)$. The addition costs $O(ms)$ and is absorbed because $s \geq 1$ and $\tau > 2$. $\square$

For an interval tree on b columns, nodes at depth ℓ have total length b and maximum length $O(b/2^\ell)$. Lemma 3.4 therefore bounds the local-policy recursion by

$$O(mb^{\tau-1} + b^\tau). \quad (8)$$

If the group lengths are $b_1, \dots, b_s$, their sum is d . Since $\tau - 1 > 1$, summing (8) over groups gives

$$O\left(m \sum_i b_i^{\tau-1} + \sum_i b_i^\tau\right) = O(md^{\tau-1} + d^\tau).$$

For the outer recursion, every recursive child has at most half as many columns as its parent. At depth ℓ , let the interval lengths be $s_1, s_2, \dots$. They satisfy

$$\sum_i s_i \leq d, \quad \max_i s_i \leq d/2^\ell.$$

Each node performs at most two cross-interval products. Its level contribution is at most

$$O\left(m \sum_i s_i^{\tau-1} + \sum_i s_i^\tau\right) \leq O\left(md^{\tau-1}2^{-\ell(\tau-2)} + d^\tau 2^{-\ell(\tau-1)}\right). \quad (9)$$

Both geometric series converge. Initializing the dense arrays costs $O(md + d^2)$; group membership can be tabulated in $O(d)$ time. Adding C proves Theorem 3.1.

The strict condition $\tau > 2$ is used only in this summation. If $\omega = 2$, every fixed $\tau > 2$ still gives the stated infimal exponent. This argument does not claim a quadratic bound at the endpoint.

The recursion can use views into V, Q, E and reuse product buffers. Apart from policy state, its storage is $O(md + d^2)$ plus the workspace of the chosen multiplication routine. A quadratic-space multiplication routine therefore yields quadratic space when $m = O(d)$.

4 Matrix products inside structured pruning

We now prove the reverse reduction. The construction has n source groups followed by n target groups. Each group has q coordinates and one designated signal coordinate. All remaining coordinates are sentinels whose only role is to control the score comparison. The reduction supplies the weights and calibration; it does not supply the matrix product hidden in them.

4.1 A transparent factor-input construction

It is useful to first see the reduction when the interaction factor is given directly. Let $A, B \in \mathbb{R}^{n \times n}$ and define

$$a = \max\{1, \max_{i,j} |A_{ij}|\}, \quad b = \max\{1, \max_{i,j} |B_{ij}|\}, \quad \hat{A} = A/a, \quad \hat{B} = B/b, \quad \eta = \frac{1}{8n}. \quad (10)$$

For $k, j \in [0, n)$, the first coordinates of source group k and target group j are $s_k = qk$ and $t_j = q(n + j)$. Write $S = (s_0, \dots, s_{n-1})$ and $\mathcal{T} = (t_0, \dots, t_{n-1})$ for these ordered signal index lists. Define a $d \times d$ matrix, $d = 2qn$, by

$$T_{s_k, t_j} = \eta \hat{B}_{kj}, \quad T_{uv} = 0 \quad \text{otherwise.} \quad (11)$$

The supports of the source and target coordinates are disjoint, so

$$T^2 = 0, \quad U = I + T, \quad X = I - T = U^{-1}. \quad (12)$$

In particular, $U^\top U = (XX^\top)^{-1}$: the orientation is that of the upper inverse-Cholesky factor used in (2).

For row i , populate the groups as follows:

	signal	small sentinels	large sentinels
source group k	$\hat{A}_{ik}$	0 repeated $r - 1$	4 repeated $q - r$
target group j	2	0 repeated r	4 repeated $q - r - 1$.

(13)

Every group has exactly q entries, including at $r = 1$ and $r = q - 1$.

Proposition 4.1 (Factor-input reduction). *The construction (10)–(13) encodes AB in one r -of- q pruning call. Its designated target outputs satisfy*

$$Q_{i, t_j} = 2 - \eta(\hat{A}\hat{B})_{ij}, \quad AB = \frac{ab}{\eta}(2J - Q_{:, \mathcal{T}}), \quad (14)$$

where $\mathcal{T} = (t_0, \dots, t_{n-1})$ and J is the $n \times n$ all-ones matrix. The weights are bounded by 4, all score cut gaps are at least $225/64$, and $\|T\|_2 \leq 1/8$.

Proof. No source coordinate has an incoming interaction. In every source group, the signal has magnitude at most 1, the small sentinels are zero, and the remaining entries have magnitude 4. Since $U_{jj} = 1$, exactly the signal and the $r - 1$ small sentinels are removed. The score cut gap is at least $16 - 1 = 15$. Hence $e_{i, s_k} = \hat{A}_{ik}$, and all other source errors vanish.

Only designated target coordinates receive these errors. Their corrected values are $2 - \eta(\hat{A}\hat{B})_{ij}$. The product entry has magnitude at most n , so these values lie in $[15/8, 17/8]$. The selector removes the r zero sentinels and retains the signal and every large sentinel. Its score cut gap is at least $(15/8)^2 = 225/64$. All target errors are zero, establishing (14). Finally,

$$\|T\|_2 \leq \|T\|_F = \eta\|\hat{B}\|_F \leq \eta n = 1/8.$$

All arrays and the decoder have $O(n^2)$ size and cost for fixed q . □

The weights above have $m = n$ rows. To obtain square weights, append zero rows to $\hat{A}$ and apply the same group construction to those rows, including all sentinels and target entries. Appending all-zero rows directly to W would not preserve the score-gap promise.

The same construction is also a lower bound for an r -of- q direct inverse-matrix kernel when that matrix is supplied: take $P = I + T + T^\top$. It is positive definite, its upper triangle is $I + T$, and its eigenvalues lie in $[7/8, 9/8]$. This observation concerns that input representation and selector; it makes no claim about encoding a damped calibration for the direct inverse-matrix kernel.

4.2 Damping and the source-to-target identity

For the raw-calibration problem, the matrix used by the algorithm is

$$U = \text{chol}_{\text{upper}}\left((XX^\top + \lambda I)^{-1}\right).$$

Here $\text{chol}_{\text{upper}}(P)$ denotes the unique upper-triangular matrix R with positive diagonal satisfying $R^\top R = P$. Even with the same $X = I - T$, a positive λ changes U . In particular, source coordinates can now interact with one another. We keep the calibration construction and reduce each source signal weight by another factor of n :

$$\text{source signal in row } i, \text{ group } k : \quad W_{i,s_k} = \hat{A}_{ik}/n. \quad (15)$$

All small sentinels, large sentinels, and target entries remain as in (13). Set $c = 1 + \lambda$.

Every sentinel coordinate is isolated in the covariance and has diagonal c . Removing those isolated coordinates from the notation, the covariance on the ordered source and target signals is

$$H_{\text{sig}} = \begin{pmatrix} cI + \eta^2 \hat{B} \hat{B}^\top & -\eta \hat{B} \\ -\eta \hat{B}^\top & cI \end{pmatrix}. \quad (16)$$

The restricted matrix retains the source and target labels S and T , so subscripts such as SS and ST below designate its source–source and source–target blocks. Equation (16) is an identity used in the proof. The encoder writes $X = I - T$ directly and does not form the displayed matrix product.

Lemma 4.2 (The transport identity). *Let*

$$K = cI + \frac{\eta^2 \lambda}{c} \hat{B} \hat{B}^\top. \quad (17)$$

On the signal coordinates, the upper inverse-Cholesky factor has the form

$$U_{\text{sig}} = \begin{pmatrix} R & Z \\ 0 & D \end{pmatrix}, \quad R^\top R = K^{-1}, \quad Z = \frac{\eta}{c} R \hat{B}, \quad D = c^{-1/2} I. \quad (18)$$

Every sentinel has factor diagonal $c^{-1/2}$ and no interactions with other coordinates.

Proof. The Schur complement of the bottom-right block of (16) is K . The leading and off-diagonal blocks of H_{sig}^{-1} are therefore

$$(H_{\text{sig}}^{-1})_{SS} = K^{-1}, \quad (H_{\text{sig}}^{-1})_{ST} = \frac{\eta}{c} K^{-1} \hat{B}.$$

Comparing them with $U_{\text{sig}}^\top U_{\text{sig}}$ gives $R^\top R = K^{-1}$ and

$$R^\top Z = \frac{\eta}{c} R^\top R \hat{B},$$

which proves the formula for Z . The Schur complement of the leading block of H_{sig}^{-1} is $c^{-1}I$, giving $D^\top D = c^{-1}I$. The positive-diagonal upper factor is $D = c^{-1/2}I$. Isolated sentinel coordinates introduce no fill in Cholesky elimination, so interleaving them among signal coordinates leaves these identities unchanged in the original signal order. $\square$

If every source signal were removed, the error matrix on the source signals would be

$$E_S^* = \frac{\hat{A}}{n} R^{-1}. \quad (19)$$

This follows from forward column processing: $E_S^* R = \hat{A}/n$ is exactly the recurrence with all source outputs equal to zero. Combining (19) with Lemma 4.2 yields

$$\underbrace{\frac{\hat{A}}{n} R^{-1}}_{\text{source errors transport to targets}} \underbrace{\frac{\eta}{c} R \hat{B}}_{\text{source errors transport to targets}} = \frac{\eta}{cn} \hat{A} \hat{B}. \quad (20)$$

The remaining task is to prove that the algorithm actually follows the trajectory used in this identity.

Lemma 4.3 (Forcing the source trajectory). *In the damped construction, the actual selector removes every source signal and the $r - 1$ zero sentinels beside it. Its source cut gaps are at least $959c/64$.*

Proof. Since $\|\hat{B}\|_2 \leq n$ and $\eta = 1/(8n)$,

$$\lambda_{\max}(K) \leq c + \frac{\lambda}{64c} \leq \frac{65c}{64}. \quad (21)$$

Moreover $R^{-1} R^{-\top} = K$. The Euclidean norm of every row of $\hat{A}/n$ is at most $1/\sqrt{n}$, so (19) gives

$$\|E_{i,S}^*\|_2^2 = \frac{\hat{A}_{i,:}}{n} K \frac{\hat{A}_{i,:}^\top}{n} \leq \frac{65c}{64n}. \quad (22)$$

Induct through the source groups. Assume the earlier groups have followed the hypothetical trajectory. The current signal is the first coordinate of its group, so its normalized pruning score is exactly $(E_{i,s_k}^*)^2$. By (22) it is at most $65c/(64n)$. The $r - 1$ small sentinels have score zero, while each of the $q - r$ large sentinels has score $4^2/(c^{-1/2})^2 = 16c$. Thus the actual rule removes precisely the proposed set, with gap at least

$$16c - \frac{65c}{64n} \geq \frac{959c}{64}.$$

This proves the induction, including its first group. The argument uses the hypothetical trajectory only to bound a candidate score, then verifies that the actual decision agrees with it. $\square$

Theorem 4.4 (Damped raw-calibration reduction). *For every $\lambda \geq 0$, use the normalization (10), T from (11), calibration $X = I - T$, and the weights in (13) with source replacement (15). The designated outputs satisfy*

$$Q_{i,t_j} = 2 - \frac{\eta}{cn} (\hat{A} \hat{B})_{ij}, \quad AB = \frac{abcn}{\eta} (2J - Q_{:,T}). \quad (23)$$

It has $|W_{ij}| \leq 4$, cut gaps at least $225c/64$, and

$$\text{spec}(XX^\top + \lambda I) \subseteq [\lambda + 49/64, \lambda + 81/64], \quad \kappa_2(XX^\top + \lambda I) \leq 81/49. \quad (24)$$

The encoder and decoder each use $O(n^2)$ operations for fixed r, q .

Proof. Lemma 4.3 validates the errors in (20). The corrected target signals consequently equal the first expression in (23). Since $|(\hat{A}\hat{B})_{ij}| \leq n$, they lie in

$$[2 - \eta/c, 2 + \eta/c] \subseteq [15/8, 17/8].$$

All target and sentinel factor diagonals are $c^{-1/2}$ by Lemma 4.2. The rule therefore removes the r zero sentinels and retains the signal, with gap at least $225c/64$. Every target error is zero, so no subsequent target decision changes these values. This proves the decoder.

The construction has $X = I - T$ and $\|T\|_2 \leq 1/8$. For every vector z ,

$$\frac{7}{8}\|z\|_2 \leq \|Xz\|_2 \leq \frac{9}{8}\|z\|_2.$$

Squaring the singular-value bounds and adding λI proves the spectral interval in (24). Its ratio is maximized at $\lambda = 0$ and equals $81/49$ there. All weights have the required bound, including the additional rows used for square weights as described after Proposition 4.1. Finally, the encoder writes X from the entries of $\hat{B}$ and W from those of $\hat{A}$ and fixed sentinels. It does not compute K, R, Z , or a Gram matrix. Those objects occur only in the proof. The decoder is entrywise, giving both quadratic costs. $\square$

The proof holds for every nonnegative value of λ , not only for a fixed constant. It therefore also applies when damping is a prescribed multiple of the average diagonal of $XX^\top$: that scalar is computable as a multiple of $\|X\|_F^2/d$ in quadratic time and can be used in the decoder. If the convention multiplies $XX^\top$ by a fixed positive constant, rescaling the calibration supplies the same covariance.

4.3 What the margins do and do not guarantee

The reduction's score gaps show that its computational difficulty does not rely on deciding nearly tied scores. Its condition number shows that this difficulty does not rely on a nearly singular covariance. Neither property alone gives a forward-error analysis for an arbitrary floating-point execution.

There is nevertheless a useful precision statement about the output itself.

Proposition 4.5 (Decoding from approximate outputs). *Suppose A, B have integer entries bounded in magnitude by an integer $M \geq 1$, and set $a = b = M$ in the damped construction. If an approximate designated output $\tilde{Q}$ satisfies*

$$\|\tilde{Q} - Q; \tau\|_{\max} < \frac{1}{16(1 + \lambda)n^2M^2},$$

then rounding the decoded values in (23) recovers AB exactly. For the factor-input construction of Proposition 4.1, the sufficient threshold is $1/(16nM^2)$.

Proof. The damped decoder amplifies entrywise error by

$$\frac{M^2cn}{\eta} = 8M^2cn^2.$$

The hypothesis makes the decoded error strictly smaller than $1/2$, so the nearest integer is unique and equals the true product entry. The factor-input decoder has amplification $M^2/\eta = 8nM^2$. Without a rounding primitive, each integer can be identified using $O(\log(2 + nM^2))$ comparisons because its magnitude is at most nM^2 . $\square$

This is a reduction for bounded-integer matrix multiplication at a specified output accuracy. It is not a bit-complexity upper bound for pruning, a constant-error hardness theorem, or a claim that fast matrix multiplication realizes that accuracy at fixed precision.

5 Lower bounds for column and block selection

The structured reduction controls comparisons within each row. A column selector compares rows instead, while a whole-group selector compares entries with different factor diagonals at an earlier common snapshot. These changes require different forcing arguments. They do not remove the matrix product carried by the numerical output.

We distinguish three rules. The *column-half* rule removes exactly half the rows of each corrected column. The *group-half* rule removes exactly half the entries in each prescribed group, using its entry snapshot. The *reference threshold* rule sorts the N scores of a prescribed block, takes the zero-based entry at index $\lfloor N/2 \rfloor$, and removes every score at most that value. The last rule is the unstructured reference predicate at sparsity parameter $1/2$ [7]; it generally removes more than half the entries.

Theorem 5.1 (Selection-rule lower bounds). *For square weights with even width, column-half and group-half pruning have infimal exponent ω under the inverse-Cholesky kernel, both with the factor supplied and with raw square calibration at every prescribed $\lambda \geq 0$. The group-half reduction works for every prescribed contiguous partition, including groups crossing the source/target boundary.*

For every fixed block width $q \geq 2$, the reference threshold rule has the same exponent. Its hard instances have $d = 2qn$ and remove exactly $N/2 + 1$ entries in every block, using the stated index and predicate.

Each lower bound uses one pruning call with quadratic encoding and decoding, $|W_{ij}| \leq 4$, covariance condition number at most $81/49$, and a score cut gap at least $1 + \lambda$. The same selector problems have exponent ω for a supplied direct inverse matrix, and for the raw direct inverse-matrix kernel at $\lambda = 0$; the latter constructions have cut gap at least 1.

5.1 Rows as sentinels

For column-half and group-half, set $m = d = 2n$. There are n source columns followed by n target columns, now with $s_k = k$ and $t_j = n + j$. The lists S and T refer to these source and target indices. Use the normalization (10), and let F have entries $F_{s_k, t_j} = \eta \hat{B}_{kj}$ and zero elsewhere. Supply the raw calibration $X = I - F$. Its signal factor is exactly (18), with $c = 1 + \lambda$ and no isolated coordinates. Define the first n rows to be active and set

$$W_{i, s_k} = \begin{cases} \hat{A}_{ik}/n, & i < n, \\ 4, & i \geq n, \end{cases} \quad W_{i, t_j} = \begin{cases} 2, & i < n, \\ 0, & i \geq n. \end{cases} \quad (25)$$

The remaining rows serve as sentinels. In addition to (22), the source factor satisfies

$$\|R\|_2 \leq c^{-1/2}, \quad \frac{64}{65c} \leq R_{kk}^2 \leq \frac{1}{c}. \quad (26)$$

Indeed, $cI \preceq K \preceq 65cI/64$; the lower diagonal bound follows from $(R^{-1})_{kk} = 1/R_{kk}$ and $\|R^{-1}\|_2^2 \leq 65c/64$.

For column-half, induct over the sources. The hypothetical active error is still $E^* = (\hat{A}/n)R^{-1}$, so the candidate score in source column k is $(E_{ik}^*)^2 \leq 65c/(64n)$. Every sentinel row has current value 4 and score at least $16c$, because its earlier source entries were retained. The selector therefore removes precisely the n active entries. This validates the source trajectory. Cancellation (20) then gives active target values in $[15/8, 17/8]$; their scores are at least $225c/64$, while the other n target entries are zero. All target errors vanish, and

$$AB = \frac{abcn}{\eta}(2J - Q_{[0, n), T}). \quad (27)$$

The minimum cut gap is at least $225c/64$.

5.2 Controlling an entire earlier snapshot

For group-half, the score of a source coordinate may be inspected before earlier source coordinates in the same group have been processed. The preceding columnwise bound cannot be used at that earlier time.

Induct over the prescribed groups. Suppose all preceding groups have removed the active source entries and the sentinel target zeros, retaining their complements. Let P be the source prefix preceding the current group, and let I be the remaining source suffix. On an active row, the unprocessed source snapshot obeys

$$V_{i,I} = \frac{\hat{A}_{i,I}}{n} - E_{i,P}^* R_{P,I} = E_{i,I}^* R_{I,I}. \quad (28)$$

Consequently every source score seen by this group is bounded by

$$\frac{V_{i,k}^2}{R_{kk}^2} \leq \frac{\|E_{i,I}^*\|_2^2 \|R_{I,I}\|_2^2}{R_{kk}^2} \leq \frac{4225c}{4096n}. \quad (29)$$

For a target column j , the active snapshot value is $2 - E_{i,P}^* Z_{P,j}$. Since

$$\|Z_{:,j}\|_2 \leq \frac{1}{8c\sqrt{cn}}, \quad |E_{i,P}^* Z_{P,j}| \leq \frac{\sqrt{65}}{64cn} < \frac{1}{4},$$

every such target score is at least $49c/16$. Sentinel source scores remain at least $16c$, and sentinel target scores remain zero.

Thus each column contributes exactly n designated low entries. Across the group these constitute precisely half its entries, even if the group contains both sources and targets. Every low score is at most $4225c/(4096n)$, every high score is at least $49c/16$, and the common cut gap is at least

$$\frac{49c}{16} - \frac{4225c}{4096} = \frac{8319c}{4096} > 2c.$$

The group oracle selects exactly the proposed set. Executing its installed mask produces the hypothetical source errors and zero target errors, closing the induction. The decoder (27) follows. Singleton groups and a single group containing the entire layer are both included.

5.3 A guard for the actual threshold index

An exact-half low set does not by itself establish hardness for the reference threshold. When N is even, its zero-based index $N/2$ selects the first entry above such a set. We place an additional, isolated entry exactly at that index.

Fix $q \geq 2$ and take $m = d = 2qn$, with n source groups followed by n target groups, each of width q . The first column of each group is its signal, so here $s_k = qk$ and $t_j = q(n + j)$, with S and $\mathcal{T}$ updated accordingly. Define F only between signal coordinates as above and let $X = I - F$; all other coordinates are isolated. Use n active rows and $(2q - 1)n$ sentinel rows, with signal values as in (25). In the $m(q - 1)$ isolated entries of each group, place the following counts, in any fixed encoder order:

Group	Entries equal to 0	One guard	Entries equal to 4
Source	$(q^2 - 1)n$	3/2	$(q - 1)^2 n - 1$
Target	$(q - 1)^2 n$	3/2	$(q^2 - 1)n - 1$

These counts are nonnegative for every $n \geq 1, q \geq 2$ and sum to $2q(q-1)n$ in each row of the table. The guard has score $9c/4$. Its column is isolated, so its nonzero pruning error has no effect on any other column.

A source signal is inspected at the beginning of its group. The columnwise induction therefore bounds its active scores by $65c/(64n)$; its sentinel rows have scores at least $16c$. The n active entries together with the $(q^2 - 1)n$ isolated zeros give exactly q^2n scores below the guard. All other scores exceed it. At a target group, the $(2q - 1)n$ zero signal entries and $(q - 1)^2n$ isolated zeros again give exactly q^2n scores below the guard. Active target scores are at least $225c/64 > 9c/4$.

Each block has $N = mq = 2q^2n$ entries. Its sorted entry at index $N/2 = q^2n$ is therefore the unique guard, and the reference predicate removes exactly $N/2 + 1$ entries. The gap between the largest removed and smallest retained scores is at least

$$\frac{225c}{64} - \frac{9c}{4} = \frac{81c}{64} > c.$$

The source trajectory and decoder (27) survive because every guard is isolated. This construction works for every fixed width $q \geq 2$, including odd widths. It preserves the reference parameter $p = 1/2$, its threshold index, and its inclusive comparison; the actual removal fraction is $1/2 + 1/N$.

5.4 Kernel scope and exponent equality

All three encoders write $X = I - F$ directly and do not form a Gram matrix, inverse, factor, or product. They have quadratic cost; q is fixed in the threshold construction. Since $F^2 = 0$ and $\|F\|_2 \leq 1/8$, the conditioning proof (24) applies unchanged. Every weight is bounded by four. At $\lambda = 0$, the factor is $U = I + F$ and can instead be supplied directly, giving the supplied-factor lower bounds.

For a supplied direct inverse-matrix kernel, take $P = I + F + F^\top$. Its upper triangle and diagonal equal those of $I + F$, and its eigenvalues lie in $[7/8, 9/8]$. For raw calibration with the direct inverse kernel at zero damping, use the separate identity

$$P = (XX^\top)^{-1} = (I + F)^\top(I + F), \quad P_{SS} = I, \quad P_{ST} = \eta \hat{B}, \quad P_{TT} = I + \eta^2 \hat{B}^\top \hat{B}. \quad (30)$$

Sources have no mutual interaction, so active source errors are $\hat{A}/n$. A partial target correction has magnitude at most η , while its diagonal lies in $[1, 65/64]$. Every active target score at every relevant snapshot is therefore at least

$$\frac{(15/8)^2}{(65/64)^2} = \frac{576}{169}.$$

Active source scores are at most $1/n^2$, and sentinel source scores equal 16. These bounds force the same column and group masks. They also place the threshold guard at score $9/4$, with target cut gap at least $576/169 - 9/4 = 783/676 > 1$. The extra target interactions in (30) cause no updates, because all signal target errors are zero. The decoder is (27) with $c = 1$.

This last proof does not extend by substitution to positive damping in the direct inverse kernel. There its source block is K^{-1} , and the chronological recurrence uses $M = \text{triu}(K^{-1})$, where triu keeps the diagonal and upper triangle and sets the entries below the diagonal to zero. The correction becomes

$$\frac{\eta}{cn} \hat{A} M^{-1} K^{-1} \hat{B},$$

which need not equal the desired scaled product. The raw positive-damping direct-inverse lower bound remains outside Theorem 5.1.

Finally, sorting all relevant scores costs $C = O(d^2 \log(2 + d))$ in square dimensions, for each of the three rules. The threshold index is a dimension-dependent integer, so no numerical floor operation is added to the scalar model. Theorem 3.1, with the appropriate charged preprocessing, gives exponent at most ω . Each reverse reduction has quadratic overhead, and its n^2 designated numerical outputs are distinct bilinear perturbations of 2 on an open normalized family. The output-gate argument and Lemma 6.1 therefore give exponent at least ω , as in the proof of Theorem 1.2. This completes the proof of Theorem 5.1.

6 Comparisons, square roots, and the exponent

The upper algorithm uses matrix multiplication, but a hypothetical faster pruning algorithm might branch on comparisons or use square roots. A reduction by itself would then yield a matrix-multiplication algorithm in that richer model. To obtain an exponent statement about arithmetic circuits, we must remove this mismatch.

Let ω denote the infimum of exponents for nonuniform real arithmetic circuits computing matrix multiplication. A scalar operation has one output. We allow arbitrary real constants, and do not assume an efficient uniform construction of the circuits. On the pruning side, consider nonuniform bounded computation trees whose operations are arithmetic, comparisons, and nonnegative square roots. No floor, bit extraction, or uncharged oracle is part of this model.

Lemma 6.1 (Extracting a bilinear computation). *For any positive dimensions k, n, ℓ , if a deterministic comparison-and-square-root computation calculates AB for every $A \in \mathbb{R}^{k \times n}$ and $B \in \mathbb{R}^{n \times \ell}$ using at most $F(k, n, \ell)$ operations, then a real arithmetic circuit of size $O(F(k, n, \ell))$ calculates AB , with an absolute constant overhead.*

Proof. Fix the dimensions. We first find a nonempty open set of inputs on which a single computation path is analytic. Start with the whole input space and shrink to a nonempty open neighborhood as necessary while following the program. Arithmetic gates are analytic wherever their denominators are nonzero. At a comparison, if the difference vanishes throughout the current set, follow its equality outcome. Otherwise choose a point at which it does not vanish and restrict to a neighborhood with a fixed strict sign. At a square-root gate, the radicand is nonnegative on the current path. If it vanishes identically on the current open set, replace that gate's output by zero; otherwise restrict to a neighborhood where it is positive. The computation is defined for all inputs, so divisions on a reached path have nonzero denominators. Its bounded length makes the number of restrictions finite.

Fix a base point (A_0, B_0) in the resulting open set. Evaluate the fixed path at $A_0 + tA$ and $B_0 + uB$ in the truncated algebra

$$\mathbb{R}[t, u]/(t^2, u^2).$$

Represent a scalar by four coefficients $z_{00} + z_{10}t + z_{01}u + z_{11}tu$. Addition and multiplication have constant arithmetic cost. For an inverse $w = z^{-1}$, the coefficients are

$$\begin{aligned} w_{00} &= z_{00}^{-1}, & w_{10} &= -z_{10}/z_{00}^2, & w_{01} &= -z_{01}/z_{00}^2, \\ w_{11} &= 2z_{10}z_{01}/z_{00}^3 - z_{11}/z_{00}^2. \end{aligned}$$

For a square root $w^2 = z$, let $w_{00} > 0$ be the root at the fixed base point. Then

$$w_{10} = \frac{z_{10}}{2w_{00}}, \quad w_{01} = \frac{z_{01}}{2w_{00}}, \quad w_{11} = \frac{z_{11} - 2w_{10}w_{01}}{2w_{00}}.$$

Every denominator in these formulas is a fixed nonzero constant. Identically zero roots have all coefficients zero. Thus each original gate is replaced by a constant number of arithmetic gates, with no variable divisions or square roots.

On the chosen neighborhood the output function is matrix multiplication. Its tu coefficient at $(A_0 + tA, B_0 + uB)$ is exactly AB . Extracting those coefficients gives the desired circuit. The selected path and base point can depend on the dimensions; this is why the conclusion is explicitly nonuniform. $\square$

Proof of Theorem 1.2. For part (i), fixed r, q give $C = O(d^2)$. Theorem 3.1 gives the execution cost after U is available. Forming $XX^\top + \lambda I$ for $h = O(d)$ and computing its inverse and upper Cholesky factor also cost $O(d^\tau)$ for every admissible $\tau > 2$; Appendix A gives the recurrences. Choosing $\tau = \omega + \varepsilon$ proves the claim. Part (ii) is Theorem 4.4 with the square-weight padding described above. Proposition 4.1 proves the factor-input assertion.

For part (iii), let π be the infimal exponent for the specified nonuniform pruning model, for fixed r, q, λ . The upper bound gives $\pi \leq \omega$. An $F(d)$ -operation pruning computation, composed with the quadratic encoder and decoder, computes an arbitrary $n \times n$ matrix product in

$$O(F(2qn) + n^2)$$

operations. Lemma 6.1 implies $\omega \leq \max\{\pi, 2\}$.

For completeness, $\pi \geq 2$ does not need an assumption that output wires are charged. Restrict to the open family $|A_{ij}|, |B_{ij}| < 1$, where $a = b = 1$. The n^2 designated outputs are pairwise distinct nonconstant bilinear perturbations of 2, and none is an input coordinate or constant. On a fixed open computation path each scalar-producing gate can supply at most one of these functions. At least n^2 such gates are required. Hence $\omega \leq \pi$, completing the equality. The same reasoning uses only the promised hard family and also applies to factor input. $\square$

The theorem identifies an infimum of exponents. It does not assert that an $O(d^\omega)$ algorithm exists with no overhead, or that the path extraction in Lemma 6.1 has an efficient uniform implementation. These conventions keep the converse at exactly the strength justified by its algebraic proof.

7 The rectangular exponent

The shape of a layer is part of its complexity. When a single row interacts with a dense $d \times d$ factor, the natural cost is quadratic. Padding that row to a square matrix loses this fact. More generally, the recursive schedule must use a multiplication algorithm suited to each interval's width, even though the layer's number of rows stays fixed.

For fixed $\alpha \geq 0$, define $\rho(\alpha) = \omega(\alpha, 1, 1)$ to be the infimum exponent for multiplying a $\lceil n^\alpha \rceil \times n$ matrix by an $n \times n$ matrix in nonuniform real arithmetic circuits. Fixed multiplicative changes in the three dimensions preserve this exponent: zero padding and a fixed number of block products give both comparisons. We will use

$$\rho(\alpha) \geq \max\{2, 1 + \alpha\}, \quad \rho(0) = 2, \quad \rho(1) = \omega. \quad (31)$$

Indeed, the $\lceil n^\alpha \rceil n$ product entries are distinct functions, each different from an input or constant, and hence require that many scalar-producing gates. Every one of the n^2 variables in the second matrix affects an output and must enter an arithmetic gate. A gate has at most two input operands, giving another $n^2/2$ gates. The first barrier is necessary even when output wires and input access are free.

Theorem 7.1 (Rectangular characterization). *Fix $\alpha \geq 0$ and let $m = \Theta(d^\alpha)$, with fixed dimension-ratio constants.*

(i) *With the interaction factor supplied, every causal procedure of Section 2 has an exact execution of cost*

$$O_\varepsilon(d^{\rho(\alpha)+\varepsilon} + C) \quad (\varepsilon > 0). \quad (32)$$

(ii) *For every fixed $1 \leq r < q$, the infimal exponent of supplied-factor r -of- q pruning is exactly $\rho(\alpha)$. The lower bound already holds with $|W_{ij}| \leq 4$, $U = I + T$, $T^2 = 0$, $\|T\|_2 \leq 1/8$, and score cut gaps at least $225/64$.*

(iii) *With raw calibration having $h = O(d)$ columns, the corresponding upper bound, including preprocessing, is*

$$O_\varepsilon(d^{\max\{\omega, \rho(\alpha)\}+\varepsilon} + C). \quad (33)$$

The exponent equality in part (ii) concerns the supplied-factor representation.

7.1 Using the right shape at every scale

Choose $\gamma > \rho(\alpha)$ and an arithmetic multiplication family of cost $O(s^\gamma)$ for shape $\lceil s^\alpha \rceil \times s$ by $s \times s$. At a recursion node of width s , each transport has shape $(m \times a)(a \times b)$ with $a, b \leq s$. Split its rows into blocks of at most $\lceil s^\alpha \rceil$ and pad the two column dimensions to s . Since $m \geq c_1 d^\alpha \geq c_1 s^\alpha$, the number of row blocks, including ceilings, is $O(m/s^\alpha)$. The node cost is therefore

$$O(ms^{\gamma-\alpha}). \quad (34)$$

Dense input, padding, and addition work is absorbed by (31). This bound uses the same aspect exponent α at every scale; it does not assume that an internal node has s^α rows.

Set $p = \gamma - \alpha > 1$. At outer depth ℓ , the intervals are disjoint, their total width is at most d , and their maximum width is at most $d/2^\ell$. Thus their total transport cost is bounded by

$$O\left(m \sum_i s_i^p\right) \leq O\left(md(d/2^\ell)^{p-1}\right).$$

Summing this geometric series gives $O(md^p)$. A local recursion within a group of length b similarly costs $O(mb^p)$; summing over the groups costs at most $O(md^p)$. Because $m = \Theta(d^\alpha)$, the total is $O(d^\gamma + C)$. The transcript proof in Section 3 is unchanged: only the implementation of its transport products has changed. Taking $\gamma = \rho(\alpha) + \varepsilon$ proves part (i), including the cases in which either barrier in (31) is attained.

7.2 A rectangular reverse reduction

Let $k = \lceil n^\alpha \rceil$, $A \in \mathbb{R}^{k \times n}$, and $B \in \mathbb{R}^{n \times n}$. Use the normalization and nilpotent interaction of Proposition 4.1, with $d = 2qn$. Set $m' = \lceil d^\alpha \rceil$, append zero rows to $\hat{A}$, and rebuild the source and target groups of (13) on every row. In particular, padded rows still contain their large sentinels and target entries. The original proof acts independently on each row and gives

$$Q_{[0,k],\mathcal{T}} = 2J - \eta \hat{A} \hat{B}.$$

Here J is the $k \times n$ all-ones matrix. The decoder recovers AB from these first k rows. Encoding the dense factor costs $O(n^2)$; encoding the weights and decoding cost $O(kn)$. The construction fits the exact canonical row count $m' = \lceil d^\alpha \rceil$, with all the promises of part (ii).

Let $\pi(\alpha)$ be the pruning exponent. The reduction and Lemma 6.1, whose proof is independent of the matrix dimensions, give

$$\rho(\alpha) \leq \max\{\pi(\alpha), 2, 1 + \alpha\}. \quad (35)$$

The encoding overhead in this inequality must be accounted for, rather than discarded. Restrict to the open normalized family $|A_{ik}|, |B_{kj}| < 1$. Every pruning input is a constant or an affine function of a single input variable. On a fixed open computation path, its kn designated outputs are pairwise distinct bilinear perturbations of 2, none equal to an input coordinate. At least kn scalar-producing gates are required. Moreover, every one of the n^2 independent B entries affects these outputs and must first enter a scalar-producing gate; each gate accepts at most two such input variables. Comparisons fix the local path and cannot supply a continuously varying output in place of those gates. Hence

$$\pi(\alpha) \geq \max\{2, 1 + \alpha\}.$$

Together with (35) and part (i), this proves part (ii). This argument is needed in particular for $\alpha < 1$, where counting output entries alone would leave a gap. Adding the preprocessing cost from Appendix A proves part (iii). We make no matching raw-calibration claim at exponent ω when $\alpha < 1$.

7.3 An explicit bound from square multiplication

The exact rectangular exponent is useful as a characterization. A simpler bound needs only a square multiplication algorithm and holds for all row counts, without fixing α .

Proposition 7.2 (A tiled execution bound). *For every admissible $2 < \tau \leq 3$, supplied-factor execution costs*

$$\begin{cases} O(d^2 m^{\tau-2} + C), & 1 \leq m \leq d, \\ O(m d^{\tau-1} + C), & m \geq d. \end{cases}$$

In particular, one row has cost $O(d^2 + C)$.

Proof. At node width $s \geq m$, tile the shared and output dimensions into blocks of size at most m . The $O((s/m)^2)$ square products cost $O(s^2 m^{\tau-2})$. At $s \leq m$, tile the rows instead, at cost $O(m s^{\tau-1})$.

Suppose $m \leq d$. For the outer nodes with $s > m$, the sum of s^2 at depth ℓ is at most $d^2/2^\ell$, so their total cost is $O(d^2 m^{\tau-2})$. Stop at the first nodes of width at most m . Their disjoint intervals have total length at most d , and their entire subtrees cost

$$O\left(m \sum s^{\tau-1}\right) \leq O(d m^{\tau-1}) \leq O(d^2 m^{\tau-2}).$$

The same argument applies inside each group. Groups with $b > m$ contribute at most $O(m^{\tau-2} \sum b^2)$; groups with $b \leq m$ contribute at most $O(m^{\tau-1} \sum b)$. Both are absorbed by the stated bound. If $m \geq d$, every node uses the row-tiling bound, and the usual geometric sum gives $O(m d^{\tau-1})$. Add the charged policy work C . $\square$

8 Application to the source algorithms

There are two different numerical kernels in the sources. SparseGPT’s original algorithm and reference implementation use the positive-diagonal upper factor satisfying $U^\top U = H^{-1}$ [5, 7]. Algorithm 1 of Li et al. [8] instead writes the inverse matrix itself in the update. Theorem 3.1 applies to the upper triangle of either supplied matrix, with its own diagonal. It does not assert that these kernels produce equal outputs.

The selection rules also differ. The source comparison is summarized below; in each row the oracle is invoked on the same snapshot as the specified baseline.

Procedure	Group decision	Interaction kernel
Li et al., Algorithm 1	Per-column ranking across rows	Upper triangle of H^{-1}
SparseGPT, unstructured	Scores across a prescribed block	Upper inverse-Cholesky factor
SparseGPT, structured	r removals per row in each q -column window	Upper inverse-Cholesky factor
GPTQ, fixed order	Quantization policy at each column	Upper inverse-Cholesky factor

Table 1: Interfaces covered by the causal upper bound. Theorems 4.4 and 5.1 give matching pruning lower bounds under their stated selector parameters and kernel-specific damping assumptions. No quantization lower bound is asserted.

Corollary 8.1 (Source-relative upper bound). *For $m = O(d)$ and $O(d)$ calibration columns, the causal versions in Table 1 can be executed in $O_\varepsilon(d^{\omega+\varepsilon})$ operations whenever the matrix preprocessing is defined and the selection or quantization work totals $O(d^2 \log(2 + d))$. Their prescribed group snapshots, decisions, and numerical outputs are preserved in exact arithmetic.*

Proof. Apply Theorem 3.1 to each procedure’s own kernel and policy. Sorting the pruning scores has the stated total bound. Preprocessing is covered by Appendix A. The oracle cost and the update cost are both absorbed by every fixed $d^{\omega+\varepsilon}$ bound, using $\omega \geq 2$. $\square$

For a conventional lazy-block implementation, a sufficient condition for this interpretation is that each mask window is contained in an update block and initialized before its first column is processed. The previous update blocks have then been fully incorporated; earlier columns of the current block have updated the entire mask window. Its observed snapshot is exactly (5). We use half-open windows and the intended row/column assignments when translating the source pseudocode, avoiding its endpoint and mask-index ambiguities. The reference implementation’s whole-block unstructured selector is covered by taking that block as a group; its structured selector is covered by q -column groups when the layer and update-block widths are multiples of q . The unstructured implementation’s threshold predicate, including its tie convention and actual number of removals, is retained as part of the oracle; we do not replace it by an exact-cardinality selector. Implementation-specific preprocessing such as dead-column replacement can be executed first and charged separately. The full-rank lower-bound calibration never triggers that replacement.

The lower bounds now distinguish the same interfaces. At the one-half parameter, per-column ranking with the direct inverse-matrix kernel has a matching raw-calibration lower bound at zero damping. With the inverse-Cholesky kernel, column-half and group-half have matching raw lower bounds for every prescribed nonnegative damping. The reference unstructured predicate, without the optional quantizer, has a matching lower bound at sparsity parameter $1/2$ for every fixed block width $q \geq 2$, using full blocks and exactly $N/2 + 1$ removals per block. Its construction therefore applies to a fixed width such as 128, while keeping

the actual threshold behavior. These statements do not identify group-half with the reference predicate, nor do they assert hardness at every unstructured sparsity fraction.

For rectangular layers, Theorem 7.1 sharpens the transport cost to $O_\varepsilon(d^{\rho(\alpha)+\varepsilon} + C)$ when $m = \Theta(d^\alpha)$ and the appropriate interaction matrix is supplied. With raw calibration, the upper bound additionally charges preprocessing as in (33). The matching rectangular lower bound in this paper concerns fixed row-wise r -of- q selection.

In particular, the algorithmic update block may be replaced by a recursive schedule without changing a fixed, legitimate mask-snapshot schedule. Changing the mask groups themselves can change the result. The improvement over (1) is therefore an execution improvement for a specified causal trajectory, rather than an optimization over mask-window sizes.

For GPTQ [6], the same argument applies to a fixed column order and an installed scalar or column quantizer. Any recalibration that reads additional mutable future columns must be represented by a permitted group oracle, or charged and analyzed separately. Corollary 8.1 does not implicitly cover an arbitrary future-dependent variant of that implementation.

9 The role of adaptivity

Two observations delimit the result. First, adaptive decisions are compatible with fast transport, but their timing cannot be discarded. Second, a hard numerical output does not require a hard decision boundary.

For the first point, return to the one-row example following Definition 1.1. Its second group sees $(-4, 3)$ and removes 3. Selecting both masks at the beginning would instead use $(2, 3)$ for that group, remove the position initially equal to 2, and retain 3. Recursive execution preserves the prescribed behavior because it prepares the full second snapshot before calling its oracle.

For the second point, the source and target masks in our reduction are fixed throughout the entire normalized hard family. Every source signal is removed, every target signal is retained, and all sentinel decisions are known in advance. The lower bound nevertheless persists because the retained values recover AB . Consequently it is a lower bound for exact numerical execution, not for returning masks alone. The proof isolates the transport as the source of the arithmetic difficulty.

The rectangular characterization makes the distinction between decisions and transport quantitative. A dense factor acting on one row requires quadratic work; adding rows changes the exponent according to $\omega(\alpha, 1, 1)$. Row sentinels and isolated guards then show that several different square selection rules preserve the same algebraic difficulty. In each case the reduction forces the decisions before invoking the matrix-product identity.

Two algebraic boundaries remain. For raw rectangular calibration with $\alpha < 1$, the preprocessing upper bound can exceed the supplied-factor exponent; the present reductions do not settle that full preprocessing problem. For the raw direct inverse-matrix kernel, positive damping introduces the uncanceled factor displayed in Section 5. Its matching lower bound needs a different argument. The half-selection results also leave other unstructured removal fractions open.

A finite-precision implementation would need to combine error bounds for the recursive products with guarantees that the decisions remain unchanged; the margin promises in the lower bound do not supply such an implementation theorem. Policies that select the next column from a mutable future state require a different interface: the present recursion relies on a fixed order, and its proof does not address that additional form of adaptivity.

Within the stated interface, decisions may still occur one after another. The dense corrections between them can be organized at the appropriate rectangular matrix-multiplication cost. Well-conditioned pruning trajectories with predetermined, separated masks can already require that much algebraic work.

References

- [1] 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)*, pages 2005–2039. SIAM, 2025. doi: 10.1137/1.9781611978322.63. URL <https://arxiv.org/abs/2404.16349v3>. Updated preprint: arXiv:2404.16349v3, August 2026.
- [2] James R. Bunch and John E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974. doi: 10.1090/S0025-5718-1974-0331751-8.
- [3] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007. doi: 10.1007/s00211-007-0114-x. URL <https://arxiv.org/abs/math/0612264v3>.
- [4] Emilien Dupont, Marvin Eisenberger, Borislav Kozlovskii, Abbas Mehrabian, Francisco J. R. Ruiz, Abigail See, Renfei Zhou, Josh Alman, Virginia Vassilevska Williams, and Matej Balog. Improving the matrix multiplication exponent with modern optimization and AlphaEvolve, 2026. URL <https://arxiv.org/abs/2608.16884v1>. arXiv:2608.16884v1.
- [5] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023. URL <https://proceedings.mlr.press/v202/frantar23a.html>.
- [6] Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL <https://arxiv.org/abs/2210.17323>.
- [7] IST-DASLab. SparseGPT reference implementation. <https://github.com/IST-DASLab/sparsegpt/blob/master/sparsegpt.py>. Source snapshot accessed 13 September 2026; copy and content hash in the accompanying source audit.
- [8] Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. A tighter complexity analysis of SparseGPT, 2024. URL <https://arxiv.org/abs/2408.12151v2>. arXiv:2408.12151v2.
- [9] Arne Storjohann and Shiyun Yang. A relaxed algorithm for online matrix inversion. In *Proceedings of the 2015 ACM International Symposium on Symbolic and Algebraic Computation*. ACM, 2015. doi: 10.1145/2755996.2756672. URL <https://cs.uwaterloo.ca/~astorjoh/issac15.pdf>.
- [10] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: From alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835. SIAM, 2024. doi: 10.1137/1.9781611977912.134. URL <https://arxiv.org/abs/2307.07970v2>.

A Preprocessing at matrix-multiplication cost

We include the elementary recurrences so that matrix preprocessing is an explicit cost in Theorem 1.2. The use of multiplication to accelerate these operations is classical [2]; nothing about the argument requires pruning structure.

For an invertible upper triangular block matrix,

$$\begin{pmatrix} A & B \\ 0 & C \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{pmatrix}.$$

With balanced blocks, triangular inversion satisfies

$$T(s) \leq 2T(\lceil s/2 \rceil) + O(s^\tau) = O(s^\tau).$$

Unequal terminal dimensions can be handled directly or by constant-factor padding.

For a symmetric positive definite matrix

$$P = \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix},$$

both A and $S = C - B^\top A^{-1}B$ are positive definite. Recursively compute A^{-1} and S^{-1} and use

$$P^{-1} = \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}B^\top A^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}B^\top A^{-1} & S^{-1} \end{pmatrix}.$$

The remaining work at size s is $O(s^\tau)$, so inversion has the same recurrence and bound. Positive definiteness ensures that no pivoting or extra hypothesis is needed for these block inverses.

To form the upper Cholesky factor $P = U^\top U$, first compute the factor U_{11} of A , then set

$$U_{12} = U_{11}^{-\top} B,$$

and recursively factor $C - U_{12}^\top U_{12}$. The latter matrix is positive definite. The triangular inversion already has cost $O(s^\tau)$, and the remaining matrix products have that cost as well. Thus the Cholesky cost is again $O(s^\tau)$. The leaf square roots have positive arguments.

If $X \in \mathbb{R}^{d \times h}$ with $h \leq c_0 d$, its Gram matrix can be formed by $O(c_0)$ padded square products. Adding λI , inversion, and Cholesky therefore cost $O(d^\tau)$ for fixed c_0 . If h is unrestricted, a dependence on h is necessary; Corollary 8.1 explicitly assumes $h = O(d)$.

B Exact diagnostic implementations

The source bundle contains rational eager and recursive interpreters that compare weights, errors, masks, and every group-entry snapshot. A seven-product Strassen kernel supplies additional exact checks. These implementations are mathematical diagnostics, not optimized GPU kernels.

The prespecified checks completed as follows; paired counts mean raw inverse-Cholesky / raw direct inverse at zero damping:

Check	Cases
Eager/recursive exact transcripts	2,268
Exact transcripts with the Strassen kernel	48
Undamped matrix-product reductions	120
Damped raw-calibration reductions	288
Group partitions with each ordered pair transported once	4,095
Explicit failure of premature mask selection	1
Square-tiled rectangular products	216
Rectangular supplied-factor reductions	225
Current-column half	48 / 12
Whole-group half	576 / 144
Earlier-snapshot column-half	528 / 132
Reference threshold, raw inverse-Cholesky	144
Reference threshold, raw direct inverse at $\lambda = 0$	36

The transcript checks include every composition of $d \leq 6$, several row counts, signed and nonunit diagonals, exact ties, pruning, quantization, and a stateful nonlinear policy. The partition checks cover every composition through $d = 12$. The damped checks use $n \leq 4$, $q \in \{2, 3, 4\}$, every $1 \leq r < q$, and $\lambda \in \{0, 1/10, 1, 100\}$.

For damped inputs, generic rational elimination and diagonal factorization give $H^{-1} = R_0^\top \text{diag}(D) R_0$, with R_0 unit upper triangular. Since $U = \text{diag}(\sqrt{D}) R_0$, the true inverse-Cholesky trajectory is evaluated exactly using normalized errors $v_j - q_j$, transport R_0 , and scores v_j^2/D_j . The checks verify exact decoding and score cut gaps without approximate square roots.

Commands, hashes, and reports accompany the manuscript. Finite diagnostics can expose errors; the universal statements rest on the proofs above. Independent proof audits are recorded separately. No neural-network accuracy or wall-clock experiment is reported.

Rectangular product dimensions range from one through six; reductions use $n \leq 4$, row counts from $1, 2, n, 2n, n^2$, and tiled Strassen transport. New selector checks form the full raw covariance and inverse, checking both transcripts, decoding, gaps, and removal counts. Half-selector runs use every partition for $n \leq 3$ and six at $n = 4$. Current-column scoring uses singleton groups; earlier-snapshot column scoring is counted separately. Threshold checks use $n \leq 3$ and widths 2 through 5. Damping is $\{0, 1/10, 1, 100\}$ for inverse-Cholesky and 0 for direct inverse.