LoGIK: Accelerating SVD approximation with Low-precision Gram Iterative Krylov

Anonymous authors
Paper under double-blind review

Abstract

Low-rank KV cache compression reduces memory by projecting each request's cache onto a compact channel subspace. Adapting this subspace to the realized cache is essential for accurate approximation, but requires online basis extraction, which can become a major bottleneck. Exact SVD provides the optimal rank-r subspace but is too costly to run for every request, while faster approximations face their own trade-offs: lightweight methods converge slowly, whereas more accurate Krylov methods repeatedly access the full cache, eroding their speed advantage. We propose **Low-precision Gram Iterative Krylov (LoGIK)**, a Gram-space reformulation of Block Krylov that performs refinement in the much smaller channel space. LoGIK forms the channel Gram matrix once, avoiding subsequent cache traversals and tall token-space orthogonalization. We further accelerate Gram construction with low-precision computation, selectively falling back to FP32 when needed to preserve the retained subspace. This design yields substantial accuracy–latency gains: on a 32k-token Llama-3.1-8B K cache at rank 16, LoGIK matches Block Krylov accuracy while reducing basis-extraction latency by $15.78\times$. Integrated into existing KV-compression pipelines, it reduces compression latency by up to $5.1\times$ while preserving downstream quality.

1 Introduction

Low-rank KV cache compression reduces the cache by representing token states in a shared low-dimensional channel subspace (Chang et al., 2025a; Kang et al., 2024; Chang et al., 2025b; Yankun et al., 2025). The quality of this representation depends on identifying the dominant channel directions of the realized cache. For input-adaptive methods, these directions must therefore be extracted online for every request. Truncated SVD provides the optimal rank-r basis under Frobenius reconstruction error, but its cost makes it impractical for online compression.

This motivates approximate methods that recover the dominant subspace at lower cost. Power iteration offers a simple solution, but may require repeated refinement to obtain an accurate basis. Block Krylov improves upon this by retaining candidates from multiple refinement depths and searching over their joint span (Musco & Musco, 2015). The improved approximation, however, comes with a cost: constructing this richer subspace repeatedly processes the full cache and orthogonalizes tall token-space intermediates. As the context grows, more computation is therefore spent in the token dimension, even though the final result is only a small set of shared channel directions. This mismatch becomes increasingly costly in long-context KV compression.

To resolve this mismatch, we introduce **Low-precision Gram Iterative Krylov (LoGIK)**, a new channel-space formulation of Block Krylov for KV cache compression. LoGIK first aggregates information across all tokens into a small channel Gram matrix, then performs Krylov refinement entirely in this channel space. This avoids repeated passes over the full cache and costly token-space orthogonalization, while preserving the richer search space of Block Krylov. With token-space refinement removed, Gram construction becomes the only step that still scales with context length. LoGIK accelerates this step with group-wise INT4/FP4 computation and selectively falls back to FP32 when quantization threatens the retained subspace, using the weakest retained direction and the observed Gram distortion as reliability signals.

Empirically, LoGIK delivers substantial improvements in the accuracy–latency trade-off. On 32k-token KV caches, it matches the approximation quality of Block Krylov while reducing basis-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

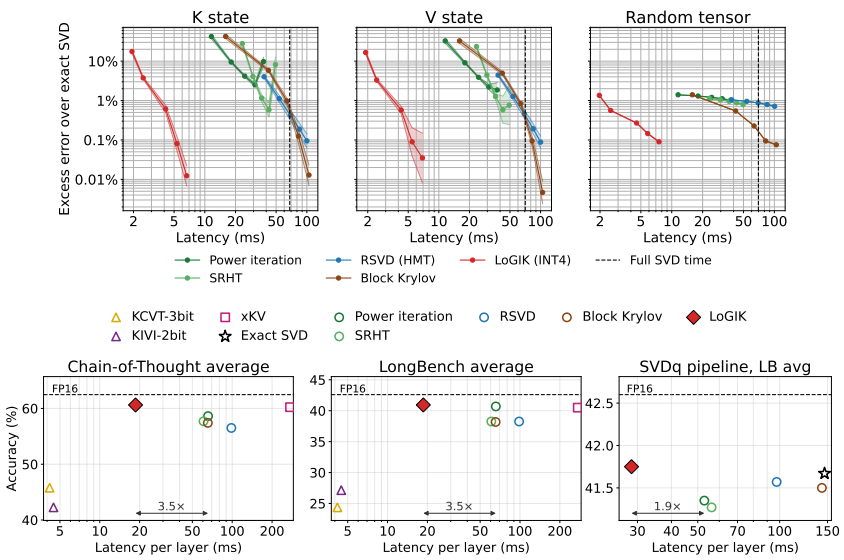

Figure 1: Top: basis-extraction time versus excess error on the Llama-3.1-8B K and V states and a StudentT tensor of the same shape (32k tokens, rank 16, mean of 32 layers, ± 1 s.d.; each point a loop count). Bottom: downstream accuracy versus per-layer compression time on Llama3-8B under GEAR (left two panels) and SVDq (right), against FP16 (dashed). LoGIK occupies the low-time, low-error corner of both.

extraction latency by up to $15.78\times$, and runs over $10\times$ faster than exact SVD. These gains translate directly to existing KV-compression pipelines, where LoGIK reduces compression latency by up to $5.1\times$ while preserving downstream quality.

Our contributions are:

- **Gram-space Krylov refinement.** We reformulate Block Krylov basis extraction in channel space, replacing repeated token-space refinement with a single Gram construction.
- **Selective low-precision execution.** We accelerate Gram construction with INT4 tensor cores and derive a retained-subspace criterion for selecting INT4 or FP32 per layer.
- **End-to-end validation.** Across three LLMs and two KV-compression pipelines, LoGIK reduces KV-compression latency by 3.58–$4.39\times$ in GEAR and $5.1\times$ in SVDq while preserving downstream quality.

2 Related Works

KV cache compression. KV cache compression broadly follows two directions: quantization reduces the precision of cached elements (Liu et al., 2024; Hooper et al., 2024; Ashkboos et al., 2024; Zandieh et al., 2026), while low-rank methods represent the cache in a compact subspace. Among the latter, PALU (Chang et al., 2025a) factorizes model weights offline and therefore avoids per-request basis extraction, at the cost of using a fixed, input-independent subspace. GEAR (Kang et al., 2024), xKV (Chang et al., 2025b), SVDq (Yankun et al., 2025), and ShadowKV (Sun et al., 2025) instead adapt the basis to the realized cache, making basis extraction part of the online compression cost. LoGIK targets this shared runtime bottleneck without changing the surrounding compression scheme. Token-eviction methods (Zhang et al., 2023; Li et al., 2024; Xiao et al., 2024) reduce the number of retained tokens rather than the channel dimension and are complementary to LoGIK.

SVD approximation. Input-adaptive low-rank methods typically approximate the dominant singular subspace rather than compute a full SVD. Randomized SVD (Halko et al., 2011), simultaneous iteration (Saad, 2011), and the power iteration used by GEAR refine a low-dimensional random block through repeated applications of the cache and its transpose, which single-pass sketches (Liberty, 2013; Tropp et al., 2017; Nakatsukasa, 2020) omit. Block Lanczos (Golub & Kahan, 1965; Golub et al., 1981) and Block Krylov (Musco & Musco, 2015) retain candidates from multiple

refinement depths, improving the search space but still performing the refinement in token space. NysBKI (Tropp & Webber, 2023) replaces QR with Cholesky-based orthogonalization for positive-semidefinite problems, but continues to apply the Gram implicitly through repeated accesses to the input. The Gram correspondence of Epperly (2024) establishes the equivalence between Krylov constructions on a matrix and its Gram. LoGIK exploits the highly rectangular geometry of KV caches to form the small channel Gram once and carry out subsequent Krylov refinement entirely in channel space.

Low-precision GEMM and Gram orthogonalization. CholeskyQR2 (Fukaya et al., 2014; Yamamoto et al., 2015), communication-avoiding Krylov (Hoemmen et al., 2010), and LOBPCG (Duersch et al., 2018) use block Gram matrices to reduce orthogonalization cost, but still construct Krylov vectors in the original high-dimensional space. Prior mixed-precision work is also cautious about lowering the Gram itself: mixed-precision CholQR (Yamazaki et al., 2015) emphasizes the numerical sensitivity introduced by Gram formation, while tensor-core RandNLA (Ootomo & Yokota, 2023; Baboulin et al., 2024) lowers the precision of the random projection instead. LoGIK takes a different view tailored to low-rank compression: only the retained leading subspace must remain reliable. This motivates computing the cache-touching Gram at low precision when that subspace is sufficiently robust.

3 Problem Formulation and Computational Opportunity

3.1 Low-Rank KV Compression as Basis Extraction

Consider a key or value cache $A \in \mathbb{R}^{S\times D}$ with S tokens and channel dimension D. Low-rank compression projects all token states onto a shared r-dimensional channel subspace. For an orthonormal basis $V \in \mathbb{R}^{D\times r}$,

$$A \approx (AV)V^\top, \qquad V^\top V = I_r, \tag{1}$$

where V contains the shared channel directions and AV their per-token coordinates. Our target is the basis V; coordinate formation and subsequent compression are handled separately.

The projection error can be written as

$$\|A - AVV^\top\|_F^2 = \mathrm{tr}(A^\top A) - \mathrm{tr}(V^\top A^\top AV). \tag{2}$$

Therefore, basis extraction depends on the cache only through the channel Gram matrix $M = A^\top A \in \mathbb{R}^{D\times D}$. Minimizing the reconstruction error is equivalent to maximizing $\mathrm{tr}(V^\top MV)$, whose solution is the leading r eigenvectors of M, or equivalently the leading r right singular vectors of A (Eckart & Young, 1936). The problem is thus not to compute a full SVD of the $S \times D$ cache, but to efficiently recover its dominant r-dimensional channel subspace.

3.2 From Power Iteration to Block Krylov

Randomized SVD methods approximate the dominant subspace without computing a full decomposition (Halko et al., 2011). Starting from a random block $G \in \mathbb{R}^{D\times r}$, the token-space iteration generates

$$Y_j = (AA^\top)^j AG, \qquad j = 0, \ldots, q-1, \tag{3}$$

where q is the number of candidate blocks. Each application of $AA^\top$ increasingly emphasizes directions associated with larger singular values. Power iteration uses only the deepest iterate Y_{q-1}, which can converge slowly when the leading singular values are close. Block Krylov instead retains all iteration depths and searches their joint span (Musco & Musco, 2015):

$$\mathcal{K}_q^{(S)} = \mathrm{range}\big[AG,\ (AA^\top)AG,\ \ldots,\ (AA^\top)^{q-1}AG\big]. \tag{4}$$

This richer candidate space allows the final approximation to combine information from multiple refinement depths rather than relying on a single iterate.

Let $k = \dim \mathcal{K}_q^{(S)} \le \min(qr, \mathrm{rank}(A))$ and let $Q_S \in \mathbb{R}^{S\times k}$ be an orthonormal basis for this space. Projecting the cache gives $B = Q_S^\top A \in \mathbb{R}^{k\times D}$, whose leading r right singular vectors approximate the dominant right singular subspace of A. Block Krylov thus improves accuracy by searching over candidates from multiple refinements, but constructing this richer space is where its cost arises.

Algorithm 1 Block Krylov

Require: $\mathbf{A} \in \mathbb{R}^{S\times D}$, rank r, block count q
Ensure: $\mathbf{V} \in \mathbb{R}^{D\times r}$

$\mathbf{Y}_S \leftarrow \mathbf{A}\,\mathrm{randn}(D, r)$ ▷ 1 pass over $\mathbf{A}$
$\mathbf{K}_S \leftarrow [\,]$
for $i = 0$ to $q-1$ **do**
  $\mathbf{Y}_S \leftarrow \mathrm{orth}(\mathbf{Y}_S)$ ▷ tall QR
  $\mathbf{K}_S \leftarrow [\mathbf{K}_S,\ \mathbf{Y}_S]$
  if $i < q-1$ **then**
    $\mathbf{Y}_S \leftarrow \mathbf{A}(\mathbf{A}^T\mathbf{Y}_S)$ ▷ 2 passes over $\mathbf{A}$
  end if
end for
$\mathbf{Q}_S \leftarrow \mathrm{orth}(\mathbf{K}_S)$ ▷ tall QR
$\mathbf{B} \leftarrow \mathbf{Q}_S^T\mathbf{A}$ ▷ 1 pass over $\mathbf{A}$
$\mathbf{V} \leftarrow$ top-r right singular vectors$(\mathbf{B})$

Algorithm 2 LoGIK (Ours)

Require: $\mathbf{A} \in \mathbb{R}^{S\times D}$, rank r, block count q, precision p
Ensure: $\mathbf{V} \in \mathbb{R}^{D\times r}$, Ritz values $\mathbf{w}$

$\mathbf{M} \leftarrow \mathrm{Gram}_p(\mathbf{A})$ ▷ 1 pass over $\mathbf{A}$
$\mathbf{Y}_D \leftarrow \mathrm{randn}(D, r)$; $\mathbf{K}_D \leftarrow [\,]$
for $i = 0$ to $q-1$ **do**
  $\mathbf{Y}_D \leftarrow \mathrm{orth}_{\mathbf{M}}(\mathbf{Y}_D - \mathbf{K}_D(\mathbf{K}_D^T\mathbf{M}\mathbf{Y}_D))$
  $\mathbf{K}_D \leftarrow [\mathbf{K}_D,\ \mathbf{Y}_D]$
  if $i < q-1$ **then**
    $\mathbf{Y}_D \leftarrow \mathbf{M}\mathbf{Y}_D$ ▷ $D \times D$ product
  end if
end for
$\mathbf{Q}_D \leftarrow \mathrm{orth}_{\mathbf{M}}(\mathbf{K}_D)$ ▷ Cholesky whitening
$\mathbf{B} \leftarrow \mathbf{Q}_D^T\mathbf{M}$ ▷ computed without $\mathbf{A}$
$\mathbf{w}, \mathbf{Z} \leftarrow \mathrm{eigh}(\mathbf{B}\mathbf{B}^T)_{:r}$; $\mathbf{V} \leftarrow \mathbf{B}^T\mathbf{Z}\,\mathrm{diag}(\mathbf{w})^{-1/2}$

3.3 The Cost of Krylov Refinement

Constructing q Krylov blocks requires repeated work on the full cache. As shown in Algorithm 1, the initial block requires one multiplication by A, each of the remaining $q-1$ blocks applies both $A^\top$ and A, and the final projection $B = Q_S^\top A$ accesses the cache once more. In total, Block Krylov therefore requires $2q$ applications of A or $A^\top$.

The refinement also incurs token-space orthogonalization: each Krylov block has size $S \times r$, while the accumulated basis grows to $S \times k$. Thus, increasing the Krylov depth couples better spectral refinement to both additional cache traversals and increasingly costly tall orthogonalization. For long-context KV caches with $S \gg D$, this is a computational mismatch: the desired output is only a $D \times r$ channel basis, yet refining it repeatedly processes the $S \times D$ cache and maintains S-dimensional intermediates. This motivates moving the same Krylov refinement into the channel dimension, without sacrificing its richer candidate space.

4 LoGIK: Krylov Refinement in Channel Space

LoGIK represents each token-space Krylov candidate implicitly in channel space through the Gram matrix $M = A^\top A$. As illustrated by Algorithms 1 and 2, this representation moves all three components of Block Krylov refinement into the channel dimension: the update $A(A^\top Y_S)$ becomes MY_D, tall QR is replaced by M-weighted orthogonalization, and the final projection $Q_S^\top A$ becomes $Q_D^\top M$. Thus, after forming the Gram once, LoGIK constructs the same Krylov candidate space and solves the reduced spectral problem without revisiting the $S \times D$ cache. We establish these three correspondences below.

Krylov update. Let $G \in \mathbb{R}^{D\times r}$ be the random starting block. The j-th token-space Krylov block satisfies

$$(AA^\top)^j AG = A(A^\top A)^j G = AM^j G. \tag{5}$$

Hence every token-space candidate can be represented implicitly as AY_D for a channel-space coefficient block Y_D, and the update $A(A^\top Y_S)$ reduces to $Y_D \leftarrow MY_D$. Once M is formed, additional Krylov blocks therefore require only channel-space computation.

Orthogonalization. The implicit representation must also preserve the geometry of the token-space candidates. Since

$$(AY_D)^\top (AY_D) = Y_D^\top M Y_D, \tag{6}$$

their inner products can be computed entirely in channel space using the M-weighted inner product. Let $LL^\top = Y_D^\top M Y_D$ be its Cholesky factorization and define $\mathrm{orth}_M(Y_D) = Y_D L^{-\top}$. Then

$$\mathrm{orth}_M(Y_D)^\top M\,\mathrm{orth}_M(Y_D) = L^{-1}(Y_D^\top M Y_D)L^{-\top} = I. \tag{7}$$

Thus, $A\,\mathrm{orth}_M(Y_D)$ is orthonormal in token space, allowing LoGIK to replace the tall token-space QR with Cholesky whitening in the much smaller channel space.

Final projection. Let Q_D denote the final M-orthonormal basis, so that $Q_D^\top M Q_D = I$. Its implicit token-space counterpart $Q_S = AQ_D$ is therefore orthonormal, and the projected matrix becomes

$$B = Q_S^\top A = Q_D^\top A^\top A = Q_D^\top M. \tag{8}$$

Table 1: Computational structure of three approaches to rank-r basis extraction, with $k = \dim \mathcal{K}_q \leq qr$ the Krylov dimension. EVD denotes eigenvalue decomposition.

Method	Token-axis work	Refinement state	Final spectral solve
Block Krylov	Repeated; grows with q	$S \times k$	$k \times D$ SVD
Gram + full EVD	One Gram construction	—	$D \times D$ EVD
LoGIK	One Gram construction	$D \times k$	$k \times k$ EVD

Thus, the final Rayleigh–Ritz projection is also computed entirely from the Gram matrix, without revisiting the cache.

4.1 Computational Payoff and Operating Regime

Table 1 summarizes the resulting cost structure. Block Krylov keeps the final spectral solve small, but each additional Krylov block revisits the $S \times D$ cache and expands an $S \times k$ token-space basis. LoGIK instead pays the token-axis cost once to form $M = A^\top A$; thereafter, refinement maintains only a $D \times k$ state and requires no further computation that scales with S.

Explicit Gram formation, however, is not always arithmetically cheaper. Considering only dense products with the cache, Block Krylov costs approximately $\mathcal{O}(qSDr)$, whereas forming the Gram costs $\mathcal{O}(SD^2)$, a ratio of roughly $D/(qr)$. LoGIK trades this one-time channel-correlation cost for eliminating repeated cache traversals and tall token-space orthogonalization. Once M is available, the remaining solver costs

$$\mathcal{O}\big(D^2k + Dk^2 + k^3\big), \qquad k \leq qr, \tag{9}$$

independently of S. Thus, increasing the Krylov depth improves the candidate subspace without proportionally increasing work on the cache.

A natural alternative is to form M once and diagonalize it directly (Gram+full EVD). This shares LoGIK's one-time token-axis cost, but solves a full $D \times D$ eigenproblem regardless of the requested rank. LoGIK instead restricts the search to a k-dimensional Krylov subspace and solves only the resulting $k \times k$ projected eigenproblem. Its reduced solve is therefore advantageous when $k \ll D$, while the benefit diminishes as k approaches D. We quantify this boundary with rank and channel-dimension sweeps in Section 6.4.

4.2 Low-Precision Acceleration of the Gram Bottleneck

After the channel-space reformulation, Gram construction is the only remaining operation whose cost scales with the token length S. As shown in Figure 2, removing the repeated token-space products and tall QR makes the FP32 Gram the dominant cost of LoGIK. We therefore accelerate this remaining bottleneck with a group-wise INT4 tensor-core kernel.

We use symmetric signed INT4 quantization with levels $[-7, 7]$. For each token group g and channel c, we set $s_{g,c} = \max_t |(A_g)_{tc}|/7$ and quantize

$$(Q_g)_{tc} = \mathrm{clip}_{[-7,7]}\left(\mathrm{round}\left(\frac{(A_g)_{tc}}{s_{g,c}}\right)\right). \tag{10}$$

The per-group, per-channel scale limits the influence of token-wise outliers while preserving a regular GEMM structure. We use groups of $S_g = 16{,}384$ tokens; this grouping also determines the quantization statistics analyzed in Section 5.

Let $D_g = \mathrm{diag}(\mathbf{s}_g)$. The resulting Gram approximation is

$$\widetilde{M} = \sum_g D_g(Q_g^\top Q_g)D_g \approx A^\top A. \tag{11}$$

The products $Q_g^\top Q_g$ are evaluated on INT4 tensor cores (NVFP4 on Blackwell, Appendix K), after which the scales are restored and the group contributions are accumulated in FP32. Low precision is therefore confined to Gram formation; all subsequent Krylov refinement operates on the resulting floating-point matrix.

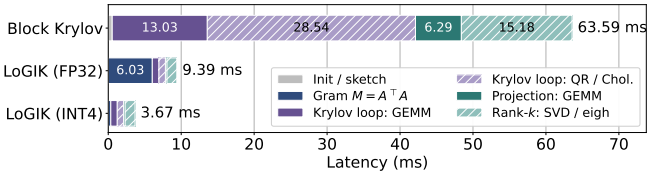

Figure 2: Per-stage latency of Block Krylov and LoGIK on the Llama-3.1-8B K state (32k tokens, rank 16, $q = 3$; separate run from Table 2). Channel-space refinement removes the token-axis stages, leaving the Gram, which the INT4 kernel then reduces.

In this setting, channel-space reformulation removes the dominant token-space GEMM and tall-QR costs, leaving the 6.03 ms FP32 Gram as the main bottleneck. The INT4 kernel reduces this stage to 0.34 ms and the total basis-extraction time from 9.39 ms to 3.67 ms.

5 When Is the Low-Precision Gram Reliable?

Further Krylov refinement can better solve the eigenproblem of $\widetilde{M}$, but cannot recover information lost during Gram formation. We therefore ask when the quantization error is small enough to preserve the retained subspace. Throughout, $\lambda_1 \geq \cdots \geq \lambda_D \geq 0$ are the eigenvalues of $M = A^\top A$, with $\lambda_i = \sigma_i^2$.

Modeling the Gram noise floor. Write the dequantized cache as $A+E$. The corresponding Gram perturbation is $\widetilde{M} - M = A^\top E + E^\top A + E^\top E$. Under the standard zero-mean rounding model, the cross terms largely cancel as they accumulate over tokens, whereas the squared quantization error in $E^\top E$ adds constructively. For a long cache, the resulting perturbation is therefore dominated by accumulated quantization energy.

Let ν denote the representative quantization noise-to-signal energy ratio. The total noise energy is approximately $\nu \ \mathrm{tr}(M)$; spreading this energy over the D channel directions gives

$$\phi = \nu \frac{\mathrm{tr}(M)}{D} = \frac{\nu \zeta \lambda_1}{D}, \qquad \zeta = \frac{\mathrm{tr}(M)}{\lambda_1} = \sum_i \left(\frac{\sigma_i}{\sigma_1}\right)^2. \tag{12}$$

Here, ζ is the stable rank of A, measuring how broadly the spectral energy is distributed relative to the dominant direction. We call ϕ the *Gram noise floor*: the background spectral level introduced by accumulated quantization noise, below which a direction is no longer distinguishable from the noise. Appendix E derives the floor in detail.

Stability of the retained subspace. For rank-r compression, the weakest direction that must remain distinguishable from the noise floor is the r-th eigen-direction, with $\lambda_r = \sigma_r^2$. Quantization perturbs the Gram eigenvectors by mixing retained directions with the discarded tail, and this mixing becomes large when the perturbation approaches the spectral strength of the weakest retained direction. The relevant comparison is therefore between λ_r and ϕ.

The perturbation analysis in Appendix E shows that, up to a tail-dependent factor, the excess reconstruction error grows with the ratio between these two scales. Requiring the retained edge to remain sufficiently above the noise floor gives $\lambda_r \gtrsim \kappa\phi$, where the safety margin κ is set by the target excess error and by the separation from the discarded spectrum. Substituting $\lambda_r/\lambda_1 = (\sigma_r/\sigma_1)^2$ and Equation (12) yields

$$\boxed{\frac{\sigma_1}{\sigma_r} \lesssim \sqrt{\frac{D}{\kappa \zeta \nu}}}. \tag{13}$$

The criterion makes the relevant trade-off explicit. A larger retained spectral ratio σ_1/σ_r means that the weakest retained direction lies closer to the quantization floor and is therefore more vulnerable. Higher quantization noise ν or broader spectral energy ζ raises that floor, while a larger channel dimension D spreads the accumulated noise over more directions. The relevant ratio is σ_1/σ_r, not the full condition number $\sigma_1/\sigma_{\min}$, because directions below rank r are discarded. On synthetic

spectra the INT4 failure boundary sits where the criterion places it and, unlike the FP32 boundary, does not move with the loop count (Figure 4 in Appendix E).

From the criterion to a precision decision. The criterion captures the dominant failure mode, but real quantization error can deviate from the independent-noise model. We therefore combine the retained spectral ratio with the realized Gram distortion:

$$\rho = \sqrt{\frac{w_1}{w_r}}, \qquad \epsilon_M = \frac{\|M_{\mathrm{INT4}} - M_{\mathrm{FP32}}\|_F}{\|M_{\mathrm{FP32}}\|_F}. \tag{14}$$

Here, $w_1 \geq \cdots \geq w_r$ are the leading Ritz values returned by LoGIK's reduced spectral solve, which approximate the leading Gram eigenvalues $\lambda_i = \sigma_i^2$. Thus, $\rho \approx \sigma_1/\sigma_r$ directly estimates the retained spectral ratio appearing in Equation (13), without requiring an exact SVD or an additional eigendecomposition. In contrast, ϵ_M measures the realized distortion of the INT4 Gram and catches structured quantization error not captured by the noise model.

We use INT4 when $\rho \leq 30$ and $\epsilon_M \leq 0.1$, and FP32 otherwise. The thresholds are selected empirically to separate the INT4-safe from the INT4-sensitive layer-states over the 184 measured across three models (Figure 6), rather than being implied directly by Equation (13). The resulting precision choice is cached for each model, layer-state, rank, and quantization configuration. Subsequent calls therefore bypass calibration and directly form the Gram at the cached precision before running LoGIK, so the FP32 reference Gram and the additional INT4 comparison are paid only once. Appendix F gives the threshold calibration and full procedure.

6 Experiments

We evaluate LoGIK along three axes: basis-extraction accuracy and latency, downstream KV-compression performance after replacing only the SVD engine, and scaling with token length, target rank, and channel dimension. We additionally evaluate the adaptive-precision policy to determine when the INT4 Gram can be used without compromising the retained subspace.

6.1 Setup

Basis extraction. We collect the K and V states of every layer from a 32,768-token NarrativeQA prompt for Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Qwen2.5-7B-Instruct (Qwen et al., 2025), yielding 184 layer-states. Each state is evaluated head-wise with $D = 128$ and target rank $r = 16$. For an orthonormal rank-r basis $\widehat{V}$ we report the excess reconstruction error $\mathrm{excess}(\widehat{V}) = \|A - A\widehat{V}\widehat{V}^\top\|_F^2 / \sum_{i>r} \lambda_i - 1$, using an FP64 SVD to compute the optimal rank-r residual. We compare exact SVD, power iteration, simultaneous iteration, randomized SVD, SRHT (Tropp, 2011), Block Krylov, Gram+EVD, and LoGIK; randomized iterative methods use three seeds and one to five Krylov blocks. Timing is measured on a single A100-80GB GPU with all heads of layer 15 batched together. These measurements isolate basis extraction; projection and cache quantization are included only in the pipeline experiments.

Downstream compression. We replace only the SVD engine in GEAR and SVDq, leaving the remaining compression pipeline unchanged. GEAR quantizes the residual after subtracting a low-rank approximation of K and V, whereas SVDq rotates K into a singular-vector basis and quantizes latent channels with nonuniform bit widths. We evaluate Llama3-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen2-7B-Instruct (Yang et al., 2024) on GSM8K (Cobbe et al., 2021), AQuA (Ling et al., 2017), BBH (Suzgun et al., 2023), and the six LongBench (Bai et al., 2024) categories using the settings of the original pipelines. In GEAR, all SVD-based engines share the same rank, block count, and residual quantizer; in SVDq, they share the same bit-allocation schedule and retained latent channels. Reported latency measures KV compression only and excludes shared attention and MLP computation.

6.2 Basis-Extraction Time–Accuracy Trade-off

LoGIK shifts the basis-extraction time–accuracy frontier in two stages (Figure 1 top, Table 2). With an FP32 Gram, it reaches 1% excess in 10.68 ms, compared with 42.55–69.08 ms for the token-space

Table 2: SVD approximation methods on the Llama-3.1-8B K state at 32k tokens, rank 16. Memory traffic counts reads of $\mathbf{A}$ and q is the loop count; blank cells never reach 1%. Bold is best in a column, underline second.

Method	Memory traffic	Extra memory	Time to 1% (ms)	Lowest excess	Time to Lowest (ms)
exact SVD	full decomposition	$O(SD)$	67.96	**0**	67.96
Power iteration	$2q$	$S \times r$		2.50%	31.04
Simultaneous iteration	$2q$	$S \times r$		1.10%	65.19
RSVD	$2q + 1$	$S \times r$	69.08	0.095%	100.12
SRHT	2	$O(SD)$	42.55	0.585%	42.55
Block Krylov	$2q$	$S \times rq$	63.76	0.013%	104.96
Gram + eigh (INT4, CPU)	1	$D \times D$	<u>8.21</u>	<u>0.005%</u>	<u>8.21</u>
Gram + eigh (FP32, CPU)	1	$D \times D$	13.68	**0**	13.68
LoGIK (FP32)	1	$D^2 + D \times rq$	10.68	<u>0.005%</u>	13.87
LoGIK (INT4)	1	$D^2 + D \times rq$	**4.13**	0.012%	**6.65**

methods that reach this target, isolating the gain from the channel-space reformulation. Replacing only Gram construction with INT4 further reduces this time to 4.13 ms. At five Krylov blocks, INT4 LoGIK reaches 0.012% excess in 6.65 ms, closely matching Block Krylov's 0.013% in 104.96 ms at $15.78\times$ lower latency. The same operating point is $10.22\times$ faster than the 67.96 ms exact SVD.

Gram+EVD provides the complementary operating point. Solving the full $D \times D$ eigenproblem yields lower error (INT4 Gram+EVD reaches 0.005% excess in 8.21 ms, and FP32 reaches the optimal residual in 13.68 ms), but pays for a full spectral solve regardless of the requested rank. LoGIK instead solves only the $k \times k$ projected Krylov problem, giving lower latency when $k \ll D$; this advantage diminishes as k approaches D, consistent with Section 4.1.

Stage-level timing supports this decomposition: the channel-space reformulation removes the token-space products and tall QR, leaving Gram construction as the dominant FP32 stage, while INT4 removes most of this remaining cost. Because these timings come from a separate run, we use them only to compare the relative cost structure.

6.3 Downstream KV-Compression Results

GEAR. With the remaining GEAR pipeline fixed, LoGIK achieves a chain-of-thought (Wei et al., 2022) average of 60.62 and a LongBench average of 40.93 on Llama3-8B, compared with 58.63 and 40.69 for the original power-iteration engine (Table 3 top, Figure 1 bottom). At 32k tokens, it reduces compression latency by $3.58\times$, $3.60\times$, and $4.39\times$ on Llama, Mistral, and Qwen, respectively, without changing the compressed KV representation.

SVDq. SVDq provides a complementary test because its nonuniform bit schedule retains only a subset of the singular directions. Under the evaluated $\bar{b} = 2$ schedule $(8, 4, 4, 0, 0, 0, 0, 0)$, only 384 of the $D = 1024$ latent channels in Llama3-8B receive nonzero bits. Since the zero-bit channels contribute nothing to reconstruction, directly computing these 384 leading directions is equivalent to forming the full basis and discarding the remainder; the bit allocation and compressed representation are unchanged.

With this substitution, LoGIK reduces per-layer compression latency to 28.5 ms, versus 146.2 ms for exact SVD and 97.5 ms for RSVD, corresponding to $5.1\times$ and $3.4\times$ reductions. Its LongBench average remains comparable at 41.75, versus 41.67 for exact SVD and 41.35–41.57 for the iterative baselines. These small score differences should not be interpreted as evidence that an approximate basis is intrinsically superior to exact SVD; rather, LoGIK preserves SVDq's downstream quality while reducing the cost of extracting the directions that survive quantization.

6.4 Scaling and the Operating Regime

The scaling sweeps match the operating regime predicted in Section 4.1. As token length grows from 1k to 32k, INT4 LoGIK increases from 3.61 to 4.30 ms to reach 1% excess, whereas exact SVD grows from 32.53 to 66.88 ms. Increasing the target rank from 2 to 32 raises LoGIK from 2.65 to 7.03 ms as the reduced Krylov space grows, while Gram+EVD remains near 9–10 ms because it always diagonalizes the full 128×128 Gram. The channel-dimension sweep shows the complementary boundary: increasing D from 64 to 256 raises full Gram diagonalization from 2.80 to 39.6 ms, while LoGIK remains near 5.33 ms in this experiment; full EVD is faster at $D = 64$, but becomes more expensive as D grows.

Table 3: Downstream accuracy (%) on Llama3-8B when only the SVD engine changes: GEAR (top; power iteration is the original GEAR) and SVDq at $\bar{b} = 2$ with compression latency (bottom; exact SVD reproduces the original SVDq). Bold is best excluding FP and exact SVD, underline second. LoGIK leads both, at the lowest latency.

	Chain of Thought				LongBench						
Method	GSM8K	AQuA	BBH	Avg.	Single	Multi	Sum.	Few-shot	Synth.	Code	Avg.
FP	78.01	50.79	58.62	62.47	29.93	34.89	24.10	58.91	52.69	55.09	42.60
KCVT-3bit	58.23	35.43	43.70	45.79	16.50	15.19	20.56	48.47	4.77	40.70	24.37
KIVI-2bit	58.38	30.31	38.10	42.26	21.22	19.22	20.66	52.97	7.94	40.86	27.15
PALU-50%	71.87	43.70	54.32	56.63	**30.79**	32.68	22.80	57.76	45.99	53.79	40.64
xKV	76.57	**50.00**	54.10	60.22	30.02	32.61	23.44	57.22	47.67	52.02	40.50
Power iteration	75.89	47.64	52.35	58.63	29.24	32.51	23.53	57.85	47.06	**53.97**	40.69
SRHT	74.30	47.64	51.16	57.70	26.45	29.22	23.48	57.86	45.57	46.95	38.26
RSVD	73.31	44.09	52.10	56.50	26.25	28.66	23.21	58.08	45.63	47.72	38.26
Block Krylov	72.55	49.21	50.47	57.41	26.02	29.09	**23.59**	**58.25**	45.26	46.86	38.18
LoGIK	**77.48**	**50.00**	**54.38**	**60.62**	30.06	**33.09**	23.53	57.85	**48.03**	53.04	**40.93**

Method	GSM8K	Single	Multi	Sum.	Few-shot	Synth.	Code	LongBench avg.	Latency (ms)
FP	78.01	29.93	34.89	24.10	58.91	52.69	55.09	42.60	–
SVDq + Exact SVD	78.17	28.03	32.59	24.05	58.31	52.00	55.06	41.67	146.2
SVDq + Power iteration	**78.77**	27.31	32.47	24.05	58.11	51.46	54.69	41.35	52.9
SVDq + SRHT	77.63	27.17	32.39	23.73	57.96	51.76	54.59	41.27	56.2
SVDq + RSVD	78.24	27.52	33.08	23.87	58.16	51.72	**55.05**	41.57	97.5
SVDq + Block Krylov	78.01	**27.79**	32.44	**24.07**	58.05	51.71	54.95	41.50	143.0
SVDq + LoGIK	78.24	27.77	**33.48**	23.99	**58.37**	**51.90**	55.01	**41.75**	**28.5**

Together, these results show that LoGIK is most effective when the requested Krylov space remains well below the full channel space. The weak measured dependence on S and D over these ranges should not be interpreted as asymptotic independence: Gram formation remains $\mathcal{O}(SD^2)$, and the channel-space refinement retains the dimension dependence in Equation (9).

6.5 Effect of the Precision Decision

We isolate the effect of the precision policy by comparing fixed INT4, fixed FP32, and adaptively selected precision across the 92 K states. We then test the stability of the resulting decisions by recomputing the detector statistics on two additional prompt datasets.

The policy assigns INT4 to 88 of the 92 K states and retains FP32 for four Qwen states. On the INT4-selected states, the INT4 and FP32 time–error curves closely agree, while INT4 runs more than twice as fast. The four FP32 fallbacks coincide with the states where fixed INT4 degrades: using INT4 everywhere leaves the five-block Qwen K excess at 1.90%, whereas selectively restoring FP32 reduces it to 0.08% at roughly half the latency of an FP32-only path. Two Qwen K layers remain inaccurate even with an FP32 Gram, confirming that the policy targets precision-induced failures rather than finite-Krylov error.

To test stability across prompts, we recompute the decision statistics for all 184 K/V layer-states using GovReport and RepoBench-P. GovReport reproduces all NarrativeQA decisions, while RepoBench-P changes only one additional state from INT4 to FP32. This measures agreement after recalibration on the new prompt distribution, rather than invariance of a frozen decision under arbitrary distribution shift.

7 Conclusion

We present LoGIK, which moves Block Krylov refinement for tall KV caches from the token dimension into the channel Gram space. This removes repeated cache access and tall orthogonalization, while an INT4 Gram kernel further reduces the remaining token-dependent cost. A selective precision policy retains FP32 only when Gram quantization is unreliable. On a 32k-token Llama-3.1-8B K cache, LoGIK matches Block Krylov accuracy $15.78\times$ faster and exact SVD $10.22\times$ faster, while reducing GEAR compression latency by up to $4.39\times$.

AI USAGE STATEMENT

We used generative AI tools to refine material we had already produced: assisting with translation, refining hypotheses, and providing feedback on our research methodology and experiments. We wrote the methodology, the manuscript draft in this paper ourselves. We reviewed every suggestion before adopting it and take responsibility for all claims, numbers, and artifacts in this paper.

REFERENCES

Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, et al. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*, 2025.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Marc Baboulin, Simplice Donfack, Oguz Kaya, Theo Mary, and Matthieu Robeyns. Mixed precision randomized low-rank approximation with gpu tensor cores. In *European Conference on Parallel Processing*, pp. 31–44. Springer, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed Abdelfattah, and Kai-Chiang Wu. Palu: Kv-cache compression with low-rank projection. In *International Conference on Learning Representations*, volume 2025, pp. 50222–50249, 2025a.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Hung-Yueh Chiang, Yash Akhauri, Xilai Dai, Huiqiang Jiang, Yucheng Li, Luis Ceze, Kai-Chiang Wu, et al. xkv: Cross-layer kv-cache compression via aligned singular vector extraction. *arXiv preprint arXiv:2503.18893*, 2025b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

Jed A Duersch, Meiyue Shao, Chao Yang, and Ming Gu. A robust and efficient implementation of lobpcg. *SIAM Journal on Scientific Computing*, 40(5):C655–C676, 2018.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Vage Egiazarian, Roberto Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, et al. Bridging the gap between promise and performance for microscaling fp4 quantization. In *International Conference on Learning Representations*, volume 2026, pp. 113529–113563, 2026.

Ethan N. Epperly. Low-rank approximation toolbox: The gram correspondence. Blog post, `https://www.ethanepperly.com/index.php/2024/12/07/low-rank-approximation-toolbox-the-gram-correspondence/`, December 2024. Accessed: 2026-09-01.

Takeshi Fukaya, Yuji Nakatsukasa, Yuka Yanagisawa, and Yusaku Yamamoto. Choleskyqr2: a simple and communication-avoiding algorithm for computing a tall-skinny qr factorization on a large-scale parallel system. In *2014 5th workshop on latest advances in scalable algorithms for large-scale systems*, pp. 31–38. IEEE, 2014.

Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2 (2):205–224, 1965.

Gene H Golub, Franklin T Luk, and Michael L Overton. A block lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software (TOMS)*, 7(2):149–169, 1981.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53 (2):217–288, 2011.

Mark Hoemmen et al. *Communication-avoiding Krylov subspace methods.* PhD thesis, University of California, Berkeley, USA, 2010.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*, 2024.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 581–588, 2013.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 158–167, 2017.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In *International Conference on Machine Learning (ICML)*, 2024.

Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. *Advances in neural information processing systems*, 28, 2015.

Yuji Nakatsukasa. Fast and stable randomized low-rank matrix approximation. *arXiv preprint arXiv:2009.11392*, 2020.

NVIDIA Corporation. cuSOLVER library documentation. `https://docs.nvidia.com/cuda/cusolver/`, 2025. Accessed: 2026-08-10.

Hiroyuki Ootomo and Rio Yokota. Mixed-precision random projection for randnla on tensor cores. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pp. 1–11, 2023.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 57355–57373, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011.

Joel A Tropp and Robert J Webber. Randomized algorithms for low-rank matrix approximation: Design, analysis, and applications. *arXiv preprint arXiv:2306.12418*, 2023.

Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 38(4): 1454–1485, 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Yusaku Yamamoto, Yuji Nakatsukasa, Yuka Yanagisawa, and Takeshi Fukaya. Roundoff error analysis of the choleskyqr2 algorithm. *Electron. Trans. Numer. Anal*, 44(01):306–326, 2015.

Ichitaro Yamazaki, Stanimire Tomov, and Jack Dongarra. Mixed-precision cholesky qr factorization and its case studies on multicore cpu with multiple gpus. *SIAM Journal on Scientific Computing*, 37(3):C307–C330, 2015.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Hong Yankun, Li Xing, Zhen Hui-Ling, Yu Xianzhi, Liu Wulong, and Yuan Mingxuan. Svdq: 1.25-bit and 410x key cache compression for llm attention. *arXiv preprint arXiv:2502.15304*, 2025.

Yi Yu, Tengyao Wang, and Richard J Samworth. A useful variant of the davis–kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. In *International Conference on Learning Representations*, volume 2026, pp. 56418–56439, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

A LIMITATIONS

Adaptive precision addresses Gram-quantization error, but not failures of the finite Krylov approximation; sink-dominated Qwen K layers 0 and 27 retain 1.52% and 0.52% excess even with an FP32 Gram (Appendix G). The speedup of the 4-bit Gram requires 4-bit tensor cores, whose format differs by GPU generation: INT4 on Ampere and Ada, NVFP4 on Blackwell (Appendix K). Hopper supports neither, so there the Gram runs in INT8, which widens the applicable range (Table 6) but gives a smaller speedup. Downstream evaluation covers GSM8K, AQuA, BBH, and LongBench and excludes synthetic long-context benchmarks such as RULER (Hsieh et al., 2024).

B TWO-PASS FULL SVD

Even when $\mathbf{U}$ and $\mathbf{\Sigma}$ are also required, the only additional cost for LoGIK is one more product along the token axis. Forming $\mathbf{W} = \mathbf{AV}$ gives $\mathbf{W}^T\mathbf{W} = \mathbf{V}^T\mathbf{MV}$, so the Gram of $\mathbf{W}$ comes out without further computation on $\mathbf{A}$, and orthogonalizing $\mathbf{W}$ with that $r \times r$ Cholesky QR yields $\mathbf{U}$ (Algorithm 3). One product with $\mathbf{A}$ therefore suffices when only $\mathbf{V}$ is needed, and two give the full SVD. Any method that outputs an $S \times r$ matrix $\mathbf{U}$ must apply $\mathbf{A}$ at least once, so two are a lower bound.

Algorithm 3 Two-pass full SVD

Require: $\mathbf{A}$, the outputs $\mathbf{V}$ and $\mathbf{M}$ of LoGIK
Ensure: $\mathbf{U} \in \mathbb{R}^{S\times r}$, $\mathbf{s} \in \mathbb{R}^r$, $\mathbf{V} \in \mathbb{R}^{D\times r}$

1: $\mathbf{W} \leftarrow \mathbf{AV}$ ▷ second and last token-axis pass
2: $\mathbf{LL}^T \leftarrow \mathrm{cholesky}(\mathbf{V}^T\mathbf{MV})$ ▷ Gram of $\mathbf{W}$, without $\mathbf{A}$
3: $\mathbf{U} \leftarrow \mathbf{WL}^{-T}$ ▷ Cholesky QR
4: $\mathbf{U}_r, \mathbf{s}, \mathbf{V}_r \leftarrow \mathrm{svd}(\mathbf{L}^T)$ ▷ $r \times r$
5: **return** $\mathbf{UU}_r$, $\mathbf{s}$, $\mathbf{VV}_r$

C INCREMENTAL GRAM

Incremental updates during decoding follow directly from the Gram reformulation. Since $\mathbf{M} = \sum_t \mathbf{a}_t\mathbf{a}_t^T$ is a sum of per-token outer products, a new token $\mathbf{a}$ updates it as $\mathbf{M} \leftarrow \mathbf{M} + \mathbf{aa}^T$ in $O(D^2)$, and the basis is re-extracted from the updated $\mathbf{M}$ without further computation on the cache. On the 4-bit path the channel scales inside a group are fixed once the group is full, so an in-progress group accumulates as an fp16 partial Gram and is committed as a 4-bit partial Gram once the group fills. Token-space methods must rebuild an $S \times rq$ basis for every new token, so they cannot support this update.

D COMPARISON WITH GRAM + EIGH

Gram + eigh cannot go below roughly 8ms on Llama, and 60 of the 86 frontier points belong to LoGIK. This section shows that both facts have the same cause, the size of the final eigendecomposition.

Accuracy and time. Gram + eigh, which forms $\mathbf{M}$ and finishes with $\mathrm{eigh}(\mathbf{M})$ without iterating, beats LoGIK on accuracy. The eigenvectors of an fp32 Gram are eigenvectors of the exact $\mathbf{M}$, so its excess error sits at the same Eckart–Young floor as the exact SVD in all nine panels, and LoGIK never reaches that floor no matter how large the loop count. In exchange, its time has the floor of a $D \times D$ eigendecomposition. Each method thus holds one floor, Gram + eigh on accuracy and LoGIK on time.

Where the time floor comes from. The floor comes from the batched eigendecomposition of cuSOLVER breaking off at size 32. As Table 4 shows, sizes up to 32 run in a single batched Jacobi kernel at 0.23ms for 8 heads, but from 33 on each matrix runs separately and the time jumps to 6.45ms. The CPU round trip instead grows smoothly with size and is faster than the GPU path from 33 on, though it still costs about 7ms at 128. Gram + eigh, which must always pay for a 128×128 eigendecomposition, therefore cannot go below roughly 8ms on Llama.

Table 4: Time (ms) of a symmetric eigendecomposition on a batch of 8 matrices. GPU is the batched cuSOLVER (NVIDIA Corporation, 2025) path and CPU is the round trip including transfers.

Matrix size	8	16	32	33	48	64	128	256
GPU	0.12	0.14	0.23	6.45	7.79	8.02	18.91	53.29
CPU round trip	0.12	0.19	0.82	0.89	1.33	1.73	6.98	29.14

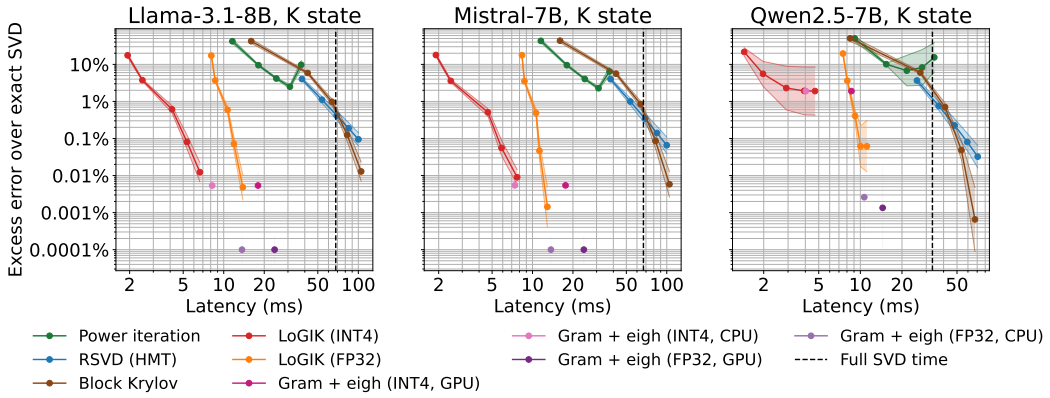

Figure 3: Time versus excess error on the K states of three models, including the four variants of Gram + eigh (Gram precision INT4/fp32, eigendecomposition placed on GPU/CPU).

Where the two diverge. The final eigendecomposition of LoGIK has size rq, so it stays on the batched Jacobi path whenever rq is at most 32, and in this low-budget regime the gap widens several-fold. When rq grows to 80, as at loop 5, the advantage thins to 1.56ms on Llama and reverses on Mistral, where LoGIK takes 7.69ms against 7.39ms. The 86 frontier points across the nine panels of three models therefore split into 60 for the LoGIK family, 23 for Gram + eigh, and 3 for the exact SVD. The LoGIK family enters as three curves, taking 37 points with the automatic precision, 16 with INT4, and 7 with fp32. On the panels where the detector sends every layer to INT4 those two curves are two runs of the same computation, so the 37 and the 16 divide by run-to-run noise and only their sum is meaningful. Figure 3 draws the K panel of each model, three of the nine panels in all. The rank and channel sweeps of Appendix J show this boundary moving with the relative size of rq and D.

E Supplement to the Gram perturbation analysis

The three approximations of the main text follow from two properties of rounding noise, diagonal dominance and a channel average of $1/D$. This section writes out that derivation, proves an upper bound of the same form for an arbitrary perturbation, and confirms the size of the floor, the value $\kappa = 8$, and the extrapolation on synthetic matrices. The measurements add two further results. The condition number $\sigma_1/\sigma_{\min}$ does not enter the sensitivity, and the limit of fp32 is set by the iteration rather than by the Gram.

Synthetic matrices. The measurements use 32768×128 synthetic matrices with prescribed singular values. To vary the ratio σ_1/σ_r that appears in the condition separately from the condition number $\sigma_1/\sigma_{\min}$, we build three families. The tail family keeps the top 16 flat and makes only the discarded tail steep, which raises $\sigma_1/\sigma_{\min}$ alone; the power family sweeps σ_1/σ_{16} itself; and the spike family lifts only the first direction and leaves the rest flat, which emulates an attention sink.

Deriving the noise floor. The main text writes the Gram perturbation as $\mathbf{\Delta} = \mathbf{A}^T\mathbf{E} + \mathbf{E}^T\mathbf{A} + \mathbf{E}^T\mathbf{E}$ and summarizes the quantization statistics by a single noise-to-signal energy ratio ν. For token group g and channel c, with $s_{g,c}$ the quantization step of Equation (10) and $a_{g,c}$ the RMS magnitude of the corresponding entries of $\mathbf{A}$, the local ratio is

$$\nu_{g,c} = \frac{s_{g,c}^2}{12\, a_{g,c}^2} = \frac{1}{12}\left(\frac{\max_t |(\mathbf{A}_g)_{tc}|}{7\, a_{g,c}}\right)^2, \tag{15}$$

and ν is a representative value of $\nu_{g,c}$. Quantization noise therefore depends not only on the bit width but on the maximum-to-RMS ratio inside each group. Under this model the perturbation collapses to its diagonal,

$$\mathbf{\Delta} \approx \nu \, \mathrm{diag}(\mathbf{M}), \tag{16}$$

because, among the components of $\mathbf{\Delta}$, only the diagonal of $\mathbf{E}^T\mathbf{E}$ grows in proportion to S. We assume the rounding noise E_{sc} of channel c has mean 0 and variance $s_c^2/12$ and is independent of the data and across channels. The expectation of the diagonal $(\mathbf{E}^T\mathbf{E})_{cc} = \sum_s E_{sc}^2$ is $Ss_c^2/12 = \nu \, Sa_c^2 = \nu \, M_{cc}$. The off-diagonal $(\mathbf{E}^T\mathbf{E})_{cc'} = \sum_s E_{sc}E_{sc'}$ has mean 0 and standard deviation $\sqrt{S}\, s_c s_{c'}/12$ and is therefore of order $S^{-1/2}$ relative to the diagonal, while the cross term $(\mathbf{A}^T\mathbf{E})_{cc'} = \sum_s A_{sc}E_{sc'}$ has mean 0 and standard deviation $\sqrt{S}\, a_c s_{c'}/\sqrt{12}$ and is of order $\sqrt{\nu/S}$ relative to M_{cc}. At 32k tokens both orders are below 10^{-2}, so Equation (16) holds.

Equation (12) applies first-order perturbation to this. The shift of the eigenvalue λ_i is $\delta\lambda_i = \mathbf{v}_i^T\mathbf{\Delta}\mathbf{v}_i = \nu \sum_c M_{cc}v_{ic}^2$, and if the tail eigenvectors are not aligned with the channel axes the channel average of v_{ic}^2 is $1/D$, so $\delta\lambda_i \approx \nu \, \mathrm{tr}(\mathbf{M})/D$. Substituting $\mathrm{tr}(\mathbf{M}) = \sum_i \lambda_i = \zeta\lambda_1$ gives ϕ. The leading eigenvalues move by the same ϕ, but this is negligible as long as $\lambda_r \gg \phi$.

Deriving the second-order term. The main text states that, up to a tail-dependent factor, the excess error grows with the ratio between the noise floor and the retained edge,

$$\text{excess} \;\propto\; \left(\frac{\phi}{\lambda_r}\right)^2. \tag{17}$$

Equation (17) counts the energy lost when a leading direction mixes with a tail direction. A leading direction $\mathbf{v}_i$ ($i \le r$) mixes with a tail direction $\mathbf{v}_j$ ($j > r$) by an angle $\theta_{ij} \approx \Delta_{ji}/(\lambda_i - \lambda_j)$ with $\Delta_{ji} = \mathbf{v}_j^T\mathbf{\Delta}\mathbf{v}_i$, and the residual energy then grows by $(\lambda_i - \lambda_j)\theta_{ij}^2$ by the same calculation as in the proof of Proposition 1. The energy lost is therefore $\sum_{i\le r}\sum_{j>r}\Delta_{ji}^2/(\lambda_i - \lambda_j)$. Here the isotropic part of $\mathbf{\Delta}$ contributes only to Δ_{ii}, so $\sum_{j>r}\Delta_{ji}^2 \le \|(\mathbf{\Delta} - \phi\mathbf{I})\mathbf{v}_i\|^2 = \nu^2\sum_c(M_{cc} - \mathrm{tr}(\mathbf{M})/D)^2 v_{ic}^2$, which is of order ϕ^2 once the spread of the channel energies is a constant fraction of their mean; on the steep spectra where the boundary lies $\lambda_j \ll \lambda_r$, so the gap is of order λ_r and the $i = r$ term dominates the sum. The energy lost is thus of order ϕ^2/λ_r, and dividing it by the optimal residual $\sum_{j>r}\lambda_j$ gives the excess error

$$\text{excess} \approx \left(\frac{\phi}{\lambda_r}\right)^2 \cdot \frac{\lambda_r}{\sum_{j>r}\lambda_j}. \tag{18}$$

The trailing factor is a spectral factor set by the shape of the tail and is constant once the data are fixed; Equation (17) is the leading factor of this expression, and the trailing factor is absorbed into κ together with the error tolerance. The bound below has the same structure: the perturbation divided by the gap, squared, times a spectral factor.

A rigorous upper bound. Write the top r eigenvectors as $\mathbf{V}_r$, the top r eigen-subspace of the perturbed $\tilde{\mathbf{M}} = \mathbf{M} + \mathbf{\Delta}$ as $\tilde{\mathbf{V}}_r$, and the excess error as

$$\mathrm{excess}(\tilde{\mathbf{V}}_r) = \frac{\left\|\mathbf{A} - \mathbf{A}\tilde{\mathbf{V}}_r\tilde{\mathbf{V}}_r^T\right\|_F^2}{\sum_{i>r}\lambda_i} - 1. \tag{19}$$

Proposition 1. *If $\lambda_r > \lambda_{r+1}$, then*

$$\mathrm{excess}(\tilde{\mathbf{V}}_r) \le \frac{4\,\|\mathbf{\Delta}\|_F^2}{(\lambda_r - \lambda_{r+1})^2} \cdot \frac{\lambda_1 - \lambda_D}{\sum_{i>r}\lambda_i}. \tag{20}$$

Proof. Let $\mathbf{\Theta}$ be the principal angles between $\mathbf{V}_r$ and $\tilde{\mathbf{V}}_r$. A variant of the Davis–Kahan theorem (Davis & Kahan, 1970; Yu et al., 2015) gives $\|\sin\mathbf{\Theta}\|_F \le 2\,\|\mathbf{\Delta}\|_F/(\lambda_r - \lambda_{r+1})$. On the other hand, for any orthonormal basis $\tilde{\mathbf{V}}$ we have $\left\|\mathbf{A} - \mathbf{A}\tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\right\|_F^2 = \mathrm{tr}(\mathbf{M}) - \mathrm{tr}(\tilde{\mathbf{V}}^T\mathbf{M}\tilde{\mathbf{V}})$, and setting $c_i = \|\mathbf{v}_i^T\tilde{\mathbf{V}}\|^2 \in [0,1]$ with $\sum_i c_i = r$ gives $\mathrm{tr}(\tilde{\mathbf{V}}^T\mathbf{M}\tilde{\mathbf{V}}) = \sum_i \lambda_i c_i$. The difference from the

Table 5: The 4-bit Gram noise floor on 32768×128 synthetic matrices with prescribed σ_1/σ_{16}. Here ϕ is the mean tail eigenvalue rise actually measured with the INT4 Gram, and the excess is measured from the top 16 eigenvectors of the INT4 Gram.

σ_1/σ_{16}	ϕ/λ_1	λ_{16}/ϕ	INT4 excess
4	1.2×10^{-3}	51	0.007%
8	5.5×10^{-4}	28	0.036%
16	3.7×10^{-4}	10.5	0.50%
21	3.4×10^{-4}	6.6	1.9%
32	3.0×10^{-4}	3.3	18%
64	2.4×10^{-4}	1.0	98%

Table 6: Predicted and measured applicability range of the 4-bit Gram as a function of precision and channel dimension. The measurement is the interval of σ_1/σ_{16} over which the loop 3 excess crosses 1% on power family synthetic matrices, INT8 rounds to 8 bits with the same group-channel scales, and NVFP4 uses the block-scaled format of Appendix K.

Setting	What changes	Predicted limit	Measured limit
INT4, $D = 128$	reference	about 20	between 16 and 32
INT4, $D = 64$	floor $\times 2$	about 14	between 11 and 16
INT4, $D = 256$	floor $\times 1/2$	about 28	between 23 and 32
INT8, $D = 128$	ν shrinks 330$\times$	about 360	between 256 and 338
NVFP4, $D = 128$	ν shrinks 3.3$\times$	about 36	between 32 and 64

optimal basis is $\sum_{i \le r} \lambda_i (1 - c_i) - \sum_{i > r} \lambda_i c_i$, and since $\sum_{i \le r} (1 - c_i) = \sum_{i > r} c_i = \|\sin \mathbf{\Theta}\|_F^2$, this difference is at most $(\lambda_1 - \lambda_D) \|\sin \mathbf{\Theta}\|_F^2$. Combining the two inequalities yields the claim. □

This proposition gives the shape of the condition. Writing the gap as $\lambda_r - \lambda_{r+1} = g\lambda_r$, the right-hand side is $(4/g^2)(\|\mathbf{\Delta}\|_F / \lambda_r)^2$ times the spectral factor $(\lambda_1 - \lambda_D)/\sum_{i>r} \lambda_i$, so the excess error falls with the square of the perturbation ratio $\|\mathbf{\Delta}\|_F / \lambda_r$. The quantities that enter the sensitivity are λ_r and $\mathbf{\Delta}$. The condition number $\sigma_1/\sigma_{\min}$ enters only through the spectral factor $(\lambda_1 - \lambda_D)/\sum_{i>r} \lambda_i$: were the tail flat at $\lambda_{\min}$, that factor would be $(\sigma_1/\sigma_{\min})^2/(D - r)$. The tails of real caches and of our synthetic families decay instead, so the factor is set by λ_{r+1} and the condition number drops out. The constants, however, are loose. Davis–Kahan counts even the rotation between λ_r and λ_{r+1} as subspace error, and $\|\mathbf{\Delta}\|_F$ counts even the identity component that rotates no eigenvector. We therefore use this proposition to choose the variables of the condition, and set the constants in the main text by resolving the structure of the perturbation.

Measuring the floor and extrapolating. The synthetic matrices reproduce both the size of the floor and κ. On the power family with prescribed σ_1/σ_{16}, measuring ν from Equation (15) gives 0.029 to 0.030, over 90% of $\mathbf{\Delta}$ is $\mathbf{E}^T\mathbf{E}$, and over 98% of that is diagonal, so Equation (16) holds. The deviation is what rotates. Write $\mathbf{\Delta} \approx \nu \operatorname{diag}(\mathbf{M})$ as $\phi\mathbf{I} + (\mathbf{\Delta} - \phi\mathbf{I})$. The first term shifts every eigenvalue alike and leaves the eigenvectors in place, so the mixing in Equation (17) is produced entirely by the second. Its entries carry the spread of the channel energies M_{cc} about their mean, and they are themselves of order ϕ once that spread is a constant fraction of the mean. In Table 5 the tail eigenvalue rise ϕ actually measured with the INT4 Gram sits where Equation (12) predicts, at $3 \times 10^{-4}\,\lambda_1$, and the excess error rises as λ_{16}/ϕ shrinks, crossing 1% at $\lambda_{16}/\phi \approx 8$. This is the $\kappa = 8$ of the main text. The basis in that table is the top 16 eigenvectors of the INT4 Gram, so its excess is the floor the iteration approaches rather than a bound on it; an iteration on the same noisy Gram is not held to those eigenvectors and can land slightly below, as at $\sigma_1/\sigma_{16} = 16$ in Figure 4.

Equation (13) also predicts where the limit moves when the precision and the channel dimension change. The floor ϕ is inversely proportional to D, so doubling D raises the limit by $\sqrt{2}$, and INT8 shrinks the step 18$\times$, which makes ν about 330 times smaller and raises the limit 18$\times$. NVFP4 has no uniform step, so we measure its ν directly on the same matrices, as the energy of the rounding error relative to that of the signal. The measured ν is 0.0090 against 0.029 to 0.030 for INT4, 3.3 times smaller, which raises the limit 1.8$\times$. In Table 6 all four predictions either fall inside the

Table 7: Loop 3 excess error and the condition number of the Krylov block Gram as a function of σ_1/σ_{16} on power family synthetic matrices.

σ_1/σ_{16}	fp32 excess	INT4 excess	Block Gram cond.
2	0.400%	0.419%	1.5×10^2
8	0.102%	0.161%	7.1×10^3
16	0.026%	0.522%	3.4×10^4
32	0.006%	17.14%	1.5×10^5
64	0.001%	112.9%	8.1×10^5
128	0.001%	374.7%	2.1×10^6
194	0.002%	756.0%	1.6×10^{31}
256	0.014%	1133%	1.6×10^{31}
512	0.989%	3131%	1.6×10^{31}
1024	11.39%	9610%	1.8×10^{31}

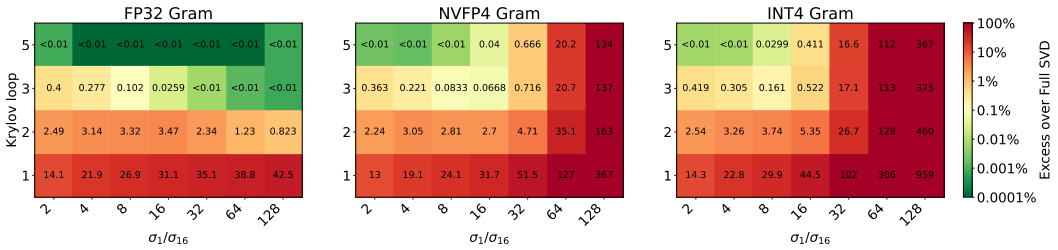

Figure 4: Excess error by loop count and spectral ratio on synthetic matrices with prescribed σ_1/σ_{16}. The panels use an fp32, NVFP4, and INT4 Gram from left to right and share one color scale. The NVFP4 panel is measured on an RTX PRO 6000 Blackwell, the other two on an A100. The fp32 boundary moves with the loop count, while the two 4-bit boundaries stay fixed, the NVFP4 boundary one grid step above the INT4 boundary.

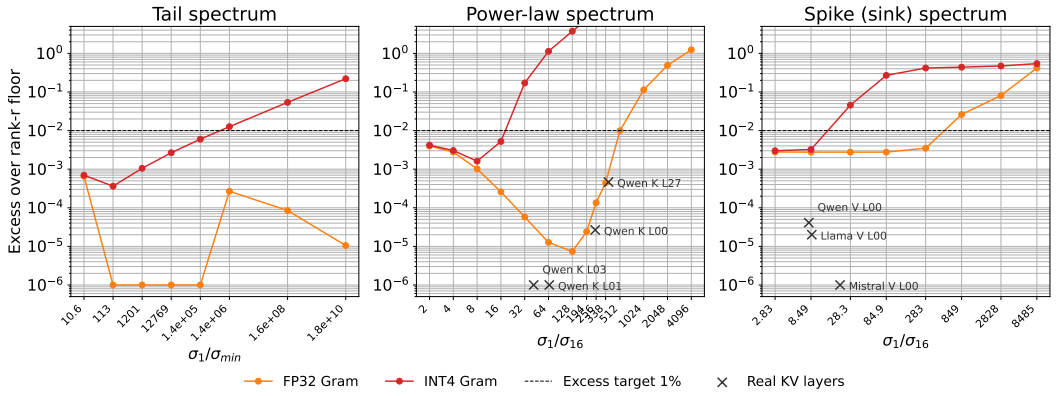

Figure 5: Failure boundaries of the fp32 and INT4 Gram, seen through the loop 3 excess error, on the tail, power, and spike families of synthetic matrices. Only the tail family uses $\sigma_1/\sigma_{\min}$ on the horizontal axis while the other two use σ_1/σ_{16}; the positions of the real layers belonging to each family are marked alongside.

measured interval or sit just above it. The derivation is therefore not fitted to the single point of 4 bits and $D = 128$.

The condition-number cliff and the accuracy cliff. Forming $\mathbf{A}^T\mathbf{A}$ explicitly squares the condition number and has long been avoided in numerical linear algebra. We narrow that rule rather than reject it. Solving a linear system needs every direction, so the whole of $\sigma_1/\sigma_{\min}$ enters. Proposition 1 says instead that a rank r approximation needs only the directions above σ_r, so the effective condition number is σ_1/σ_r. The squaring still applies at $\sigma_1/\sigma_r \approx \sqrt{1/u_{32}}$, as the fp32 Gram limit below confirms. The tail family shows this by measurement. Raising $\sigma_1/\sigma_{\min}$ by nine orders of magnitude from 10 to 1.8×10^{10} keeps the loop 3 excess below 0.07%. On real caches, layer 0 of

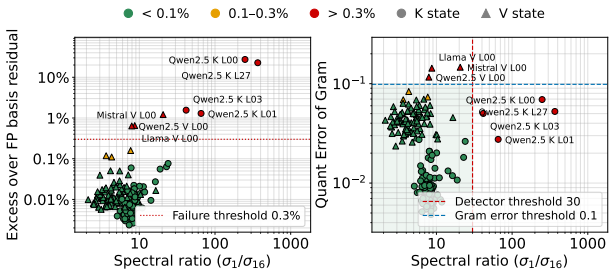

Figure 6: Integrity map of the 4-bit Gram over 184 layer-states from three models. Horizontal axis σ_1/σ_{16}, vertical axis the relative error $\|\mathbf{M}_{\text{int4}} - \mathbf{M}\|_F / \|\mathbf{M}\|_F$, color the excess error of the rank 16 basis from the 4-bit Gram. The rectangle marks the two detector thresholds.

the Llama K cache has the largest $\sigma_1/\sigma_{\min}$ at 1.1×10^4, yet its σ_1/σ_{16} is 7.8 and its fp32 excess stays at 0.017%.

What creates the boundary is σ_1/σ_{16}, and its location differs by precision. As Table 7 shows, on the power family INT4 breaks down between 16 and 32 while fp32 stays within 1% up to 512 and breaks down at 1024. In between, the condition number of the Krylov block Gram $\mathbf{K}^T\mathbf{M}\mathbf{K}$ jumps from 10^6 to 10^{31} as soon as σ_1/σ_{16} exceeds 128, yet the fp32 excess is still 0.001%, because the directions being lost would be discarded anyway. The condition-number cliff and the accuracy cliff are therefore not in the same place. Figure 5 shows the boundaries for the three families.

The floor of the fp32 Gram. The fp32 Gram sits inside the same framework, but its floor has a different source. Comparing a Gram formed in fp32 arithmetic against an fp64 Gram gives $\|\mathbf{\Delta}\|_F$ at 6×10^{-8} times λ_1, which is the fp32 unit roundoff $u_{32} = 2^{-24}$ itself, and the tail eigenvalue rise is around 10^{-9} times λ_1. Running an eigendecomposition on this Gram gives an excess of 0.0024% at $\sigma_1/\sigma_{16} = 1024$ and 0.80% at 4096, so the limit of the fp32 Gram itself is $\sigma_1/\sigma_{16} \approx \sqrt{1/u_{32}} \approx 4100$, where λ_{16} meets $u_{32}\lambda_1$. Yet fp32 on the power family already breaks down at 1024. The failure boundary of fp32 is therefore set by the iteration rather than by the Gram. The iteration injects arithmetic rounding every time it multiplies by $\mathbf{M}$ and reorthogonalizes, and that rounding accumulates with the loop count, so the fp32 boundary in Figure 4 sits in a different place for every loop count. The INT4 boundary is in the same place at loop 3 and loop 5, because the floor that the storage precision leaves in the Gram is static and does not change with the loop count.

F Detector details

The two thresholds 30 and 0.1 sit where the gap is widest between the group that breaks down at 4 bits and the group that tolerates it, across 184 layer-states. Figure 6 maps that gap and the rectangle the two thresholds cut out of it. This section describes that location and why the decision does not fluctuate with the loop count or the prompt. Algorithm 4 states the full procedure.

Algorithm 4 Per-layer precision decision

Require: $\mathbf{A}$, rank r, loop q, layer key ℓ, cache $\mathcal{C}$, thresholds $\tau = 30$, $\eta = 0.1$
if $\ell \in \mathcal{C}$ **then**
  return $\text{GramKrylov}(\mathbf{A}, r, q, \mathcal{C}[\ell])$
end if
$\mathbf{V}, \mathbf{w}, \mathbf{M} \leftarrow \text{GramKrylov}(\mathbf{A}, r, \max(q, 3), \text{fp32})$ ▷ first result is always exact
$\rho \leftarrow \sqrt{w_1/w_r}$ ▷ Ritz values, hence free
$\epsilon \leftarrow \|\text{Gram}_{\text{int4}}(\mathbf{A}) - \mathbf{M}\|_F / \|\mathbf{M}\|_F$ ▷ one INT4 Gram
$\mathcal{C}[\ell] \leftarrow$ INT4 **if** $\rho \le \tau$ **and** $\epsilon \le \eta$ **else** fp32
return $\mathbf{V}$

Thresholds. The threshold 30 on the first axis lies between the two groups. Measuring the Ritz ratio $\sqrt{w_1/w_{16}}$ of the fp32 Gram at loop 3 or higher across 184 layer-states, the Qwen K layers 0, 1, 3, and 27 that break down at 4 bits fall between 41.5 and 367.1, while the maximum over the remaining 180 is 24.0. The value 30 sits where the gap between the two groups is widest, and it is consistent with both the roughly 20 that Equation (13) predicts and the 16-to-32 failure boundary of

Table 8: Agreement of the precision decision when the prompt dataset changes. The reference is NarrativeQA at 32k tokens, and the last column is the largest relative movement of the Ritz ratio among the 184.

Dataset	Matching layer-states (of 184)	Max relative change in Ritz ratio
NarrativeQA (reference)	184	0
GovReport	184	46.5%
RepoBench-P	183	47.0%

the synthetic power family. The threshold 0.1 on the second axis is set the same way. The Gram relative error runs from 0.116 to 0.146 on the V layer 0 of all three models while the maximum over the rest is 0.083, so 0.1 lies between the two groups.

Why the decision is made at loop 3 or higher. The ratio stabilizes from loop 3 on. When the loop count is low the Krylov subspace has not converged, so the smallest Ritz value sits below its true value and the ratio is inflated accordingly, which makes the same layer receive different decisions at different loop counts. We therefore raise the loop count of the first call to at least 3 for the decision, and when the requested loop count is smaller we still return the fp32 result of that call unchanged. The raised loop count is paid once per layer.

Dataset invariance. The decision holds when the prompt changes. We re-dump the KV of the three models with 32768-token prompts from LongBench drawn from two dissimilar sources, the government reports of GovReport and the code of RepoBench-P, and measure the same two quantities again (Table 8). On GovReport all 184 decisions match the reference. On RepoBench-P one further layer, Mistral V layer 1, moves to fp32, caught not by its Ritz ratio but by its Gram relative error rising from 0.083 to 0.159, so the shift is in the safe direction. The ratio itself moves by up to 47%, but that movement occurs in layers whose ratio lies between 2 and 10, while Qwen K layers 3 and 19, which sit near the threshold, stay in place.

G Failures outside the detector and a remedy

Of the four Qwen K layers that the detector returned to fp32, two do not fully recover even in fp32. The cause is not the precision but the spectrum.

Qwen K layers 0 and 27 retain a loop 5 excess of 1.52% and 0.52% on the fp32 path, and those two layers alone account for most of the 0.08% that remains there. On spike family synthetic matrices, raising the ratio of the first direction keeps the fp32 loop 3 excess flat at 0.35% up to $\sigma_1/\sigma_{16} = 283$, then raises it to 2.61% at 849 and 8.07% at 2830. The fp32 sink boundary is therefore near 500, and the two layers, with σ_1/σ_{16} of 249 and 367, sit just before it. Layers with an attention sink land here for structural reasons, so a few can be expected in any model. The remedy is to append one final power step against the actual $\mathbf{A}$, in these layers only.

H Downstream pipeline settings

We state the settings of the two downstream pipelines and the GEAR results for all three models.

GEAR pipeline. LoGIK enters the SVD engine slot of the GEAR family pipeline: after prefill it obtains $\mathbf{V}$ for the K and V caches of every layer and quantizes to 2 bits with group size 64 the residual left after subtracting the low-rank component, and during decoding it collects new tokens up to a streaming gap of 64 and compresses them the same way. Evaluation runs on the simulated compression harness of GEAR. The SVD engine rows all share rank 4, loop 3, and the quantization settings above, and differ only in the engine that extracts the basis. The results for all three models are in Table 9.

SVDq pipeline. We replace only the SVD engine in the Llama3-8B pipeline of SVDq and, following the original work, apply the $\bar{b} = 2$ schedule $(8, 4, 4, 0, 0, 0, 0, 0)$ to pre-RoPE keys of $d = 1024$ formed by concatenating eight KV heads. The zero-bit groups of this schedule cut the latent channels

Table 9: Downstream accuracy (%) of the GEAR pipeline across all three models. The settings are those of Table 3 (top), and the Llama3-8B block repeats that table. Bold is best in a column excluding FP, underline second.

		Chain of Thought				LongBench						
Model	Method	GSM8K	AQuA	BBH	Avg.	Single	Multi	Sum.	Few-shot	Synth.	Code	Avg.
Llama3-8B	FP	78.01	50.79	58.62	62.47	29.93	34.89	24.10	58.91	52.69	55.09	42.60
	KCVT-3bit	58.23	35.43	43.70	45.79	16.50	15.19	20.56	48.47	4.77	40.70	24.37
	KIVI-2bit	58.38	30.31	38.10	42.26	21.22	19.22	20.66	52.97	7.94	40.86	27.15
	PALU-50%	71.87	43.70	54.32	56.63	**30.79**	32.68	22.80	57.76	45.99	53.79	40.64
	xKV	76.57	**50.00**	54.10	60.22	30.02	32.61	23.44	57.22	47.67	52.02	40.50
	Power iteration	75.89	47.64	52.35	58.63	29.24	32.51	23.53	57.85	47.06	**53.97**	40.69
	SRHT	74.30	47.64	51.16	57.70	26.45	29.22	23.48	57.86	45.57	46.95	38.26
	RSVD	73.31	44.09	52.10	56.50	26.25	28.66	23.21	58.08	45.63	47.72	38.26
	Block Krylov	72.55	49.21	50.47	57.41	26.02	29.09	**23.59**	**58.25**	45.26	46.86	38.18
	LoGIK	**77.48**	**50.00**	**54.38**	**60.62**	30.06	**33.09**	23.53	57.85	**48.03**	53.04	**40.93**
Mistral-7B	FP	55.95	33.46	47.74	45.72	39.02	37.40	26.20	63.05	66.67	59.03	48.56
	KCVT-3bit	44.81	33.86	42.84	40.50	32.12	25.79	25.37	58.88	43.94	52.36	39.74
	KIVI-2bit	41.09	25.20	35.88	34.06	31.74	29.16	24.77	57.14	28.48	52.86	37.36
	PALU-50%	54.13	**37.80**	46.56	**46.16**	**39.30**	**37.56**	25.38	62.97	66.67	58.87	48.46
	xKV	53.90	31.89	46.04	43.94	38.92	37.04	24.56	62.97	66.31	59.62	48.24
	Power iteration	54.06	32.68	42.80	43.18	38.92	37.33	25.84	62.97	66.50	58.92	48.41
	SRHT	54.97	30.31	43.84	43.04	39.28	37.15	25.98	62.98	66.50	59.11	48.50
	RSVD	53.15	32.28	43.44	42.96	38.95	37.23	25.78	62.94	66.33	58.96	48.37
	Block Krylov	55.34	32.28	44.13	43.92	39.19	37.00	25.82	62.96	66.50	58.90	48.40
	LoGIK	**55.88**	33.46	**46.96**	45.43	39.14	37.14	**26.18**	**63.02**	**67.01**	**60.18**	**48.78**
Qwen2-7B	FP	71.65	57.09	50.63	59.79	38.42	15.37	25.56	63.76	51.42	59.42	42.33
	KCVT-3bit	65.05	56.69	40.47	54.07	31.95	16.66	23.40	57.75	18.81	50.80	33.23
	KIVI-2bit	57.77	45.67	37.61	47.02	31.52	**19.74**	22.45	55.54	15.55	47.77	32.10
	PALU-50%	65.43	**57.48**	48.31	57.07	38.05	15.72	24.98	63.57	51.00	59.17	42.08
	xKV	70.43	53.94	**48.37**	57.58	**38.51**	15.94	24.84	63.57	51.33	59.11	42.22
	Power iteration	66.11	53.94	46.29	55.45	**38.51**	15.15	24.78	63.69	50.92	59.36	42.07
	SRHT	68.39	56.30	47.81	57.50	38.35	14.96	25.10	63.74	50.54	59.16	41.98
	RSVD	68.84	53.54	47.45	56.61	38.26	15.00	24.92	63.72	50.54	59.18	41.94
	Block Krylov	69.67	54.72	47.26	57.22	38.33	15.05	25.07	63.69	50.37	59.06	41.93
	LoGIK	**70.58**	55.51	48.34	**58.14**	**38.51**	16.30	**25.91**	**64.01**	**51.99**	**59.42**	**42.69**

Table 10: Bit schedules from Tables 2 and 5 of the SVDq paper. One group is $d/8$ latent channels, and a zero-bit group is exactly the truncation. Here r is the number of surviving latent channels, based on $d = 1024$ for Llama3-8B.

$\bar{b}$	Schedule b	r	Comp. ratio $16/\bar{b}$	Source
3	(8,4,4,4,2,2,0,0)	768	5.33	orig. Table 2
2.25	(8,4,4,2,0,0,0,0)	512	7.11	orig. Table 5
2	(8,4,4,0,0,0,0,0)	384	8.00	orig. Table 2
1.75	(8,4,2,0,0,0,0,0)	384	9.14	orig. Table 5
1.25	(4,4,2,0,0,0,0,0)	384	12.80	orig. Table 5

down to 384, so the engines build only those 384, which produces the same output as the original path (Appendix I). The loop count is 2.

I SVDq REPRODUCTION AND THE EQUIVALENCE OF TRUNCATION

The SVD engine of SVDq can be swapped. This section shows that we reproduce the original work faithfully and that its truncation, being a zero-bit group, produces the same output as an engine that builds only the top r directions. The engine therefore changes only the basis, and that difference shows up when $\bar{b}$ is small.

Faithful implementation. SVDq (Yankun et al., 2025) concatenates the keys of all KV heads in a layer to form $\mathbf{K} \in \mathbb{R}^{s\times d}$ with $d = n_{\text{kv}} d_h$, subtracts the per-channel mean $\bar{\mathbf{K}} \in \mathbb{R}^d$, and obtains the latent representation $\mathcal{P}_{\mathbf{V}}(\mathbf{K}) = \mathbf{KV}$ from $\mathbf{K} = \mathbf{UDV}^H$. The d latent channels are split into eight groups of $d/8$, each group receives one integer bit width b_i, and each channel is asymmetrically quantized by a min-max over the token axis before being restored as $\hat{\mathbf{K}} = \mathcal{D}_b(\mathcal{Q}_b(\mathcal{P}_{\mathbf{V}}(\mathbf{K})))\mathbf{V}^H + \bar{\mathbf{K}}$. The SVD is applied to the keys before RoPE, and the value cache is left uncompressed because its low-rank structure is weak. The mean bit width is $\bar{b} = \frac{1}{8}\sum_i b_i$ and the compression ratio is $16/\bar{b}$.

We use the five schedules of the original Tables 2 and 5 exactly (Table 10), and Llama3-8B has eight KV heads, giving $d = 1024$.

Truncation is a zero-bit group. SVDq is not a truncated SVD that takes the rank as an input. It finishes the full decomposition and then assigns $b_i = 0$ to the trailing groups, and the original work itself calls this step "truncation". At first there appears to be no slot in which to swap the SVD engine, but the two computations produce the same result. When the schedule is non-increasing and $b_r > 0 = b_{r+1}$, the reconstruction decomposes as

$$\hat{\mathbf{K}} = \sum_{j=1}^{d} \mathcal{D}_{b_j}\big(\mathcal{Q}_{b_j}(\mathbf{K}\mathbf{v}_j)\big)\,\mathbf{v}_j^H + \bar{\mathbf{K}} = \sum_{j=1}^{r} \mathcal{D}_{b_j}\big(\mathcal{Q}_{b_j}(\mathbf{K}\mathbf{v}_j)\big)\,\mathbf{v}_j^H + \bar{\mathbf{K}}. \tag{21}$$

Terms with $b_j = 0$ are exactly zero, so discarding the trailing $d - r$ channels and building only the top r give the same $\hat{\mathbf{K}}$. The original work therefore requires a full SVD for implementation convenience rather than for accuracy, and placing an engine that emits only the top r in that slot does not change the method. Note that r is set by the schedule, not chosen by us.

This equivalence is not confined to the formulas. We apply four of the five schedules to the NarrativeQA 32k key caches of Llama3-8B-Instruct (layers 0, 8, 16, 31) and compare the outputs of the full SVD path and the top r path in fp32. The maximum absolute difference is exactly 0 at $\bar{b} = 3, 2, 1.25$, and below 10^{-6} in relative terms at $\bar{b} = 2.25$, where floating-point SVD chooses a different basis among nearly coincident singular values. The fifth schedule, $\bar{b} = 1.75$, keeps the same $r = 384$ as $\bar{b} = 2$, so its truncation is one already checked. The exact SVD row of Table 3 (bottom) therefore carries the same numbers as the original SVDq, and what separates it from the approximate engine rows is only the way those r directions are chosen.

The engine changes only the basis. The schedule is a fixed integer vector that does not depend on the data, so every engine row uses the same bit allocation, the same number of latent channels r, and the same storage of $\bar{b} \cdot d$ bits per token. Basis storage is $d \times r$ in every row as well, and is in fact smaller than the original work's storage of the full $\mathbf{V} \in \mathbb{R}^{d\times d}$. However, since one group is $d/8$ channels, r moves only in multiples of $d/8$, and since a group bit width is capped at 8, a larger $\bar{b}$ makes the truncation shallower: $\bar{b} = 3$ gives $r \geq 768$ and $\bar{b} = 2$ gives $r = 384$. When the rank approaches d, the Krylov space and the sketch become the whole space and every engine produces the same basis as the full decomposition, so the differences between engines appear on the small $\bar{b}$ side. That is why the main text takes $\bar{b} = 2$ as its primary setting.

J ADDITIONAL EXPERIMENTS

We collect the remaining measurements referenced by the main text along one axis. The cost of LoGIK sits in the single Gram line, that line alone grows with the token length S, and everything else is tied to size rq and independent of the channel dimension D.

Storage format and Gram kernel. The time of that single Gram line is set by how much it reads. Figure 7 varies the storage format (fp32, fp16, int4) and the Gram kernel independently on layer 15 of Llama. The INT4 kernel takes 0.358ms from fp32 storage and 0.043ms from int4 storage. From fp32 storage it must quantize on the way in, so it moves 302MB against the 134MB of the tensor itself. Only when the cache is already 4-bit does it read the 16.8MB as stored. The traffic benefit is therefore complete only when the storage format matches the kernel. In the same settings the fp32 kernel takes 6.03ms to 7.19ms, and even with 4-bit storage it requires a dequantization pass that brings the volume read back to 285MB, so it gains nothing from the storage format. The excess error of all three paths is identical at 0.444%, so the information is already fixed at storage time and the kernel adds no error on top of it.

Token-length sweep. On the length axis the entire Gram family is flat and only token-space methods grow expensive. We truncate the first S entries of the 32768-token dump into a real cache and measure the time to reach 1% excess error. Between 1k and 32k, Table 11 shows LoGIK moving only from 3.61ms to 4.30ms and Gram + eigh only from 9.2ms to 10.1ms, while the exact SVD doubles from 32.53ms to 66.88ms (Figure 8, right). Gram + eigh finishes its eigendecomposition on

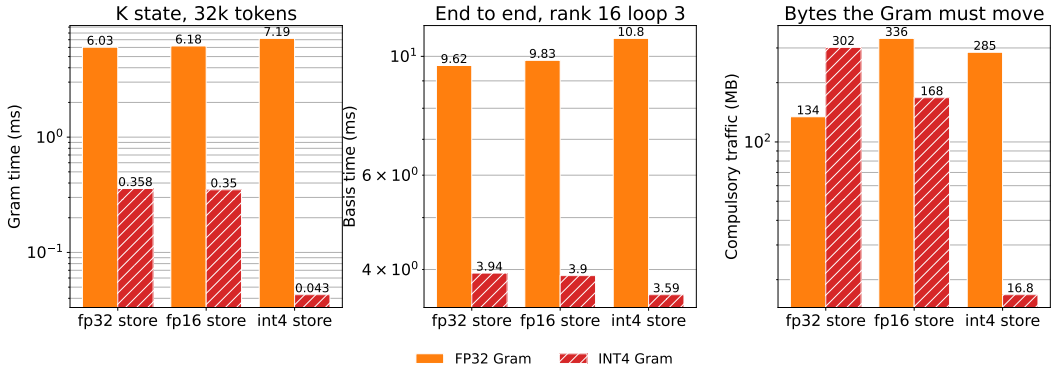

Figure 7: Gram time and bytes read as a function of the cache storage format and the Gram kernel, on Llama-3.1-8B layer 15 at 32k tokens. Bytes count the quantization pass as well as the GEMM. This is a separate run from Table 2 and Figure 2, so quantities they share move by a few percent between them.

Table 11: Time (ms) to reach 1% excess error by token length on the Llama-3.1-8B K state at rank 16. This sweep and Table 12 are each a separate run, from Table 2 and from each other, and the right panel of Figure 8 is drawn from this one; at the shared setting of 32k tokens and rank 16 the GPU-side times reproduce that table within 5%.

S	LoGIK (INT4)	Gram + eigh (INT4, CPU)	RSVD	Block Krylov	exact SVD	vs. SVD
1k	3.61	9.23	16.87	27.11	32.53	9.0×
2k	3.62	9.32	18.47	30.54	33.59	9.3×
4k	3.56	9.34	24.32	34.58	38.76	10.9×
8k	3.64	9.66	32.44	44.11	44.49	12.2×
16k	3.77	9.63	51.97	62.81	60.93	16.2×
32k	4.30	10.12	69.68	63.38	66.88	15.6×

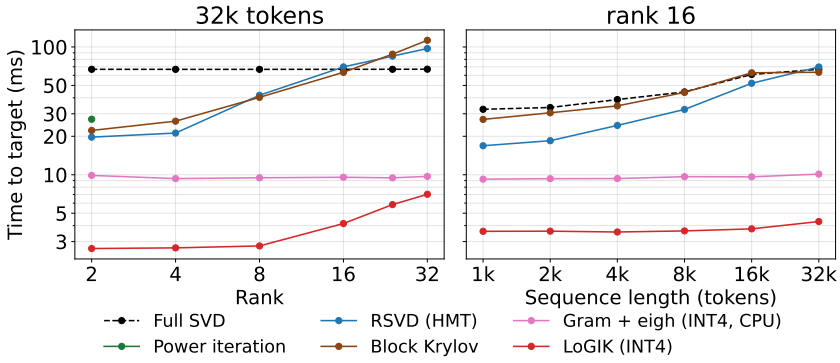

Figure 8: Time to reach 1% excess error on the Llama-3.1-8B K state. The left panel varies the rank at 32k tokens and the right panel varies the token length at rank 16. The exact SVD is pinned at about 67ms regardless of rank and rises with length, 2.06× over a 32× range.

the host, so it is the least reproducible column across runs: the same setting reads 8.21ms in Table 2, against 9.2ms to 10.1ms across the cells here that should be constant.

Rank sweep. On the rank axis the time of LoGIK follows the size rq of the final eigendecomposition. Table 12 lists the time to reach 1% excess error from rank 2 to 32 on the Llama K state at 32k tokens. Gram + eigh stays between 9.3ms and 9.9ms because its single 128×128 eigendecomposition does not depend on the rank. LoGIK always reaches the target at loop 3, so its rq is three times the rank and its time does rise with the rank, yet even at rank 32 it stays below Gram + eigh at 7.03ms. The jump from 2.77ms to 4.15ms as the rank goes from 8 to 16, where rq crosses 32 from 24 to 48, is the batched Jacobi boundary (Figure 8, left).

Table 12: Time (ms) to reach 1% excess error by rank on the Llama-3.1-8B K state at 32k tokens. Cells that never reach 1% are left blank. This is another run of the same sweep as Table 11, and the left panel of Figure 8 is drawn from it, so the two tables move by a few percent at the setting they share.

rank	rq	LoGIK (INT4)	Gram + eigh (INT4, CPU)	Block Krylov	RSVD	Power iteration
2	6	2.65	9.88	22.22	19.69	27.18
4	12	2.68	9.33	26.27	21.20	
8	24	2.77	9.47	40.34	41.87	
16	48	4.15	9.56	63.31	69.81	
24	72	5.84	9.46	87.90	84.78	
32	96	7.03	9.71	112.72	97.18	

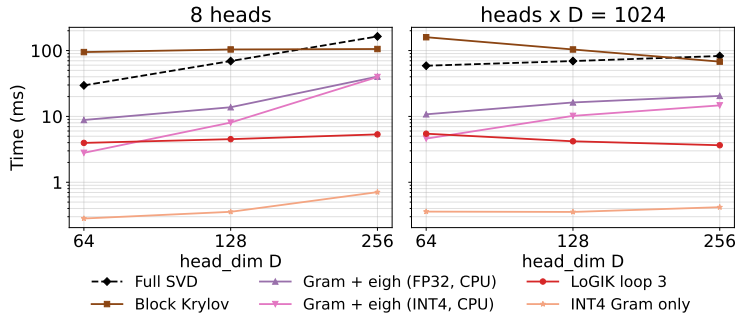

Figure 9: Basis extraction time as a function of the channel dimension D on synthetic tensors (32k tokens, rank 16). The left panel fixes 8 heads and varies D alone, and the right panel fixes the product of head count and D at 1024.

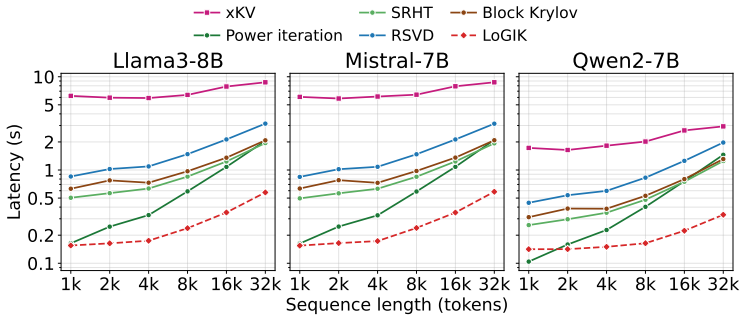

Figure 10: KV compression latency of the SVD family methods of Table 3 (top) measured by sequence length. This is the time to compress the full cache of every layer, 32 for Llama and Mistral and 28 for Qwen, and shared costs such as attention and MLP are excluded (A100-80GB, median of 5 runs over 8 layers).

Channel-dimension sweep. On the channel dimension D as well, only LoGIK is insensitive, for the same reason. All three models have head dimension 128, so we measure with synthetic tensors of prescribed singular values. The left panel of Figure 9 varies D over 64, 128, and 256 with 8 heads at 32k tokens. The eigendecomposition cost of Gram + eigh scales with the cube of D, so at $D = 64$ it is cheaper than LoGIK at 2.80ms, is overtaken at 128, and pays 39.6ms at 256. LoGIK stays at 5.33ms throughout, because its eigendecomposition is tied to size rq. The right panel grows the head dimension while reducing the head count at fixed total KV volume (head count $\times$ $D = 1024$). The exact SVD and Gram + eigh grow more expensive, Block Krylov falls with the head count but never below 67.9ms, and LoGIK drops from 5.46ms to 3.65ms. The eigendecomposition that sets its cost is independent of D, so it receives only the benefit of a falling head count.

KV compression latency. The same gap appears inside the pipeline. Figure 10 gives the KV compression latency of the SVD engines of the GEAR pipeline measured by sequence length. The five engines share a storage format, so their KV memory coincides and only the cost of extracting

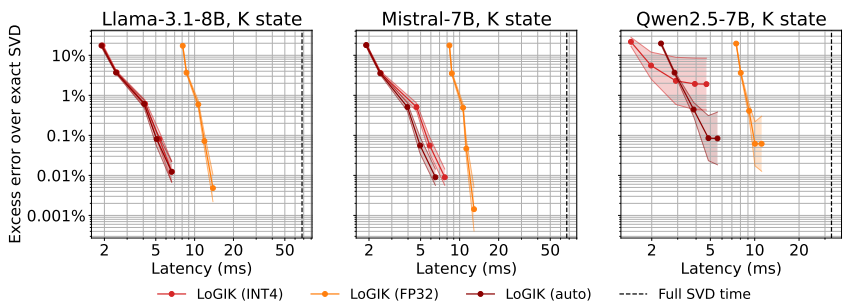

Figure 11: Time versus excess error by precision for LoGIK on the K states of three models: INT4, fp32, and the precision chosen by the detector.

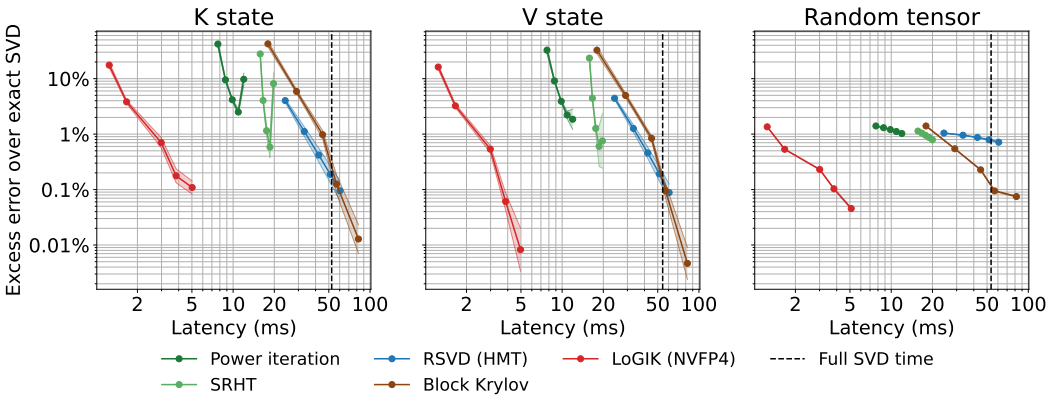

Figure 12: Basis-extraction time versus excess error with an NVFP4 Gram on an RTX PRO 6000 Blackwell. The setting matches Figure 1 (top) except for the GPU and the Gram format, and all methods run on this GPU.

the basis separates them here. At 32k, LoGIK is 3.58× faster than power iteration on Llama, 3.60× on Mistral, and 4.39× on Qwen.

Curves by precision. Figure 11 gives time versus excess error on the K states of three models for INT4, fp32, and the precision chosen by the detector. The detector leaves only four layers in fp32, all in the Qwen K cache, and on the remaining 88 the INT4 curve closely tracks the fp32 curve.

K NVFP4 Gram on Blackwell

The Gram kernel of LoGIK does not depend on INT4. Blackwell GPUs provide block-scaled FP4 tensor cores (Rouhani et al., 2023; Abecassis et al., 2025; Egiazarian et al., 2026), and on them LoGIK forms the same Gram in the NVFP4 format and keeps its speedup.

Format and kernel. NVFP4 stores each element as a 4-bit E2M1 value, gives every block of 16 elements an E4M3 scale, and adds one fp32 scale per head. We place the blocks along the token axis, so each block holds 16 tokens of one channel, and compute $\mathbf{Q}^T\mathbf{Q}$ with the block-scaled FP4 MMA in CUTLASS. Only the Gram formation changes; the Krylov refinement in channel space stays in fp32 as in Algorithm 2.

Time and accuracy. In Figure 12, LoGIK with the NVFP4 Gram is the only method on the frontier for all three tensors. At loop 3 it reaches 0.70% excess error on the K state in 2.98ms, while Block Krylov needs 45.07ms for 0.98%, 15× longer. A separate per-stage timing at the same setting puts the Gram at 0.17ms with the NVFP4 kernel against 1.15ms in fp32. On synthetic matrices the NVFP4 Gram crosses 1% excess between σ_1/σ_{16} of 32 and 64 (Figure 4), one grid step above INT4. This interval contains the value of about 36 that Equation (13) predicts (Table 6).