SMOOTHGPTQ: OPTIMIZING DIAGONAL SCALING FOR GPTQ WEIGHT-ACTIVATION QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diagonal smoothing, as in SmoothQuant, is a useful tool for balancing the errors incurred from weights quantization and activations quantization. For weight quantization the error can be further reduced by accounting for the activations second-order statistics via GPTQ. Prior work has considered the optimization of diagonal smoothing parameters for round-to-nearest (RTN) quantization. In this work we develop a framework, termed SmoothGPTQ, for optimizing those parameters for GPTQ-quantized weights and RTN-quantized activations. We derive an approximate expression for the quantization mean-squared error (MSE) as a function of the diagonal smoothing matrix S. This expression is convex in log-scale and is therefore amenable to minimization, and its accuracy at high rates is established through experiments. Applying the developed SmoothGPTQ algorithm to various LLMs gives state-of-the-art results for W4A4 INT quantization across various benchmarks.

1 INTRODUCTION

Quantization plays an important role in efficient deployment of large language models (LLMs) allowing for decreasing memory footprint, memory traffic, communication costs and faster matrix multiplication. This paper focuses on quantization of the weight matrix $W \in \mathbb{R}^{a\times n}$ and activation matrix $X \in \mathbb{R}^{n\times b}$ to integer entries with bounded support, so that matrix multiplication (MatMul) can be carried out by INT multipliers. We focus on minimizing the mean squared error (MSE) for a single linear projection WX. This objective is by far the most popular one Ordentlich & Polyanskiy (2026b), though, others exist Sun et al. (2025); Tseng et al. (2025).

One of the early observations Xiao et al. (2023) in LLM post training quantization (PTQ) literature was that the trivial identity

$$WX = (WS^{-1})(SX),$$

which holds for any invertible matrix $S \in \mathbb{R}^{n\times n}$, entails a rich set of opportunities for improving quantization quality. The source of improvement is that it is often possible to find $S \neq I_n$ such that the empirical distribution of the rows of WS^{-1} and the columns of SX are much better matched to the applied quantizer. Note however that using $S \neq I_n$ has one important disadvantage: the activations X have to be multiplied by S prior to quantization, and since this must be computed in real-time during inference, the corresponding overhead may be large.[1] Furthermore, the matrix S typically needs to be stored together with the model parameters, and loaded whenever the corresponding projection is computed, entailing additional cost.

A particularly useful choice is to take S as an efficiently implementable random rotation Chee et al. (2023); Tseng et al. (2024); Ashkboos et al. (2024); Liu et al. (2025), where using the randomized Hadamard transform Tropp (2011); Ben-Basat et al. (2026) is the canonical choice. The benefit of a random rotation is that it balances the energy of a vector in $\mathbb{R}^n$ among all n coordinates, such that they all have the same variance. In fact, together with a suitable high-dimensional quantizer, a random rotation allows to attain optimal rate-distortion tradeoff within the class of all matrices W/X with fixed rows/columns energy Ordentlich & Polyanskiy (2024). While the Hadamard transform can be implemented with only $O(n \log n)$ arithmetic operations per vector in $\mathbb{R}^n$, even this additional

[1] The weights are quantized offline, so the cost of computing WS^{-1} prior to quantization has no effect on inference time.

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.

work is often prohibitive in applications with tight latency or hardware constraints. There is a variety of schemes in the literature that consider an even richer structure for S including learned dense transformations Ma et al. (2024), tensor-structured transformations such as the Kronecker products used in FlatQuant Sun et al. (2025) and block-diagonal S with dense blocks Lin et al. (2024); Chen et al. (2025), but even with fused specialized kernels, those may be prohibitive in some applications.

Taking S as a diagonal matrix stands out as a highly appealing choice as it incurs negligible overhead in terms of storage and FLOPS alike: It requires only n spacing coefficients and, when applied explicitly, n multiplications per activation vector. Often even this cost can be avoided: when X is produced by LayerNorm or RMSNorm, the diagonal transformation can be absorbed into the normalization layer's existing per-channel parameters. This approach was initiated with SmoothQuant Xiao et al. (2023), and remained a dominant technique for throughput-sensitive PTQ.

Previous work considered the problem of optimizing the diagonal spacing factors $s_1, \dots, s_n$ for minimizing the MSE under absmax scaling followed by **round-to-nearest (RTN) INT quantization** for the rows of WS^{-1} and columns of SX. In particular, SmoothQuant Xiao et al. (2023) addresses this tradeoff through a heuristic law $s_i \propto (\max_j |W_{ji}|)^{1-\alpha_{\mathrm{SQ}}}/(\max_{k\in\mathrm{cal}} |X_{ik}|)^{\alpha_{\mathrm{SQ}}}$, where α_{SQ} is a design parameter selected by trial and error. Under an independent additive quantization-noise model, the matrix-product mean squared error (MSE) admits a simple expression in terms of the operand energies and quantization-noise variances; see, e.g., Ordentlich & Polyanskiy (2026a). Recently, Sao et al. (2026) formulated the selection of positive diagonal scales under such a model as a convex optimization problem in logarithmic coordinates. OmniQuant Shao et al. (2023) considered the objective of *jointly optimizing multiple diagonal spacing matrices* participating in each transformer block for minimizing the MSE between the quantized and unquantized outputs of the block.[2] This objective is not convex, but the OmniQuant optimization nevertheless provides appealing results in practice Shao et al. (2023).

A major shortcoming of the approaches above is that they are designed to optimize RTN quantization, whose quantization error typically has a nearly isotropic covariance matrix. Activation vectors on the other hand are often very far from isotropic, and favor certain directions in $\mathbb{R}^n$. One can therefore obtain smaller distortion by quantizing the rows of the weight matrix in a way that will minimize the quantization error energy in those directions, at the cost of allowing for more error energy in directions where activations are weak Ordentlich & Polyanskiy (2026b). This observation is at the heart of the GPTQ algorithm Frantar et al. (2023), which is a landmark PTQ algorithm. In fact, when the second order statistics of the activations are known, under the additive noise model, GPTQ attains distortion smaller or equal to that of RTN Chee et al. (2023). Consequently, no matter what matrix S is used, it is better to quantize WS^{-1} with GPTQ instead of RTN (under the additive noise model). The original SmoothQuant work already suggested combining its diagonal scaling with GPTQ for improved weight quantization (Xiao et al., 2023, Appendix A). Indeed, this combination is implemented in practical pipelines such as LLM Compressor's W8A8 recipe LLM Compressor contributors (2026), which applies SmoothQuant scaling followed by GPTQ weight quantization and RTN activation quantization. However, spacing coefficients $s_1, \dots, s_n$ that were optimal for RTN quantization may no longer be optimal for GPTQ quantization. The goal of this work is to develop a procedure for optimizing those scale coefficienrs for the latter setup.

Our main contributions include:

- We develop an analytic approximation $\tilde{D}(S)$ for the MSE when quantizing WS^{-1} using GPTQ and SX using RTN. This expression involves the diagonal entries in the Cholesky decomposition of $\Sigma_X = \mathbb{E}[XX^\top]$ rather than $\mathrm{diag}(\Sigma_X)$. Experiments with Llama-3.1-8B-Instruct validate the accuracy of this expression for high-rate quantization. An appealing feature of $\tilde{D}(S)$ is that it is convex in $\log s_1, \dots, \log s_n$ and can therefore be efficiently optimized.
- We introduce SmoothGPTQ: a PTQ method with diagonal S chosen to optimize $\tilde{D}(S)$ for each projection, followed by GPTQ for the smoothed weights and RTN for the smoothed activations.
- SmoothGPTQ demonstrates significant improvements in the MSE, KL divergence and perplexity with respect to GPTQ, SmoothQuant, and SmoothQuant followed by GPTQ with

[2] OmniQuant also avoids the absmax scaling for weight matrices and optimizes their scaling coefficients.

the default choices of S. At W6A6, SmoothGPTQ further achieves lower output MSE than OmniQuant in 20 of 32 transformer blocks, despite OmniQuant explicitly optimizing a reconstruction loss during calibration, whereas SmoothGPTQ optimizes only our analytically derived distortion proxy. SmoothGPTQ also attains smaller KL divergence and perplexity than OmniQuant.

- Extensive experiments show that for W4A4 quantization with popular group sizes, SmoothGPTQ either attains state-of-the-art (SOTA) results or is competitive with SOTA methods that require rotations. The advantage of SmoothGPTQ over such schemes is that it does not require specialized fused kernels, and results in negligible overhead.

1.1 Related Work

Optimizing diagonal scaling. SmoothQuant Xiao et al. (2023) selects diagonal scales through a heuristic balance of weight and activation maxima. OmniQuant Shao et al. (2023) learns equivalent transformations and weight-clipping parameters by minimizing block-output reconstruction error. An alternative is to optimize an analytic distortion model: independent additive-noise models yield tractable expressions for matrix-product MSE Ordentlich & Polyanskiy (2026a), and Sao et al. (2026) formulate diagonal scaling under such a model as a convex problem in logarithmic coordinates. Our contribution is to account explicitly for GPTQ's feedback in the scaling objective and show that the resulting high-rate approximation remains convex in log-scales. Another work addressing optimized diagonal scaling for NVFP4 quantization was submitted to this conference (Anonymous, 2027).

Feedback-based weight quantization. GPTQ Frantar et al. (2023) uses activation second-order statistics to compensate for rounding errors through updates to the remaining weights. QuIP Chee et al. (2023) establishes its equivalence to LDLQ and analyzes the distortion of this linear-feedback quantizer. Its connection to Babai's nearest-plane algorithm is developed by Chen et al. (2026); Birnick (2026), while Ordentlich & Polyanskiy (2026b) analyze the corresponding SIC procedure in the high-rate regime. These results explain why the weight-error contribution involves Cholesky pivots. We use this established quantizer and optimize the diagonal transformation preceding it, accounting also for activation quantization and the quantization ranges of both operands.

WaterSIC and rate allocation. WaterSIC Lifar et al. (2026) is particularly close to our work: it adapts per-coordinate quantization spacings to the innovation variances of the activations and combines SIC with unequal rate allocation. Its weight-only formulation uses entropy coding of potentially unbounded integer outputs to control the compressed model size. Our setting instead requires bounded, fixed-width integer representations for both weights and activations, so that their product can be computed using INT arithmetic. Here, changing a weight coordinate's effective spacing also changes the activation range through SX. The resulting balance between weight and activation errors, under per-vector or group scaling, is the optimization problem addressed in this paper.

2 Problem Formulation and Approximate Distortion Expression

Let $W \in \mathbb{R}^{a\times n}$ be a fixed weight matrix and $X \in \mathbb{R}^{n\times b}$ a calibration matrix whose columns are activation vectors. For an integer $R \geq 2$, we quantize the entries of both W and X to $\mathrm{INT}_R = \mathbb{Z}\cap[-2^{R-1}, 2^{R-1})$, with diagonal row and column scale matrices Γ and Λ. The goal is to minimize the empirical output MSE

$$D(S) = \frac{1}{ab}\|WX - \widehat{WX}\|_F^2, \qquad \widehat{WX} = \Gamma\hat{\bar{W}}\hat{\bar{X}}\Lambda. \tag{1}$$

Let $S = \mathrm{diag}(s_1, \ldots, s_n)$ be a diagonal matrix with positive entries and write $\bar{W} = WS^{-1}$, and $\bar{X} = SX$, so $WX = \bar{W}\bar{X}$.

For the derivation, let $\Sigma_X = \frac{1}{b}XX^\top$ and assume it is positive definite. Consider its Cholesky decomposition $\Sigma_X = U^\top U$ for upper-triangular $U \in \mathbb{R}^{n\times n}$ with diagonal entries $u_i = U_{ii}$. Smooth-

ing transforms Σ_X to

$$\frac{1}{b}\bar{X}\bar{X}^\top = S\Sigma_X S = (US)^\top(US). \tag{2}$$

Thus the triangular factor governing weight quantization is US, with diagonal $u_i s_i$. In practice, we may estimate Σ_X from a larger calibration set and apply damping, as described in Section 4.

2.1 Quantization Method and Approximate Distortion

Write $W_j = (W_{j,:})^\top$, $\bar{W}_j = S^{-1}W_j$, and $\bar{X}_k = SX_k$. We use RTN for online activation quantization and clipped SIC/GPTQ for offline weight quantization. With $\beta_R = 2^{R-1} - 1$ and margin $\tau > 0$, their scales are set to

$$\gamma_j = (1+\tau)\frac{\|\bar{W}_j\|_\infty}{\beta_R}, \qquad \lambda_k = \frac{\|\bar{X}_k\|_\infty}{\beta_R}. \tag{3}$$

The diagonal entries of Γ and Λ are γ_j and λ_k, respectively. We assume positive scales; an identically zero vector can be represented by zero codes and any positive scale.

Quantization rules. RTN quantizes each activation coordinate independently, setting $\hat{\bar{X}}_k = \mathrm{round}(\bar{X}_k/\lambda_k) \in \mathrm{INT}_R^n$. Weight quantization can instead exploit activation correlations. Ignoring quantization errors of X for the moment, for row j, reconstructing $\bar{W}_j$ as $\gamma_j z$ gives the average squared output error

$$\frac{1}{b}\sum_{k=1}^{b}\left[(\bar{W}_j - \gamma_j z)^\top \bar{X}_k\right]^2 = \|US(\bar{W}_j - \gamma_j z)\|_2^2 = \gamma_j^2\|y - Lz\|_2^2, \tag{4}$$

where $L = US$ and $y = US\bar{W}_j/\gamma_j$. Thus, we seek integer codes whose residual is small after multiplication by the triangular factor L.

SIC approximately solves this problem by choosing the coordinates of z sequentially, from n down to 1. Use y as the running residual, initialized as above. At step i, the coordinates $z_{i+1}, \ldots, z_n$ have already been fixed. Because L is upper triangular, the ith residual coordinate depends only on these decisions and z_i, so rounding y_i/L_{ii} minimizes its magnitude. After choosing z_i, subtract $z_i L_{:,i}$ from y. This update accounts for the chosen code in every remaining coordinate before the next rounding decision; later decisions cannot change residual coordinates already settled. Algorithm 1 gives the resulting procedure, and we set $\hat{\bar{W}}_j = z_{\mathrm{SIC}}$.

The algorithm as written allows unrestricted integer codes. For our finite codebook, we clip each rounded decision to INT_R *before* subtracting $z_i L_{:,i}$, so subsequent decisions account for the clipped value. Clipping only the completed output would generally give a different result. With the prescribed metric, grids, and ordering, this clipped SIC procedure is equivalent to GPTQ. Appendix A describes its implementation using existing packages such as `llmcompressor`.

Algorithm 1 SIC

Inputs: $y \in \mathbb{R}^n$ and upper triangular $L \in \mathbb{R}^{n\times n}$
Outputs: $z_{\mathrm{SIC}} \in \mathbb{Z}^n$.

for $i = n : 1$ **do**
  $z_{\mathrm{SIC},i} \leftarrow \mathrm{round}(y_i/\ell_i)$ ▷ ℓ_i is the ith diagonal entry of L
  $y \leftarrow y - z_{\mathrm{SIC},i} \cdot L_{:,i}$ ▷ $L_{:,i}$ is the ith column entry of L
end for

Error decomposition and noise model. Let $e_{\bar{W}_j} = \gamma_j\hat{\bar{W}}_j - \bar{W}_j$ and $e_{\bar{X}_k} = \lambda_k\hat{\bar{X}}_k - \bar{X}_k$. The error in output entry (j,k) is

$$e_{jk} = e_{\bar{W}_j}^\top \bar{X}_k + \bar{W}_j^\top e_{\bar{X}_k} + e_{\bar{W}_j}^\top e_{\bar{X}_k}. \tag{5}$$

The first two terms describe weight and activation errors propagated through the other matrix. We assume their cross terms average to negligible values. Neglecting also the $(e_{\bar{W}_j}^\top e_{\bar{X}_k})^2$ term gives

$$D(S) \approx \frac{1}{a}\sum_{j=1}^{a} \|USe_{\bar{W}_j}\|_2^2 + \frac{1}{b}\sum_{k=1}^{b} e_{\bar{X}_k}^\top \left(\frac{1}{a}\bar{W}^\top\bar{W}\right) e_{\bar{X}_k}. \tag{6}$$

Thus the weight error is measured in the transformed activation metric, whereas the activation error is weighted by the smoothed weights.

For RTN, we model $e_{\bar{X}_k}$ as uniform on $\lambda_k[-1/2, 1/2)^n$. For unclipped SIC, the residual satisfies (Ordentlich & Polyanskiy, 2026b, Lemma 1)

$$USe_{\bar{W}_j} \in \operatorname{diag}(\gamma_j US)[-1/2, 1/2)^n, \tag{7}$$

and we model it as uniform on this cell. In contrast to RTN, whose modeled weight error is isotropic, the SIC quantization error $e_{\bar{W}_j}$ is modeled as an isotropic random vector that underwent the linear transformation $\gamma_j \cdot S^{-1}U^{-1}\operatorname{diag}(U)S$.

We assume that clipping has negligible effect on this residual model. The margin τ enlarges the weight grid to reduce clipping; for fixed S and U, any positive margin eventually avoids clipping as R increases, but it does not guarantee its absence at finite rates. Under the noise model, the squared product error $(e_{\bar{W}_j}^\top e_{\bar{X}_k})^2$ omitted from (6) is $O(\beta_R^{-4})$, compared with the leading $O(\beta_R^{-2})$ terms. These are high-rate approximations, whose empirical accuracy we test in Section 5.1. Appendix G gives the full derivation and clipping argument.

The model gives the two leading distortion contributions:

$$\mathbb{E}\|USe_{\bar{W}_j}\|_2^2 \approx \frac{\gamma_j^2}{12}\sum_{i=1}^{n}(u_i s_i)^2, \tag{8}$$

$$\mathbb{E}\left[e_{\bar{X}_k}^\top \left(\frac{1}{a}\bar{W}^\top\bar{W}\right) e_{\bar{X}_k}\right] \approx \frac{\lambda_k^2}{12a}\|\bar{W}\|_F^2. \tag{9}$$

The first expression is the key effect of feedback: it uses Cholesky diagonal entries u_i^2 rather than marginal activation energies $(\Sigma_X)_{ii}$. Averaging over rows and columns and substituting the quantization scales yields $D(S) \approx \tilde{D}(S)/(12\beta_R^2)$, where $\sigma_{W_i}^2 = \frac{1}{a}\sum_j W_{ji}^2$ and

$$\tilde{D}(S) = \sum_{i=1}^{n}\left[(1+\tau)^2(u_i s_i)^2 \frac{1}{a}\sum_{j=1}^{a}\|\bar{W}_j\|_\infty^2 + \frac{\sigma_{W_i}^2}{s_i^2}\frac{1}{b}\sum_{k=1}^{b}\|\bar{X}_k\|_\infty^2\right]. \tag{10}$$

2.2 Group Scaling

We can assign a separate quantization scale to every g consecutive entries of each weight row and activation column. Assume g divides n and partition the shared feature dimension into n/g groups $\mathcal{I}_h = \{(h-1)g+1, \ldots, hg\}$. For group h, set

$$\gamma_{jh} = \frac{1+\tau}{\beta_R}\max_{i\in\mathcal{I}_h}\frac{|W_{ji}|}{s_i}, \qquad \lambda_{kh} = \frac{1}{\beta_R}\max_{i\in\mathcal{I}_h} s_i|X_{ik}|. \tag{11}$$

We assume these group maxima are positive. Weights are quantized by clipped SIC with coordinate $i \in \mathcal{I}_h$ reconstructed on the grid $\gamma_{jh}\mathrm{INT}_R$, and activations by RTN with scale λ_{kh} within each group. Writing $\Gamma_h = \operatorname{diag}(\gamma_{1h}, \ldots, \gamma_{ah})$ and $\Lambda_h = \operatorname{diag}(\lambda_{1h}, \ldots, \lambda_{bh})$, the reconstructed product is

$$\widehat{WX} = \sum_{h=1}^{n/g}\Gamma_h\hat{\bar{W}}_{:,\mathcal{I}_h}\hat{\bar{X}}_{\mathcal{I}_h,:}\Lambda_h. \tag{12}$$

Under the same noise assumptions and neglecting of the second-order error term as above, the weight contribution to the distortion is $\frac{1}{12a}\sum_{j,h}\gamma_{jh}^2\sum_{i\in\mathcal{I}_h}(u_i s_i)^2$, and the activation contribution

is $\frac{1}{12b}\sum_{k,h}\lambda_{kh}^2\sum_{i\in\mathcal{I}_h}\sigma_{W_i}^2/s_i^2$. Consequently, the rescaled proxy for group size g separates as

$$D(S) \approx \frac{1}{12\beta_R^2}\tilde{D}_g(S), \qquad \tilde{D}_g(S) = \sum_{h=1}^{n/g}\tilde{D}_{g,h}(S_h), \tag{13}$$

$$\tilde{D}_{g,h}(S_h) = \sum_{i\in\mathcal{I}_h}\left[(1+\tau)^2(u_i s_i)^2\frac{1}{a}\sum_{j=1}^{a}\max_{\ell\in\mathcal{I}_h}\frac{W_{j\ell}^2}{s_\ell^2} + \frac{\sigma_{W_i}^2}{s_i^2}\frac{1}{b}\sum_{k=1}^{b}\max_{\ell\in\mathcal{I}_h}X_{\ell k}^2 s_\ell^2\right], \tag{14}$$

where $S_h = \mathrm{diag}(s_i : i \in \mathcal{I}_h)$. In particular, $\tilde{D}_n(S) = \tilde{D}(S)$. The coefficients u_i are taken from the Cholesky factor of the full covariance Σ_X, and SIC feedback may propagate across group boundaries.

Each term $\tilde{D}_{g,h}(S_h)$ depends only on the g smoothing coefficients in its own group. Thus minimizing $\tilde{D}_g(S)$ reduces to n/g independent optimization problems, whose solutions can be assembled into S. Each has the same form as the per-vector proxy, with feature indices restricted to $\mathcal{I}_h$, so the log-scale formulation and optimization in Section 3 apply separately to each group. The redundant common scale can likewise be removed independently in each group by imposing $\sum_{i\in\mathcal{I}_h} v_i = 0$.

3 Optimizing the Smoothing Matrix

We minimize $\tilde{D}(S)$ for fixed W, X, Cholesky diagonal $\{u_i\}$, and margin τ. We write the objective for $g = n$; the same formulation applies independently within each group.

3.1 Distortion proxy in log-scales

Set $v_i = \ln s_i^2$ and $S(v) = \mathrm{diag}(e^{v_1/2}, \ldots, e^{v_n/2})$. Then

$$\begin{aligned} F(v) &\triangleq \tilde{D}(S(v)) \\ &= \sum_{i=1}^{n}\left[(1+\tau)^2 u_i^2 e^{v_i}\frac{1}{a}\sum_{j=1}^{a}\max_{\ell\in[n]}W_{j\ell}^2 e^{-v_\ell} + \sigma_{W_i}^2 e^{-v_i}\frac{1}{b}\sum_{k=1}^{b}\max_{\ell\in[n]}X_{\ell k}^2 e^{v_\ell}\right]. \end{aligned} \tag{15}$$

3.2 Convexity and optimization in log-scales

The following proposition parallels the log-scale convexity result of Sao et al. (2026), here for a distortion proxy that accounts for GPTQ weight quantization and RTN activation quantization.

Proposition 3.1. *The function $v \mapsto F(v)$ defined in* (15) *is convex on $\mathbb{R}^n$.*

Proof. Distribute each nonnegative prefactor through the corresponding sum and maximum to obtain

$$F(v) = \frac{(1+\tau)^2}{a}\sum_{j=1}^{a}\max_{\ell\in[n]}\left\{\sum_{i=1}^{n}u_i^2 W_{j\ell}^2 e^{v_i - v_\ell}\right\} + \frac{1}{b}\sum_{k=1}^{b}\max_{\ell\in[n]}\left\{\sum_{i=1}^{n}\sigma_{W_i}^2 X_{\ell k}^2 e^{v_\ell - v_i}\right\}. \tag{16}$$

For each fixed j, ℓ or k, ℓ, the expression inside braces is a nonnegative weighted sum of exponentials of affine functions of v, and is therefore convex. A pointwise maximum of convex functions is convex, and the coefficients $(1+\tau)^2/a$ and $1/b$ are positive. Hence F is convex. This establishes convexity of the rescaled distortion proxy in log-scales. □

Since $F(v + c\mathbf{1}) = F(v)$, we remove the redundant common scale by solving

$$\min_{v\in\mathbb{R}^n:\,\sum_{i=1}^{n}v_i=0} F(v). \tag{17}$$

If $\Sigma_X \succ 0$ and every column of W is nonzero, this problem has a unique global minimizer (Proposition E.1, Appendix E).

4 THE SMOOTHGPTQ ALGORITHM

Parameters and calibration data. We specify bit width $R \geq 2$, margin $\tau > 0$, damping $\delta \geq 0$, group size $g \mid n$, autocorrelation sample count $N_\Sigma \geq b$, optimization-pool size b, minibatch size $1 \leq B \leq b$, and peak step size $\eta > 0$. Forward passes through the full-precision model on calibration texts supply input vectors X_t at each distinct projection input. We compute

$$C = \frac{1}{N_\Sigma} \sum_{t=1}^{N_\Sigma} X_t X_t^\top, \qquad \Sigma_X = C + \delta \frac{\operatorname{trace}(C)}{n} I_n, \tag{18}$$

and factor $\Sigma_X = U^\top U \succ 0$ once, using the same factor for the objective coefficients and SIC. Damping may depend on R. A retained subset of b activation vectors supplies the activation term in F.

Balancing matrices that share an input. Projections sharing X, such as Q/K/V or gate/up, share S and reuse the quantized activations. For m matrices $W^{(r)} \in \mathbb{R}^{a_r \times n}$, let $A = \sum_r a_r$. To balance their influence, normalize nonzero rows and rescale:

$$B^{(r)}_{j,:} = \frac{W^{(r)}_{j,:}}{\|W^{(r)}_{j,:}\|_2}, \qquad \xi_r = \frac{A}{m a_r}, \qquad W^{(r)}_{\mathrm{opt}} = \sqrt{\xi_r} B^{(r)}. \tag{19}$$

Concatenating these matrices for optimization gives each projection equal aggregate row weight, since $\|W^{(r)}_{\mathrm{opt}}\|_F^2 = A/m$. Actual quantization uses the original matrices $W^{(r)} S^{-1}$.

Optimizing the smoothing matrix. Calibration must estimate $\mathbb{E}\|SX\|_\infty^2$ as S changes. Capturing sufficiently many activation patterns motivates a large pool with $b \gg n$, but processing the entire pool at every update is expensive. We therefore use projected stochastic subgradients with fresh uniform minibatches of $B \ll b$ activation vectors and all weight rows, targeting the full-pool convex objective. For $g < n$, optimization separates across groups. Algorithm 2 in Appendix F specifies initialization, calibration-only step-size selection, updates, and averaging.

Offline weight quantization. With S fixed, quantize each original WS^{-1} by clipped SIC with factor US and the scales in (3), or their group counterparts. The equivalent GPTQ implementation in Appendix A uses the prescribed metric $S\Sigma_X S$ and retains feedback across group boundaries. Store the integer codes and weight scales.

Online activation quantization. Form SX and apply absmax-scaled RTN to each vector or feature group. Projections sharing an input reuse its codes and scales. Rescale the integer products by their weight and activation scales and sum group contributions as in Section 2.2; zero groups use zero codes and any positive scale.

5 EXPERIMENTS

We first validate the distortion approximation and optimization on Llama-3.1-8B-Instruct, then report downstream benchmark accuracy. GPTQ denotes $S = I$ and SQ denotes SmoothQuant. Detailed protocols and numerical settings are deferred to Appendix D.

5.1 METHOD VALIDATION

These experiments use FineWeb calibration texts and separate held-out texts. SmoothGPTQ uses $N_\Sigma = 10^6$ vectors for estimation of activation autocorrelation matrix, an optimization pool of $b = 100{,}000$, and minibatches of $B = 5{,}000$; calibration budgets differ across methods. Both weights and activations are quantized.

Distortion prediction and optimization. Figure 1 shows that optimizing S approximately halves MSE for the layer-20 output projection at $R = 6$, with predictions closely tracking empirical distortion. Across all 32 layers, the approximation is accurate at $R = 6, 8$ for most projection types, with a larger residual discrepancy for down projections (Table 1). Both use $\delta = 0$; Appendix D.6 repeats them with $\delta = 0.05$, used in the remaining validation experiments.

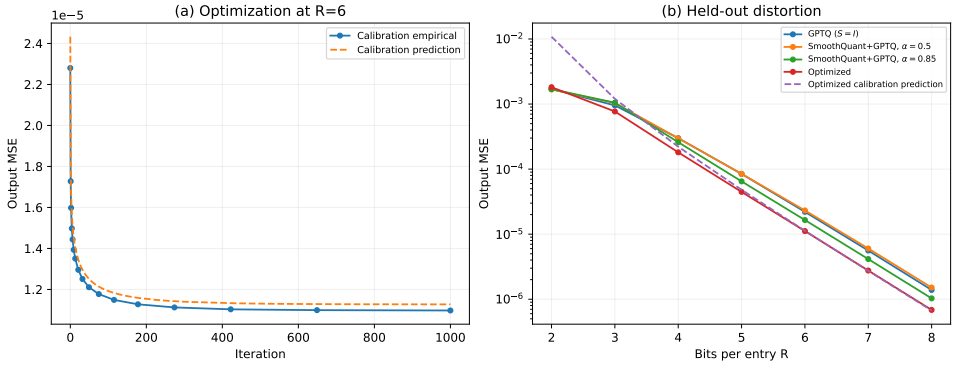


Figure 1: Layer-20 O projection, $g = n$, $\delta = 0$. Left: calibration MSE and prediction during optimization from $S = I$, $R = 6$. Right: held-out MSE and optimized calibration prediction versus bit width.

Table 1: Held-out MSE/calibration-predicted MSE at optimized S, averaged over 32 layers; $g = n$, $\delta = 0$. Values near one indicate accurate prediction.

R	Q	K	V	O	GATE	UP	DOWN
4	1.0894	1.0501	1.0315	0.9036	1.0511	1.0454	0.5075
6	1.0238	1.0296	1.0024	0.9999	1.0298	1.0278	0.9637
8	1.0112	1.0173	1.0050	1.0194	1.0301	1.0278	1.1166

Projection and end-to-end distortion. The full sweep covers all seven projection types in all 32 layers with $g = n$. Optimized scaling gives the lowest mean MSE for six of seven projection types at $R = 6$ (Table 2; V is the exception. Note that it shares the same S with Q and K), and the lowest full-model KL for $R \geq 4$ (Table 3).

Table 2: Held-out MSE at $R = 6$, $g = n$: mean $\pm$ standard deviation across 32 layers, in the indicated row units.

Projection	Unit	GPTQ	SQ+GPTQ $\alpha = 0.5$	SQ+GPTQ $\alpha = 0.85$	Optimized
Q	10^{-3}	5.71 ± 2.15	2.02 ± 0.656	2.72 ± 1.09	1.44 ± 0.478
K	10^{-2}	1.25 ± 0.499	0.496 ± 0.171	0.987 ± 0.455	0.428 ± 0.16
V	10^{-3}	2.36 ± 1.58	0.861 ± 0.784	0.465 ± 0.422	0.495 ± 0.456
O	10^{-5}	7.45 ± 18.6	6.18 ± 10	2.94 ± 3.87	2 ± 2.64
GATE	10^{-4}	8.97 ± 5.29	5.74 ± 3.65	5.43 ± 3.11	3.86 ± 2.23
UP	10^{-4}	5.83 ± 3.39	3.67 ± 2.43	2.63 ± 2.52	2.08 ± 1.27
DOWN	10^{-3}	1.82 ± 4.85	1.18 ± 2.01	1.13 ± 2.28	0.533 ± 0.709

Table 3: Full-model KL (nats/token) from the FP16 reference; R-bit weights and activations, $g = n$.

R	GPTQ	SQ+GPTQ $\alpha = 0.5$	SQ+GPTQ $\alpha = 0.85$	Optimized
2	11.037	11.907	10.865	11.919
3	10.074	12.061	10.375	11.377
4	3.7953	6.2703	9.873	3.1924
5	0.62037	0.69593	1.8281	0.37156
6	0.14786	0.1429	0.28211	0.094754
7	0.04597	0.040256	0.065692	0.028554
8	0.013115	0.01455	0.01843	0.0093525

Grouped quantization and OmniQuant. With INT4-G32, our scheme reduces KL by 27.5% relative to GPTQ (Table 4) and gives the lowest mean MSE for every projection type (Appendix C).

In a separate W6A6, $g = n$ comparison, it reduces KL by 28.0% relative to OmniQuant Shao et al. (2023), improves perplexity (Table 5), and gives lower held-out block-output MSE in 20 of 32 layers.

Table 4: INT4-G32: KL from FP16 (nats/token) and perplexity (PPL).

Method	KL	PPL
FP16	0	10.9167
GPTQ ($S = I$)	0.154212	12.4156
SQ ($\alpha = 0.5$), RTN	0.186521	12.7982
SQ ($\alpha = 0.85$), RTN	0.222819	13.2213
SQ+GPTQ ($\alpha = 0.5$)	0.172134	12.5919
SQ+GPTQ ($\alpha = 0.85$)	0.283997	13.8184
SmoothGPTQ	**0.111854**	**11.9655**

Table 5: W6A6, $g = n$: KL from BF16 (nats/token) and PPL.

Method	KL	PPL
SmoothGPTQ	**0.098385**	**11.8516**
OmniQuant	0.136668	12.4010

5.2 Benchmarking

Table 6 reports Llama-3.1-8B-Instruct accuracy on MMLU-CoT, GSM8K, HellaSwag, and WinoGrande. SmoothGPTQ with INT4-G32 reaches 75.64% average accuracy, recovering 95.83% of the FP16 baseline. These results use a separate SlimPajama calibration and optimizer configuration; competing results are reproduced from Egiazarian et al. (2026), so this is not a matched comparison with the FineWeb experiments. Appendix D.7 gives the configuration and the expanded comparison across formats. Tables 7–10 in Appendix B report the broader Llama benchmark results. In all models we have tested, SmoothGPTQ is comparable to or outperforms MR-GPTQ Egiazarian et al. (2026) for NVINT4 (this corresponds to group size $g = 16$, with an E4M3 scale per group). The advantage of SmoothGPTQ over the latter, is that it does not require a Hadamard transform, and is therefore cheaper to implement. Remarkably, SmoothGPTQ with NVINT4 Is competitive with or outperforms the NVFP4 schemes reported in Egiazarian et al. (2026), over the models we tested.

Table 6: Llama-3.1-8B-Instruct benchmark accuracy (%). All quantization schemes use INT4 for weights and activations alike, with g=32. Results for RTN,RTN+HT and GPTQ taken from Egiazarian et al. (2026).

Method	MMLU-CoT	GSM8K	HellaSwag	WinoGrande	Avg.	Recovery
FP16	72.76	85.06	80.01	77.90	78.93	100
RTN	65.96	74.68	77.62	74.19	73.11	92.63
RTN+HT	68.30	79.61	77.60	73.48	74.75	94.71
GPTQ	66.36	76.65	77.38	72.48	73.21	92.75
SmoothGPTQ	67.90	79.91	78.67	76.09	**75.64**	**95.83**

AI use statement

In this work, we used generative AI tools for assistance with writing and exposition, for the optimization algorithm for $\tilde{D}(S)$ described in Appendix F, for the formulation and proof of Proposition E.1, for the detailed implementation of the SIC algorithm using GPTQ (described in Appendix A), and for the experiments and code. We have not used generative AI tools for the conceptualization of SmoothGPTQ, for deriving the expression for $\tilde{D}(S)$ and for the design of the SmoothGPTQ algorithm. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

An anonymized implementation of SmoothGPTQ, built on `llm-compressor` and including a vLLM plugin for running the quantized checkpoints, is available at `https://anonymous.4open.science/r/smoothgptq-ECCE/`; its README maps Algorithm 2 and Appendix A to the code and lists the settings of Appendix D.7. The code for the validation experiments of Sec-

tion 5.1 is provided in the supplementary material (`SmoothGPTQ-public.zip`); their protocols, seeds and sample budgets are given in Appendices D and F.

REFERENCES

Anonymous. OSFP4: Joint optimization of diagonal smoothing and block scales for nvfp4 quantization. Submitted to ICLR 2027; under review, 2027.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. doi: 10.1007/BF02579403. URL https://doi.org/10.1007/BF02579403.

Ran Ben-Basat, William Kuszmaul, Michael Mitzenmacher, Amit Portnoy, and Shay Vargaftik. Quantizing with randomized hadamard transforms: Efficient heuristic now proven. *arXiv preprint arXiv:2605.06014*, 2026.

Johann Birnick. The lattice geometry of neural network quantization—a short equivalence proof of GPTQ and Babai's algorithm. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.01077.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36:4396–4429, 2023.

Jiale Chen, Vage Egiazarian, Roberto L Castro, Torsten Hoefler, and Dan Alistarh. Wush: Near-optimal adaptive transforms for llm quantization. *arXiv preprint arXiv:2512.00956*, 2025.

Jiale Chen, Yalda Shabanzadeh, Elvir Crnčević, Torsten Hoefler, and Dan Alistarh. The geometry of LLM quantization: GPTQ as Babai's nearest plane algorithm. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=NFB4QGGS65.

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.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Egor Lifar, Semyon Savkin, Or Ordentlich, and Yury Polyanskiy. WaterSIC: Information-theoretically (near) optimal linear layer quantization. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.04956.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.01721.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025.

LLM Compressor contributors. INT8 weight and activation quantization, 2026. URL https://docs.vllm.ai/projects/llm-compressor/en/latest/examples/quantization_w8a8_int8/. LLM Compressor documentation. Accessed September 13, 2026.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. AffineQuant: Affine transformation quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2403.12544.

Or Ordentlich and Yury Polyanskiy. Optimal quantization for matrix multiplication. *arXiv preprint arXiv:2410.13780*, 2024.

Or Ordentlich and Yury Polyanskiy. High-rate quantized matrix multiplication I. *arXiv preprint arXiv:2601.17187*, 2026a. URL https://arxiv.org/abs/2601.17187.

Or Ordentlich and Yury Polyanskiy. High-rate quantized matrix multiplication II. *arXiv preprint arXiv:2605.13768*, 2026b. URL https://arxiv.org/abs/2605.13768.

Piyush Sao, Narasinga Miniskar, Pedro Valero-Lara, Keita Teranishi, and Sudip Seal. Contraction-gauge preconditioning for quantized matrix multiplication. *arXiv preprint arXiv:2607.18745*, 2026. URL https://arxiv.org/abs/2607.18745.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. FlatQuant: Flatness matters for LLM quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2410.09426.

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

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding. *arXiv preprint arXiv:2505.22988*, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.

A Implementing Clipped SIC Using GPTQ

We describe how to obtain $Z_{\mathrm{SIC}} \in \mathrm{INT}_R^{n\times a}$ using an existing GPTQ implementation, with $\hat{\bar{W}} = Z_{\mathrm{SIC}}^\top$. Fix S and the row scales γ_j from (3), and define

$$A = \Gamma^{-1}\bar{W} \in \mathbb{R}^{a\times n}, \qquad L = US, \qquad H = L^\top L = S\Sigma_X S. \tag{20}$$

The construction assumes $\gamma_j > 0$. An identically zero weight row can be assigned any positive scale and is reconstructed as zero. Define the scalar integer quantizer

$$q_R(t) = \min\left\{2^{R-1}-1, \max\left\{-2^{R-1}, \mathrm{round}(t)\right\}\right\}. \tag{21}$$

Clipped SIC applies Algorithm 1 to each column of $Y = LA^\top$, replacing the rounding step by $z_i \leftarrow q_R(y_i/L_{ii})$. The update $y \leftarrow y - z_i L_{:,i}$ uses this clipped value. Clipping the completed output of unrestricted SIC would in general give a different result.

GPTQ inputs and output. Let $\mathrm{GPTQ}_{q_R}(B, G)$ denote GPTQ applied to a weight matrix $B \in \mathbb{R}^{a\times n}$ with Hessian $G \succ 0$, processing columns in the order $1, \dots, n$ and using the fixed quantizer (21) for every entry. Its output has the same shape as B and is integer-valued, possibly stored in a floating-point tensor. Let $J \in \mathbb{R}^{n\times n}$ reverse the coordinates, so that $Je_i = e_{n+1-i}$ and $J^\top = J^{-1} = J$. Run GPTQ with

$$B = AJ = \Gamma^{-1}\bar{W}J, \qquad G = JHJ = JS\Sigma_X SJ, \tag{22}$$

and restore the original coordinate order:

$$Q = \mathrm{GPTQ}_{q_R}(B, G), \tag{23}$$

$$Z_{\mathrm{SIC}} = JQ^\top, \qquad \hat{\bar{W}} = QJ. \tag{24}$$

Thus GPTQ receives an $a \times n$ weight matrix, rather than its transpose. The factor L used to initialize SIC is represented by the Hessian G in the GPTQ call. The permutation is needed only during offline quantization; the recovered weights use the original feature order.

Why the procedures agree. GPTQ compensates each quantization decision by minimizing the quadratic error over the remaining unquantized coordinates Frantar et al. (2023); Chen et al. (2026); Birnick (2026). For a row j, put $y^{(0)} = LA_j$, where $A_j = (A_{j,:})^\top$. Once coordinates $i+1, \dots, n$ have been fixed, triangularity of L implies that the real-valued minimizer over the remaining coordinates satisfies

$$t_i = \frac{y_i^{(0)} - \sum_{k=i+1}^{n} L_{ik} z_k}{L_{ii}}. \tag{25}$$

Indeed, the first i residual coordinates can be set to zero by solving the leading triangular system, while the remaining residual coordinates no longer depend on the free variables. Both reverse-order GPTQ and SIC therefore choose $z_i = q_R(t_i)$ and continue with the same fixed coordinates. Induction gives (24), including when clipping is active. This is an exact-arithmetic statement with the same rounding convention in both procedures; floating-point implementations can differ near rounding thresholds.

Required configuration. The following settings specify the GPTQ black box used above.

1. *Fixed quantization grid.* Use signed R-bit integers with the full range in (21), scale one, and zero point zero. Disable scale estimation or optimization after forming B. A preset that recomputes absmax or group scales generally implements a different quantizer. Use the same rule for rounding ties as in SIC.
2. *Fixed column order.* Disable activation-based ordering and any other column permutation inside GPTQ. The input reversal J already implements the order required by SIC.
3. *Matching Hessian.* Supply G without further damping. If regularization is needed, the effective metric used by GPTQ must be the same metric used to define L. In particular, when Σ_X already includes damping, form $G = JS\Sigma_X SJ$ from that matrix and disable additional damping. A positive global multiplier on G does not change the decisions.
4. *Clipped feedback.* Use q_R at each decision and propagate the error relative to that clipped value. GPTQ may batch its error updates in blocks; with a fixed grid, this preserves the same sequential decisions in exact arithmetic. This computational blocking is distinct from using different quantization scales for different groups of columns.

If the interface constructs its Hessian from calibration activations, supply $X_{\mathrm{rev}} = JSX$, since

$$\frac{1}{b} X_{\mathrm{rev}} X_{\mathrm{rev}}^\top = JS\Sigma_X SJ \tag{26}$$

for the empirical covariance. In a samples-by-features interface, supply $X_{\mathrm{rev}}^\top$. Use the unquantized smoothed activations for this calibration. Any regularization must still satisfy the matching-metric requirement above.

Using prescribed row scales directly. An equivalent implementation avoids explicitly normalizing the weights. Supply $B = \bar{W}J$ with the same Hessian G, and fix the quantizer for row j to

$$q_{R,j}(t) = \gamma_j q_R(t/\gamma_j). \tag{27}$$

If GPTQ returns the dequantized weight matrix $Q_{\mathrm{deq}} \in \mathbb{R}^{a\times n}$, recover

$$Z_{\mathrm{SIC}} = JQ_{\mathrm{deq}}^{\top}\Gamma^{-1}, \qquad \Gamma\hat{\bar{W}} = Q_{\mathrm{deq}}J. \tag{28}$$

Positive scaling of an entire row commutes with the GPTQ error updates, so these are the same integer decisions. Numerically, round the recovered codes to integers to remove floating-point division error.

Example: `llmcompressor`. The package's low-level `quantize_weight` routine accepts a module and an explicit Hessian, reads quantization parameters from `module.weight_observer.get_qparams()`, and returns dequantized weights.[3] A wrapper can therefore set the module weights to $\bar{W}J$, use channel-wise signed integer quantization, and supply a fixed observer returning scales $(\gamma_j)_{j=1}^{a}$ as an $a \times 1$ tensor, zero points zero, and no global scale. Set `quant_args.actorder=None` and `percdamp=0`, and pass a copy of G, which the routine modifies. Extract `weight` from the returned parameter dictionary and recover the codes as above. Check that `used_rtn_fallback` is false: an inversion failure followed by RTN does not implement SIC. This wrapper reuses the GPTQ updates; the package's automatic scale selection must be bypassed to preserve the prescribed grid.

A.1 Quantization with Group Scales

Once S has been optimized and fixed, form $\bar{W} = WS^{-1}$ and compute the weight scales γ_{jh} from Section 2.2. These scales remain fixed throughout quantization. To execute clipped SIC directly, initialize $y = US\bar{W}_j$ for each row j and, for $i = n, \dots, 1$ with $h(i) = \lceil i/g \rceil$, apply

$$z_i = q_R\left(\frac{y_i}{u_i s_i \gamma_{j,h(i)}}\right), \tag{29}$$

$$y \leftarrow y - \gamma_{j,h(i)} z_i (US)_{:,i}. \tag{30}$$

Store $\hat{\bar{W}}_{ji} = z_i$ together with the group scales. The update uses the reconstructed value $\gamma_{j,h(i)}z_i$ and propagates its effect across group boundaries.

Execution using GPTQ. Use the direct-scale construction above with $B = \bar{W}J$ and $G = JS\Sigma_X SJ$. Since reversal maps group r to original group $n/g + 1 - r$, assign the reversed scale table

$$\gamma^{\mathrm{rev}}_{jr} = \gamma_{j,n/g+1-r}, \qquad r = 1, \dots, n/g. \tag{31}$$

For each entry in row j and reversed group r, use the fixed quantizer $t \mapsto \gamma^{\mathrm{rev}}_{jr} q_R(t/\gamma^{\mathrm{rev}}_{jr})$. Run a single GPTQ sweep over all n columns with the full Hessian, retaining error compensation across groups. Independent optimization of the smoothing coefficients in Section 2.2 does not justify separate GPTQ calls on the groups. The same sequential-minimization argument given above establishes equivalence to clipped SIC with these coordinate-dependent grids.

In `llmcompressor`, select `QuantizationStrategy.GROUP` and `group_size=g`. Supply the fixed observer with the $a \times (n/g)$ scale table γ^{rev}, zero points zero of the same shape, and no global scale. Keep activation ordering disabled and the Hessian damping configuration as specified above. After obtaining the dequantized output Q_{deq}, undo reversal and recover the codes group by group:

$$\hat{\bar{W}}_{:,\mathcal{I}_h} = \Gamma_h^{-1}(Q_{\mathrm{deq}}J)_{:,\mathcal{I}_h}. \tag{32}$$

Round these recovered codes to integers to remove floating-point division error.

[3] Implementation: `https://github.com/vllm-project/llm-compressor/blob/main/src/llmcompressor/modifiers/gptq/gptq_quantize.py`.

Online activation quantization and multiplication. For each incoming activation vector X_k, form $\bar{X}_k = SX_k$, compute $\lambda_{kh} = \max_{i \in \mathcal{I}_h} |\bar{X}_{ik}|/\beta_R$, and set $\hat{\bar{X}}_{ik} = q_R(\bar{X}_{ik}/\lambda_{kh})$ for $i \in \mathcal{I}_h$. An all-zero activation group is represented by zero codes and any positive scale. Compute the integer matrix products within each group, rescale each by its weight and activation scales, and sum the resulting contributions as in Section 2.2.

B Additional Benchmark Results

Table 7: Performance of Llama-3.2-1B-Instruct for different weight & activation quantization settings.

Format	Quantization	MMLU	GSM8k	HellaSwag	WinoGrande	Avg.	Recovery%
–	FP16	46.20	46.32	59.78	61.56	53.47	–
INT8	RTN	45.90	44.20	59.80	61.30	52.80	99.55
	GPTQ	45.40	44.90	59.60	60.10	52.50	98.99
FP8	RTN	46.10	44.70	59.50	59.50	52.50	98.99
	GPTQ	45.80	45.00	59.10	60.60	52.63	99.22
NVFP4	RTN	36.08	31.39	54.77	57.22	44.87	83.91
	RTN+Had16	32.80	25.02	56.24	59.04	43.28	80.94
	RTN+Had128	38.28	29.95	54.27	58.41	45.23	84.59
	GPTQ	37.79	29.80	55.48	60.22	45.82	85.71
	GPTQ+Had16	38.99	32.98	56.66	58.17	46.70	87.35
	GPTQ+Had128	35.47	31.16	57.02	59.19	45.71	85.50
	QAT	27.85	38.51	57.52	60.30	46.05	86.12
	QAT+Had16	32.72	37.60	57.53	58.41	46.57	87.09
NVINT4	RTN	37.33	26.08	52.62	58.56	43.65	81.64
	RTN+Had16	33.41	32.52	57.12	59.19	45.56	85.21
	GPTQ	37.15	27.60	55.94	59.35	45.01	84.19
	MR-GPTQ	36.69	33.36	57.95	58.96	46.74	87.42
	SmoothGPTQ	36.73	36.16	58.74	59.83	47.86	89.52

Table 8: Performance of Llama-3.2-3B-Instruct for different weight & activation quantization settings.

Format	Quantization	MMLU	GSM8k	HellaSwag	WinoGrande	Avg.	Recovery%
–	FP16	64.43	78.01	73.42	70.09	71.49	–
INT8	RTN	64.00	77.70	73.30	69.60	71.15	99.47
	GPTQ	64.00	77.90	73.40	69.90	71.30	99.67
FP8	RTN	63.40	77.70	73.00	69.70	70.95	99.19
	GPTQ	64.10	77.80	73.00	69.90	71.20	99.54
NVFP4	RTN	60.62	70.43	70.99	68.03	67.52	94.45
	RTN+Had16	59.91	64.82	69.77	65.59	65.02	90.96
	RTN+Had128	54.34	67.48	69.69	66.93	64.61	90.38
	GPTQ	61.76	70.36	71.07	69.93	68.28	95.51
	GPTQ+Had16	60.26	68.76	71.05	67.80	66.97	93.68
	GPTQ+Had128	60.19	70.89	70.97	68.19	67.56	94.51
	MicroQAT+Had16	60.66	69.98	70.55	67.01	67.05	93.79
	QAT	62.06	75.06	71.27	67.96	69.09	96.64
	QAT+Had16	62.03	72.93	70.95	66.46	68.09	95.25
NVINT4	RTN	60.22	71.65	70.92	66.77	67.39	94.27
	RTN+Had16	56.75	71.95	69.24	66.61	66.14	92.52
	GPTQ	60.50	71.42	27.01	51.07	52.50	73.44
	MR-GPTQ	60.80	72.25	71.74	70.48	68.82	96.27
	SmoothGPTQ	61.64	73.24	72.02	69.30	69.05	96.58

Table 9: Performance of Llama-3.1-8B-Instruct for different weight & activation quantization settings.

Format	Quantization	MMLU-CoT	GSM8k	HellaSwag	WinoGrande	Avg.	Recovery%
–	FP16	72.80	85.10	80.00	77.90	78.90	–
INT8	RTN	72.50	84.80	80.20	77.40	78.73	99.74
	GPTQ	72.40	84.40	80.00	77.30	78.53	99.48
FP8	RTN	72.40	84.70	79.80	77.70	78.65	99.64
	GPTQ	71.80	84.50	79.90	78.10	78.58	99.55
NVFP4	RTN	68.70	78.70	78.40	73.40	74.80	94.80
	RTN+Had	67.00	77.40	77.30	74.40	74.00	93.80
	RTN+Had128	66.60	77.00	77.50	75.50	74.10	93.90
	GPTQ	68.60	79.60	78.70	75.50	75.60	95.70
	GPTQ+Had	69.40	79.60	78.40	75.10	75.60	95.80
	GPTQ+Had128	68.90	79.50	78.30	73.60	75.10	95.10
	QAT	68.20	79.80	78.90	74.40	75.30	95.40
	QAT+Had	68.90	81.60	79.00	75.10	76.10	96.50
NVINT4	RTN	68.56	78.17	78.64	75.14	75.13	95.18
	RTN+Had16	68.59	81.73	78.38	74.27	75.74	95.96
	GPTQ	68.69	81.58	77.59	73.40	75.32	95.42
	MR-GPTQ	69.71	82.26	79.14	75.53	76.66	97.12
	SmoothGPTQ	69.50	81.35	79.33	76.01	76.55	97.02

Table 10: Performance of Llama-3.3-70B-Instruct for different weight & activation quantization settings.

Format	Quantization	MMLU	GSM8k	HellaSwag	WinoGrande	Avg.	Recovery%
–	FP16	86.55	95.07	86.22	84.93	88.19	–
NVFP4	RTN	85.50	93.48	85.63	83.27	86.97	98.61
	RTN+Had16	85.02	93.63	84.97	83.82	86.86	98.49
	RTN+Had128	85.24	91.81	84.91	83.35	86.33	97.89
	GPTQ	85.54	94.09	85.49	84.37	87.37	99.07
	GPTQ+Had16	85.58	93.40	85.45	82.40	86.71	98.32
	GPTQ+Had128	85.59	94.16	85.56	84.77	87.52	99.24
NVINT4	SmoothGPTQ	85.42	95.07	86.05	84.21	87.69	99.43

C GROUPED PROJECTION DISTORTION

Table 11 reports the mean local distortion for each of the six methods in Section 5.1. Each entry averages the 32 layer-level distortions with equal weight per layer, using 5,000 common held-out reference-model input vectors per projection. Values are rounded to one decimal place after division by the row's unit; standard deviations are omitted. These are absolute output MSE values, not errors normalized by output energy. Percentage comparisons in the main text use the unrounded means.

Table 11: Mean INT4-G32 local distortion across all 32 layers. Multiply each entry by the unit in its row. SQ denotes SmoothQuant; all methods use activation RTN. Optimized scaling uses the 100K calibration pool and 5K minibatches; the baselines retain their original calibration budgets.

Projection	Unit	GPTQ $S = I$	SQ+RTN $\alpha = 0.5$	SQ+RTN $\alpha = 0.85$	SQ+GPTQ $\alpha = 0.5$	SQ+GPTQ $\alpha = 0.85$	SmoothGPTQ
Q	10^{-3}	8.4	7.2	8.1	5.1	5.7	4.3
K	10^{-2}	1.8	1.8	2.3	1.2	1.6	1.1
V	10^{-3}	3.7	2.8	2.2	2.3	1.8	1.8
O	10^{-5}	9.6	12.3	10.9	8.6	7.3	6.0
GATE	10^{-3}	2.1	2.6	2.5	2.2	2.0	1.8
UP	10^{-3}	1.4	1.7	1.5	1.4	1.2	1.2
DOWN	10^{-4}	6.6	8.5	12.1	6.7	9.2	3.5

D EXPERIMENTAL DETAILS

This appendix documents the protocols and numerical results underlying Figure 1 and Tables 1–6. Quantization and evaluation settings are matched within each comparison, while calibration procedures and sample budgets may differ across methods.

D.1 SINGLE-LAYER DISTORTION AND CALIBRATION ACCURACY

We first examine the distortion approximation and smoothing optimization in isolation, using the attention output-projection matrix of layer 20 (zero-based indexing) of Llama-3.1-8B-Instruct, $W \in \mathbb{R}^{4096\times4096}$. We collect activations from FineWeb texts in 2,048-token contexts, excluding the first 128 positions and special tokens from sampling, with seed 20260910. One million calibration token positions provide the autocorrelation estimate Σ_{raw}. For Figure 1, we set $\Sigma_X = \Sigma_{\mathrm{raw}}$ ($\delta = 0$), without adding numerical regularization. This same undamped estimate supplies the coefficients u_i in the distortion proxy and the metric for SIC/GPTQ. Appendix D.6 gives the corresponding experiment with $\Sigma_X = \Sigma_{\mathrm{raw}} + \delta\,\mathrm{tr}(\Sigma_{\mathrm{raw}})I_n/n$ and $\delta = 0.05$. Our smoothing optimization uses $b = 100{,}000$ calibration positions with minibatches of $B = 5{,}000$. The SmoothQuant baselines retain their original statistics from 5,000 calibration positions; their calibration budget is therefore smaller. Evaluation uses 5,000 token positions from 25 FineWeb documents disjoint from the calibration corpus, with 200 positions sampled per document. All activations are collected from the original model using BF16 inference. Both weights and activations use R-bit quantization with per-vector scaling $g = n = 4096$ and weight safety margin $\tau = 0.05$. We simulate clipped SIC and activation RTN in double precision with unquantized scales and measure the full output MSE, including all cross terms.

Optimization. We run Algorithm 2 on the original weight matrix for 1,000 minibatch updates and return the average of the final 200 log-scale iterates. For Figure 1, we deliberately use $\alpha = 0$, hence $S = I$ ($v = 0$), to illustrate improvement along the trajectory; the default initialization for the remaining experiments is $\alpha = 1/2$.

Step-size selection. We retain the peak rate $\eta \approx 79.5181$ selected by the preliminary calibration-only search in Appendix F.3 with $\delta = 0.05$, so the damping comparison does not introduce a new rate search. The 100K-pool run uses this rate for the first 500 updates, followed by linear decay to $\eta/10$ over the next 500. No stability retry was needed for this figure.

Baselines. We compare the optimized scales with $S = I$, labeled GPTQ, and SmoothQuant scaling Xiao et al. (2023) with balancing parameters $\alpha_{\mathrm{SQ}} \in \{0.5, 0.85\}$, labeled SmoothQuant+GPTQ. In our scaling convention, the latter uses $s_i \propto (\max_j |W_{ji}|)^{1-\alpha_{\mathrm{SQ}}}/(\max_{k\in\mathrm{cal}} |X_{ik}|)^{\alpha_{\mathrm{SQ}}}$. All choices use the same weight SIC, activation RTN, and safety margin. Each method's S is reused for every R, since the factor $1/(12\beta_R^2)$ does not affect the minimizer of the proxy.

Calibration and held-out comparisons. Figure 1(a) compares the calibration prediction $\tilde{D}_g(S)/(12\beta_R^2)$ with empirical distortion on all 100,000 calibration vectors, at $R = 6$. The prediction uses the large-sample undamped covariance described above. It is recorded at every iteration; empirical distortion is evaluated at 17 checkpoints, including initialization, with more checkpoints near the beginning. Through update 800, each checkpoint uses the current iterate; thereafter it uses the running average of log iterates from update 801 onward, so the final point is the returned S. Figure 1(b) instead measures empirical distortion on the 5,000 held-out vectors for $R = 2, \ldots, 8$. It also shows the calibration-derived prediction for the optimized scales, allowing the generalization of that prediction to be assessed separately from its calibration accuracy. Activation group scales are computed from each incoming vector in both comparisons.

The calibration-derived prediction differs from measured calibration distortion by at most 6.69% at the plotted checkpoints. At $R = 6$, optimization reduces measured calibration distortion from 2.2803×10^{-5} to 1.0973×10^{-5}, a 51.88% reduction relative to $S = I$; the final prediction is 1.1272×10^{-5}. The recorded full-pool proxy improves by 0.0115% over the last 100 updates. Held-out distortion decreases from 2.2046×10^{-5} for GPTQ to 1.1166×10^{-5} for the optimized method, a 49.35% reduction. SmoothQuant+GPTQ attains held-out distortions of 2.3174×10^{-5}

and 1.6491×10^{-5} for $\alpha_{\mathrm{SQ}} = 0.5$ and 0.85, respectively. The optimized method's held-out distortion is 0.991 times its calibration-derived prediction.

D.2 Full-Sweep Distortion and Model Distributions

We summarize the completed sweep over all seven projection types in all 32 layers of Llama-3.1-8B-Instruct. The four methods are GPTQ with $S = I$, SmoothQuant+GPTQ with $\alpha_{\mathrm{SQ}} = 0.5$ and 0.85, and the optimized scaling from Section 4. Both weights and activations use $R \in \{2, \dots, 8\}$ bits with per-vector scales ($g = n$), $\tau = 0.05$, and damping $\delta = 0.05$. The integer codebook is $\{-2^{R-1}, \dots, 2^{R-1} - 1\}$, and scales are unquantized. The same million-token covariance estimate is used for the proxy coefficients and SIC/GPTQ.

Optimization uses $b = 100{,}000$ calibration vectors, fresh minibatches of $B = 5{,}000$, initialization $\alpha = 1/2$, and the 1,000-update schedule and tail averaging in Algorithm 2. The Q/K/V and gate/up matrices share S and use the row balancing in Section 4; the separate output and down projections use their original rows. The GPTQ and SmoothQuant baselines retain their original settings; SmoothQuant statistics use $2n$ calibration vectors and the original weights. Thus, quantization and evaluation settings are matched, while calibration sample budgets differ. Actual quantization and reported distortion always use the original matrices.

Projection distortion. For each method, rate, and projection type, we first measure $D_{\ell,p} = \|W_{\ell,p}X_{\ell,p} - \widehat{W_{\ell,p}X_{\ell,p}}\|_F^2/(a_{\ell,p}b_{\mathrm{eval}})$ separately in each layer, using the same 5,000 held-out activation vectors collected from the original model in BF16. We then report

$$\overline{D}_p = \frac{1}{32}\sum_{\ell=0}^{31} D_{\ell,p}, \qquad s_p = \left(\frac{1}{31}\sum_{\ell=0}^{31}(D_{\ell,p} - \overline{D}_p)^2\right)^{1/2}.$$

Thus, the standard deviation describes variation across layers, rather than sampling uncertainty in the mean. These local distortions use common reference-model inputs; upstream quantization errors are included only in the end-to-end evaluation. Table 2 gives the $R = 6$ local-distortion summary.

End-to-end KL divergence. We compare the next-token distributions on identical held-out FineWeb prefixes. The reference model runs in FP16. Quantized models retain the completed sweep's FP32 emulation of the reconstructed integer weights and activations, with TF32 disabled and unquantized scales. All 224 decoder projections are quantized; embeddings, the vocabulary projection, and nonlinear operations remain unquantized. No separate KV-cache quantizer is applied. We score all 48,000 non-special target tokens at positions 128–2047 in the same 25 evaluation documents. For each position, we sum over the full vocabulary and then average over positions:

$$\overline{D}_{\mathrm{KL}} = \frac{1}{T}\sum_{t=1}^{T}\sum_{z\in\mathcal{V}} P_{\mathrm{FP16}}(z \mid c_t)\log\frac{P_{\mathrm{FP16}}(z \mid c_t)}{P_{\mathrm{quantized}}(z \mid c_t)}, \qquad T = 48{,}000.$$

Here c_t is the common reference-text prefix; log-softmax and KL accumulation use FP64. This metric is measured in nats per token and is not inferred from perplexity. As a numerical control, the unquantized FP32 model has divergence 7.3359×10^{-6} from the FP16 reference.

D.3 Matched INT4-G32 Comparison

We next compare six methods with 4-bit weights and activations and group size $g = 32$: GPTQ with $S = I$, SmoothQuant with $\alpha_{\mathrm{SQ}} \in \{0.5, 0.85\}$ and weight RTN, the same two SmoothQuant choices followed by GPTQ, and our optimized scaling followed by GPTQ. Activation quantization uses RTN throughout. Within the GPTQ comparisons, the deployed quantization procedure differs only through S; the calibration procedures and sample budgets used to choose S differ. The RTN baselines additionally replace weight error compensation by independent rounding.

Protocol. All 224 decoder projections of Llama-3.1-8B-Instruct are quantized. We retain the million-token FineWeb covariance estimate, damping $\delta = 0.05$, a $b = 100{,}000$ optimization pool

with $B = 5{,}000$ minibatches, initialization $\alpha = 1/2$, and shared-matrix row balancing from Section D.2. Each group follows the 1,000-update minibatch schedule, tail averaging, and calibration-only peak-rate selection of Appendix F.3. Q/K/V share S, as do gate/up; the weight matrices used for quantization are the originals. Each method's S is normalized to have geometric mean one within each group. Both SmoothQuant variants retain their original statistics from $2n$ calibration vectors and use the original weights; their RTN and GPTQ versions share identical smoothing scales.

These completed measurements use the balanced grid $\{-\beta_R, \ldots, \beta_R\}$, with $\beta_R = 2^{R-1} - 1$; thus INT4-G32 uses $[-7, 7]$. Before BF16 rounding, the weight step in group h of row j is $(1 + \tau)\|\bar{W}_{j,\mathcal{I}_h}\|_\infty/\beta_R$, with $\tau = 0.05$, and the activation step is $\|\bar{X}_{\mathcal{I}_h,k}\|_\infty/\beta_R$. Weight steps are fixed before GPTQ, while activation steps are computed per incoming token and group. GPTQ uses the same full damped covariance for all methods, processes coordinates in reverse order, and retains feedback across groups. No rotations or additional Hessian correction are applied.

Evaluation uses FP16 model arithmetic with BF16 group scales and FP64 quantization decisions and error accumulation; TF32 is disabled. Embeddings and the vocabulary projection remain in FP16. Local distortion is computed from FP16 linear outputs on 5,000 common reference-model activation vectors, sampled at 200 positions from each of 25 held-out FineWeb documents. The calibration activations remain those collected from the original BF16 model. End-to-end KL uses the same 48,000 target positions and full-vocabulary definition as Section D.2, with a fresh FP16 reference in the same evaluation environment. Consequently, these grouped measurements differ in grid and arithmetic from the earlier $g = n$ sweep.

Results. Table 4 reports end-to-end KL and perplexity. Optimized scaling gives the lowest KL among the six methods: 0.111854 nats/token, a 27.5% reduction from GPTQ with $S = I$ and a 35.0% reduction from the better SmoothQuant+GPTQ baseline ($\alpha_{\mathrm{SQ}} = 0.5$). It also gives the lowest mean local distortion for each of the seven projection types; Appendix C presents all six methods in one table of rounded means. The gain for V over SmoothQuant+GPTQ with $\alpha_{\mathrm{SQ}} = 0.85$ is small, approximately 1.1%, so the benefit is not uniform across projection types. These results support optimizing S under matched grouped quantization; they describe one fixed calibration/evaluation split rather than a significance test or a comparison across downstream benchmarks.

D.4 Comparison with OmniQuant

We compare our scheme with OmniQuant Shao et al. (2023) on all 32 decoder blocks of Llama-3.1-8B-Instruct using W6A6 and $g = n$. Both methods use the balanced integer grid $[-31, 31]$, BF16 quantization steps, and BF16 matrix multiplications; embeddings, the final vocabulary projection, and attention matrix multiplications remain unquantized. Our scheme uses the million-token autocorrelation estimate, $\delta = \tau = 0.05$, a 100,000-vector pool with 5,000-vector minibatches, initialization $\alpha = 1/2$, the schedule and averaging of Algorithm 2, and shared-matrix balancing. OmniQuant learns its equivalent transformations and clipping parameters for 20 epochs on 128 FineWeb documents of length 2,048, using AdamW with batch size one, learning rates 0.005 for transformation parameters and 0.01 for clipping parameters, and no weight decay. Shifts are disabled for this Llama model; OmniQuant's learned parameters and folded normalization gains are retained in FP32. Each block is calibrated independently on inputs from the original BF16 model.

Table 5 reports full-model forward KL and perplexity on 48,000 target positions from 25 held-out FineWeb documents, using a common BF16 reference. Our scheme reduces KL by 28.0% relative to OmniQuant and also achieves lower perplexity. On 5,000 common held-out token positions, our scheme achieves smaller block-output MSE in 20 of the 32 layers. This MSE compares each quantized block with its BF16 counterpart on the same reference-model inputs, whereas KL and perplexity evaluate the model with all quantized blocks installed.

The BF16 reference has perplexity 10.9177 on the common evaluation set.

D.5 Accuracy of the Calibration Prediction

We evaluate the distortion approximation at the optimized scaling $S_\star$ for $R \in \{4, 6, 8\}$ and $g = n$. For each projection and layer, we divide its empirical distortion on 5,000 held-out activation vectors by its calibration-derived prediction $\tilde{D}(S_\star)/(12\beta_R^2)$, then average these ratios over the 32

layers. The prediction uses all $b = 100{,}000$ calibration vectors at the returned $S_\star$ and the million-token covariance estimate with $\delta = 0$ for both optimization and SIC/GPTQ; $\tau = 0.05$. Shared-matrix balancing is used to select $S_\star$, but each projection's prediction is evaluated using its original, unbalanced weights. The balanced integer grids, BF16 quantization steps, and FP16 local evaluation arithmetic follow Section D.3.

The optimization uses initialization $\alpha = 1/2$, the same 100K pool and 5K minibatches, and the previously selected peak rates.

Table 1 shows mean ratios within 3.1% of one at $R = 6, 8$ for all projection types except down. The down-projection mean ratios are 0.508, 0.964, and 1.117 at $R = 4, 6, 8$, respectively. Thus, the approximation is accurate at high rates for most projections, while the residual down-projection discrepancy persists without damping.

D.6 Effect of Autocorrelation Damping

Figure 2 and Table 12 repeat the main-text Figure 1 and Table 1 with $\delta = 0.05$. For each damping value, the same covariance supplies the optimization coefficients and SIC/GPTQ feedback. We reoptimize S while retaining each experiment's sample positions, minibatches, initialization, peak-rate source, quantization conventions, and evaluation arithmetic. Figure baselines retain their original smoothing scales but are requantized with the corresponding autocorrelation.

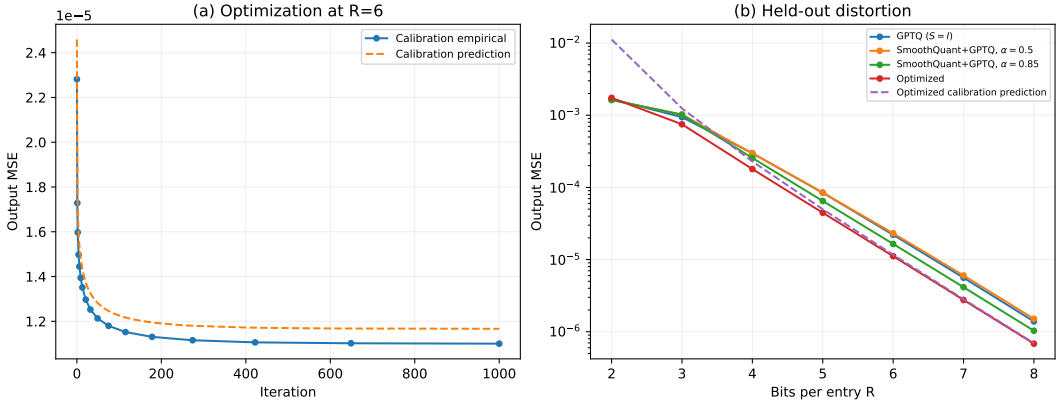


Figure 2: Damped counterpart of Figure 1: layer-20 O projection, $g = n$, $\delta = 0.05$. **(a)** Calibration MSE and prediction during optimization from $S = I$, at $R = 6$. **(b)** Held-out MSE versus bit width, with the optimized calibration prediction.

Table 12: Damped counterpart of Table 1: mean layer-wise held-out empirical/calibration-predicted MSE ratio at optimized $S_\star$, with $g = n$ and $\delta = 0.05$.

Projection	$R = 4$	$R = 6$	$R = 8$
Q	1.0025	0.9529	0.9402
K	0.9573	0.9409	0.9308
V	0.9783	0.9570	0.9622
O	0.8394	0.9429	0.9627
GATE	0.9978	0.9771	0.9774
UP	1.0009	0.9835	0.9838
DOWN	0.4941	0.9475	1.0991

For Figure 2, the calibration-derived prediction differs from measured calibration distortion by at most 7.77% at the plotted checkpoints. At $R = 6$, optimization reduces measured calibration distortion from 2.2817×10^{-5} to 1.1002×10^{-5}, a 51.78% reduction relative to $S = I$; the final prediction is 1.1666×10^{-5}. The recorded full-pool proxy improves by 0.0103% over the last 100 updates. Held-out distortion decreases from 2.2034×10^{-5} for GPTQ to 1.1142×10^{-5} for the optimized method, a 49.43% reduction. SmoothQuant+GPTQ attains held-out distortions of 2.3145×10^{-5} and 1.6476×10^{-5} for $\alpha_{\mathrm{SQ}} = 0.5$ and 0.85, respectively. The optimized method's held-out distortion is 0.955 times its calibration-derived prediction.

Removing damping improves the final calibration empirical/predicted ratio from 0.943 to 0.974 and the held-out ratio at $R = 6$ from 0.955 to 0.991, while empirical distortion changes little. Across layers, removing damping improves the high-rate prediction accuracy for Q/K/V/O, whereas gate/up shift from slight overprediction to approximately 3% underprediction. The down-projection ratio improves modestly at $R = 6$ but moves farther from one at $R = 8$. Damping therefore explains part of the prediction mismatch, but does not explain the remaining down-projection discrepancy.

D.7 Benchmarking Protocol and Expanded Comparison

The following protocol applies to Table 6, a separate benchmark run from the FineWeb validation experiments. We retain its original optimizer and calibration settings here.

Model and quantization formats. We evaluate SmoothGPTQ on Llama-3.1-8B-Instruct. All linear projections of the 32 decoder blocks (`q/k/v/o`, `gate/up/down`) are quantized to 4-bit weights and 4-bit input activations; the embedding and `lm_head` remain in BF16. Here we report **INT4-G32**: symmetric INT4 on the balanced grid $[-7, 7]$ with one BF16 scale per 32 contiguous elements, for both weights (static) and activations (dynamic, per token-group). No Hadamard or other rotation is applied.

Calibration. We calibrate on 1,248 sequences of 2,048 tokens drawn from SlimPajama (**?**), class-balanced across its six RedPajama source domains (208 sequences each from C4, CommonCrawl, Wikipedia, GitHub, StackExchange and ArXiv; sampling seed 42).

Method configuration. SmoothGPTQ first optimizes per-channel smoothing scales with the group-aware separable objective (matching the scale granularity of the target format), using Algorithm 2 with the settings of Section 4: $\delta = \tau = 0.05$, $\alpha = 1/2$, a pool of 100,000 calibration activation vectors sampled per layer, minibatches of 5,000, and 1,000 updates; the average of the last 200 log-scale iterates is retained. GPTQ then quantizes the smoothed weights with block size 128, processing columns last-to-first so that the effective per-column error weights coincide with the Cholesky diagonal used by the smoothing objective. GPTQ uses the same damped covariance $S\Sigma_X S$ as the smoothing objective, with no additional Hessian correction.

Evaluation. We follow the evaluation protocol of Egiazarian et al. (2026): WinoGrande (5-shot, accuracy), HellaSwag (10-shot, normalized accuracy), GSM8K (8-shot, strict-match exact match, Llama chat template) and MMLU-CoT (0-shot, Llama chat template), all via lm-evaluation-harness on vLLM. Quantized models are executed with exact fake-quantization kernels (BF16 GEMM over dequantized weights and group-quantized activations), reproducing the arithmetic of the corresponding W4A4 kernels. Baseline (FP16) and competing method results are taken from Egiazarian et al. (2026); recovery is computed against their FP16 baseline average of 78.93.

Expanded comparison underlying Table 6. Original results across integer and floating-point formats; all rows except ours are reproduced from Egiazarian et al. (2026).

Format	Method	MMLU-CoT	GSM8k	HellaSwag	WinoGrande	Avg.	Recovery %
Baseline	FP16	72.76	85.06	80.01	77.90	78.93	100
INT8	RTN	72.50	84.80	80.20	77.40	78.73	99.74
	GPTQ	72.40	84.40	80.00	77.30	78.53	99.48
FP8	RTN	72.40	84.70	79.80	77.70	78.65	99.64
	GPTQ	71.80	84.50	79.90	78.10	78.58	99.55
INT4	RTN	65.96	74.68	77.62	74.19	73.11	92.63
	RTN+HT	68.30	79.61	77.60	73.48	74.75	94.71
	GPTQ	66.36	76.65	77.38	72.48	73.21	92.75
	SmoothGPTQ	67.90	79.91	78.67	76.09	**75.64**	**95.83**
NVINT4	RTN	68.56	78.17	78.64	75.14	75.13	95.18
	RTN+HT	68.59	81.73	78.38	74.27	75.74	95.96
	GPTQ	68.69	81.58	77.59	73.40	75.32	95.42
	MR-GPTQ	69.71	82.26	79.14	75.53	**76.66**	**97.12**
	SmoothGPTQ	69.50	81.35	79.33	76.01	76.55	96.98
NVFP4	RTN	68.26	78.39	78.15	74.11	74.73	94.67
	RTN+HT	67.41	78.01	77.31	73.48	74.05	93.82
	QuaRot	66.50	77.40	77.25	75.14	74.10	93.80
	SpinQuant	66.50	76.10	76.96	75.32	73.70	93.40
	SmoothQuant	68.90	79.50	79.50	74.70	75.70	95.90
	GPTQ	68.85	82.60	78.26	74.51	75.72	95.92
	MR-GPTQ	69.12	80.80	78.17	75.24	75.84	96.08

E Existence and Uniqueness of the Optimal Log-Scales

Proposition E.1. *Assume Σ_X is positive definite and every column of W is nonzero. Then F is strictly convex on the hyperplane $\sum_{i=1}^n v_i = 0$, and problem* (17) *has a unique global minimizer.*

Proof of Proposition E.1. We first establish strict convexity on the hyperplane $\sum_i v_i = 0$. Since every column of W is nonzero, $W \neq 0$. Choose a nonzero row j of W and, for each ℓ with $W_{j\ell} \neq 0$, define

$$f_{j\ell}(v) = W_{j\ell}^2 \sum_{i=1}^n u_i^2 e^{v_i - v_\ell}. \tag{33}$$

For any nonzero direction $d \in \mathbb{R}^n$ satisfying $\sum_i d_i = 0$,

$$\frac{d^2}{dt^2} f_{j\ell}(v + td)\bigg|_{t=0} = W_{j\ell}^2 \sum_{i=1}^n u_i^2 e^{v_i - v_\ell}(d_i - d_\ell)^2 > 0. \tag{34}$$

Indeed, positive definiteness of Σ_X gives $u_i^2 > 0$ for every i, so equality would require $d_i = d_\ell$ for all i, contradicting $d \neq 0$ and $\sum_i d_i = 0$. Thus each $f_{j\ell}$ is strictly convex on the constraint hyperplane. The contribution of row j is a positive multiple of the finite maximum of these strictly convex functions; branches with $W_{j\ell} = 0$ can be omitted since they are identically zero and the other branches are positive. This maximum is strictly convex, and adding the remaining convex terms preserves strict convexity. Consequently, there is at most one minimizer of (17).

To establish existence, use the assumption that every column of W is nonzero, equivalently $\sigma_{W_i}^2 > 0$ for all i. The weight contribution alone gives the lower bound

$$F(v) \geq (1+\tau)^2 \left(\min_i u_i^2\right) \left(\min_i \sigma_{W_i}^2\right) e^{\max_i v_i - \min_i v_i}. \tag{35}$$

On $\sum_i v_i = 0$, the range $\max_i v_i - \min_i v_i$ tends to infinity as $\|v\|_2 \to \infty$. Hence F is coercive on this hyperplane and, by continuity, attains its minimum there. Together with strict convexity, this proves that problem (17) has a unique global minimizer. □

F Optimization Details

This appendix specifies the optimizer used in Section 4, including the default initialization $\alpha = 1/2$ and the finite-budget implementation choices.

F.1 Stochastic subgradient updates

The maxima make F potentially nondifferentiable when maximizing indices tie, but subgradients remain available. For a fixed v, select a maximizing index for each maximum in (16) and differentiate the selected branches. To reduce the cost of the activation term, draw a fresh uniform subset $\mathcal{B}_t \subseteq [b]$ of size $B \leq b$ at each iteration and replace its average over b vectors by the average over $\mathcal{B}_t$. All weight rows still participate at every update. The resulting stochastic subgradient G_t satisfies $\mathbb{E}_{\mathcal{B}_t}[G_t \mid v] \in \partial F(v)$ for fixed v and a consistent tie rule. Thus, minibatching leaves the full-pool objective and its convexity unchanged; it changes how we estimate a subgradient.

We project updates onto the zero-sum hyperplane using

$$P(v) \triangleq v - \frac{1}{n}\left(\sum_{i=1}^n v_i\right)\mathbf{1}, \tag{36}$$

$$v^{(t)} = P\left(v^{(t-1)} - \eta\rho_t G_t\right), \qquad t = 1, \ldots, N, \tag{37}$$

where $\eta > 0$ is a problem-dependent peak step size. For the finite iteration budget used here, we keep the peak rate for the first half of the updates and decrease it linearly to one tenth over the second half:

$$\rho_t = \begin{cases} 1, & t \leq N/2, \\ 1 - 0.9\dfrac{t - N/2}{N/2}, & N/2 < t \leq N. \end{cases} \tag{38}$$

We use $N = 1{,}000$ and return the average of the final $L = N/5 = 200$ log-scale iterates,

$$\bar{v} = P\left(\frac{1}{L}\sum_{t=N-L+1}^{N} v^{(t)}\right), \qquad S = \mathrm{diag}(e^{\bar{v}_1/2}, \ldots, e^{\bar{v}_n/2}). \tag{39}$$

This averages log-scales, equivalently taking a coordinatewise geometric mean of the corresponding positive scales. We do not select the output using held-out distortion or the minimum noisy minibatch objective. The schedule and averaging are practical finite-budget choices, rather than a claim of exact convergence after N updates.

Algorithm 2 summarizes the implementation. The initialization parameter α controls the strength of variance-based scaling: $\alpha = 0$ gives $S = I$, while the default $\alpha = 1/2$ gives $s_i \propto (\sigma^2_{W_i}/\sigma^2_{X_i})^{1/4}$. For $\alpha > 0$, every column of W is assumed nonzero; the activation second moments are the diagonal entries of the prescribed positive-definite Σ_X. Here e_i denotes the ith standard basis vector, and ties may be resolved arbitrarily. For $g < n$, the algorithm applies separately to each feature group, with projection and averaging performed within that group.

Algorithm 2 Minibatch Optimization of the Smoothing Matrix

Inputs: $W \in \mathbb{R}^{a\times n}$, calibration pool $X \in \mathbb{R}^{n\times b}$, $U^\top U = \Sigma_X$, $\tau > 0$
batch size $1 \le B \le b$, peak step size $\eta > 0$, and $\alpha \in [0, 1]$ (default $1/2$)
iteration count $N \ge 10$ divisible by 10 (default 1,000); set $L = N/5$
Output: positive diagonal smoothing matrix S, or numerical failure.
for $i = 1 : n$ **do**
  $u_i \leftarrow U_{ii}, \quad \sigma^2_{W_i} \leftarrow \frac{1}{a}\sum_{j=1}^{a} W^2_{ji}, \quad \sigma^2_{X_i} \leftarrow (\Sigma_X)_{ii}$
  $v_i^{(0)} \leftarrow \alpha \ln(\sigma^2_{W_i}/\sigma^2_{X_i})$ ▷ Set to zero directly if $\alpha = 0$
end for
$v^{(0)} \leftarrow P(v^{(0)})$
Draw independent uniform subsets $\mathcal{B}_1, \ldots, \mathcal{B}_N \subseteq [b]$, each of size B
for $r = 0 : 7$ **do** ▷ Reuse initialization and batches on a stability retry
  $v \leftarrow v^{(0)}, \quad z \leftarrow 0 \in \mathbb{R}^n, \quad \text{stable} \leftarrow \text{true}$
  for $t = 1 : N$ **do**
    $G \leftarrow 0 \in \mathbb{R}^n$
    for $j = 1 : a$ **do**
      $\ell_j \leftarrow \mathrm{argmax}_{\ell\in[n]} W^2_{j\ell} e^{-v_\ell}$
      $G \leftarrow G + \frac{(1+\tau)^2}{a}\sum_{i=1}^{n} u_i^2 W^2_{j\ell_j} e^{v_i - v_{\ell_j}}(e_i - e_{\ell_j})$
    end for
    for $k \in \mathcal{B}_t$ **do**
      $r_k \leftarrow \mathrm{argmax}_{\ell\in[n]} X^2_{\ell k} e^{v_\ell}$
      $G \leftarrow G + \frac{1}{B}\sum_{i=1}^{n} \sigma^2_{W_i} X^2_{r_k k} e^{v_{r_k} - v_i}(e_{r_k} - e_i)$
    end for
    $v \leftarrow P(v - 2^{-r}\eta\rho_t G)$ ▷ ρ_t is given by (38)
    if v is nonfinite or $\|v\|_\infty > 100$ **then**
      stable $\leftarrow$ false; **break**
    end if
    if $t > N - L$ **then**
      $z \leftarrow z + v$
    end if
  end for
  if stable **then**
    $\bar{v} \leftarrow P(z/L)$; **return** $S = \mathrm{diag}(e^{\bar{v}_1/2}, \ldots, e^{\bar{v}_n/2})$
  end if
end for
Report numerical failure if all eight trials fail.

F.2 Optimization sample size

We use a fixed pool of $b = 100{,}000$ activation vectors and a fresh minibatch of $B = 5{,}000$ vectors at each iteration. Sampling is uniform without replacement within a minibatch; vectors may reappear in later minibatches. A larger pool exposes the optimizer to more activation patterns, while the per-update activation cost is governed by B rather than b. It does not guarantee that the empirical maxima

generalize to every future input. All calibration predictions reported in this paper are computed over the full b-vector pool at the returned S, rather than averaged from noisy minibatch objective values.

The optimization pool is separate from the sample budget used to estimate Σ_X. We estimate the uncentered second moment from 10^6 FineWeb token positions and use the same damped estimate both to compute u_i and to execute SIC/GPTQ. The 100,000 optimization positions are drawn from this calibration corpus and include the smaller calibration subsets used for the preliminary step-size search; seed 20260914 selects the additional positions. Evaluation uses separate documents and does not participate in selecting S or the step size. We use seed 917 to generate the minibatch sequence and reuse it across optimization problems.

F.3 Peak step size and stability retries

We select a peak η separately for each feature-group optimization problem using calibration data only. For the experiments in this paper, we reuse the rates from preliminary full-batch searches on $b_0 = 2n$ calibration vectors ($b_0 = 5{,}000$ for the illustrative Figure 1). Here n is the full projection input dimension even when $g < n$. The preliminary objective F_0 has the same form as F, restricted to these b_0 activation vectors. Let $G_0^{(0)} \in \partial F_0(v^{(0)})$ and $h_0 = \|P(G_0^{(0)})\|_\infty$. For $h_0 > 0$, we test

$$\mathcal{E} = \{0.8\} \cup \left\{ \frac{c}{h_0} : c \in \{0.1, 0.3, 1, 2, 3, 4, 5\} \right\}.$$

Each candidate uses the same initialization and 1,000 full-batch updates with a constant step size. We discard trials with nonfinite objective values, subgradients, or iterates, or $\|v\|_\infty > 300$, and choose the candidate attaining the lowest best-iterate value of F_0. If $h_0 = 0$, we use $\eta = 0.8$. This search selects a peak rate; the subsequent 100K-pool run follows (38) and returns the tail average in (39).

In the minibatch run, a nonfinite iterate or $\|v\|_\infty > 100$ triggers a restart from the same initialization and minibatch sequence with the peak rate halved, for at most eight trials including the initial attempt. Nonfinite full-pool objective values at recorded checkpoints also trigger a restart. When feature groups are processed together, instability in any group restarts that block of groups and halves all its peak rates. All selected rates and retries are saved; no held-out criterion is used.

G Derivation of the Distortion Approximation

Let $W_j = (W_{j,:})^\top \in \mathbb{R}^n$ denote the jth row of the weight matrix W transposed into a column vector, $\bar{W}_j = (\bar{W}_{j,:})^\top = S^{-1}W_j \in \mathbb{R}^n$ denote its smoothed version, and $\hat{\bar{W}}_j \in \mathrm{INT}_R^n$ denote the quantized version of $\bar{W}_j$. Similarly, $\bar{X}_k = SX_k \in \mathbb{R}^n$ is the kth column of the smoothed activation matrix and $\hat{\bar{X}}_k \in \mathrm{INT}_R^n$ is its quantized version. Furthermore, denote by $\gamma_j = \Gamma(j,j)$ and $\lambda_k = \Lambda(k,k)$. Let

$$e_{\bar{W}_j} = \gamma_j \hat{\bar{W}}_j - \bar{W}_j, \tag{40}$$

$$e_{\bar{X}_k} = \lambda_k \hat{\bar{X}}_k - \bar{X}_k, \tag{41}$$

so that

$$\begin{aligned} e_{jk} &= (\widehat{WX} - WX)_{jk} && (42)\\ &= \gamma_j \hat{\bar{W}}_j^\top \hat{\bar{X}}_k \lambda_k - \bar{W}_j^\top \bar{X}_k && (43)\\ &= (\bar{W}_j + e_{\bar{W}_j})^\top (\bar{X}_k + e_{\bar{X}_k}) - \bar{W}_j^\top \bar{X}_k && (44)\\ &= e_{\bar{W}_j}^\top \bar{X}_k + \bar{W}_j^\top e_{\bar{X}_k} + e_{\bar{W}_j}^\top e_{\bar{X}_k}. && (45) \end{aligned}$$

In order to obtain a tractable expression for D, we will have to assume a simplified model for the vectors $e_{\bar{W}_j}$ and $e_{\bar{X}_k}$. As we shall see, while our simplified model is not accurate, for moderate-to-large values of R it leads to an excellent approximation for $D = \frac{1}{ab}\sum_{jk} e_{jk}^2$. The first assumption we make is that all cross terms in e_{jk}^2 are negligible next to the sum of the three squared terms. This corresponds to assuming that $e_{\bar{W}_j}$ and $e_{\bar{X}_k}$ behave like zero mean noise vectors, that are statistically independent with one another and with the signals. Under this assumption we get

$$e_{jk}^2 \approx e_{\bar{W}_j}^\top (\bar{X}_k \bar{X}_k^\top) e_{\bar{W}_j} + e_{\bar{X}_k}^\top (\bar{W}_j \bar{W}_j^\top) e_{\bar{X}_k} + e_{\bar{X}_k}^\top (e_{\bar{W}_j} e_{\bar{W}_j}^\top) e_{\bar{X}_k}. \tag{46}$$

Summing over j and k we obtain

$$\begin{aligned} D &= \frac{1}{ab}\sum_{j=1}^{a}\sum_{k=1}^{b} e_{jk}^2 \\ &\approx \frac{1}{a}\sum_{j=1}^{a} e_{\bar{W}_j}^\top \left(\frac{1}{b}\sum_{k=1}^{b} \bar{X}_k \bar{X}_k^\top\right) e_{\bar{W}_j} + \frac{1}{b}\sum_{k=1}^{b} e_{\bar{X}_k}^\top \left(\frac{1}{a}\sum_{j=1}^{a} \bar{W}_j \bar{W}_j^\top\right) e_{\bar{X}_k} + \frac{1}{ab}\sum_{j=1}^{a}\sum_{k=1}^{b} \left(e_{\bar{W}_j}^\top e_{\bar{X}_k}\right)^2 \\ &= \frac{1}{a}\sum_{j=1}^{a} e_{\bar{W}_j}^\top \left(S\Sigma_X S\right) e_{\bar{W}_j} + \frac{1}{b}\sum_{k=1}^{b} e_{\bar{X}_k}^\top \left(\frac{1}{a}\sum_{j=1}^{a} \bar{W}_j \bar{W}_j^\top\right) e_{\bar{X}_k} + \frac{1}{ab}\sum_{j=1}^{a}\sum_{k=1}^{b} \left(e_{\bar{W}_j}^\top e_{\bar{X}_k}\right)^2. \end{aligned} \tag{47}$$

It remains to specify how we quantize $\bar{X}_k$ and $\bar{W}_j$. The former must be quantized online and we therefore resort to the simplest method of RTN, whereas the weight matrix can be quantized offline, and we can therefore afford using a feedback-based sequential quantization scheme in the form of GPTQ/SIC, which shapes the spectrum of $e_{\bar{W}_j}$ according to $S\Sigma_X S$.

Round-to-nearest (RTN): RTN quantization of $\bar{X}_k$ consists of absmax scaling followed by rounding to the nearest integer. Specifically, we set $\beta_R = 2^{R-1} - 1$ and apply

$$\lambda_k = \frac{\|\bar{X}_k\|_\infty}{\beta_R}, \quad \hat{\bar{X}}_k = \mathrm{round}\left(\frac{\bar{X}_k}{\lambda_k}\right). \tag{48}$$

Since each entry of $\frac{\bar{X}_k}{\lambda_k}$ has magnitude at most β_R, we have that $\hat{\bar{X}}_k \in \mathrm{INT}_R^n$. The error therefore satisfies

$$e_{\bar{X}_k} = \lambda_k \cdot \hat{\bar{X}}_k - \bar{X}_k \in \lambda_k \cdot \left[-\frac{1}{2}, \frac{1}{2}\right)^n. \tag{49}$$

Here we will make our second simplifying assumption and model $e_{\bar{X}_k} \sim \mathrm{Uniform}\left(\lambda_k \cdot \left[-\frac{1}{2}, \frac{1}{2}\right)^n\right)$, so that

$$\mathbb{E}[e_{\bar{X}_k}] = 0, \quad \mathbb{E}[e_{\bar{X}_k} e_{\bar{X}_k}^\top] = \frac{1}{12\beta_R^2} \|\bar{X}_k\|_\infty^2 \cdot I_n, \tag{50}$$

and in particular

$$\begin{aligned}\mathbb{E}\left[e_{\bar{X}_k}^\top \left(\frac{1}{a}\sum_{j=1}^a \bar{W}_j \bar{W}_j^\top\right) e_{\bar{X}_k}\right] &= \frac{1}{12\beta_R^2}\|\bar{X}_k\|_\infty^2 \cdot \mathrm{trace}\left(\frac{1}{a}\sum_{j=1}^a \bar{W}_j \bar{W}_j^\top\right) \\ &= \frac{1}{12\beta_R^2}\|\bar{X}_k\|_\infty^2 \cdot \frac{1}{a}\|\bar{W}\|_F^2. \end{aligned} \tag{51}$$

Successive interference cancellation (SIC): For the quantization of $\bar{W}_j$ one can exploit the structure of Σ_X for improving upon RTN. Specifically, note that

$$e_{\bar{W}_j}^\top (S\Sigma_X S) e_{\bar{W}_j} = \|USe_{\bar{W}_j}\|^2 = \|US\bar{W}_j - \gamma_j US\hat{\bar{W}}_j\|^2 = \|y_j - \gamma_j US\hat{\bar{W}}_j\|^2, \tag{52}$$

where

$$y_j \triangleq US\bar{W}_j. \tag{53}$$

For a fixed γ_j the reconstructed weight vector is $\gamma_j \cdot z$, where $z = \hat{\bar{W}}_j \in \mathrm{INT}_R^n = \mathbb{Z}^n \cap [-2^{R-1}, 2^{R-1})^n$. Let us first relax the constraint on z to $z \in \mathbb{Z}^n$, and develop a method for choosing a "good" $z \in \mathbb{Z}^n$. By choosing γ_j large enough, we can ensure that the $z \in \mathbb{Z}^n$ our method chooses is also in INT_R. Under this relaxation, for a fixed γ_j, our goal is to solve

$$\operatorname*{argmin}_{z\in\mathbb{Z}^n} \|y_j - \gamma_j USz\|^2 = \operatorname*{argmin}_{z\in\mathbb{Z}^n} \left\|\frac{y_j}{\gamma_j} - USz\right\|^2. \tag{54}$$

This is the closest vector in a lattice problem, which is known to be NP-hard. The canonical way to alleviate this difficulty is to apply a sequential quantization procedure, where decision on entries of z are done greedily entry after entry. This algorithm dates back to Babai Babai (1986), and is known as the GPTQ algorithm Frantar et al. (2023) in the LLM quantization literature. For our purposes, it will be more convenient to use the SIC algorithm Ordentlich & Polyanskiy (2026b) (see also Chen et al. (2026); Birnick (2026); Lifar et al. (2026)) detailed in Algorithm 1. In order to quantize $\bar{W}_j$ to $\hat{\bar{W}}_j$ we therefore apply Algorithm 1 with

$$y = y_j \cdot \gamma_j^{-1} = US\bar{W}_j\gamma_j^{-1} = US[(\Gamma^{-1}\bar{W})_{j,:}]^\top, \quad L = US. \tag{55}$$

and set $\hat{\bar{W}}_j = z_{\mathrm{SIC}}$. In fact, we can use this algorithm to quantize the entire matrix $\bar{W}$ by applying Algorithm 1 with $Y = US(\Gamma^{-1}\bar{W})^\top$ and $L = US$, and quantizing an entire row of Y in each iteration. The SIC algorithm, including the clipping described below, can be implemented using GPTQ, which is already integrated into popular quantization packages such as `llmcompressor`, as described in Appendix A.

It is proved in (Ordentlich & Polyanskiy, 2026b, Lemma 1) that under SIC the quantization error satisfies

$$US\bar{W}_j - \gamma_j USz_{\mathrm{SIC}} \in \mathrm{diag}(\gamma_j US) \cdot \left[-\frac{1}{2}, \frac{1}{2}\right)^n. \tag{56}$$

We make our third simplifying assumption and further model it as a uniform random vector on this domain, which yields

$$\mathbb{E}\|(US)e_{\bar{W}_j}\|^2 = \frac{\gamma_j^2}{12}\sum_{i=1}^n (u_i s_i)^2. \tag{57}$$

The analysis above relied on the assumption that $z_{\mathrm{SIC}} \in \mathrm{INT}_R^n$. Clearly, this will hold true for a sufficiently large choice of γ_j, but as evident from (8) the distortion also grows with γ_j. We would

therefore like to choose as small as possible γ_j while encouraging the integers in z_{SIC} not to exceed β_R. From (56) we immediately get

$$z_{\mathrm{SIC}} \in \bar{W}_j\gamma_j^{-1} - (\gamma_j US)^{-1}\operatorname{diag}(\gamma_j US)\cdot\left[-\frac{1}{2},\frac{1}{2}\right)^n = \bar{W}_j\gamma_j^{-1} - \tilde{U}\cdot\left[-\frac{1}{2},\frac{1}{2}\right)^n, \tag{58}$$

where

$$\tilde{U} = S^{-1}U^{-1}\operatorname{diag}(U)S \tag{59}$$

is a unit-diagonal upper triangular matrix. Fix $\tau > 0$ and consider the choice

$$\gamma_j = (1+\tau)\frac{\|\bar{W}_j\|_\infty}{\beta_R}. \tag{60}$$

With this choice we have that $\beta_R - \|\bar{W}_j\gamma_j^{-1}\|_\infty = \frac{\tau}{1+\tau}\beta_R$. Since this term increases with R while $\tilde{U}\cdot\left[-\frac{1}{2},\frac{1}{2}\right)^n \subset \kappa_{\tilde{U}}\cdot[-1,1)^n$ for $\kappa_{\tilde{U}} > 0$ that does not increase with R, we see that for large R even small choice of $\tau > 0$ will ensure that $\|z_{\mathrm{SIC}}\|_\infty < \beta_R$. While this reasoning fails for moderate R, we still choose γ_j as in (3). We modify the SIC algorithm such that whenever $\mathrm{round}(y_i/\ell_i) \notin \mathrm{INT}_R$ we clip it to the nearest value in INT_R and set $z_{\mathrm{SIC},i}$ to this clipped value. Our fourth simplifying assumption is that (8) holds under the choice of γ_j from (3). Thus, we obtain

$$\mathbb{E}\|(US)e_{\bar{W}_j}\|^2 = \frac{1}{12\beta_R^2}(1+\tau)^2\|\bar{W}_j\|_\infty^2\sum_{i=1}^n (u_i s_i)^2. \tag{61}$$

Approximate Distortion Expression: The fifth and final simplifying assumption we make is that the third summand in (47) is negligible next to the sum of the first two summands. This becomes more accurate as R increases since the term we neglect is $O(\beta_R^{-4}) = O(2^{-4R})$ whereas the dominant term is $\Omega(\beta_R^{-2}) = \Omega(2^{-2R})$.

The overall approximated error is therefore

$$\begin{aligned}
D(S) &= \frac{1}{ab}\sum_{j=1}^a\sum_{k=1}^b e_{jk}^2 \\
&\approx \frac{1}{12\beta_R^2}\sum_{i=1}^n\left[(1+\tau)^2(u_i s_i)^2\frac{1}{a}\sum_{j=1}^a\|\bar{W}_j\|_\infty^2 + \frac{1}{b}\sum_{k=1}^b\|\bar{X}_k\|_\infty^2\frac{1}{a}\sum_{j=1}^a\bar{W}_{ji}^2\right] \\
&= \frac{1}{12\beta_R^2}\sum_{i=1}^n\left[(1+\tau)^2(u_i s_i)^2\frac{1}{a}\sum_{j=1}^a\|\bar{W}_j\|_\infty^2 + \frac{1}{a}\sum_{j=1}^a\frac{W_{ji}^2}{s_i^2}\frac{1}{b}\sum_{k=1}^b\|\bar{X}_k\|_\infty^2\right] \\
&= \frac{1}{12\beta_R^2}\sum_{i=1}^n\left[(1+\tau)^2(u_i s_i)^2\frac{1}{a}\sum_{j=1}^a\|\bar{W}_j\|_\infty^2 + \frac{\sigma_{W_i}^2}{s_i^2}\frac{1}{b}\sum_{k=1}^b\|\bar{X}_k\|_\infty^2\right] \\
&= \frac{1}{12\beta_R^2}\tilde{D}(S)
\end{aligned} \tag{62}$$

where

$$\sigma_{W_i}^2 \triangleq \frac{1}{a}\sum_{j=1}^a W_{ji}^2. \tag{63}$$

and

$$\tilde{D}(S) = \sum_{i=1}^n\left[(1+\tau)^2(u_i s_i)^2\frac{1}{a}\sum_{j=1}^a\|\bar{W}_j\|_\infty^2 + \frac{\sigma_{W_i}^2}{s_i^2}\frac{1}{b}\sum_{k=1}^b\|\bar{X}_k\|_\infty^2\right] \tag{64}$$

is our rescaled distortion proxy.