JARQ: JOINT ALTERNATING REFINEMENT FOR QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Group-wise post-training quantizers for large language models round weights onto a grid that is not refit to the resulting integer codes. We show that this leaves accuracy on the table: the best grid depends on the codes, input correlations couple the errors of different groups, and useful code changes often involve many codes at once. We propose JARQ (**J**oint **A**lternating **R**efinement for **Q**uantization), a plug-in refinement that starts from any group-wise quantizer and alternates a joint least-squares fit of all group scales with bounded Babai proposals that move many codes of a group together on the current grid. The problem is a bilinear box-constrained mixed-integer least-squares problem; the solver is backpropagation-free, does not increase the layer-wise objective under exact scale solves, and keeps the host's bit width, groups, zero points, and inference cost. Across Llama-2, Llama-3, and Qwen models with RTN, GPTQ, OmniQuant, and AWQ hosts, JARQ lowers perplexity in 90 of 96 comparisons, cuts three-bit RTN perplexity by up to 36%, raises mean multiple-choice accuracy in 23 of 24 configurations, and improves QEP, QuaRot, and OJBKQ outputs, at under a minute per 7B block.

1 INTRODUCTION

Post-training quantization (PTQ) makes large language models cheaper to serve by storing weights in a few bits, using only a small calibration set (Frantar et al., 2022; Lin et al., 2024; Shao et al., 2023; Xiao et al., 2023; Ashkboos et al., 2024). At three or four bits, a group-wise quantizer stores each weight as an integer code on a grid set by a per-group scale. Widely used methods obtain the codes by rounding onto a grid: GPTQ fixes the grid and rounds with sequential error compensation, AWQ rounds after activation-aware scaling, and OmniQuant learns clipping thresholds by gradient descent and rounds each weight to the current grid (Frantar et al., 2022; Lin et al., 2024; Shao et al., 2023). The final grid is not refit to the codes it produces, and those codes are not revisited.

Leaving the grid fixed is suboptimal, because the grid and the codes interact in three ways. First, the best grid depends on the codes: once some codes change, a different scale fits the layer output better, and on that grid other codes become optimal, so a grid fixed in advance goes stale (*grid staleness*). Second, the groups of a channel interact: correlated inputs mix their output errors, so the best scale of one group depends on the scales of all others (*cross-group coupling*). Third, useful code changes are often collective: within a group, correlated inputs create improvements that no single-code change can reach (*single-code stationarity*). OJBKQ captures the third effect by decoding many codes at once, but on a fixed, precomputed grid (Wang et al., 2026). Figure 1 illustrates the first and third effects on one group of two correlated weights, where refitting the scale unlocks a two-code move.

We propose JARQ (**J**oint **A**lternating **R**efinement for **Q**uantization), a plug-in refinement for any group-wise quantizer. Starting from the host's scales, codes, and zero points, it minimizes the layer's output reconstruction error by alternating two updates: a joint least-squares fit of all group scales of an output channel, and a bounded Babai (nearest-plane) proposal that can move many codes of a group at once on the current grid. The objective is a bilinear box-constrained mixed-integer least-squares (BBMILS) problem; the bit width, groups, zero points, and inference cost stay unchanged.

Our contributions are:

- We address grid staleness, cross-group coupling, and single-code stationarity together in one BBMILS refinement problem that any group-wise quantizer can enter as a host.

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

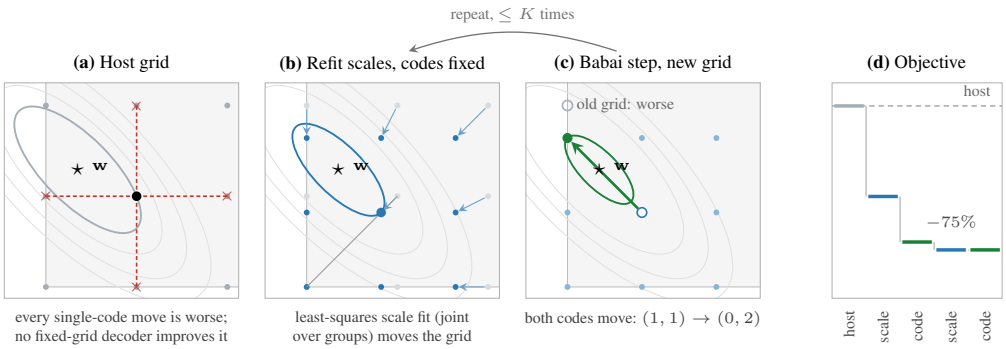


Figure 1: JARQ on one group of two weights with correlated inputs. Ellipses are error level sets around $\mathbf{w}$ ($\star$); dots are grid points. (a) The host codes are optimal on the host grid. (b) Refitting the scale moves the grid. (c) On the new grid, one Babai step moves both codes (Proposition 3.1). (d) The objective stays at or below the host level.

- We give a backpropagation-free alternating solver and show why the grid must move: codes that no search on the host grid can improve may still be improved after a scale refit (Proposition 3.1), and with exact scale solves does not increase the layer's calibration objective (Theorem 1). On full models, alternation beats a scale refit alone in all 16 comparisons, while code updates alone on the host grid raise perplexity in 5 of 16 (Section 4.4).
- Across six models, four hosts, and two bit widths, JARQ lowers perplexity in 90 of 96 comparisons and cuts RTN's W3A16 WikiText-2 perplexity by up to 36%. It raises mean multiple-choice accuracy in 23 of 24 configurations on three models and also improves QEP, QuaRot, and OJBKQ outputs at under a minute per 7B block (Section 4).

2 Related Work

Layer-wise PTQ. GPTQ and OBC reduce layer-wise reconstruction error with second-order sequential quantization (Frantar et al., 2022; Frantar & Alistarh, 2022); AWQ uses activation-aware scaling and OmniQuant learns clipping and equivalent transformations (Lin et al., 2024; Shao et al., 2023). QEP corrects the target for errors from preceding layers (Arai & Ichikawa, 2025), and QuaRot and SpinQuant rotate weights and activations (Ashkboos et al., 2024; Liu et al., 2024). AdaRound and BRECQ learn rounding by layer or block reconstruction (Nagel et al., 2020; Li et al., 2021), and SignRound/AutoRound tunes rounding offsets and clipping with signed gradient descent (Cheng et al., 2024). These methods decide the target, the representation, or the initial codes; JARQ starts from their output and re-optimizes the grid and the codes together.

Integer search on a lattice. GPTQ, run from the last to the first dimension, is Babai's nearest-plane algorithm (Babai, 1986) on a lattice defined by the Hessian (Chen et al., 2026); QuIP relates its LDLQ rounding to GPTQ, and QuIP# quantizes to lattice codebooks (Chee et al., 2023; Tseng et al., 2024). OJBKQ applies Babai–Klein decoding with precomputed scales (Wang et al., 2026); Qronos and CoreQ improve sequential rounding with error correction (Zhang et al., 2026; Cha et al., 2026); ADMM-Q alternates continuous updates with discrete projections and notes that the grid can become stale, allowing one grid refresh (Lucas et al., 2026). The other methods decode on a grid that is fixed during the search. JARQ instead refits all group scales of a channel jointly by least squares at every iteration, so each new grid opens new code improvements (Proposition 3.1).

Joint scale–code optimization. Alternating between codes and their coefficients goes back to multi-bit quantization of recurrent networks (Xu et al., 2018); HQQ fixes the scales and optimizes the zero points without calibration data by half-quadratic splitting (Badri & Shaji, 2023). For LLMs, QuantEase runs coordinate descent over codes on the layer reconstruction objective with the grid held fixed (Behdin et al., 2023). JARQ instead fits all group scales of a channel jointly under the output objective and moves a group's codes as a block on each refitted grid via Babai proposals.

3 METHODOLOGY

We first write refinement of a quantized layer as one reconstruction problem over scales and codes (Sections 3.1 and 3.2). The problem mixes a continuous grid with discrete assignments, so we solve it by alternating two updates, each designed around one of the observations in Section 1: a joint scale fit that accounts for coupled groups, and a Babai proposal that moves several codes at once. Repeating the pair lets the grid follow the codes and the codes follow the grid (Section 3.3).

3.1 PRELIMINARIES AND PROBLEM FORMULATION

Let $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_{d_{\text{out}}}] \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ denote the original weight matrix of a linear layer, and let $\mathbf{x}^\mathsf{T} \in \mathbb{R}^{1 \times d_{\text{in}}}$ denote an input row vector to this layer. Partition each column $\mathbf{w}_j$, $j = 1, \dots, d_{\text{out}}$, into G groups $\mathbf{w}_{i,j} \in \mathbb{R}^{d_i}$, $i = 1, \dots, G$, where $\sum_{i=1}^{G} d_i = d_{\text{in}}$. Groups typically have equal size, in which case $d_i = d_{\text{in}}/G$ for all i. For bit width b, define the integer set $\mathbb{B} = \{0, 1, \dots, 2^b - 1\}$. A group-wise uniform quantizer represents each weight column $\mathbf{w}_j$ by

$$\widehat{\mathbf{w}}_j = \begin{bmatrix} s_{1,j}(\mathbf{q}_{1,j} - z_{1,j}\mathbf{1}) \\ \vdots \\ s_{G,j}(\mathbf{q}_{G,j} - z_{G,j}\mathbf{1}) \end{bmatrix} = \mathbf{D}_{\mathbf{s}_j}(\mathbf{q}_j - \mathbf{z}_j), \qquad \mathbf{q}_{i,j} \in \mathbb{B}^{d_i}, \tag{1}$$

where

$$\mathbf{s}_j = \begin{bmatrix} s_{1,j} \\ \vdots \\ s_{G,j} \end{bmatrix}, \quad \mathbf{D}_{\mathbf{s}_j} = \begin{bmatrix} s_{1,j}\mathbf{I}_{d_1} & & \\ & \ddots & \\ & & s_{G,j}\mathbf{I}_{d_G} \end{bmatrix}, \quad \mathbf{q}_j = \begin{bmatrix} \mathbf{q}_{1,j} \\ \vdots \\ \mathbf{q}_{G,j} \end{bmatrix}, \quad \mathbf{z}_j = \begin{bmatrix} z_{1,j}\mathbf{1}_{d_1} \\ \vdots \\ z_{G,j}\mathbf{1}_{d_G} \end{bmatrix}.$$

The scale sets the grid spacing, the zero point sets its offset, and the integer code selects a grid value. Collect these variables as $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_{d_{\text{out}}}] \in \mathbb{R}^{G \times d_{\text{out}}}$, $\mathbf{Z} = (z_{i,j}) \in \mathbb{Z}^{G \times d_{\text{out}}}$, and $\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_{d_{\text{out}}}] \in \mathbb{B}^{d_{\text{in}} \times d_{\text{out}}}$. For later use, we write $\widehat{\mathbf{W}} = [\widehat{\mathbf{w}}_1, \dots, \widehat{\mathbf{w}}_{d_{\text{out}}}] \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$.

Let $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\mathsf{T} \in \mathbb{R}^{n \times d_{\text{in}}}$ stack the calibration input vectors to this layer in the original model as rows. Let $\widetilde{\mathbf{X}}$ stack the corresponding inputs obtained using the quantized preceding layers in the same way. They coincide for the first layer and drift apart in later layers.

Given an existing quantized layer, we refine its scales $\mathbf{S}$ and integer codes $\mathbf{Q}$ while fixing the bit width, groups, and zero points $\mathbf{Z}$. The original weight matrix $\mathbf{W}$ remains the reference. The goal is to choose scales and codes together so that the quantized layer preserves the reference response.

3.2 OBJECTIVE

We preserve the layer's response $\mathbf{Y}^\star = \widetilde{\mathbf{X}}\mathbf{W}$ to the inputs it receives in the quantized model by solving

$$\begin{aligned} &\min_{\mathbf{S},\mathbf{Q}} \mathcal{F}(\mathbf{S}, \mathbf{Q}) = \|\widetilde{\mathbf{X}}\widehat{\mathbf{W}} - \widetilde{\mathbf{X}}\mathbf{W}\|_F^2 + \nu^2 \|\widehat{\mathbf{W}} - \mathbf{W}\|_F^2, \\ &\text{s.t.} \quad \mathbf{S} \in \mathbb{R}^{G \times d_{\text{out}}}, \quad \mathbf{Q} \in \mathbb{B}^{d_{\text{in}} \times d_{\text{out}}}, \end{aligned} \tag{2}$$

where $\widehat{\mathbf{W}}$ is given by (1). The output term compares $\widetilde{\mathbf{X}}\widehat{\mathbf{W}}$ with $\widetilde{\mathbf{X}}\mathbf{W}$, so the target needs no full-precision activations. The output error accounts for correlations between input coordinates. The weight penalty ($\nu > 0$) restrains changes in directions weakly constrained by the calibration set.

Define the augmented target and design matrices

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}^\star \\ \nu\mathbf{W} \end{bmatrix} \in \mathbb{R}^{m \times d_{\text{out}}}, \qquad \mathbf{A} = \begin{bmatrix} \widetilde{\mathbf{X}} \\ \nu\mathbf{I} \end{bmatrix} \in \mathbb{R}^{m \times d_{\text{in}}}, \qquad m = n + d_{\text{in}}. \tag{3}$$

Thus $\mathbf{Y}$ contains the reference outputs followed by the weighted reference weights; it is constructed from the calibration inputs and the original layer weights. Writing $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_{d_{\text{out}}}]$, the objective becomes

$$\mathcal{F}(\mathbf{S}, \mathbf{Q}) = \|\mathbf{Y} - \mathbf{A}\widehat{\mathbf{W}}\|_F^2 = \sum_{j=1}^{d_{\text{out}}} \|\mathbf{y}_j - \mathbf{A}\widehat{\mathbf{w}}_j\|_2^2 = \sum_{j=1}^{d_{\text{out}}} \|\mathbf{y}_j - \mathbf{A}\mathbf{D}_{\mathbf{s}_j}(\mathbf{q}_j - \mathbf{z}_j)\|_2^2. \tag{4}$$

Both $\mathbf{A}$ and $\mathbf{Y}$ stay fixed during a layer's refinement. The target contains no correction for errors of preceding layers; mixing in the full-precision response $\mathbf{XW}$ yields QEP's propagation-corrected target (Arai & Ichikawa, 2025), for which all results below hold unchanged (Appendix E.5).

Because $\widehat{\mathbf{W}}$ multiplies real scales by bounded integer codes, (4) is a bilinear box-constrained mixed-integer least-squares (BBMILS) problem; fixing the scales leaves a closest-vector problem.

3.3 Alternating Update

We now approximately minimize (2). Scales determine the grid, and codes determine the assignments to it. A host settles the grid and rounds the codes onto it; but once the codes move, a different grid fits the output best, and on that grid different codes become optimal. JARQ therefore alternates the two: at iteration k, it refits $\mathbf{S}$ for the codes $\mathbf{Q}^{(k-1)}$, then updates $\mathbf{Q}$ on the new grid $\mathbf{S}^{(k)}$. The objective in (4) separates over output columns, so each column can be refined independently.

We call the PTQ method supplying the initial quantized layer the *host*. It may use rounding, error compensation, learned clipping, or another procedure. We require only its group-wise uniform representation $(\mathbf{S}^{(0)}, \mathbf{Q}^{(0)}, \mathbf{Z})$, and initialize directly from these scales, codes, and zero points. The two updates are described below, followed by their core loop in Algorithm 1. Alternation matters even when the host's codes are already optimal for the host's grid, as the following construction shows:

Proposition 3.1 (Refitting the grid unlocks descent)**.** *There exist a group problem of the form (8) and a host state $(s_0, \mathbf{q}_0)$ such that $\mathbf{q}_0$ minimizes the error over $\mathbb{B}^{d_i}$ at scale s_0, yet after a least-squares scale refit, the bounded Babai proposal changes two codes and strictly lowers the error.*

Moving the grid causes the descent; Figure 1 draws an instance, verified in Appendix G.3.

3.3.1 Updating Scales Given Codes

Groups are stored separately, but their outputs are not independent: correlated inputs make one group's error depend on the others' scales. The scale update fits all groups of a channel jointly.

Partition the design matrix consistently with the weight groups: $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_G]$, where $\mathbf{A}_i \in \mathbb{R}^{m \times d_i}$. At iteration k, in the objective $\mathcal{F}(\mathbf{S}, \mathbf{Q})$ we take $\mathbf{Q} = \mathbf{Q}^{(k-1)}$, which is obtained at iteration $k-1$. Then, from (1), we can write

$$\mathbf{A}\widehat{\mathbf{w}}_j = \mathbf{B}_j^{(k-1)}\mathbf{s}_j,$$

where

$$\mathbf{B}_j^{(k-1)} = \left[\mathbf{A}_1(\mathbf{q}_{1,j}^{(k-1)} - z_{1,j}\mathbf{1}_{d_1}), \ldots, \mathbf{A}_G(\mathbf{q}_{G,j}^{(k-1)} - z_{G,j}\mathbf{1}_{d_G})\right] \in \mathbb{R}^{m \times G}. \tag{5}$$

From (4), we see that the optimal $\mathbf{s}_j$, to be denoted by $\mathbf{s}_j^{(k)}$, should be the solution of the real least-squares problem:

$$\min_{\mathbf{s} \in \mathbb{R}^G} \|\mathbf{y}_j - \mathbf{B}_j^{(k-1)}\mathbf{s}\|_2^2,$$

Equivalently, $\mathbf{s}_j^{(k)}$ satisfies the normal equations:

$$(\mathbf{B}_j^{(k-1)})^\mathsf{T}\mathbf{B}_j^{(k-1)}\mathbf{s}_j^{(k)} = (\mathbf{B}_j^{(k-1)})^\mathsf{T}\mathbf{y}_j. \tag{6}$$

Thus all G scales of $\widehat{\mathbf{w}}_j$ are fitted jointly. We solve the normal equations by Cholesky factorization. If the Gram matrix $(\mathbf{B}_j^{(k-1)})^\mathsf{T}\mathbf{B}_j^{(k-1)}$ is nearly singular, we add $\lambda_j\mathbf{I}$ to it with $\lambda_j = 10^{-4}$; otherwise $\lambda_j = 0$ (Appendix E.2). Scales are unconstrained in sign and are evaluated as signed values; Appendix E maps negative scales to an equivalent positive form for formats that require one.

Fitting each group's scale separately ignores the off-diagonal terms of the Gram matrix and leaves an error gap that vanishes when the groups' column spaces are orthogonal and is bounded by a quantity quadratic in their coherence (Propositions G.1 and G.2); Section 4.4 measures this gap. The joint solve is cheap because G is small: a 4,096-wide input with groups of 128 weights has $G = 32$, so the $G \times G$ system costs far less than forming $\mathbf{B}_j$ (Appendix G.8).

3.3.2 Updating Integer Codes Given Scales

On a fixed grid, the best change to a group's codes often moves several codes together, because correlated inputs let one code's error offset another's. Single-code search stops at such points (Proposition 3.2); the code update below proposes all codes of a group at once from one triangular solve.

Hold $\mathbf{s}_j^{(k)}$ fixed and visit groups $i = 1, \ldots, G$. The earlier groups in the sweep already have their new codes; the later groups still have their previous codes. The target for group i is therefore

$$\mathbf{y}_j^{(i,k)} = \mathbf{y}_j - \sum_{g<i} s_{g,j}^{(k)} \mathbf{A}_g (\mathbf{q}_{g,j}^{(k)} - z_{g,j} \mathbf{1}_{d_g}) - \sum_{g>i} s_{g,j}^{(k)} \mathbf{A}_g (\mathbf{q}_{g,j}^{(k-1)} - z_{g,j} \mathbf{1}_{d_g}). \tag{7}$$

This subtracts the latest contributions of all other groups. Ideally, the new group codes would solve

$$\min_{\mathbf{q} \in \mathbb{B}^{d_i}} \|\mathbf{y}_j^{(i,k)} - s_{i,j}^{(k)} \mathbf{A}_i (\mathbf{q} - z_{i,j} \mathbf{1}_{d_i})\|_2^2. \tag{8}$$

Problem (8) is a closest-vector problem over a bounded set of lattice points. Such problems are NP-hard in general (van Emde Boas, 1981; Micciancio & Goldwasser, 2002), and searching the entire integer set is impractical for typical group sizes. We instead compute one bounded Babai nearest-plane proposal (Babai, 1986) using a triangular system from the group's QR factor.

For each group, precompute a thin QR factorization, without an additional column permutation:

$$\mathbf{A}_i = \mathbf{U}_i \mathbf{R}_i, \qquad \mathbf{U}_i^\mathsf{T} \mathbf{U}_i = \mathbf{I}, \qquad \mathbf{R}_i \in \mathbb{R}^{d_i \times d_i} \text{ upper triangular.} \tag{9}$$

Since $\nu > 0$, $\mathbf{A}$ has full column rank, so each $\mathbf{R}_i$ is nonsingular. These factors are reused across output columns and iterations. For a nonzero scale, let $\mathbf{v} = \mathbf{U}_i^\mathsf{T} \mathbf{y}_j^{(i,k)} / s_{i,j}^{(k)}$. The proposal $\mathbf{q}^{\mathrm{cand}}$ is computed backward:

$$q_\ell^{\mathrm{cand}} = \mathrm{clip}_{\mathbb{B}} \left(\mathrm{round} \left[z_{i,j} + \frac{v_\ell - \sum_{h=\ell+1}^{d_i} (\mathbf{R}_i)_{\ell h} (q_h^{\mathrm{cand}} - z_{i,j})}{(\mathbf{R}_i)_{\ell\ell}} \right] \right), \quad \ell = d_i, \ldots, 1. \tag{10}$$

Each step subtracts the contribution of the already chosen integer codes before rounding and clipping. Thus, codes with larger index, chosen first, shape the choices for smaller indices. We accept the proposal only if it gives a finite, strictly smaller residual in (8); otherwise we keep $\mathbf{q}_{i,j}^{(k-1)}$. The resulting codes define $\mathbf{B}_j^{(k)}$ for the next scale fit.

Proposition 3.2 (Descent beyond single-code updates)**.** *There exists a box-constrained group problem of the form (8), with its scale optimal for the current codes, such that every single-code change has strictly larger error even after optimally refitting the scale, while one bounded Babai proposal strictly reduces the error at the current scale.*

Correlated input coordinates can therefore make coordinated code changes beneficial beyond the reach of single-code moves with conditional scale refitting (Appendix G.2). Without clipping and under a uniform model for rounding errors, Babai's expected residual is also no larger than that of rounding the unconstrained solution, and lower by $s_{i,j}^2/12$ times the off-diagonal energy of $\mathbf{R}_i$ (Proposition G.3). The update applies the nearest-plane recursion that underlies GPTQ (Chen et al., 2026) to one group on the current grid; Remark 2 discusses the relation and the differences.

Descent guarantee. Write $\mathcal{F}_j(\mathbf{s}, \mathbf{q})$ for the jth term of (4) and $\phi_j(\mathbf{q}) = \min_{\mathbf{s}} \mathcal{F}_j(\mathbf{s}, \mathbf{q})$. We call $(\mathbf{s}, \mathbf{q})$ *scale-optimal and Babai-stable* if $\mathbf{s}$ minimizes $\mathcal{F}_j(\cdot, \mathbf{q})$ and, for every group with $|s_i| > \epsilon_{\mathrm{fp}}$, the bounded Babai proposal computed at $(\mathbf{s}, \mathbf{q})$ does not strictly lower $\mathcal{F}_j$ any further.

Theorem 1 (Monotone descent and finite termination)**.** *Fix an output column j and assume that every scale update in Algorithm 1 is an exact least-squares solve ($\lambda_j = 0$). In exact arithmetic, (i) $\mathcal{F}_j$ does not increase at any scale update or code update; (ii) ϕ_j strictly decreases over every iteration that changes a code; (iii) with an unlimited budget and no early stopping, codes change in finitely many iterations, and after the first iteration without an accepted proposal the state is a fixed point that is scale-optimal and Babai-stable.*

Corollary 1 (No worse than the host)**.** *In exact arithmetic, if every scale update in the layer is an exact least-squares solve ($\lambda_j = 0$), the returned state has an objective no larger than that of the host state, $\mathcal{F}(\mathbf{S}, \mathbf{Q}) \le \mathcal{F}(\mathbf{S}^{(0)}, \mathbf{Q}^{(0)})$, for any budget K and any stopping rule.*

Layerwise procedure. Algorithm 1 summarizes the two updates with a maximum budget of K iterations; all main experiments use $K = 3$. For efficient computation, factor $\mathbf{A} = \mathbf{U}\mathbf{R}$ and set $\overline{\mathbf{Y}} = \mathbf{U}^{\mathsf{T}}\mathbf{Y}$. The reduced objective is

$$\mathcal{J}(\mathbf{S}, \mathbf{Q}) = \|\overline{\mathbf{Y}} - \mathbf{R}\widehat{\mathbf{W}}\|_F^2. \tag{11}$$

It differs from $\mathcal{F}$ by a constant, which is zero for the main target $\mathbf{Y} = \mathbf{A}\mathbf{W}$ (Appendix E). We compute the updates with $(\mathbf{R}, \overline{\mathbf{Y}})$ in place of $(\mathbf{A}, \mathbf{Y})$, caching all QR factors and retaining the constant for stopping. The terminal scales and codes define the installed weights, whose outputs supply the next layer's calibration inputs. Each channel stops once the objective after consecutive scale updates decreases by at most $\tau = 10^{-5}$, relative or absolute (Appendix E.2). The computational cost is analyzed in Appendix G.8.

Algorithm 1 JARQ layer refinement

Input: $(\mathbf{A}, \mathbf{Y})$, host $\Theta^{(0)} = (\mathbf{S}^{(0)}, \mathbf{Q}^{(0)}, \mathbf{Z})$, budget K.
Output: refined state Θ.

$\mathbf{A} = \mathbf{U}\mathbf{R}$, $\overline{\mathbf{Y}} \leftarrow \mathbf{U}^{\mathsf{T}}\mathbf{Y}$; $(\mathbf{A}, \mathbf{Y}) \leftarrow (\mathbf{R}, \overline{\mathbf{Y}})$
$\mathbf{A}_i = \mathbf{U}_i\mathbf{R}_i$ for $i = 1, \ldots, G$; $\Theta \leftarrow \Theta^{(0)}$ ▷ cached QR
for $j = 1, \ldots, d_{\text{out}}$ **do**
  for $k = 1, \ldots, K$ **do** ▷ iterations
    $\mathbf{B}_j \leftarrow [\,\mathbf{A}_i(\mathbf{q}_{i,j} - z_{i,j}\mathbf{1})\,]_{i=1}^{G}$ ▷ scales
    $\mathbf{s}_j \leftarrow \arg\min_{\mathbf{s}} \|\mathbf{y}_j - \mathbf{B}_j\mathbf{s}\|_2^2 + \lambda_j\|\mathbf{s}\|_2^2$
    $L_j^{(k)} \leftarrow \|\mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j\|_2^2 + c_j$ ▷ objective
    if $k \geq 2$ and (18) holds **then break** ▷ early stop
    for $i = 1, \ldots, G$ with $|s_{i,j}| > \epsilon_{\text{fp}}$ **do** ▷ codes
      $\mathbf{r} \leftarrow \mathbf{y}_j - \sum_{g \neq i} s_{g,j}\mathbf{A}_g(\mathbf{q}_{g,j} - z_{g,j}\mathbf{1})$
      $\mathbf{q}^{+} \leftarrow \mathrm{Babai}_{\mathbb{B}}(\mathbf{R}_i, \mathbf{U}_i^{\mathsf{T}}\mathbf{r}/s_{i,j}, z_{i,j})$
      if $e(\mathbf{q}^{+}) < e(\mathbf{q}_{i,j})$ **then** $\mathbf{q}_{i,j} \leftarrow \mathbf{q}^{+}$
return Θ; install $\widehat{\mathbf{w}}_j = \mathbf{D}_{\mathbf{s}_j}(\mathbf{q}_j - \mathbf{z}_j)$

$e(\mathbf{q}) = \|\mathbf{r} - s_{i,j}\mathbf{A}_i(\mathbf{q} - z_{i,j}\mathbf{1})\|_2$, with nonfinite values rejected. $\lambda_j = 10^{-4}$ only if the Gram matrix $\mathbf{B}_j^{\mathsf{T}}\mathbf{B}_j$ is nearly singular, else 0. c_j is the constant dropped by the QR reduction (zero for $\mathbf{Y} = \mathbf{A}\mathbf{W}$). $\mathrm{Babai}_{\mathbb{B}}$ is the back substitution (10), rounding and clipping each code to $\mathbb{B}$.

The guarantee is per layer: it compares both states on the calibration objective with the same inputs $\widetilde{\mathbf{X}}$. Appendix G.5 gives the proofs and the ridge case. At a Babai-stable point, a group whose proposal needs no clipping is within $s_{i,j}^2 \sum_{\ell} (\mathbf{R}_i)_{\ell\ell}^2/4$ of its best real-valued fit (Proposition G.4).

4 Experiments

We test perplexity across six models, four hosts, and two bit widths, then evaluate downstream accuracy and examine the sources of improvement, including layer-wise errors and cost.

4.1 Setup

We quantize Llama-2-7B/13B (Touvron et al., 2023), Llama-3-8B (Grattafiori et al., 2024), Qwen2.5-7B (Yang et al., 2024), and Qwen3-4B/8B (Yang et al., 2025) to W3A16 and W4A16 with groups of 128 weights, using 128 WikiText-2 training sequences of 2,048 tokens. The host is RTN, GPTQ (Frantar et al., 2022), OmniQuant (Shao et al., 2023) (weight clipping only), or AWQ (Lin et al., 2024); Base is its quantized model and +Ours is JARQ applied to it with the same bit width, groups, and zero points. All refined models use $K = 3$ iterations with early stopping and $\nu = 0.6$. We report WikiText-2 (WT2) (Merity et al., 2017) and C4 (Raffel et al., 2020) perplexity, five multiple-choice tasks, and generative accuracy on GSM8K and MATH-500 (Appendix A).

4.2 Main Results

Perplexity. Table 1 shows lower perplexity in 46 of 48 WT2 comparisons and 44 of 48 C4 comparisons. The gains are largest for RTN, the only host without calibration data. At W3A16, refining RTN cuts WT2 perplexity by 36.21% on Qwen2.5-7B (11.95→7.63) and by 31.78% on Llama-3-8B (12.05→8.22). Part of the RTN gain comes from the calibration data that refinement adds, but calibrated hosts still leave room: on Qwen3-8B at W3A16, OmniQuant improves by 8.02% (11.46→10.55) and GPTQ by 1.93% (10.75→10.54). The same holds at 32B: across OmniQuant and AWQ on Qwen2.5-32B and OmniQuant on Qwen3-32B, JARQ lowers perplexity in 11 of 12 comparisons, with the largest gains at three bits (up to 3.0% on WT2; Figure 4c and Appendix B).

Downstream accuracy. Refinement raises mean multiple-choice accuracy in 23 of 24 configurations (Table 2), including all eight on Llama-3-8B, by up to +5.57 points (Qwen3-8B, RTN, W3A16). Across individual tasks, it is better in 82 of 120 comparisons; the one lower average,

Table 1: Perplexity before (Base) and after (+Ours) refinement with $K = 3$ (lower is better). Small numbers give the relative change of +Ours over Base in percent (green: lower, red: higher).

	Llama-2-7B		Llama-2-13B		Llama-3-8B		Qwen2.5-7B		Qwen3-4B		Qwen3-8B	
Method	WT2	C4	WT2	C4	WT2	C4	WT2	C4	WT2	C4	WT2	C4
W3A16												
FP16	5.472	6.973	4.884	6.468	6.136	8.881	6.848	10.442	13.638	16.626	9.715	13.290
RTN	6.663	8.405	5.521	7.179	12.048	16.444	11.954	16.054	22.447	27.013	13.456	17.582
+Ours	6.204 −6.88	8.217 −2.24	5.270 −4.54	7.158 −0.29	8.219 −31.8	12.667 −23.0	7.625 −36.2	11.916 −25.8	16.194 −27.9	21.625 −19.9	10.712 −20.4	15.103 −14.1
GPTQ	6.083	7.990	5.248	7.034	7.618	13.242	7.665	11.677	15.153	18.623	10.749	14.720
+Ours	6.014 −1.13	7.979 −0.14	5.248 −0.01	7.037 +0.04	7.649 +0.41	12.011 −9.30	7.524 −1.85	11.670 −0.05	15.089 −0.42	18.791 +0.90	10.542 −1.93	14.666 −0.36
OmniQuant	6.095	7.812	5.322	7.025	8.768	12.476	7.779	11.785	17.221	20.902	11.465	15.454
+Ours	6.045 −0.82	7.795 −0.22	5.253 −1.30	7.010 −0.22	7.662 −12.6	11.989 −3.91	7.516 −3.38	11.690 −0.81	16.480 −4.30	20.346 −2.66	10.546 −8.02	14.963 −3.18
AWQ	6.152	7.771	5.299	6.946	8.039	11.539	7.966	11.759	16.796	20.418	11.103	14.978
+Ours	5.924 −3.70	7.741 −0.38	5.220 −1.49	6.865 −1.16	7.491 −6.82	11.591 +0.45	7.518 −5.62	11.637 −1.04	15.513 −7.64	20.166 −1.23	10.533 −5.13	14.776 −1.35
W4A16												
FP16	5.472	6.973	4.884	6.468	6.136	8.881	6.848	10.442	13.638	16.626	9.715	13.290
RTN	5.726	7.246	4.984	6.588	6.727	9.636	7.227	10.886	15.968	18.291	10.137	13.643
+Ours	5.626 −1.76	7.186 −0.83	4.960 −0.48	6.583 −0.07	6.481 −3.66	9.532 −1.08	7.004 −3.09	10.697 −1.73	14.555 −8.85	17.930 −1.98	9.998 −1.37	13.627 −0.12
GPTQ	5.585	7.133	4.951	6.574	6.423	9.386	7.029	10.842	13.675	16.980	9.941	13.560
+Ours	5.564 −0.38	7.129 −0.06	4.947 −0.09	6.565 −0.13	6.416 −0.11	9.408 +0.24	7.001 −0.40	10.672 −1.57	13.553 −0.90	16.917 −0.37	9.860 −0.81	13.550 −0.08
OmniQuant	5.592	7.123	4.963	6.572	6.628	9.574	7.082	10.766	14.665	17.909	10.141	13.829
+Ours	5.584 −0.13	7.103 −0.28	4.953 −0.20	6.562 −0.15	6.496 −1.99	9.539 −0.37	6.996 −1.21	10.694 −0.67	13.728 −6.39	17.386 −2.92	9.858 −2.79	13.582 −1.79
AWQ	5.607	7.119	4.972	6.559	6.549	9.410	7.103	10.726	14.620	17.793	9.855	13.576
+Ours	5.553 −0.97	7.111 −0.12	4.955 −0.34	6.550 −0.13	6.448 −1.55	9.368 −0.45	7.000 −1.46	10.692 −0.32	13.762 −5.87	17.222 −3.21	9.879 +0.24	13.532 −0.32

Table 2: Multiple-choice accuracy (%; higher is better). Entries are Base/+Ours; Avg. is the five-task mean, followed by the change of +Ours in points (green: higher, red: lower). Bold marks the higher value; ties are not bolded.

Model	Host	W/A	ARC-E	ARC-C	MMLU	Wino.	BoolQ	**Avg.**
Llama-3-8B	FP16	W16A16	77.53	54.10	65.45	74.03	82.26	70.67
	RTN	W3A16	61.07/**67.85**	40.53/**44.45**	47.17/**53.11**	67.48/**71.03**	69.11/**75.60**	57.07/**62.41** +5.34
		W4A16	77.31/**77.78**	**51.62**/51.11	62.63/**63.57**	72.38/72.38	79.69/**81.07**	68.73/**69.18** +0.45
	GPTQ	W3A16	63.59/**69.32**	39.16/**43.00**	52.81/**57.47**	**71.35**/71.19	69.88/**70.34**	59.36/**62.26** +2.90
		W4A16	77.48/**78.75**	**53.67**/53.33	**63.71**/63.45	74.11/**74.43**	81.44/**82.51**	70.08/**70.49** +0.41
	OmniQuant	W3A16	63.80/**68.52**	40.96/**43.00**	**53.45**/52.55	68.75/**70.72**	**75.08**/71.41	60.41/**61.24** +0.83
		W4A16	**76.94**/76.64	50.77/**52.82**	63.64/**63.76**	71.98/**73.09**	80.80/**80.83**	68.83/**69.43** +0.60
	AWQ	W3A16	74.71/**76.09**	**48.63**/47.53	55.33/**57.34**	71.03/**72.06**	**80.03**/78.44	65.95/**66.29** +0.34
		W4A16	75.00/**78.07**	51.71/**52.47**	**63.40**/62.88	**72.38**/72.30	80.92/**81.38**	68.68/**69.42** +0.74
Qwen2.5-7B	FP16	W16A16	77.44	51.11	74.30	72.93	84.65	72.09
	RTN	W3A16	68.73/**70.79**	47.01/**48.21**	67.14/**67.97**	64.48/**69.38**	77.61/**82.84**	64.99/**67.84** +2.85
		W4A16	76.68/**78.75**	50.00/**51.79**	72.68/**73.14**	**71.82**/71.59	**84.01**/83.15	71.04/**71.68** +0.64
	GPTQ	W3A16	72.47/**77.10**	48.55/**50.77**	68.19/**68.74**	68.59/**69.53**	83.52/**84.25**	68.26/**70.08** +1.82
		W4A16	75.72/**77.78**	49.74/**51.96**	**73.31**/73.16	**72.22**/71.82	**83.21**/82.78	70.84/**71.50** +0.66
	OmniQuant	W3A16	77.57/**78.66**	50.34/**52.56**	**69.34**/68.93	68.03/**71.11**	82.81/**83.12**	69.62/**70.88** +1.26
		W4A16	76.60/**77.82**	49.15/**51.71**	**73.48**/72.96	**72.22**/71.90	84.98/84.98	71.29/**71.87** +0.58
	AWQ	W3A16	66.55/**70.24**	**49.57**/46.16	**69.62**/68.23	67.48/**71.51**	**81.93**/80.92	67.03/**67.41** +0.38
		W4A16	77.19/**77.48**	**51.11**/50.34	**73.32**/73.07	**72.14**/71.90	80.83/**82.54**	70.92/**71.07** +0.15
Qwen3-8B	FP16	W16A16	80.93	56.57	74.96	67.64	86.61	73.34
	RTN	W3A16	71.93/**76.14**	48.55/**52.13**	61.10/**68.49**	61.56/**68.90**	79.88/**85.20**	64.60/**70.17** +5.57
		W4A16	75.88/**78.49**	53.07/**55.38**	73.66/**74.23**	67.64/**67.96**	85.29/**86.64**	71.11/**72.54** +1.43
	GPTQ	W3A16	71.80/**78.24**	47.61/**51.45**	67.11/**68.59**	**67.88**/66.38	84.77/**85.29**	67.83/**69.99** +2.16
		W4A16	79.42/**79.84**	**55.12**/54.61	73.99/73.99	**68.51**/68.27	86.48/**87.16**	72.70/**72.77** +0.07
	OmniQuant	W3A16	**77.48**/72.56	**50.60**/48.04	68.03/**69.23**	64.88/**67.56**	**85.35**/84.74	**69.27**/68.43 −0.84
		W4A16	78.91/**79.42**	53.84/**55.12**	**73.81**/73.71	68.27/**68.67**	86.45/**86.88**	72.26/**72.76** +0.50
	AWQ	W3A16	76.09/**76.60**	50.43/**50.94**	**69.10**/69.00	64.01/**66.54**	84.71/**85.11**	68.87/**69.64** +0.77
		W4A16	**80.39**/79.67	**55.55**/55.12	73.66/**73.86**	68.59/**68.98**	86.18/**86.79**	72.87/**72.88** +0.01
Improved / evaluated configurations (Avg.)								23/24

Qwen3-8B with OmniQuant at W3A16 (−0.84), comes mainly from ARC-E. On generation, refinement raises the W4A16 host average on GSM8K and MATH-500 for all three models (Figure 2); at W3A16 it raises GSM8K accuracy in six of eight configurations, by up to 8.56 points (Table 10). As with perplexity, the gains grow as bits shrink: the change in mean multiple-choice accuracy averages +1.95 points at W3A16 and +0.52 at W4A16, and the three largest gains all come from RTN. With the RTN and GPTQ hosts, refinement raises mean accuracy in all 12 configurations.

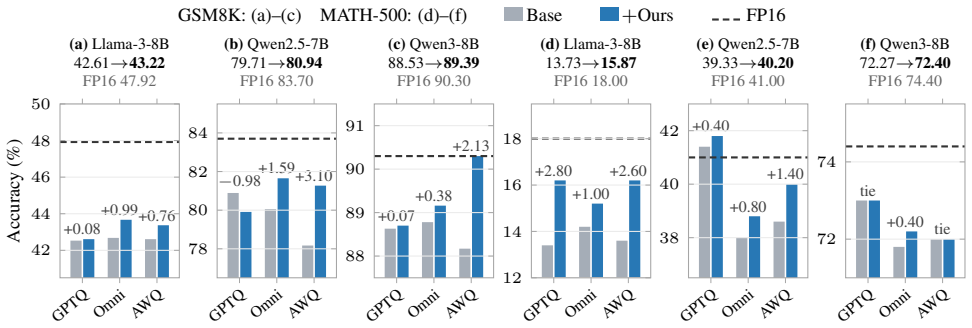


Figure 2: Generative accuracy at W4A16 on GSM8K (a–c) and MATH-500 (d–f) for three hosts: Base (gray), +Ours (blue), and FP16 (dashed). Labels give the change in points; titles give the mean over hosts. Axes are zoomed per panel. W3A16 GSM8K scores are in Table 10.

Table 3: Compatibility results for (a) QEP (Arai & Ichikawa, 2025), (b) QuaRot (Ashkboos et al., 2024) on Llama-2-7B, and (c) OJBKQ (Wang et al., 2026) (WT2 perplexity, with C4 also in (b); lower is better). Settings are in Appendix A.4.

(a) QEP

Model	W/A	Base	+Ours
Qwen2.5-7B	W3A16	7.67	**7.58**
	W4A16	7.02	**7.00**
Qwen3-8B	W3A16	11.00	**10.52**
	W4A16	9.94	**9.88**

(b) QuaRot

W/A	Data	Base	+Ours
W3A16	WT2	6.38	**6.34**
	C4	8.59	**8.46**
W4A16	WT2	5.63	**5.61**
	C4	7.46	**7.45**

(c) OJBKQ

Model	W/A	Base	+Ours
Qwen3-8B	W3A16	10.98	**10.74**
	W4A16	9.94	**9.85**
Llama-3-8B	W3A16	7.96	**7.65**
	W4A16	6.79	**6.45**

Table 4: Ablations. (a) Medians over 16 layers at W3A16 (Appendix C). (b) Full-model WT2/C4 perplexity on Llama-3-8B (L3) and Qwen3-4B (Q3); bold: lowest (Appendix C.5).

(a) Layer level

Measurement	RTN	GPTQ
Indep. scale fits: gap (%)	20.98	25.29
3-pass coord. descent: gap (%)	1.31	0.01
3-pass coord. descent: time ratio	1.29	1.32
Babai gain after 1-code search (%)	40.14	13.53
Median codes changed	30	6
Joint over Fixed, held-out (%)	12.32	5.47
Layers improved	16/16	16/16

(b) Full model (WT2/C4 perplexity)

	Host		Host	Codes only	Scales only	Full JARQ
L3	RTN	W3	12.05/16.44	8.69/14.17	8.58/13.31	**8.22/12.67**
		W4	6.73/9.64	6.59/9.61	6.49/9.58	**6.48/9.53**
	GPTQ	W3	**7.62**/13.24	7.71/12.32	7.78/12.63	7.65/**12.01**
		W4	6.42/**9.39**	6.42/9.42	6.42/9.41	**6.42**/9.41
Q3	RTN	W3	22.45/27.01	21.20/27.08	17.97/22.32	**16.19/21.62**
		W4	15.97/18.29	14.63/**17.92**	14.60/17.96	**14.55**/17.93
	GPTQ	W3	15.15/**18.62**	15.90/18.95	15.09/18.86	**15.09**/18.79
		W4	13.68/16.98	13.66/16.93	13.57/16.92	**13.55/16.92**

4.3 Compatibility with Other Quantizers

Because JARQ needs only the host's scales, codes, and zero points, it plugs into target correction, rotation, and lattice-based code search alike (Table 3). Starting from QEP+GPTQ, it lowers WT2 perplexity in all four settings, by 4.4% on Qwen3-8B at W3A16. After QuaRot+GPTQ on Llama-2-7B, it lowers both perplexities at both bit widths with the rotation fixed. After OJBKQ (Wang et al., 2026), which already searches codes with Babai–Klein decoding on a fixed grid, refitting the grid still lowers WT2 perplexity in all four settings. QuaRot and OJBKQ use C4 calibration; for Qwen3-4B GPTQ at W3A16, JARQ raises GSM8K with WT2, C4, or Pile calibration (Appendix B.3).

4.4 Where Do the Gains Come From?

Each ingredient is needed. Table 4(a) isolates the ingredients on 16 layers of Llama-2-7B and Qwen3-8B at W3A16 (Appendix C). *Groups must be fitted jointly:* fitting each group's scale on its own leaves a median error gap of 21–25% of the starting objective, caused by input correlations between groups (Proposition G.2). *Codes must move together:* after single-code search stops, one Babai proposal still lowers the objective in 40% (RTN) and 14% (GPTQ) of cases. *The grid must move:* at equal solver time, alternating updates beat code updates on the host grid in all 16 layers.

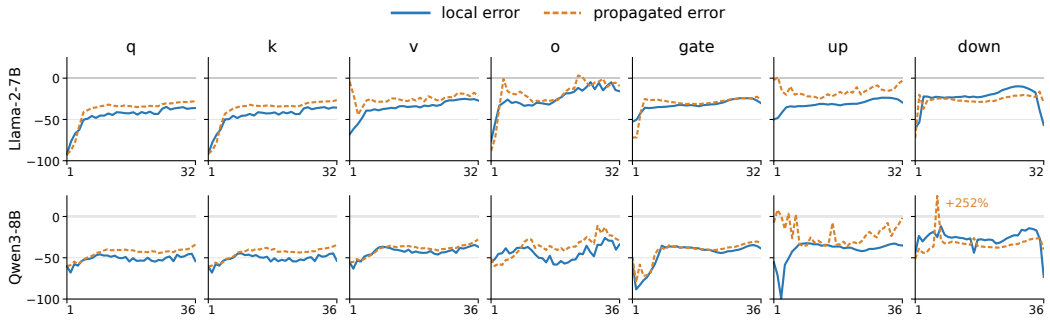


Figure 3: Per-block error change (%) from refinement with the AWQ host at W3A16 on held-out WT2, for the seven linear modules of Llama-2-7B (top) and Qwen3-8B (bottom); below zero is better. Local error uses fixed full-precision inputs, propagated error each model's own inputs. Local error falls in all 476 block–module pairs, propagated error in 468. Other settings: Appendix D.

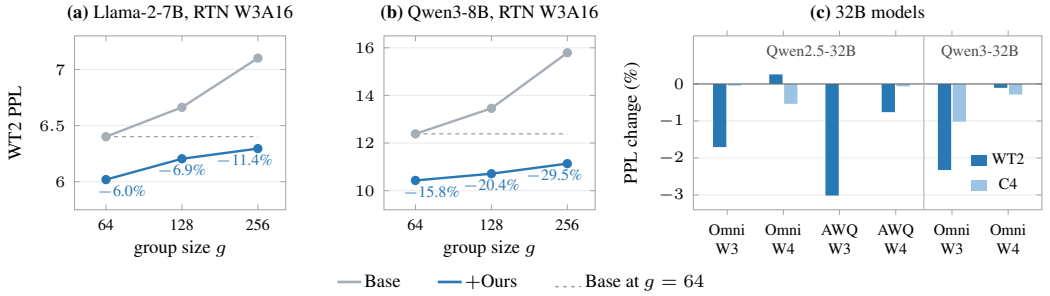


Figure 4: (a, b) WT2 perplexity of RTN at W3A16 versus group size. (c) Perplexity change from refinement on the 32B models (lower is better).

Table 4(b) supports this on full models: full JARQ beats a scale refit alone in 16 of 16 comparisons, whereas code updates on the host grid alone raise perplexity in 5 of 16.

Gains carry over to held-out data and through depth. On held-out WT2 and C4 inputs, refinement lowers the local error of 92.9% of 7,616 block–module comparisons on Llama-2-7B and Qwen3-8B. Figure 3 shows the AWQ host on both models: the local error of every projection falls at every depth, and the propagated error falls in 468 of the 476 block–module pairs. At the last block, block-output error falls in 23 of 32 comparisons, including 8/8 for RTN, 7/8 for GPTQ, and 6/8 for AWQ (Appendix D); OmniQuant, which already fits block outputs itself, gains least (2/8).

Coarser grids gain more, at modest cost. With RTN at W3A16, refinement lowers WT2 perplexity by 6.0%, 6.9%, and 11.4% on Llama-2-7B and by 15.8%, 20.4%, and 29.5% on Qwen3-8B as the group size grows from 64 to 128 to 256 (Figure 4a,b). At $g = 256$, refined RTN beats unrefined RTN at $g = 64$ on WT2 for both models while storing a quarter of the scales, and every weight-penalty strength $\nu \in \{0, 0.2, \ldots, 1\}$ improves both perplexities over RTN (Appendix F). $K = 3$ captures most gains (Figure 6) and adds 53–57 s per Llama-2-7B block and 36–38 s per Qwen3-4B block (Table 6); a full Llama-2-7B RTN run, including calibration, takes 32 min (Table 14).

5 Conclusion

Group-wise quantizers round weights onto a grid that is not refit to the resulting codes. JARQ refits all group scales jointly and moves codes in blocks on the new grid; code updates are more reliable after the grid has moved. It lowers perplexity in 90 of 96 comparisons, most at three bits, with no change to storage or inference cost. Its guarantee is per layer, and gains are smallest for strong hosts at four bits. Sharing QEP's target (Appendix E.5) and refitting zero points are next steps.

AI USE STATEMENT

Large language model assistants were used to edit the writing, check the consistency of reported numbers across tables and text, and help search the literature. All methods, experiments, results, and conclusions are the authors' own, and the authors verified all AI-assisted content.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 specify the solver; Appendix E gives all numerical settings and stopping rules; Appendix A gives models, calibration data, host settings, and evaluation protocols; and Appendix G contains all proofs of the results stated in Section 3. Code will be released upon publication.

REFERENCES

Yamato Arai and Yuma Ichikawa. Quantization error propagation: Revisiting layer-wise post-training quantization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=a3l3K9khbL.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. *arXiv preprint arXiv:2404.00456*, 2024. URL https://arxiv.org/abs/2404.00456.

László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6 (1):1–13, 1986.

Hicham Badri and Appu Shaji. Half-quadratic quantization of large machine learning models. https://dropbox.github.io/hqq_blog/, 2023.

Kayhan Behdin, Ayan Acharya, Aman Gupta, Qingquan Song, Siyu Zhu, Sathiya Keerthi, and Rahul Mazumder. QuantEase: Optimization-based quantization for language models. *arXiv preprint arXiv:2309.01885*, 2023.

Seohyeon Cha, Huancheng Chen, Dongjun Kim, Haoran Zhang, Kevin Chan, Gustavo de Veciana, and Haris Vikalo. CoreQ: Learning-free mismatch correction and successive rounding for quantization, 2026. URL https://arxiv.org/abs/2602.05902.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, 2023.

Jiale Chen, Yalda Shabanzadeh, Elvir Crnčević, Torsten Hoefler, and Dan Alistarh. The geometry of LLM quantization: GPTQ as Babai's nearest plane algorithm. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.18553. arXiv:2507.18553.

Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Kaokao Lv, and Yi Liu. Optimize weight rounding via signed gradient descent for the quantization of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

Elias Frantar and Dan Alistarh. Optimal Brain Compression: a framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4475–4488, 2022. URL https://papers.neurips.cc/paper_files/paper/2022/hash/1caf09c9f4e6b0150b06a07e77f2710c-Abstract-Conference.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022. URL https://arxiv.org/abs/2210.17323.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track*, 2021b.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *MLSys*, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

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, 2024. URL https://arxiv.org/abs/2405.16406.

Ryan Lucas, Mehdi Makni, Xiang Meng, Adam Deng, and Rahul Mazumder. ADMM-Q: An improved Hessian-based weight quantizer for post-training quantization of large language models. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=pXd1ueLloD.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, 2002.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, 2020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

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.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. In *International Conference on Machine Learning*, 2024.

Peter van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, Mathematisch Instituut, Universiteit van Amsterdam, 1981.

Xinyu Wang, Ziyu Zhao, Peng Lu, Yu Gu, and Xiao-Wen Chang. OJBKQ: Objective-joint Babai-Klein quantization, 2026. URL https://arxiv.org/abs/2602.08376.

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 *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating multi-bit quantization for recurrent neural networks. In *International Conference on Learning Representations*, 2018.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shihao Zhang, Haoyu Zhang, Ian Colbert, and Rayan Saab. Qronos: Correcting the past by shaping the future... in post-training quantization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 87775–87798, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8d22a09f18d9fc85dcbe6f482e13e1d7-Paper-Conference.pdf.

A Experimental Settings

This appendix specifies the quantization settings, the evaluated model states, and the quality and cost measurements.

A.1 Calibration and Host Quantizers

The four hosts in Table 1 use 128 WikiText-2 (WT2) training sequences of 2,048 tokens, seed 0, asymmetric weight quantization with group size $g = 128$, 3-bit or 4-bit weights, and 16-bit activations. Quality is evaluated after reconstructing the quantized weights in the model's floating-point dtype.

RTN rounds weights to the quantization grid. GPTQ uses activation ordering and damping 0.1; refinement keeps its group assignments in the reordered coordinates and restores the original coordinates when writing back the weights. OmniQuant uses 20 epochs of learnable weight clipping, with learnable equivalent transformations disabled. AWQ uses its official scale search and quantization routines with our calibration inputs. The Qwen OmniQuant and AWQ runs use native scaled dot-product attention.

Refinement starts from the host's scales, codes, and zero points. It updates scales and codes and keeps the zero points fixed. The reconstruction objective is (2) with $\nu = 0.6$. After each layer is refined, its outputs supply the inputs to the following layer. For every host, the final scales and codes define the returned weights. Numerical settings and early stopping are given in Appendix E.

A.2 Perplexity and Iteration Budget

All main experiments use an iteration budget of $K = 3$ with early stopping. This choice follows the budget sweep on Llama-2 in Figure 6, where three iterations capture most of the WT2 perplexity reduction for RTN hosts, with smaller gains from additional iterations. Base denotes the host model ($K = 0$), and +Ours denotes the model refined with $K = 3$. The QuaRot runs also use $K = 3$ and Algorithm 1, as detailed in Appendix E.4. The Llama-2-7B and Qwen3-8B RTN entries in Table 1 are the $g = 128$, $\nu = 0.6$ states of the group-size and regularization ablations (Appendix F).

Llama-3-8B. We use the original base checkpoint, `meta-llama/Meta-Llama-3-8B`. Table 1 reports all four hosts at W3A16 and W4A16. Within each configuration, WT2 and C4 are evaluated on the same model state. Table 2 evaluates the same states and the original FP16 model on five multiple-choice tasks.

Qwen3-4B. We use `Qwen/Qwen3-4B` with native scaled dot-product attention. Table 1 includes the FP16 reference and all four hosts at W3A16 and W4A16. Perplexity is measured from the saved full-model states after all 36 Transformer blocks have been processed. The host and refinement streams propagate their own preceding outputs: host calibration uses the host prefix, while refinement uses the refined prefix. WT2 and C4 are evaluated on the same saved state for each entry.

A.3 Downstream Evaluation

Multiple-choice tasks. The five tasks are ARC-Easy and ARC-Challenge (Clark et al., 2018), MMLU (Hendrycks et al., 2021a), WinoGrande (Sakaguchi et al., 2021), and BoolQ (Clark et al., 2019). ARC-Easy and ARC-Challenge use zero-shot normalized accuracy; WinoGrande and BoolQ use zero-shot accuracy; MMLU uses five-shot accuracy. Avg. is their unweighted mean,

$$\text{Avg.} = \frac{\text{ARC-E} + \text{ARC-C} + \text{MMLU} + \text{WinoGrande} + \text{BoolQ}}{5}$$

Generative tasks. Table 10 (Appendix B.4) reports GSM8K accuracy for Llama-3-8B, Qwen2.5-7B, and Qwen3-8B with GPTQ, OmniQuant, and AWQ at W3A16 and W4A16 (except GPTQ on Llama-3-8B at W3A16), together with FP16 references. Figure 2 compares GSM8K and MATH-500 at W4A16 on the same models; its panel averages are unweighted means over the three hosts for each model and task. GSM8K (Cobbe et al., 2021) uses five-shot strict exact-match scoring and MATH-500 (Hendrycks et al., 2021b; Lightman et al., 2024) uses four-shot exact-match scoring.

Evaluation settings. All tasks use the full task splits, seed 0, completion prompts, greedy generation, batch size 1, and context length 4,096. For Llama-3-8B, completion prompts include the beginning-of-sequence token. Base and +Ours ($K = 0$ and $K = 3$) use the same questions, prompts, and scoring settings. FP16 rows use the original models with the same task data and evaluation settings.

A.4 Compatibility with Other Quantizers

QEP. We use the official QEP correction followed by GPTQ, with 128 WT2 calibration sequences of 2,048 tokens, $g = 128$, seed 0, native scaled dot-product attention, and W3A16/W4A16. Refinement starts from the resulting quantized model and uses the original full-precision weights in its reconstruction target; QEP's correction step is performed before refinement. The WT2 values in Table 3(a) use $K = 3$, as in Table 1.

QuaRot. We use the official Hadamard rotation and GPTQ implementation on Llama-2-7B, with 128 C4 calibration sequences of 2,048 tokens, $g = 128$, seed 0, and W3A16/W4A16. The rotation and zero points stay fixed during refinement, and weights and inputs are represented in the same rotated coordinates. Refinement uses Algorithm 1 with $K = 3$ and $\nu = 0.6$ (Appendix E.4). Evaluation uses the QuaRot evaluator.

OJBKQ. Table 3(c) reports our reproduction of OJBKQ (Wang et al., 2026) on Qwen3-8B and Llama-3-8B at W3A16/W4A16, with 128 C4 calibration sequences of 2,048 tokens, $g = 128$, and BF16 activations. On Qwen3-8B, the reproduced WT2 perplexities match the published values (10.98 at W3A16 and 9.94 at W4A16; Table 1 of Wang et al. (2026)). Base is the reproduced OJBKQ model; +Ours refines its scales and codes through the same host interface as above, with $K = 3$ and $\nu = 0.6$. Both columns report measured WT2 perplexity.

A.5 Efficiency Measurements

We measure the first Transformer block of Qwen3-8B at W3A16 and $g = 128$, including its seven attention and MLP linear layers. Inputs are the same 128 WT2 calibration sequences of 2,048 tokens, with seed 0. Each of three repetitions uses one NVIDIA A100. Within a repetition, OmniQuant runs its 20-epoch weight clipping once, and every budget $K > 0$ refines that same result with the same inputs and with early stopping enabled. Table 5 and Figure 5 report the results.

Table 5: Time and memory for the first Qwen3-8B block at W3A16 with the OmniQuant host. Times are mean ± sample standard deviation over three repetitions; total time includes the host. Memory is the time-weighted mean over both stages.

K	Refinement time (s)	Total time (s)	Overhead	GPU memory (GiB)	CPU memory (GiB)
0	0.00 ± 0.00	128.37 ± 0.83	—	2.07	6.63
3	48.43 ± 0.29	176.80 ± 1.12	+37.73%	2.48	5.95
5	57.61 ± 0.41	185.98 ± 1.20	+44.88%	2.53	5.86
7	67.15 ± 0.74	195.52 ± 1.50	+52.31%	2.58	5.79
10	80.65 ± 0.55	209.02 ± 1.37	+62.83%	2.64	5.69

Time. The host timer includes parameter search and quantization. The refinement timer includes computing calibration responses, QR reduction, scale and code updates, proposal comparisons, output propagation, and synchronization. Model loading, input capture, warmup, serialization, and result validation are excluded. For repetition r, total time is $T_{\mathrm{total},r} = T_{\mathrm{host},r} + T_{\mathrm{ref},r}$, and overhead is $100\,\overline{T}_{\mathrm{ref}}/\overline{T}_{\mathrm{host}}$. These timings measure quantization cost, not inference latency.

Memory. GPU memory is allocated tensor memory; CPU memory is process resident memory. For repetition r, we report the time-weighted mean over the two stages,

$$\overline{M}_{\mathrm{comp},r} = \frac{T_{\mathrm{host},r}\overline{M}_{\mathrm{host},r} + T_{\mathrm{ref},r}\overline{M}_{\mathrm{ref},r}}{T_{\mathrm{host},r} + T_{\mathrm{ref},r}}, \tag{12}$$

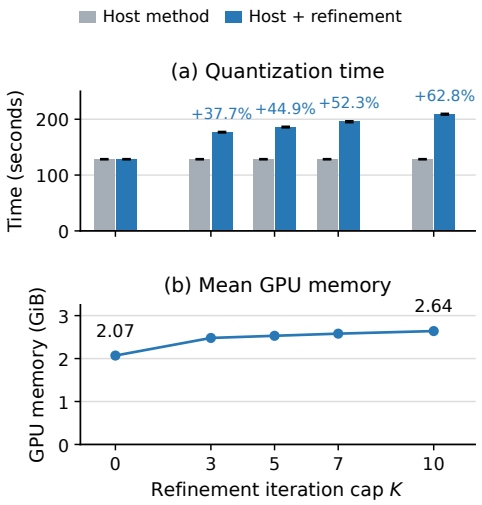


Figure 5: Cost on the first Qwen3-8B block (OmniQuant, W3A16). (a) Host time ($K = 0$) versus host-plus-refinement time (mean $\pm$ sample SD, three runs); labels give the overhead. (b) Time-weighted mean GPU memory. Values are in Table 5.

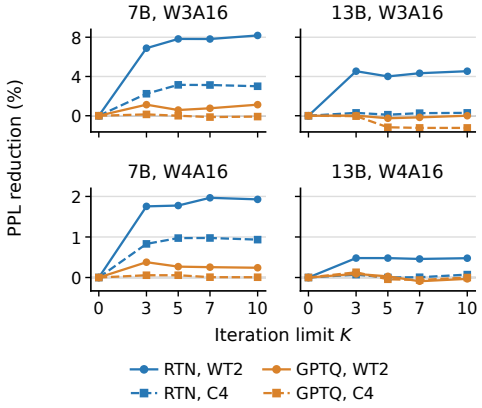


Figure 6: Llama-2 perplexity versus iteration budget K. Reduction is $100(1-\mathrm{PPL}_K/\mathrm{PPL}_0)$; higher is better.

averaged over the three repetitions. The CPU mean can decrease with K when refinement spends more time in a lower-memory stage.

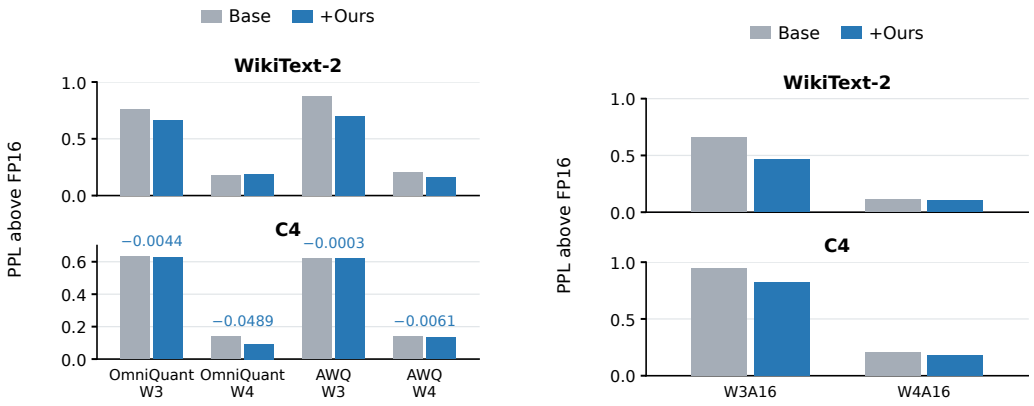
Cost per host. Table 6 measures every host on Llama-2-7B and Qwen3-4B at W3A16, $g = 128$, and $K = 3$, over the first Transformer block and over the first two blocks. Each entry is the median of the completed repetitions (one or two per entry). Refinement adds 53–57 s per Llama-2-7B block and 36–38 s per Qwen3-4B block regardless of the host, and the added time for two blocks is about twice that for one. At this rate, refining all 32 blocks of Llama-2-7B takes about 30 minutes, consistent with the full-model RTN runs in Table 14.

Table 6: Quantization time (s) of each host alone and with refinement (+Ours) at W3A16 over the first block and the first two blocks; Δ is the time added by refinement. Medians of the completed repetitions (one or two per entry).

		One block			Two blocks		
Model	Host	Host	+Ours	Δ	Host	+Ours	Δ
Llama-2-7B	RTN	4.9	61.5	56.6	7.8	119.8	112.0
	GPTQ	33.0	86.6	53.6	64.0	170.1	106.1
	AWQ	28.4	81.5	53.1	49.3	154.0	104.7
	OmniQuant	69.7	126.6	56.9	135.6	244.9	109.3
Qwen3-4B	RTN	2.8	40.3	37.5	4.8	79.4	74.6
	GPTQ	25.0	61.0	36.0	47.2	119.2	72.0
	AWQ	17.1	53.0	35.9	32.2	105.0	72.8
	OmniQuant	45.4	83.1	37.7	89.4	163.8	74.4

B Additional Results

This appendix reports perplexity on the 32B models and the full GSM8K results. Figure 7 summarizes the 32B models relative to their FP16 references.



(a) Qwen2.5-32B: OmniQuant and AWQ

(b) Qwen3-32B: OmniQuant

Figure 7: Perplexity gap to FP16, $\mathrm{PPL} - \mathrm{PPL}_{\mathrm{FP16}}$, on the 32B models (lower is better). Gray is Base ($K = 0$); blue is +Ours ($K = 3$). Axes start at zero, with a tighter range for C4 in (a). Labels give $\mathrm{PPL}_{+\mathrm{Ours}} - \mathrm{PPL}_{\mathrm{Base}}$. Absolute values are in Tables 7 and 8.

B.1 Qwen2.5-32B Perplexity

We use the calibration and host settings of Appendix A.1. Evaluation covers all complete, nonoverlapping 2,048-token WT2 test windows and 256 C4 validation windows sampled with seed 0. The evaluated weights use FP16, with native scaled dot-product attention and FP32 logits and loss. WT2 and C4 are evaluated on the same model state within each configuration.

Table 7 compares each host with its refinement at $K = 3$ and with the FP16 reference. WT2 perplexity decreases in three of four configurations, by up to 3.02%, and C4 perplexity in all four, by up to 0.54%.

Table 7: Qwen2.5-32B perplexity (lower is better). Base uses $K = 0$ and +Ours uses $K = 3$. Bold marks the better value in each Base/+Ours pair (compared before rounding).

		WT2		C4	
Host	W/A	Base	+Ours	Base	+Ours
FP16	W16A16	5.0176		8.9503	
OmniQuant	W3A16	5.7804	**5.6818**	9.5809	**9.5765**
	W4A16	**5.1951**	5.2085	9.0920	**9.0431**
AWQ	W3A16	5.8960	**5.7181**	9.5722	**9.5719**
	W4A16	5.2222	**5.1824**	9.0894	**9.0833**

B.2 Qwen3-32B Perplexity

We quantize the Qwen3-32B dense model with the OmniQuant host at W3A16 and W4A16. Table 8 compares each host with its refinement at $K = 3$. Perplexity decreases on WT2 and C4 at both bit widths, with larger reductions at W3A16.

B.3 Calibration Corpus

Table 9 varies only the calibration corpus for GPTQ on Qwen3-4B at W3A16, using 128 sequences of 2,048 tokens from WT2, C4, or the Pile (Gao et al., 2020). For each corpus, JARQ ($K = 3$) starts from the GPTQ model calibrated on that corpus and uses the same calibration data. GSM8K uses five-shot strict exact match and MATH-500 four-shot Minerva exact match, with the settings

Table 8: Qwen3-32B perplexity with the OmniQuant host (lower is better). Base uses $K = 0$ and +Ours uses $K = 3$. Bold marks the better value in each Base/+Ours pair (compared before rounding).

	WT2		C4	
W/A	Base	+Ours	Base	+Ours
W3A16	8.2562	**8.0641**	11.7171	**11.5977**
W4A16	7.7106	**7.7025**	10.9813	**10.9500**

of Appendix A.3. Refinement raises GSM8K accuracy for all three corpora, by 6.52, 6.21, and 2.04 points. MATH-500 accuracy rises with C4 and Pile calibration and falls by 1.2 points with WT2.

Table 9: GSM8K and MATH-500 accuracy (%; higher is better) of Qwen3-4B with GPTQ at W3A16 under three calibration corpora. Bold marks the better value in each Base/+Ours pair.

	GSM8K		MATH-500	
Calibration	Base	+Ours	Base	+Ours
WT2	59.67	**66.19**	**29.4**	28.2
C4	61.49	**67.70**	29.8	**30.2**
Pile	69.98	**72.02**	42.4	**45.6**

B.4 GSM8K Accuracy

Table 10 reports GSM8K accuracy for Llama-3-8B, Qwen2.5-7B, and Qwen3-8B with the GPTQ, OmniQuant, and AWQ hosts at W3A16 and W4A16; the W4A16 pairs are plotted in Figure 2. Refinement raises accuracy in 14 of the 17 reported pairs: six of eight at W3A16 (one tie) and eight of nine at W4A16. Evaluation settings are in Appendix A.3.

C Component Ablations

C.1 Setup

We test 16 linear layers from Llama-2-7B and Qwen3-8B with the RTN and GPTQ hosts, sampling 32 output channels per layer at W3A16 and $g = 128$. Alternating refinement uses an iteration budget of $K = 3$. Inputs come from the full-precision model and stay fixed during each test. Separate held-out inputs measure reconstruction beyond the samples used for fitting.

For each layer, we sum the objective over the sampled channels before computing relative differences, and we report medians across layers. Timings use one synchronized FP64 measurement after warmup on an NVIDIA RTX A6000; scale-update timers include constructing the scale system. Table 4 collects the results of the three tests below.

C.2 Joint Scale Fitting

With codes fixed, independent fitting matches each group's contribution separately, coordinate descent updates one scale at a time against the full residual, and joint fitting solves for all scales at once. The reported gap is $100(L_{\text{method}} - L_{\text{joint}})/L_{\text{base}}$, where L_{base} is the objective before scale fitting. The first block of Table 4 compares the reconstruction error and solve time of the three fits with the same fixed codes.

C.3 Multi-Code Updates

Each case is one output channel and one group of 128 weights. With the scales and the other groups fixed, we run single-code updates until no one-code change strictly lowers the objective; a separate FP64 check tests every allowed code for every coordinate. A Babai proposal is then computed from

Table 10: GSM8K accuracy (%; higher is better). Gray marks FP16 references and blue shading marks +Ours. Bold marks the better value in each Base/+Ours pair; ties are not bolded. — denotes a configuration not reported.

Model	Host	W/A	Base	+Ours
Llama-3-8B	FP16	W16A16	47.92	
	GPTQ	W3A16	—	—
		W4A16	42.53	**42.61**
	OmniQuant	W3A16	20.17	**22.29**
		W4A16	42.68	**43.67**
	AWQ	W3A16	21.30	21.30
		W4A16	42.61	**43.37**
Qwen2.5-7B	FP16	W16A16	83.70	
	GPTQ	W3A16	69.22	**69.37**
		W4A16	**80.89**	79.91
	OmniQuant	W3A16	60.05	**68.61**
		W4A16	80.06	**81.65**
	AWQ	W3A16	67.17	**70.74**
		W4A16	78.17	**81.27**
Qwen3-8B	FP16	W16A16	90.30	
	GPTQ	W3A16	75.21	**80.67**
		W4A16	88.63	**88.70**
	OmniQuant	W3A16	81.05	**81.20**
		W4A16	88.78	**89.16**
	AWQ	W3A16	**82.56**	81.43
		W4A16	88.17	**90.30**

this state. The second block of Table 4 reports the number and share of accepted proposals and the minimum and median number of codes they change.

C.4 Fixed-Grid versus Joint Refinement

Fixed keeps the host scales and updates only codes; Joint alternates scale and code updates using Algorithm 1. Both start from the same weights. Fixed receives the time used by Joint and returns its last complete sweep within that time; shared factorization and diagnostic work are excluded. The last block of Table 4 reports the median held-out error reduction and the number of layers improved.

C.5 Full-Model Component Ablation

Table 11 applies each update alone to the full model on Llama-3-8B and Qwen3-4B, with the settings of Table 1. *Codes only* runs one code sweep of Algorithm 1 on the host grid; *Scales only* runs one joint scale fit with the host codes. Both variants start independently from the same host state and propagate their own outputs block by block. With the codes fixed, the joint fit has a unique solution, so repeating it changes nothing; the iteration budget of full JARQ is studied in Figure 6.

Each update alone is effective for the RTN host: on Llama-3-8B at W3A16, codes only lowers WT2 perplexity by 27.9% and scales only by 28.8%, so the two partly correct the same error. Their interaction depends on the grid. Codes only raises perplexity in 5 of the 16 comparisons, including WT2 for both GPTQ settings at W3A16, whose codes already come from a nearest-plane recursion on a related lattice. Full JARQ, which alternates both updates, is below Scales only in all 16 comparisons: by 4.2% and 9.9% on WT2 for RTN at W3A16, and by at most 0.3% on WT2 at W4A16. Full JARQ gives the lowest WT2 perplexity in 7 of the 8 settings; on Llama-3-8B with GPTQ at W3A16, no variant improves WT2 over the host, while full JARQ gives the lowest C4 perplexity.

Table 11: Full-model component ablation (WT2/C4 perplexity; lower is better). Bold marks the lowest value per setting and corpus.

Model	Host	W/A	Host	Codes only	Scales only	Full JARQ
Llama-3-8B	RTN	W3A16	12.048/16.444	8.689/14.171	8.580/13.311	**8.219/12.667**
		W4A16	6.727/9.636	6.586/9.605	6.490/9.583	**6.481/9.532**
	GPTQ	W3A16	**7.618**/13.242	7.714/12.318	7.784/12.628	7.649/**12.011**
		W4A16	6.423/**9.386**	6.419/9.416	6.417/9.411	**6.416**/9.408
Qwen3-4B	RTN	W3A16	22.447/27.013	21.205/27.078	17.968/22.316	**16.194/21.625**
		W4A16	15.968/18.291	14.629/**17.924**	14.598/17.965	**14.555**/17.930
	GPTQ	W3A16	15.153/**18.623**	15.903/18.953	15.091/18.859	**15.089**/18.791
		W4A16	13.675/16.980	13.665/16.932	13.570/16.923	**13.553/16.917**

D LAYER-WISE ERROR PROFILES

This appendix tracks how refinement changes errors inside the model, from single linear modules to the output of each Transformer block.

D.1 SETUP

We profile every block and linear module of Llama-2-7B (32 blocks) and Qwen3-8B (36 blocks) with the RTN, GPTQ, AWQ, and OmniQuant hosts at W3A16 and W4A16, comparing each host with host+JARQ ($K = 3$, $\nu = 0.6$, $g = 128$). Calibration uses the 128 WT2 training sequences of the main experiments. Errors are measured on held-out data: 32 sequences of 2,048 tokens each from the WT2 test set and the C4 validation set, over all tokens and output channels. The 16 host, bit-width, and model configurations cover 544 block states and 3,808 matrices, giving 7,616 paired module measurements over the two corpora.

Each portrait page contains one 7×8 linear-module grid for one model and metric. Rows are the seven projections of a block (`q`, `k`, `v`, `o`, `gate`, `up`, `down`); the four left columns are the hosts on WT2 and the four right columns the same hosts on C4. Blue and orange mark W3A16 and W4A16; solid and dashed lines mark the host and host+JARQ. The horizontal axis is the block index; all vertical axes are logarithmic with limits set per panel.

D.2 METRICS

Squared errors are summed over all tokens and output channels; NMSE divides the total squared error by the total energy of the full-precision reference. Products are computed in FP32 from FP16 activations and summed in FP64.

Block-output NMSE. The full-precision and quantized models each propagate the same token sequences, and

$$E^{\mathrm{block}}_{\ell,s} = \frac{\|\mathbf{H}^s_\ell - \mathbf{H}^{\mathrm{FP}}_\ell\|_F^2}{\|\mathbf{H}^{\mathrm{FP}}_\ell\|_F^2} \tag{13}$$

compares the hidden state after block ℓ, including normalization, nonlinearities, and residual connections, for state s (host or host+JARQ); the last block is measured before the final normalization and output head.

Local linear NMSE. Both states receive the same full-precision input to module a:

$$E^{\mathrm{local}}_{\ell,a,s} = \frac{\|\mathbf{X}^{\mathrm{ref}}_{\ell,a}(\mathbf{W}^s_{\ell,a} - \mathbf{W}^{\mathrm{ref}}_{\ell,a})\|_F^2}{\|\mathbf{Y}^{\mathrm{ref}}_{\ell,a}\|_F^2}, \qquad \mathbf{Y}^{\mathrm{ref}} = \mathbf{X}^{\mathrm{ref}}\mathbf{W}^{\mathrm{ref}} + \mathbf{b}^{\mathrm{ref}}. \tag{14}$$

This isolates the module's own weight error. For AWQ, the reference block uses the unclipped, transformed full-precision weights fed by the original full-precision block inputs, so both states are compared in the same coordinates.

Propagated output SSE. Each state feeds the module its own propagated input:

$$S^{\mathrm{out}}_{\ell,a,s} = \|\mathbf{X}^{\mathrm{FP}}_{\ell,a}\mathbf{W}^{\mathrm{FP}}_{\ell,a} - \mathbf{X}^s_{\ell,a}\mathbf{W}^s_{\ell,a}\|_F^2, \tag{15}$$

so the error includes the effect of all preceding modules.

Combined error. As a supplementary metric, we add a weight penalty with coefficient 0.6,

$$C_{\ell,a,s} = S^{\mathrm{out}}_{\ell,a,s} + 0.6\|\mathbf{W}^{\mathrm{FP}}_{\ell,a} - \mathbf{W}^s_{\ell,a}\|_F^2, \qquad C^{\mathrm{block}}_{\ell,s} = \sum_a C_{\ell,a,s}, \tag{16}$$

where the sum runs over the seven modules of block ℓ and the weight penalty is counted once per matrix and corpus. Unlike the refinement objective (2), which uses the calibration inputs $\widetilde{\mathbf{X}}$ and coefficient $\nu^2 = 0.36$, these metrics use held-out inputs. For AWQ, propagated quantities also contain the difference between folded and original coordinates, so their raw magnitudes are not comparable across hosts or modules.

D.3 RESULTS

Table 12 counts the pairs in which host+JARQ has strictly lower error than the host.

Table 12: Pairs in which refinement lowers the error. The first three columns count all modules and blocks of both models, bit widths, and corpora; the last two count the final block only.

Host	Local NMSE	Linear output SSE	Linear combined	Final block NMSE	Final block combined
RTN	1,818/1,904	1,888/1,904	1,888/1,904	8/8	8/8
GPTQ	1,739/1,904	1,447/1,904	1,443/1,904	7/8	6/8
AWQ	1,721/1,904	1,648/1,904	1,644/1,904	6/8	8/8
OmniQuant	1,794/1,904	1,456/1,904	1,456/1,904	2/8	3/8

Local gains hold on held-out data. Local NMSE decreases in 7,072 of 7,616 module pairs (92.9%) and in at least 90% of pairs for every host, including OmniQuant (Figures 8 and 9). The reduction in the calibration objective therefore carries over to held-out inputs.

Propagation depends on the host. At the final block, block-output NMSE decreases in 23 of 32 comparisons: 8/8 for RTN, 7/8 for GPTQ, 6/8 for AWQ, and 2/8 for OmniQuant. A likely reason is that OmniQuant optimizes block outputs directly, and refinement toward the layer target can undo part of that compensation. For Llama-2-7B with OmniQuant at W3A16 on C4, all seven local NMSEs of the last block decrease, and `down_proj` falls by 47.51% (Figure 8), while its propagated SSE rises by 37.72% (Figure 10) and the final block-output NMSE rises from 0.08358 to 0.10799.

Two propagation-sensitive modules. All 14 of the 7,616 module pairs in which refinement more than doubles the propagated SSE come from the `down_proj` of one early block per model: block 2 of Llama-2-7B and block 7 of Qwen3-8B. There, the host's propagated SSE is 77–1,133 and 6–34 times that of the preceding block, and refinement raises it in 9 and 13 of the 16 host, bit-width, and corpus settings. At block 7 of Qwen3-8B, local NMSE still falls in all 13 of those settings: the refined weights fit better, but these modules amplify the small input drift left by the refined preceding blocks. The main target $\widetilde{\mathbf{X}}\mathbf{W}$ ignores such drift by design; the propagation-corrected target of Appendix E.5 accounts for it.

AWQ coordinates. For Qwen3-8B with AWQ at W3A16 on C4, the propagated SSE of the third block's `up_proj` changes from 3.30×10^{14} to 3.32×10^{14} (Figure 11), while its aligned local NMSE falls by 98.75% (Figure 9). Large raw SSE values in AWQ panels reflect its folded coordinates rather than larger quantization error.

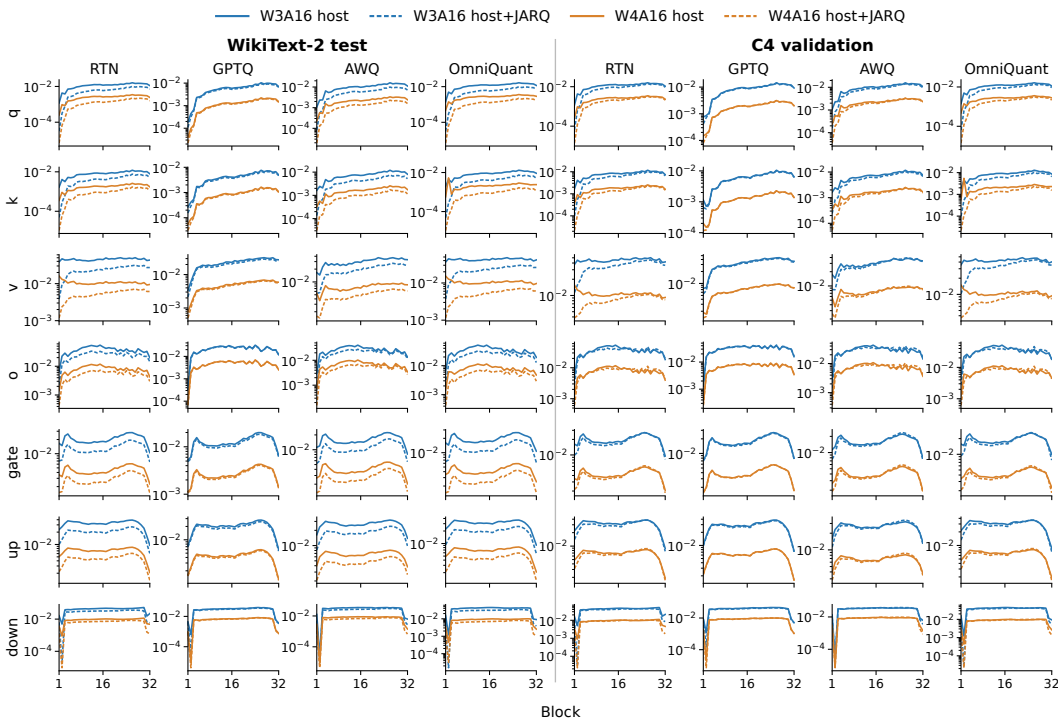


Figure 8: Local linear NMSE (14) for Llama-2-7B. Rows are the seven projections; the left and right halves are WT2 and C4. At the final block, refinement lowers it in 104/112 module pairs (RTN 26/28, GPTQ 24/28, AWQ 26/28, OmniQuant 28/28).

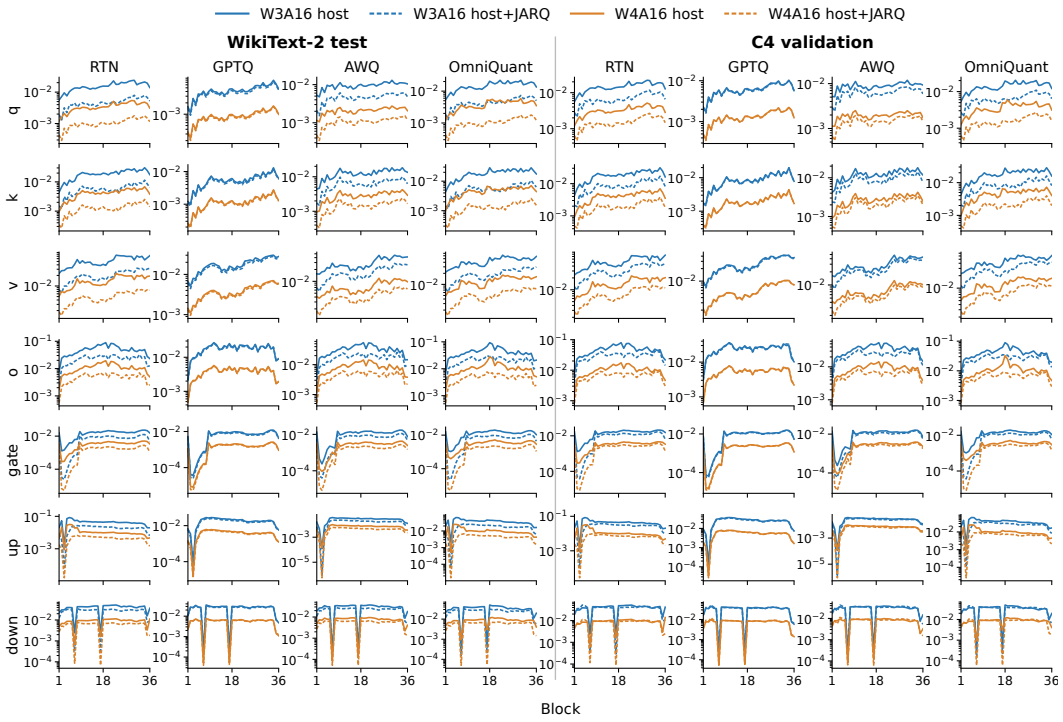


Figure 9: Local linear NMSE (14) for Qwen3-8B, as in Figure 8. At the final block, refinement lowers it in 104/112 module pairs (RTN 28/28, GPTQ 20/28, AWQ 28/28, OmniQuant 28/28).

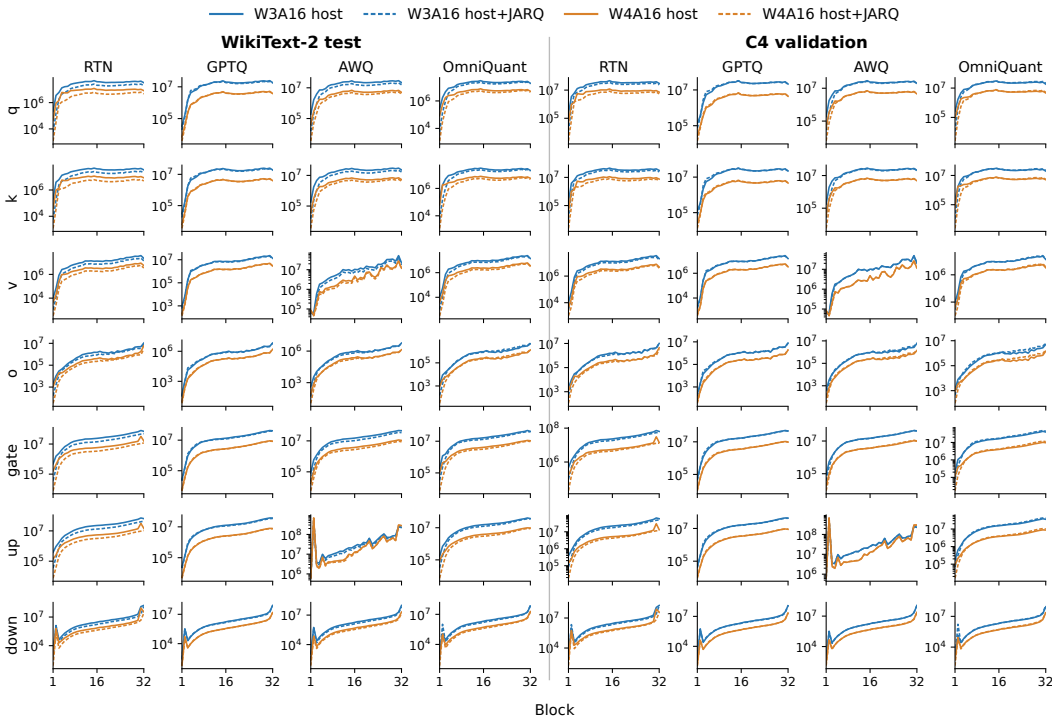


Figure 10: Propagated output SSE (15) for Llama-2-7B, laid out as in Figure 8. At the final block, refinement lowers it in 75/112 module pairs (RTN 27/28, GPTQ 16/28, AWQ 21/28, OmniQuant 11/28).

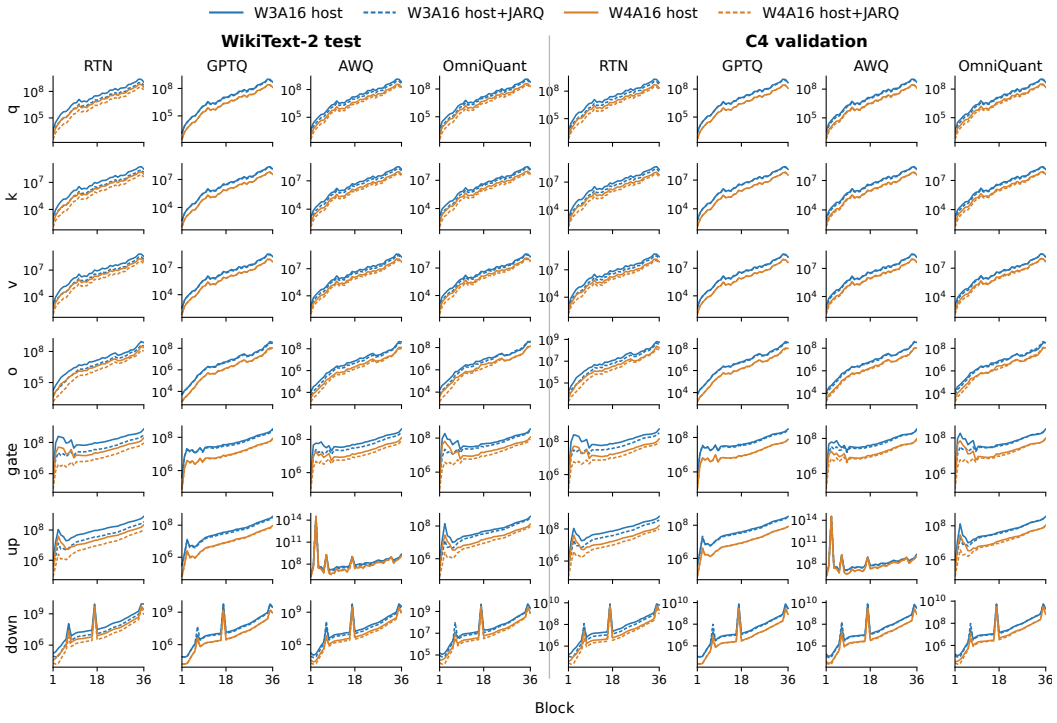


Figure 11: Propagated output SSE (15) for Qwen3-8B, laid out as in Figure 8. At the final block, refinement lowers it in 105/112 module pairs (RTN 28/28, GPTQ 28/28, AWQ 24/28, OmniQuant 25/28).

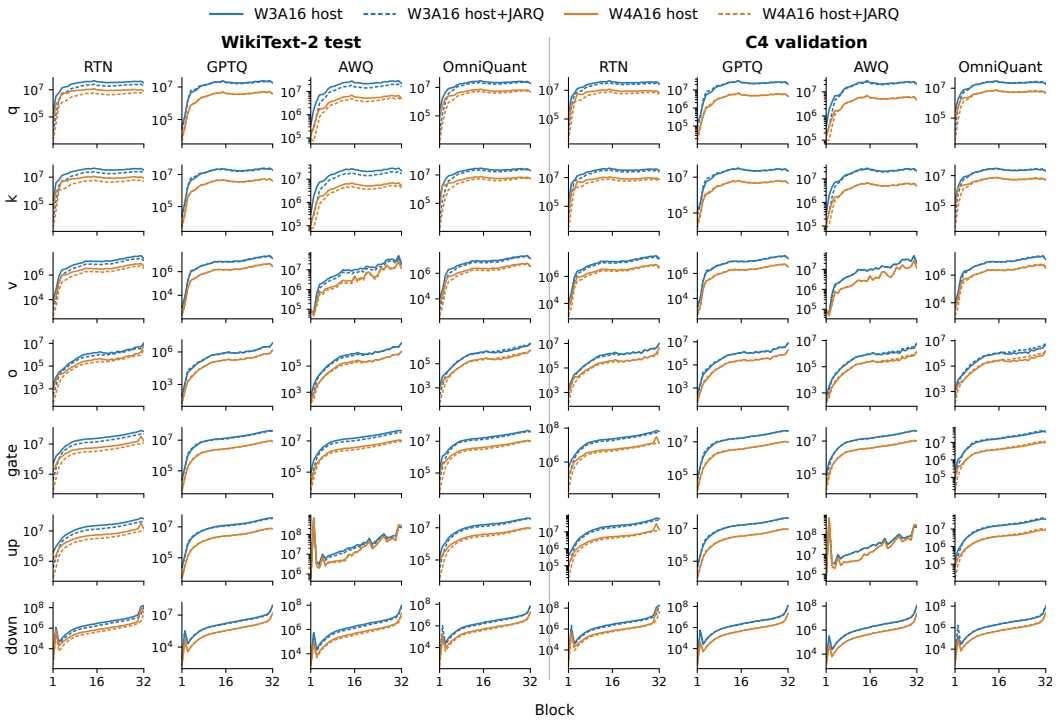


Figure 12: Per-module combined error (16) for Llama-2-7B, laid out as in Figure 8. At the final block, refinement lowers it in 75/112 module pairs (RTN 27/28, GPTQ 16/28, AWQ 21/28, OmniQuant 11/28).

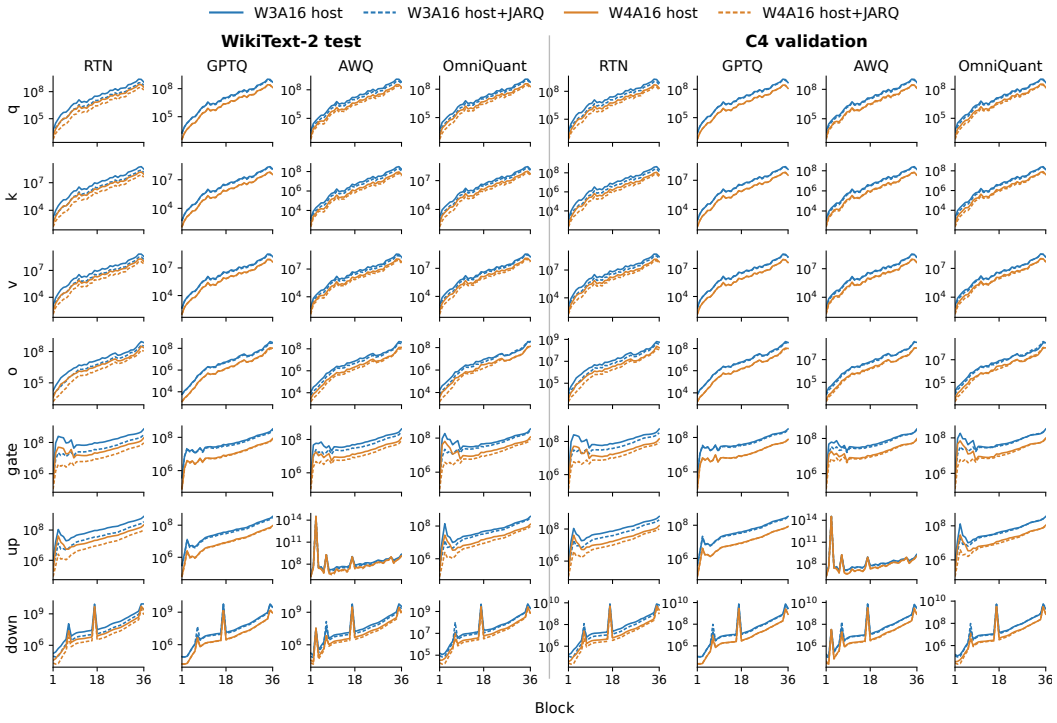


Figure 13: Per-module combined error (16) for Qwen3-8B, laid out as in Figure 8. At the final block, refinement lowers it in 105/112 module pairs (RTN 28/28, GPTQ 28/28, AWQ 24/28, OmniQuant 25/28).

E Implementation Details

E.1 Reconstruction after QR Reduction

For the thin QR factorization $\mathbf{A} = \mathbf{U}\mathbf{R}$ and $\overline{\mathbf{Y}} = \mathbf{U}^\mathsf{T}\mathbf{Y}$, orthogonality gives

$$\|\mathbf{Y} - \mathbf{A}\widehat{\mathbf{W}}\|_F^2 = \|\overline{\mathbf{Y}} - \mathbf{R}\widehat{\mathbf{W}}\|_F^2 + \|(\mathbf{I} - \mathbf{U}\mathbf{U}^\mathsf{T})\mathbf{Y}\|_F^2. \tag{17}$$

The second term does not depend on the quantized weights, so the original objective (2) and the reduced objective (11) have the same minimizers. Appending $\nu\mathbf{I}$ to the activation matrix makes $\mathbf{A}$ full column rank when $\nu > 0$.

E.2 Numerical Updates and Stopping

Order of updates. Algorithm 1 summarizes the layerwise alternating updates; the stopping rule below can stop a channel before its iteration budget is exhausted. Within a layer, the input factors and reconstruction target stay fixed. Each iteration fits the scales and then proposes group-level integer updates. A code proposal is accepted only if it lowers the reconstruction objective. The refined layer's outputs become the inputs to the next layer. Group QR uses no additional column ordering; any ordering introduced by GPTQ is preserved during refinement and inverted when the weights are written back.

Solving for scales. The scale system is solved in FP32. Let $\mathbf{G}_j = \mathbf{B}_j^\mathsf{T}\mathbf{B}_j$. We symmetrize this matrix before factorization. If its spectrum is nonfinite, its smallest eigenvalue is at most $r_{\mathrm{cond}} \max(|\lambda_{\max}(\mathbf{G}_j)|, \epsilon_{\mathrm{fp}})$, or Cholesky factorization fails, we add $\lambda\mathbf{I}$ and factorize again, with $\lambda = 10^{-4}$, $r_{\mathrm{cond}} = 10^{-10}$, and ϵ_{fp} equal to FP32 machine epsilon. If factorization still fails, the solve stops with a numerical error. Thus $\lambda_j = 0$ for the ordinary solve and $\lambda_j = 10^{-4}$ when stabilization is triggered. The stabilized branch solves a ridge-regularized scale subproblem, not the exact least-squares problem in (6), and its result is kept without a separate acceptance test on the original reconstruction objective; iteration-wise monotonicity is therefore not guaranteed for this branch (Remark 1).

Early stopping. Let $L_j^{(k)}$ be the full reconstruction objective immediately after the scale update, including the constant term in (17). For $k \geq 2$, an output channel stops before the next code update when

$$\frac{L_j^{(k-1)} - L_j^{(k)}}{\max(|L_j^{(k-1)}|, \epsilon_{\mathrm{fp}})} \leq \tau \quad \text{or} \quad |L_j^{(k-1)} - L_j^{(k)}| \leq \tau, \qquad \tau = 10^{-5}. \tag{18}$$

The new scales are kept. At termination, the current scales and codes define the returned weights, which are cast to the model's weight dtype and installed in the layer.

Small scales. Code proposals are computed only when $|s_{i,j}|$ exceeds FP32 machine epsilon; groups with smaller scales keep their codes. A proposal is rejected if its residual is nonfinite or if it does not lower the conditional reconstruction objective.

E.3 Representing Negative Scales

All quality evaluations use the final signed scales directly. For a deployment format that supports the resulting scales and zero points, a negative scale can be converted to a positive one without changing the weights it represents. Let $q_{\max} = 2^b - 1$. For $s_{i,j} < 0$, set

$$s'_{i,j} = -s_{i,j}, \qquad \mathbf{q}'_{i,j} = q_{\max}\mathbf{1} - \mathbf{q}_{i,j}, \qquad z'_{i,j} = q_{\max} - z_{i,j}. \tag{19}$$

The transformed codes remain in the same allowed integer range, and

$$s'_{i,j}(\mathbf{q}'_{i,j} - z'_{i,j}\mathbf{1}) = (-s_{i,j})(-\mathbf{q}_{i,j} + z_{i,j}\mathbf{1}) = s_{i,j}(\mathbf{q}_{i,j} - z_{i,j}\mathbf{1}). \tag{20}$$

This optional conversion is applied after refinement; zero points stay fixed during optimization. Fitting the offset jointly with the scale, as an affine term $s_{i,j}z_{i,j}$, would keep the scale step linear but would yield real-valued offsets, whereas group-wise formats store integer zero points; fixing $\mathbf{Z}$ keeps the host's format.

E.4 Refinement with QuaRot

The QuaRot results in Table 3(b) use Algorithm 1 in the rotated coordinates, starting directly from the scales, codes, and zero points of QuaRot+GPTQ. Each iteration jointly fits the scales and then performs group-wise bounded Babai updates using already rounded higher-index codes during back substitution, as in (10). The iteration budget is $K = 3$, and the reconstruction objective is (2) with $\nu = 0.6$. The Hadamard rotation, group assignments, and zero points stay fixed. Numerical stabilization, stopping, and code-proposal acceptance follow Appendix E.2. The final scales and codes define the weights installed in the rotated model.

E.5 Propagation-Corrected Targets and QEP

The main target $\mathbf{Y}^\star = \widetilde{\mathbf{X}}\mathbf{W}$ asks each layer to reproduce the original weights' response to the inputs it actually receives. It does not compensate for errors introduced by preceding layers, so the gains in Section 4 come from the scale and code updates, not from a corrected target. A target can also absorb those errors. Keeping the design matrix in (3), replace the activation part of $\mathbf{Y}$ with

$$\mathbf{Y}^\star(\mu) = (1-\mu)\mathbf{X}\mathbf{W} + \mu\widetilde{\mathbf{X}}\mathbf{W}, \qquad \mu \in [0,1], \tag{21}$$

where $\mathbf{X}$ and $\widetilde{\mathbf{X}}$ stack the inputs from full-precision and quantized preceding layers, respectively. Setting $\mu = 1$ recovers the main target.

Equivalence to QEP's corrected weights. Let $\mathbf{H} = \widetilde{\mathbf{X}}^\mathsf{T}\widetilde{\mathbf{X}} + \nu^2\mathbf{I}$, which is nonsingular for $\nu > 0$, and let $\mathbf{Y}(\mu)$ be the augmented target with activation part (21). The normal equations of $\min_{\widehat{\mathbf{W}}} \|\mathbf{Y}(\mu) - \mathbf{A}\widehat{\mathbf{W}}\|_F^2$ over real matrices give

$$\mathbf{W}_{\mathrm{corr}} = \mathbf{W} + (1-\mu)\,\mathbf{H}^{-1}\widetilde{\mathbf{X}}^\mathsf{T}(\mathbf{X} - \widetilde{\mathbf{X}})\mathbf{W}, \tag{22}$$

which has the form of QEP's propagation-corrected weights (Arai & Ichikawa, 2025) with correction strength $\alpha = 1 - \mu$ and damping ν^2; QEP's default $\alpha = 0.5$ corresponds to $\mu = 0.5$. Because $\mathbf{Y}(\mu) - \mathbf{A}\mathbf{W}_{\mathrm{corr}}$ is orthogonal to the range of $\mathbf{A}$, for every $\widehat{\mathbf{W}}$,

$$\|\mathbf{Y}(\mu) - \mathbf{A}\widehat{\mathbf{W}}\|_F^2 = \|\mathbf{A}(\mathbf{W}_{\mathrm{corr}} - \widehat{\mathbf{W}})\|_F^2 + \|\mathbf{Y}(\mu) - \mathbf{A}\mathbf{W}_{\mathrm{corr}}\|_F^2. \tag{23}$$

Refining with target (21) is therefore the main problem with $\mathbf{W}$ replaced by $\mathbf{W}_{\mathrm{corr}}$, plus a constant that Algorithm 1 carries in c_j. Theorem 1, Corollary 1, and the propositions of Appendix G need only a target that is fixed within the layer, so they hold for every $\mu \in [0,1]$. In this form, QEP chooses the target and JARQ solves it on a moving grid. All experiments in this paper use $\mu = 1$.

F Group Size and Regularization Ablations

We run two full-model ablations on Llama-2-7B and Qwen3-8B with the RTN host at W3A16 and W4A16. Both use 128 WT2 training sequences of 2,048 tokens, seed 0, and an iteration budget of $K = 3$ with early stopping. Each model is evaluated on the WT2 test set and 256 C4 validation windows. All runs share the same calibration and evaluation data; the $g = 128$, $\nu = 0.6$ states are shared by the two ablations, giving 44 distinct model states.

F.1 Group Size

We vary the group size $g \in \{64, 128, 256\}$ while fixing $\nu = 0.6$. Each group size has its own RTN Base ($K = 0$). Table 13 shows that refinement lowers both WT2 and C4 perplexity in all 12 model–bit-width–group-size configurations. The main experiments use $g = 128$, the common setting for group-wise LLM quantization used in AWQ (Lin et al., 2024).

Runtime. Table 14 compares the runtime of $g = 64$ and $g = 128$. Relative to $g = 128$, $g = 64$ gives 0.57–3.95% lower refined perplexity across the eight model–bit-width–corpus comparisons and takes 2.45–2.74 times as long.

The timer covers calibration input capture, RTN quantization, refinement, diagnostics and progress logging, and block propagation; model loading and perplexity evaluation are excluded. Each entry is one full-model measurement, with a different timing scope from the block-level measurements in Appendix A.5.

Table 13: Group-size ablation with the RTN host, $\nu = 0.6$, and $K = 3$. Entries are Base/+Ours perplexity (lower is better). Bold marks the better value in each pair (compared before rounding).

Model	W/A	Data	$g = 64$	$g = 128$	$g = 256$
Llama-2-7B	W3A16	WT2	6.4009/**6.0186**	6.6630/**6.2043**	7.1019/**6.2951**
		C4	8.1095/**7.8924**	8.4050/**8.2166**	8.9828/**8.4944**
	W4A16	WT2	5.6780/**5.5938**	5.7261/**5.6256**	5.7495/**5.6474**
		C4	7.2114/**7.1347**	7.2458/**7.1857**	7.2983/**7.2414**
Qwen3-8B	W3A16	WT2	12.3856/**10.4292**	13.4564/**10.7123**	15.7918/**11.1354**
		C4	16.0018/**14.5767**	17.5825/**15.1033**	20.3924/**15.8302**
	W4A16	WT2	9.9778/**9.8458**	10.1369/**9.9980**	10.4434/**10.0580**
		C4	13.5253/**13.4867**	13.6431/**13.6271**	13.8796/**13.7751**

Table 14: Full-model quantization and refinement time for the RTN host with $\nu = 0.6$ and $K = 3$. Increase is $100(T_{64}/T_{128} - 1)$.

Model	W/A	$g = 128$ (min)	$g = 64$ (min)	Increase
Llama-2-7B	W3A16	32.13	87.96	$+173.78\%$
	W4A16	32.06	87.04	$+171.50\%$
Qwen3-8B	W3A16	38.54	95.36	$+147.43\%$
	W4A16	39.24	96.00	$+144.64\%$

F.2 Regularization Strength

We fix $g = 128$ and vary $\nu \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ in (3); the weight-error coefficient is ν^2. Every value, including $\nu = 0$, uses $K = 3$; only the RTN Base uses $K = 0$. The conditional 10^{-4} stabilization of the scale solve (Appendix E.2) is unchanged. Table 15 reports all six strengths.

Table 15: Regularization-strength ablation with the RTN host, $g = 128$, and $K = 3$. Entries are WT2/C4 perplexity (lower is better). Bold marks the best of the six strengths for each model, bit width, and corpus (compared before rounding).

	Llama-2-7B		Qwen3-8B	
ν	W3A16	W4A16	W3A16	W4A16
Base ($K = 0$)	6.6630/8.4050	5.7261/7.2458	13.4564/17.5825	10.1369/13.6431
0.0	6.2254/8.2287	**5.6203**/7.1873	**10.5207/15.0312**	9.9718/13.5872
0.2	6.1879/8.2263	5.6242/**7.1837**	10.6558/15.1019	9.9430/13.5646
0.4	6.1944/8.2391	5.6295/7.1875	10.7394/15.1621	**9.9146/13.5575**
0.6	6.2043/8.2166	5.6256/7.1857	10.7123/15.1033	9.9980/13.6271
0.8	6.1906/8.2149	5.6340/7.1848	10.7197/15.1473	10.0092/13.6307
1.0	**6.1830/8.2041**	5.6354/7.1875	10.7658/15.2126	9.9385/13.5743

Refinement improves both perplexities over RTN at every tested strength. The best strength depends on the model and bit width: Llama-2-7B at W3A16 reaches its lowest values at $\nu = 1.0$, whereas Qwen3-8B at W3A16 and W4A16 reaches them at $\nu = 0$ and $\nu = 0.4$, respectively.

G Analysis of the Refinement Updates

This appendix proves the results stated in Section 3.3, gives two further propositions on the scale and code updates, relates the code update to GPTQ, and derives the computational cost of the refinement procedure.

G.1 JOINT SCALE FITTING

Proposition G.1 (Error gap for joint scale fitting)**.** *Fix the codes of output column j and write $f_j(\mathbf{s}) = \|\mathbf{y}_j - \mathbf{B}_j\mathbf{s}\|_2^2$. For any least-squares minimizer $\mathbf{s}_j^\star$ and any $\widetilde{\mathbf{s}}_j \in \mathbb{R}^G$,*

$$f_j(\widetilde{\mathbf{s}}_j) - f_j(\mathbf{s}_j^\star) = \|\mathbf{B}_j(\widetilde{\mathbf{s}}_j - \mathbf{s}_j^\star)\|_2^2 \geq 0. \tag{24}$$

Equality holds exactly when $\mathbf{B}_j(\widetilde{\mathbf{s}}_j - \mathbf{s}_j^\star) = \mathbf{0}$.

Proof of Proposition G.1. Let $\mathbf{r}_j^\star = \mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j^\star$ and $\boldsymbol{\delta} = \widetilde{\mathbf{s}}_j - \mathbf{s}_j^\star$. The least-squares normal equations imply $\mathbf{B}_j^\mathsf{T}\mathbf{r}_j^\star = \mathbf{0}$. Hence

$$\begin{aligned} f_j(\widetilde{\mathbf{s}}_j) &= \|\mathbf{r}_j^\star - \mathbf{B}_j\boldsymbol{\delta}\|_2^2 \\ &= \|\mathbf{r}_j^\star\|_2^2 - 2\boldsymbol{\delta}^\mathsf{T}\mathbf{B}_j^\mathsf{T}\mathbf{r}_j^\star + \|\mathbf{B}_j\boldsymbol{\delta}\|_2^2 \\ &= f_j(\mathbf{s}_j^\star) + \|\mathbf{B}_j\boldsymbol{\delta}\|_2^2. \end{aligned}$$

The final term is nonnegative and vanishes exactly under the stated condition. The argument does not require $\mathbf{B}_j$ to have full column rank. □

When do separate group fits suffice? For the main target, $\mathbf{y}_j = \mathbf{A}\mathbf{w}_j$. Write $\mathbf{c}_{i,j} = \mathbf{q}_{i,j} - z_{i,j}\mathbf{1}$ and $\mathbf{e}_{i,j} = \mathbf{w}_{i,j} - s_{i,j}\mathbf{c}_{i,j}$. Then

$$f_j(\mathbf{s}_j) = \sum_i \|\mathbf{A}_i\mathbf{e}_{i,j}\|_2^2 + 2\sum_{i<h} \mathbf{e}_{i,j}^\mathsf{T}\mathbf{A}_i^\mathsf{T}\mathbf{A}_h\mathbf{e}_{h,j}. \tag{25}$$

If $\mathbf{A}_i^\mathsf{T}\mathbf{A}_h = \mathbf{0}$ for every $i \neq h$, minimizing each group's reconstruction error separately also minimizes the full objective. In general the cross terms remain, so separate fits need not satisfy the joint normal equations. Proposition G.1 measures their resulting error gap, and Proposition G.2 below expresses it through the coupling between groups. The scale-fitting comparisons in Table 4 quantify this effect on model layers; their reported gaps are normalized by the host objective.

The independent fit in Table 4 fits each group's scale to that group's own contribution $\mathbf{A}_i\mathbf{w}_{i,j}$:

$$\widetilde{s}_{i,j} = \arg\min_{s\in\mathbb{R}} \|\mathbf{A}_i(\mathbf{w}_{i,j} - s\,\mathbf{c}_{i,j})\|_2^2 = \frac{\mathbf{b}_i^\mathsf{T}\mathbf{A}_i\mathbf{w}_{i,j}}{\|\mathbf{b}_i\|_2^2}, \qquad \mathbf{b}_i = \mathbf{A}_i\mathbf{c}_{i,j}, \tag{26}$$

where $\mathbf{b}_i$ is the ith column of $\mathbf{B}_j$. We use this definition below. Fitting each scale alone to the full target, $\mathbf{b}_i^\mathsf{T}\mathbf{y}_j/\|\mathbf{b}_i\|_2^2$, is a different estimator and is not the one reported in the table.

Proposition G.2 (Gap of independent scale fits)**.** *Let $\mathbf{y}_j = \mathbf{A}\mathbf{w}_j$, fix the codes, and assume every $\mathbf{b}_i \neq \mathbf{0}$ and $\mathbf{G}_j = \mathbf{B}_j^\mathsf{T}\mathbf{B}_j$ is nonsingular. Let $\mathbf{e}_{h,j} = \mathbf{w}_{h,j} - \widetilde{s}_{h,j}\mathbf{c}_{h,j}$, $\mathbf{N} = \mathbf{B}_j\mathbf{D}_j^{-1/2}$ with $\mathbf{D}_j = \mathrm{diag}(\mathbf{G}_j)$, and let $\mathbf{C}_j = \mathbf{N}^\mathsf{T}\mathbf{N} = \mathbf{D}_j^{-1/2}\mathbf{G}_j\mathbf{D}_j^{-1/2}$ be the normalized Gram matrix, with columns $\mathbf{n}_i = \mathbf{b}_i/\|\mathbf{b}_i\|_2$ of $\mathbf{N}$. Let $\Delta_j = f_j(\widetilde{\mathbf{s}}_j) - \min_{\mathbf{s}} f_j(\mathbf{s})$.*

(a) $\Delta_j = \widehat{\mathbf{g}}^\mathsf{T}\mathbf{C}_j^{-1}\widehat{\mathbf{g}}$*, where* $\widehat{g}_i = \sum_{h\neq i} \mathbf{n}_i^\mathsf{T}\mathbf{A}_h\mathbf{e}_{h,j}$.

(b) $\|\widehat{\mathbf{g}}\|_2^2/\lambda_{\max}(\mathbf{C}_j) \leq \Delta_j \leq \|\widehat{\mathbf{g}}\|_2^2/\lambda_{\min}(\mathbf{C}_j)$.

(c) Let $\boldsymbol{\Gamma} \in \mathbb{R}^{G\times G}$ have zero diagonal and entries $\Gamma_{ih} = \|\mathbf{U}_i^\mathsf{T}\mathbf{U}_h\|_2$ for $i \neq h$, with $\mathbf{U}_i$ from (9), and let $\gamma = \|\boldsymbol{\Gamma}\|_2$. Then $\|\mathbf{C}_j - \mathbf{I}\|_2 \leq \gamma$ and

$$\Delta_j \leq \frac{\gamma^2}{\lambda_{\min}(\mathbf{C}_j)} \sum_{h=1}^G \|\mathbf{A}_h\mathbf{e}_{h,j}\|_2^2 \leq \frac{\gamma^2}{1-\gamma} \sum_{h=1}^G \|\mathbf{A}_h\mathbf{e}_{h,j}\|_2^2 \quad \text{if } \gamma < 1. \tag{27}$$

The entry Γ_{ih} is the cosine of the smallest principal angle between the column spaces of $\mathbf{A}_i$ and $\mathbf{A}_h$; it depends on the inputs and ν but not on the codes. The sum in (27) is the first term of (25) at the independent fit. Thus the gap is zero when the group column spaces are orthogonal and grows at most quadratically with their coherence. With many correlated groups, γ can exceed one and part (c) becomes uninformative, while the exact expression in part (a) still holds.

Proof of Proposition G.2. Let $\widetilde{\mathbf{r}} = \mathbf{y}_j - \mathbf{B}_j\widetilde{\mathbf{s}}_j$. The unique minimizer is $\mathbf{s}_j^\star = \mathbf{G}_j^{-1}\mathbf{B}_j^\mathsf{T}\mathbf{y}_j$, so $\widetilde{\mathbf{s}}_j - \mathbf{s}_j^\star = -\mathbf{G}_j^{-1}\mathbf{B}_j^\mathsf{T}\widetilde{\mathbf{r}}$. By Proposition G.1 and $\mathbf{B}_j = \mathbf{N}\mathbf{D}_j^{1/2}$,

$$\Delta_j = \widetilde{\mathbf{r}}^\mathsf{T}\mathbf{B}_j\mathbf{G}_j^{-1}\mathbf{B}_j^\mathsf{T}\widetilde{\mathbf{r}} = \widetilde{\mathbf{r}}^\mathsf{T}\mathbf{N}\mathbf{C}_j^{-1}\mathbf{N}^\mathsf{T}\widetilde{\mathbf{r}}.$$

Since $\mathbf{y}_j = \sum_h \mathbf{A}_h\mathbf{w}_{h,j}$, we have $\widetilde{\mathbf{r}} = \sum_h \mathbf{A}_h\mathbf{e}_{h,j}$. The optimality condition of (26) gives $\mathbf{b}_i^\mathsf{T}\mathbf{A}_i\mathbf{e}_{i,j} = 0$, so $\mathbf{n}_i^\mathsf{T}\widetilde{\mathbf{r}} = \widehat{g}_i$. This proves (a). The matrix $\mathbf{C}_j$ is positive definite because $\mathbf{G}_j$ is, and (b) follows from the Rayleigh quotient. For (c), $\mathbf{n}_i$ is a unit vector in the range of $\mathbf{U}_i$ and $\mathbf{A}_h\mathbf{e}_{h,j}$ lies in the range of $\mathbf{U}_h$. Hence $|\mathbf{n}_i^\mathsf{T}\mathbf{A}_h\mathbf{e}_{h,j}| \le \Gamma_{ih}\|\mathbf{A}_h\mathbf{e}_{h,j}\|_2$ and $|(\mathbf{C}_j)_{ih}| = |\mathbf{n}_i^\mathsf{T}\mathbf{n}_h| \le \Gamma_{ih}$ for $i \neq h$. Let $\boldsymbol{\rho}$ have entries $\rho_h = \|\mathbf{A}_h\mathbf{e}_{h,j}\|_2$. Then $\|\widehat{\mathbf{g}}\|_2 \le \|\boldsymbol{\Gamma}\boldsymbol{\rho}\|_2 \le \gamma\|\boldsymbol{\rho}\|_2$. Since $\boldsymbol{\Gamma}$ is symmetric and entrywise nonnegative, for any $\mathbf{x}$, $|\mathbf{x}^\mathsf{T}(\mathbf{C}_j - \mathbf{I})\mathbf{x}| \le |\mathbf{x}|^\mathsf{T}\boldsymbol{\Gamma}|\mathbf{x}| \le \gamma\|\mathbf{x}\|_2^2$, where $|\mathbf{x}|$ is taken entrywise. Thus $\|\mathbf{C}_j - \mathbf{I}\|_2 \le \gamma$ and $\lambda_{\min}(\mathbf{C}_j) \ge 1 - \gamma$. Combining these bounds with (b) gives (27). □

For the other estimator, $\mathbf{b}_i^\mathsf{T}\mathbf{y}_j/\|\mathbf{b}_i\|_2^2$, the same argument gives the gap $\widehat{\mathbf{b}}^\mathsf{T}(\mathbf{C}_j - \mathbf{I})\mathbf{C}_j^{-1}(\mathbf{C}_j - \mathbf{I})\widehat{\mathbf{b}}$ with $\widehat{\mathbf{b}} = \mathbf{D}_j^{-1/2}\mathbf{B}_j^\mathsf{T}\mathbf{y}_j$, which is again zero when $\mathbf{C}_j = \mathbf{I}$.

G.2 A Single-Code Minimum with an Improving Babai Proposal

Proof of Proposition 3.2. Consider one group of two weights with a scalar scale s, zero point $z = 0$, and two-bit codes. Its reduced reconstruction problem is

$$f(s, \mathbf{q}) = \|\mathbf{y} - s\mathbf{R}\mathbf{q}\|_2^2, \qquad \mathbf{R} = \begin{bmatrix} 1 & -1 \\ 0 & 1/10 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} -1 \\ 3/10 \end{bmatrix}, \qquad \mathbf{q} \in \{0,1,2,3\}^2. \tag{28}$$

The matrix $\mathbf{R}$ is nonsingular. For a nonzero code vector, the optimal scale and the corresponding error are

$$s^\star(\mathbf{q}) = \frac{(\mathbf{R}\mathbf{q})^\mathsf{T}\mathbf{y}}{\|\mathbf{R}\mathbf{q}\|_2^2}, \qquad \phi(\mathbf{q}) = \min_s f(s, \mathbf{q}) = \|\mathbf{y}\|_2^2 - \frac{((\mathbf{R}\mathbf{q})^\mathsf{T}\mathbf{y})^2}{\|\mathbf{R}\mathbf{q}\|_2^2}. \tag{29}$$

At $\mathbf{q}_0 = (1,2)^\mathsf{T}$, these give $s_0 = 53/52$ and $\phi(\mathbf{q}_0) = 1/104$. Table 16 enumerates all six single-code neighbors, and Figure 14 shows the refitted error of every code pair. Every neighbor has larger error even after its scale is refitted optimally, since their minimum is $9/409 > 1/104$.

Table 16: All single-code neighbors of $(1,2)^\mathsf{T}$ in (28), with an exact scale refit for each candidate. The current state's error is $1/104$.

Changed coordinate	New codes $\mathbf{q}^\mathsf{T}$	Refitted error $\phi(\mathbf{q})$
First	$(0,2)$	$4/101$
	$(2,2)$	1
	$(3,2)$	$25/104$
Second	$(1,0)$	$9/100$
	$(1,1)$	1
	$(1,3)$	$9/409$

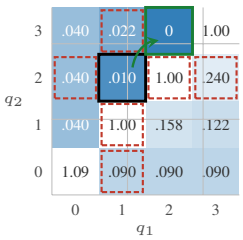


Figure 14: Error after refitting the scale for each code pair in (28) (darker is lower). Black: current codes $(1,2)$; dashed: single-code moves; green: the Babai step to $(2,3)$.

At the current scale s_0, bounded Babai back substitution gives

$$q_2^+ = \mathrm{clip}_{\{0,1,2,3\}}\left(\mathrm{round}\,\frac{156}{53}\right) = 3, \qquad q_1^+ = \mathrm{clip}_{\{0,1,2,3\}}\left(\mathrm{round}\,\frac{107}{53}\right) = 2. \tag{30}$$

Thus $\mathbf{q}^+ = (2,3)^\mathsf{T}$ changes both codes and satisfies

$$f(s_0, \mathbf{q}^+) = \frac{109}{270400} < \frac{1}{104} = f(s_0, \mathbf{q}_0). \tag{31}$$

The proposal is therefore accepted. The next scale fit gives $s^+ = 1$ and zero error because $\mathbf{y} = \mathbf{R}(2,3)^\mathsf{T}$. □

This construction is compatible with the main reconstruction target: $\mathbf{y} = \mathbf{R}\mathbf{w}$ for $\mathbf{w} = (2,3)^\mathsf{T}$. For any fixed $\nu > 0$ and a sufficiently large constant c, the matrix $c^2\mathbf{R}^\mathsf{T}\mathbf{R} - \nu^2\mathbf{I}$ is positive definite. Choosing $\widetilde{\mathbf{X}}$ with $\widetilde{\mathbf{X}}^\mathsf{T}\widetilde{\mathbf{X}} = c^2\mathbf{R}^\mathsf{T}\mathbf{R} - \nu^2\mathbf{I}$ makes the augmented objective in (2) equal to $c^2 f(s, \mathbf{q})$, so the scale fits, Babai codes, and strict error comparisons are unchanged.

G.3 Refitting the Grid of a Fixed-Grid Optimum

Proof of Proposition 3.1. Take one group of two weights with zero point $z = 0$, two-bit codes $\mathbb{B} = \{0,1,2,3\}$, and

$$\mathbf{R} = \begin{bmatrix} 1 & \rho \\ 0 & \sqrt{1-\rho^2} \end{bmatrix}, \qquad \rho = \tfrac{18}{25}, \qquad \mathbf{y} = \mathbf{R}\mathbf{w}, \qquad \mathbf{w} = \left(\tfrac{21}{50}, \tfrac{31}{20}\right)^\mathsf{T}. \tag{32}$$

Then $f(s, \mathbf{q}) = \|\mathbf{y} - s\mathbf{R}\mathbf{q}\|_2^2 = (\mathbf{w} - s\mathbf{q})^\mathsf{T}\mathbf{M}(\mathbf{w} - s\mathbf{q})$ with $\mathbf{M} = \mathbf{R}^\mathsf{T}\mathbf{R} = \left[\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right]$, so the inputs of the two weights have correlation 0.72.

Host state. Let $s_0 = 6/5$ and $\mathbf{q}_0 = (1,1)^\mathsf{T}$, so $f(s_0, \mathbf{q}_0) = 16889/50000 \approx 0.338$. Evaluating all 16 codes at s_0 gives the next smallest value $f(s_0, (0,2)^\mathsf{T}) = 19241/50000 \approx 0.385$; hence $\mathbf{q}_0$ minimizes $f(s_0, \cdot)$ over $\mathbb{B}^2$.

Scale refit. For $\mathbf{q}_0 = (1,1)^\mathsf{T}$, the least-squares scale is $s_1 = (\mathbf{q}_0^\mathsf{T}\mathbf{M}\mathbf{w})/(\mathbf{q}_0^\mathsf{T}\mathbf{M}\mathbf{q}_0) = (w_1 + w_2)/2 = 197/200$, and $f(s_1, \mathbf{q}_0) = 89383/500000 \approx 0.179$.

Babai proposal at s_1. Back substitution on $\mathbf{R}\mathbf{q} \approx \mathbf{y}/s_1$ gives the continuous values $w_2/s_1 = 310/197 \approx 1.574$, so $q_2^+ = 2$, and then $w_1/s_1 + \rho(w_2/s_1 - 2) = 588/4925 \approx 0.119$, so $q_1^+ = 0$; neither value needs clipping. The proposal $\mathbf{q}^+ = (0,2)^\mathsf{T}$ changes both codes and has $f(s_1, \mathbf{q}^+) = 3087/31250 \approx 0.099 < f(s_1, \mathbf{q}_0)$, so it is accepted. □

At s_0 the same back substitution returns $\mathbf{q}_0$ itself ($w_2/s_0 = 31/24$ and $14/25$ round to 1 and 1), in line with $\mathbf{q}_0$ being optimal on the host grid. One more scale refit gives $s_2 = 4631/5000$ and $f(s_2, \mathbf{q}^+) = 132741/1562500 \approx 0.085$, a 75% reduction from the host state; the Babai proposal at s_2 is $\mathbf{q}^+$ again, so the state is scale-optimal and Babai-stable. The scaling argument at the end of Appendix G.2 realizes this instance with the augmented design of (2) for any fixed $\nu > 0$.

G.4 Babai Proposals versus Coordinatewise Rounding

Fix an output column j, a group i, and write $s = s_{i,j} \neq 0$, $z = z_{i,j}$, and $\mathbf{t} = \mathbf{y}_j^{(i,k)}$ for the conditional target in (7). With $\mathbf{v} = \mathbf{U}_i^\mathsf{T}\mathbf{t}/s$ as in (10) and $\eta = \|(\mathbf{I} - \mathbf{U}_i\mathbf{U}_i^\mathsf{T})\mathbf{t}\|_2^2$, the group error in (8) is

$$E(\mathbf{q}) = \|\mathbf{t} - s\mathbf{A}_i(\mathbf{q} - z\mathbf{1})\|_2^2 = s^2\|\mathbf{v} - \mathbf{R}_i(\mathbf{q} - z\mathbf{1})\|_2^2 + \eta. \tag{33}$$

Its unconstrained real minimizer is $\boldsymbol{\xi} = z\mathbf{1} + \mathbf{R}_i^{-1}\mathbf{v}$. Let $\mathbf{q}^\mathrm{B}$ be the Babai proposal (10) and $\mathbf{q}^\mathrm{R} = \mathrm{round}(\boldsymbol{\xi})$ the coordinatewise rounding of $\boldsymbol{\xi}$.

Proposition G.3 (Babai versus coordinatewise rounding)**.** *(a) If no clipping is active in (10), then every entry of $\mathbf{v} - \mathbf{R}_i(\mathbf{q}^\mathrm{B} - z\mathbf{1})$ has magnitude at most $|(\mathbf{R}_i)_{\ell\ell}|/2$, and*

$$E(\mathbf{q}^\mathrm{B}) \leq \eta + \frac{s^2}{4}\sum_{\ell=1}^{d_i}(\mathbf{R}_i)_{\ell\ell}^2.$$

(b) Ignore the code bounds and suppose the fractional parts of $\boldsymbol{\xi}$ are independent and uniform on $[0, 1)$. Then

$$\mathbb{E}\, E(\mathbf{q}^{\mathrm{B}}) = \eta + \frac{s^2}{12}\sum_{\ell}(\mathbf{R}_i)_{\ell\ell}^2, \qquad \mathbb{E}\, E(\mathbf{q}^{\mathrm{R}}) = \eta + \frac{s^2}{12}\|\mathbf{R}_i\|_F^2,$$

and the difference is $\frac{s^2}{12}\sum_{\ell<h}(\mathbf{R}_i)_{\ell h}^2 = \frac{s^2}{12}\big(\|\mathbf{A}_i\|_F^2 - \sum_{\ell}(\mathbf{R}_i)_{\ell\ell}^2\big) \geq 0$, with equality if and only if $\mathbf{A}_i$ has orthogonal columns.

Part (a) is deterministic and is the classical nearest-plane bound (Babai, 1986); Chen et al. (2026) note the same bound for GPTQ without clipping. Part (b) is exact when $\mathbf{v}$ is uniformly distributed modulo the lattice $\mathbf{R}_i\mathbb{Z}^{d_i}$ and otherwise follows the standard uniform model of rounding error; it compares expected errors.

Proof of Proposition G.3. Write $\mathbf{R} = \mathbf{R}_i$, $\mathbf{c} = \mathbf{q}^{\mathrm{B}} - z\mathbf{1}$, and let $x_\ell = (v_\ell - \sum_{h>\ell} R_{\ell h}c_h)/R_{\ell\ell}$. Since z is an integer and no clipping is active, $c_\ell = \mathrm{round}(x_\ell)$. The ℓth entry of $\mathbf{v} - \mathbf{R}\mathbf{c}$ is $v_\ell - \sum_{h\geq\ell} R_{\ell h}c_h = R_{\ell\ell}(x_\ell - c_\ell)$, and $|x_\ell - c_\ell| \leq 1/2$. Summing squares and using (33) gives (a).

For (b), write $\boldsymbol{\xi} = \mathbf{m} + \mathbf{u}$ with $\mathbf{m}$ integer and $\mathbf{u}$ uniform on $[0, 1)^{d_i}$. Since $v_\ell = \sum_{h\geq\ell} R_{\ell h}(\xi_h - z)$,

$$x_\ell = u_\ell + (m_\ell - z) + \sum_{h>\ell}\frac{R_{\ell h}}{R_{\ell\ell}}(\xi_h - z - c_h),$$

where the last sum depends only on u_h for $h > \ell$. Conditional on these, x_ℓ is uniform modulo one, so $\delta_\ell = x_\ell - c_\ell$ is uniform on $[-1/2, 1/2]$ and independent of δ_h for $h > \ell$. Hence $\mathbb{E}\,\delta_\ell^2 = 1/12$ and $\mathbb{E}\|\mathbf{v} - \mathbf{R}\mathbf{c}\|_2^2 = \sum_\ell R_{\ell\ell}^2/12$. For rounding, $\mathbf{v} - \mathbf{R}(\mathbf{q}^{\mathrm{R}} - z\mathbf{1}) = \mathbf{R}(\boldsymbol{\xi} - \mathbf{q}^{\mathrm{R}})$, whose entries $\xi_h - q_h^{\mathrm{R}}$ are independent, uniform on $[-1/2, 1/2]$, and have mean zero. Thus the expectation is $\|\mathbf{R}\|_F^2/12$. The difference is the strictly upper-triangular energy of $\mathbf{R}$. Because $\mathbf{U}_i$ has orthonormal columns, $\|\mathbf{R}\|_F = \|\mathbf{A}_i\|_F$. The difference vanishes if and only if $\mathbf{R}$ is diagonal. For nonsingular upper-triangular $\mathbf{R}$, this holds if and only if $\mathbf{A}_i^{\mathsf{T}}\mathbf{A}_i = \mathbf{R}^{\mathsf{T}}\mathbf{R}$ is diagonal, by induction on the rows of $\mathbf{R}$. □

Proposition 3.2 shows that coordinated code changes can help; Proposition G.3 gives the expected size of the advantage over coordinatewise rounding: it is proportional to the within-group off-diagonal energy of $\mathbf{R}_i$, which is produced by correlated input coordinates. RTN assigns each code by rounding its own weight, while GPTQ codes already come from a nearest-plane recursion on the full Hessian lattice (Chen et al., 2026). The higher Babai improvement rate for the RTN host than for the GPTQ host in Table 4 (40.14% versus 13.53%) is consistent with this view.

G.5 Monotone Descent and Finite Termination

The objective (4) is a sum of per-column terms $\mathcal{F}_j$, and column j's updates change only $(\mathbf{s}_j, \mathbf{q}_j)$. It therefore suffices to consider one column. We write $(\mathbf{s}^{(k)}, \mathbf{q}^{(k)})$ for its state after iteration k, with $(\mathbf{s}^{(0)}, \mathbf{q}^{(0)})$ from the host, and drop the index j.

Proof of Theorem 1. Scale update. Without the ridge term, the Cholesky solve returns the minimizer of $\mathcal{F}(\cdot, \mathbf{q}^{(k-1)})$, which is unique because $\lambda_j = 0$ means $\mathbf{G}_j$ passed the Cholesky test and is positive definite. Hence

$$\mathcal{F}(\mathbf{s}^{(k)}, \mathbf{q}^{(k-1)}) = \phi(\mathbf{q}^{(k-1)}) \leq \mathcal{F}(\mathbf{s}^{(k-1)}, \mathbf{q}^{(k-1)}).$$

Code update. With the scales and all other groups fixed, $\mathcal{F}$ as a function of $\mathbf{q}_i$ equals the conditional residual in (8), since (7) subtracts exactly the current contributions of the other groups. The QR reduction in (17) and the projection onto $\mathbf{U}_i$ in (33) change this residual only by constants independent of $\mathbf{q}_i$. The acceptance test therefore accepts a proposal exactly when it strictly lowers $\mathcal{F}$. Rejected proposals and skipped groups with $|s_i| \leq \epsilon_{\mathrm{fp}}$ leave the state unchanged. This proves (i). For $|s_i| > \epsilon_{\mathrm{fp}}$ the proposal is computed at the current scale.

Profiled objective. Codes change only through accepted proposals. If iteration k accepts one, then $\phi(\mathbf{q}^{(k)}) \leq \mathcal{F}(\mathbf{s}^{(k)}, \mathbf{q}^{(k)}) < \mathcal{F}(\mathbf{s}^{(k)}, \mathbf{q}^{(k-1)}) = \phi(\mathbf{q}^{(k-1)})$, which proves (ii).

Termination. By (ii), the code vectors after code-changing iterations have strictly decreasing values of ϕ, so none repeats. The set $\mathbb{B}^{d_{\text{in}}}$ is finite, so at most $|\mathbb{B}|^{d_{\text{in}}} - 1$ iterations change a code. Consider the first iteration k that accepts no proposal. Throughout its sweep the state is $(\mathbf{s}^{(k)}, \mathbf{q}^{(k-1)})$, so every proposal was computed at this state and rejected, and $\mathbf{s}^{(k)}$ minimizes $\mathcal{F}(\cdot, \mathbf{q}^{(k-1)})$. The state is thus scale-optimal and Babai-stable. The next scale update receives the same codes and returns the same unique minimizer, so the same proposals are rejected again. This proves (iii). □

Early stopping. Under the exact-solve assumption, $L_j^{(k)}$ in (18) includes the constant in (17) and therefore equals $\phi(\mathbf{q}^{(k-1)})$, so it is nonincreasing in k. If iteration $k-1$ changes no code, then $L_j^{(k)} = L_j^{(k-1)}$ and the rule stops the column. Hence a column stops at the latest one iteration after reaching the fixed point in Theorem 1. It may stop earlier, when the decrease falls below τ; that state satisfies (i) but need not be Babai-stable.

Proof of Corollary 1. For $K = 0$ the host state is returned. Otherwise, the host scales are feasible for the first scale update, so $\mathcal{F}_j(\mathbf{s}_j^{(1)}, \mathbf{q}_j^{(0)}) \le \mathcal{F}_j(\mathbf{s}_j^{(0)}, \mathbf{q}_j^{(0)})$. By Theorem 1(i), later half-steps do not increase $\mathcal{F}_j$, and early stopping returns one of these states. Summing over columns gives the result. □

Both states are evaluated with the refined model's inputs $\widetilde{\mathbf{X}}$ and the same target. Since $\widetilde{\mathbf{X}}$ depends on the preceding layers, the guarantee is per layer; FP32 scale solves and the final cast to the model's weight dtype can perturb it by rounding.

Remark 1 (Making the guarantee unconditional)**.** *When the ridge branch is triggered, the scales $(\mathbf{G}_j + \lambda\mathbf{I})^{-1}\mathbf{B}_j^{\mathsf{T}}\mathbf{y}_j$ need not lower $\mathcal{F}_j$ relative to the previous scales, so Theorem 1(i) can fail at that update. Suppose the ridge solution were kept only when it does not increase $\mathcal{F}_j$, and the previous scales retained otherwise. Then (i) and Corollary 1 would hold without the exact-solve assumption. With $\tau > 0$, $L_j^{(k)}$ would be nonincreasing and bounded below, so the absolute test in (18) would stop each column within $L_j^{(1)}/\tau + 2$ iterations. Part (iii) would still require exact scale solves.*

G.6 Quality of Babai-Stable Codes

Proposition G.4 (Error at a Babai-stable point)**.** *Let $(\mathbf{s}_j, \mathbf{q}_j)$ be scale-optimal and Babai-stable for column j, let i be a group with $|s_{i,j}| > \epsilon_{\text{fp}}$, and let $\mathbf{r}$ be its conditional target, the residual after subtracting the other groups' contributions. If the bounded Babai proposal for this group needs no clipping, then*

$$\|\mathbf{r} - s_{i,j}\mathbf{A}_i(\mathbf{q}_{i,j} - z_{i,j}\mathbf{1}_{d_i})\|_2^2 \le \min_{\mathbf{u}\in\mathbb{R}^{d_i}} \|\mathbf{r} - \mathbf{A}_i\mathbf{u}\|_2^2 + \frac{s_{i,j}^2}{4}\sum_{\ell=1}^{d_i}(\mathbf{R}_i)_{\ell\ell}^2. \tag{34}$$

Proof. Write $s = s_{i,j}$, $z = z_{i,j}$, and $\mathbf{A}_i = \mathbf{U}_i\mathbf{R}_i$. For any codes $\mathbf{q}$, orthogonality gives

$$\|\mathbf{r} - s\mathbf{A}_i(\mathbf{q} - z\mathbf{1})\|_2^2 = \|(\mathbf{I} - \mathbf{U}_i\mathbf{U}_i^{\mathsf{T}})\mathbf{r}\|_2^2 + s^2\|\mathbf{v} - \mathbf{R}_i(\mathbf{q} - z\mathbf{1})\|_2^2, \qquad \mathbf{v} = \mathbf{U}_i^{\mathsf{T}}\mathbf{r}/s.$$

Since $\mathbf{R}_i$ is nonsingular, the first term equals $\min_{\mathbf{u}} \|\mathbf{r} - \mathbf{A}_i\mathbf{u}\|_2^2$. For the unclipped proposal $\mathbf{q}^+$, let $t_\ell = \big(v_\ell - \sum_{h>\ell}(\mathbf{R}_i)_{\ell h}(q_h^+ - z)\big)/(\mathbf{R}_i)_{\ell\ell}$, so that $q_\ell^+ = \text{round}(z + t_\ell)$ by (10) and $|t_\ell - (q_\ell^+ - z)| \le 1/2$. The ℓth entry of $\mathbf{v} - \mathbf{R}_i(\mathbf{q}^+ - z\mathbf{1})$ is $(\mathbf{R}_i)_{\ell\ell}\big(t_\ell - (q_\ell^+ - z)\big)$, so its squared norm is at most $\sum_\ell (\mathbf{R}_i)_{\ell\ell}^2/4$. Hence $\mathbf{q}^+$ satisfies (34). With the other groups and the scales fixed, $\mathcal{F}_j$ differs from the group's conditional error by a constant, and Babai stability means that $\mathbf{q}^+$ does not strictly lower $\mathcal{F}_j$. The current codes therefore have conditional error no larger than that of $\mathbf{q}^+$. □

The bound is deterministic and needs no distributional assumption. It compares the codes with the best real-valued weights for the group on the conditional target and shrinks with the grid spacing $s_{i,j}$.

G.7 Relation to GPTQ as Babai's Algorithm

Remark 2 (Relation to GPTQ). *Chen et al. (2026) show that GPTQ, executed from the last to the first dimension, is Babai's nearest-plane algorithm for a closest-vector problem on a lattice defined by the Hessian, on a fixed quantization grid. The code update (10) applies the same recursion to a single group: the lattice is generated by $s_{i,j}\mathbf{R}_i$ at the current scale, and the target is the conditional residual (7), from which the latest contributions of all other groups are subtracted. With $K = 0$,* JARQ *returns the host state, whether or not the host is GPTQ. The differences from GPTQ are as follows. (i) The grid changes: all scales of a column are refit jointly given the codes, and the codes are decoded again on the new grid. (ii) Each proposal is accepted only if it lowers $\mathcal{F}_j$, which gives Theorem 1; GPTQ applies each rounding decision once. (iii) Decoding is repeated across groups and iterations, so a group's codes respond to later changes in other groups. When the host is GPTQ, its codes are already nearest-plane points of a closely related lattice, which is consistent with the lower Babai improvement rate for GPTQ in Table 4 (Appendix G.4). The lattices differ in the calibration inputs, the regularization, and the grouping, so this is not an identity.*

G.8 Computational Complexity

Let $d = d_{\text{in}}$, $p = d_{\text{out}}$, and assume equal-sized groups of g weights, so $G = d/g$. We count dense arithmetic for QR reduction followed by cached group factors and incremental residual updates. The counts cover one linear layer's refinement; host quantization, activation collection, and model evaluation are separate costs.

Shared preprocessing. Thin QR of the $(n+d)\times d$ augmented matrix costs $O((n+d)d^2)$. Forming $\overline{\mathbf{Y}} = \mathbf{U}^\mathsf{T}\mathbf{Y}$ as in Algorithm 1, equivalently solving $\mathbf{R}^\mathsf{T}\overline{\mathbf{Y}} = \mathbf{A}^\mathsf{T}\mathbf{Y}$, costs $O((n+d)dp + d^2p)$. For the main target, $\mathbf{Y} = \mathbf{AW}$, so the identity $\overline{\mathbf{Y}} = \mathbf{RW}$ permits a target-projection shortcut costing $O(d^2p)$ instead. The orthogonal residual in (17) is zero in exact arithmetic for this target. QR factorization of each reduced $d \times g$ group matrix costs $O(dg^2)$, or $O(d^2g)$ across all groups. These factors are reused across output columns and iterations.

One iteration for one output column. Constructing $\mathbf{B}_j \in \mathbb{R}^{d\times G}$ costs $O(d^2)$. Forming its Gram matrix costs $O(dG^2)$, and the scale solve costs $O(G^3)$, including the spectral check and Cholesky factorization. The right-hand side costs $O(dG)$. For code updates, cache the full residual and update it whenever a group proposal is accepted. Projection onto a group's QR basis and updating its contribution each cost $O(dg)$; bounded Babai back substitution costs $O(g^2)$. A complete group sweep therefore costs $O(d^2 + dg)$, dominated by the residual updates rather than the back substitution.

Total time and workspace. For an iteration budget $K \geq 1$, the resulting bound is

$$T_{\text{ref}} = O\big((n+d)(d^2+dp) + Kp\left[d^2 + dG^2 + G^3\right]\big). \tag{35}$$

Here $g \leq d$ and $K \geq 1$ allow the d^2p, d^2g, and $Kpdg$ terms to be absorbed; early stopping reduces the actual work. Using the main-target shortcut above removes the $(n+d)dp$ term, giving $O((n+d)d^2 + Kp[d^2 + dG^2 + G^3])$. The shared reduced matrix and group factors occupy $O(d^2)$ storage. Processing one output column at a time requires an additional $O(dG + G^2 + d)$ workspace for the scale system and residuals, excluding the stored model, host states, calibration inputs, and outputs. Batching columns increases this workspace with the batch size.

Since $G = d/g$, halving the group size multiplies the dG^2 term by four and the G^3 term by eight, while reducing the dg back-substitution term. This explains why smaller groups can increase refinement time, as observed in Table 14.