The Geometry of LLM Quantization: GPTQ as Babai's Nearest Plane Algorithm

Jiale Chen[†], **Yalda Shabanzadeh**[†], **Torsten Hoefler**[‡], **Dan Alistarh**[†]
[†]*Institute of Science and Technology Austria (ISTA)*, [‡]*ETH Zürich*
JIALE.CHEN@IST.AC.AT

Editors: List of editors' names

Abstract

Quantizing the weights of large language models (LLMs) from 16-bit to lower bitwidth is the de facto approach to deploy massive transformers onto more affordable accelerators. GPTQ emerged as one of the standard methods for one-shot post-training quantization at LLM scale. Yet, its inner workings are described as a sequence of ad-hoc algebraic updates that obscure any geometric meaning or worst-case guarantees. In this work, we show that, when executed back-to-front (from the last to first dimension) for a linear layer, GPTQ is mathematically identical to Babai's nearest plane algorithm for the classical closest vector problem (CVP) on a lattice defined by the Hessian matrix of the layer's inputs. This equivalence is based on a sophisticated mathematical argument, and has two analytical consequences: (i) the GPTQ error propagation step gains an intuitive geometric interpretation; (ii) GPTQ inherits the error upper bound of Babai's algorithm under the no-clipping condition. Taken together, these results place GPTQ on a firm theoretical footing and open the door to importing decades of progress in lattice algorithms towards the design of future quantization algorithms for billion-parameter models.

Keywords: LLM, Quantization, Lattice Algorithm, Closest Vector Problem

1. Introduction

Post-training quantization has emerged as the default practical solution for reducing the footprint of GPT-scale models, without retraining. Among a growing family of methods, GPTQ (Frantar et al., 2023) was the first to push one-shot quantization down to the 4-bit regime, while retaining near-baseline accuracies. Despite its (relative) age, the method is still very popular, and still yields state-of-the-art results in some regimes (Kurtic et al., 2024).

The GPTQ algorithm was originally presented as a sequence of algebraic operations, applied greedily. This paper is the first[1] to provide a geometric interpretation for GPTQ, which implies new error bounds. Our main results are (i) the GPTQ optimization problem, i.e. linear-layer quantization with the L2 objective on the output, is equivalent to the closest vector problem (CVP) w.r.t. L2 distance; (ii) GPTQ executed from the last to first dimension is the same as Babai's nearest plane algorithm on the basis of the factorized Hessian matrix, without LLL basis reduction, and this finding holds independently of whether weight clipping is used; and (iii) the worst-case layer-wise error in thr no-clipping setting is bound tightly by the trace of the diagonal matrix of the LDL decomposition of the Hessian matrix.

This lattice perspective explains GPTQ's advantage: viewing activations as a lattice basis, each GPTQ step is the orthogonal projection onto the nearest hyperplane. Conceptually, the work sits squarely within the workshop scope by tying representational geometry and symmetry-structured (lattice) algorithms to practical quantization of neural networks.

[1] The concurrent work of Birnick (2025) appeared on arXiv later than our preprint.

2. Related Work

Second-order compression (pruning and quantization). The idea of using Hessian information to guide parameter removal dates back to Optimal Brain Damage (LeCun et al., 1989) and Optimal Brain Surgeon (OBS) (Hassibi et al., 1993). Optimal Brain Compression (OBC) (Frantar and Alistarh, 2022) generalizes OBS to the post-training setting and unifies structured pruning and quantization (also called Optimal Brain Quantizer, OBQ) under a single exact solver. GPTQ (Frantar et al., 2023) inherits OBQ's error propagation method but applies it in a fixed order, so that the inverse Hessian can be shared and only needs to be computed once. GPTQ only has cubic computational complexity in the column/row dimension, making it suitable for LLMs. QuIP (Chee et al., 2023) proves an error guarantee for GPTQ and proposes the LDLQ method as an equivalent variant of GPTQ.

Lattices, CVP algorithms, and hardness. The closest vector problem (CVP) is NP-complete to approximate within any constant factor under polynomial-time reductions (van Emde Boas, 1981; Micciancio and Goldwasser, 2002; Dinur et al., 2003), motivating decades of approximation algorithms. Babai's nearest plane heuristic (Babai, 1986) delivers a solution in polynomial time and, when preceded by LLL basis reduction (Lenstra et al., 1982), enjoys a $2^{O(n)}$ approximation.

3. Preliminaries and Notations

3.1. Linear-Layer Quantization Problem

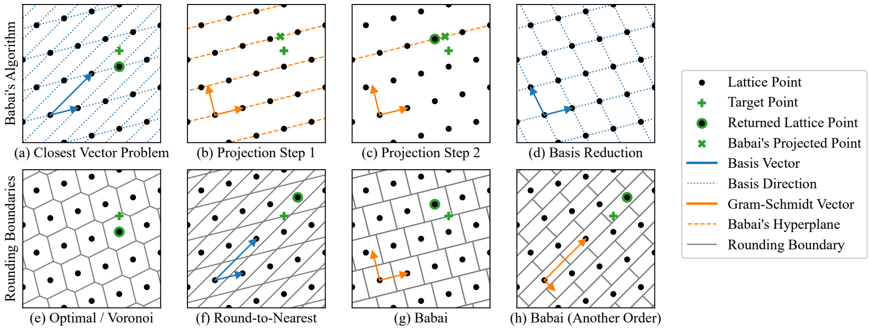
Problem. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n \times c}$ be the sampled calibration input data of batch size n and input dimension c with $\boldsymbol{x}_i \in \mathbb{R}^c$ and $n \geq c = \operatorname{rank}(\boldsymbol{X})$. Let $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_r] \in \mathbb{R}^{c \times r}$ be the linear layer weights of input dimension c and output dimension r with $\boldsymbol{w}_i \in \mathbb{R}^c$. Let $\boldsymbol{S} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_r] \in \mathbb{R}^{c \times r}$ be the non-zero quantization scales with $\boldsymbol{s}_i \in \mathbb{R}^c_{\neq 0}$. Here we consider a general case that applies to any grouping pattern: each weight element[2] $\boldsymbol{w}_i[j]$ has its own scaling factor $\boldsymbol{s}_i[j]$. Assume $\boldsymbol{S}$ is statically computed using methods like AbsMax or MSE before any weight updates. Let $\mathbb{Z}^\dagger \subseteq \mathbb{Z}$ be the quantization grid (representable integers). In the clipping cases, e.g., for INT4 format, $\mathbb{Z}^\dagger = \{-8, \ldots, -1, 0, 1, \ldots, 7\}$. In the no-clipping cases, $\mathbb{Z}^\dagger = \mathbb{Z}$, which allows any integers as the quantization results. Let $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_r] \in {\mathbb{Z}^\dagger}^{c \times r}$ be the (unknown) quantized integers with $\boldsymbol{z}_i \in {\mathbb{Z}^\dagger}^c$. Denote $\boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_r] \in \mathbb{R}^{c \times r}$ as the dequantized weights with $\boldsymbol{q}_i = \operatorname{diag}(\boldsymbol{s}_i)\, \boldsymbol{z}_i \in \mathbb{R}^c$. The goal is to minimize the L2 error on the layer output $\boldsymbol{X}\boldsymbol{W} \in \mathbb{R}^{n \times r}$: $\|\boldsymbol{X}\boldsymbol{Q} - \boldsymbol{X}\boldsymbol{W}\|_{\mathrm{F}}^2 = \sum_{i=1}^r \|\boldsymbol{X} \operatorname{diag}(\boldsymbol{s}_i)\, \boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2$, i.e, finding $\operatorname{argmin}_{\boldsymbol{z}_i \in {\mathbb{Z}^\dagger}^c} \|\boldsymbol{X} \operatorname{diag}(\boldsymbol{s}_i)\, \boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2$ for all $1 \leq i \leq r$.

GPTQ algorithm. For each weight vector $\boldsymbol{w}_i$, the algorithm sequentially picks one weight $\boldsymbol{w}_i[j]$ at a time, quantizes it via round-to-nearest as $\boldsymbol{q}_i[j]$, and then optimally updates the remaining unquantized weights via an OBQ update step $\boldsymbol{w}_i[j'] \leftarrow \boldsymbol{w}_i[j'] + \Delta\boldsymbol{w}_i[j']$ for all j' in the unquantized indices set J with $\Delta\boldsymbol{w}_i[j'] \leftarrow \frac{\left(\boldsymbol{X}[:,J]^\top \boldsymbol{X}[:,J]\right)^{-1}[j',j]}{\left(\boldsymbol{X}[:,J]^\top \boldsymbol{X}[:,J]\right)^{-1}[j,j]} \left(\boldsymbol{q}_i[j] - \boldsymbol{w}_i[j]\right)$. Algorithm 1 in Section C.1 is the pseudocode of GPTQ.

[2] We use Python-style indexing inside square brackets to select elements and sub-matrices.

3.2. The Closest Vector Problem (CVP)

Problem. Let $\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_c] \in \mathbb{R}^{n \times c}$ be a set of c basis vectors of dimension n with $\boldsymbol{b}_j \in \mathbb{R}^n$ and $n \geq c = \operatorname{rank}(\boldsymbol{B})$. Let $\boldsymbol{y} \in \mathbb{R}^n$ be an external target vector to approximate. Let $\boldsymbol{z} \in \mathbb{Z}^c$ be the (unknown) integer vector representing the basis combinations of the lattice vector. The goal is to find the vector on the lattice defined by the basis $\boldsymbol{B}$ that is the closest to the external vector $\boldsymbol{y}$, i.e., finding $\operatorname{argmin}_{\boldsymbol{z} \in \mathbb{Z}^c} \|\boldsymbol{B}\boldsymbol{z} - \boldsymbol{y}\|^2$. A visualization of a two-dimensional CVP is shown in Figure 1 (a).



(a) Closest Vector Problem (b) Projection Step 1 (c) Projection Step 2 (d) Basis Reduction

(e) Optimal / Voronoi (f) Round-to-Nearest (g) Babai (h) Babai (Another Order)

Figure 1: **Upper row:** (a) CVP in a two-dimensional lattice; (b-c) The projection steps in Babai's nearest plane algorithm without basis reduction; (d) Basis reduction can find a shorter, more orthogonal basis that can potentially improve the results. **Lower row:** rounding boundaries of (e) optimal rounding or Voronoi cells; (f) round-to-nearest (RTN); (g) Babai's nearest plane algorithm without basis reduction; (h) Babai's algorithm without basis reduction under reversed basis ordering.

Babai's nearest plane algorithm. Algorithm 2 in Section C.1 is the pseudocode of Babai's nearest plane algorithm to solve CVP, which iteratively projects a target vector onto the nearest hyperplane and rounds the coefficient. Figure 1 (b-c) visualize the projection steps, and Figure 1 (d) visualizes the basis reduction step that can be executed before the projections. Figure 1 also shows the rounding boundaries of the optimal (e), round-to-nearest (RTN) (f), and Babai's algorithm without basis reduction (g-h). Compared to RTN, Babai's algorithm generates rectangular partitions and thus has a smaller worst-case error. Babai's algorithm has been proven to have an error bound.

4. Main Results

4.1. Equivalence of Quantization and CVP

The L2 objective quantization problem $\operatorname{argmin}_{\boldsymbol{z}_i \in \mathbb{Z}^{\dagger c}} \|\boldsymbol{X} \operatorname{diag}(\boldsymbol{s}_i) \boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2$ and a CVP with the L2 distance $\operatorname{argmin}_{\boldsymbol{z} \in \mathbb{Z}^c} \|\boldsymbol{B}\boldsymbol{z} - \boldsymbol{y}\|^2$ share the same solution ($\boldsymbol{z} = \boldsymbol{z}_i$) whenever the structural conditions $\boldsymbol{B} = \boldsymbol{X} \operatorname{diag}(\boldsymbol{s}_i)$ and $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}_i$ hold and the solution domain matches. To ensure the solution domain matches, we can either disable the clipping in the quantization setup (setting $\mathbb{Z}^\dagger = \mathbb{Z}$) or enable the clipping in the CVP setup (making $\boldsymbol{z} \in {\mathbb{Z}^{\dagger}}^{c}$).

4.2. GPTQ and Babai's Algorithm

Section C.2 describes the quantization procedures using Babai's algorithm. By default, GPTQ (Algorithm 1) runs from the first to the last dimension ($j \leftarrow 1$ to c) while Babai's algorithm (Algorithm 3) runs from the last to the first dimension ($j \leftarrow c$ to 1).

Geometric interpretation. Section B shows that each intermediate weight vector produced by GPTQ can be viewed as Babai's residual in activation space: if we regard the floating-point weight vector as a target point and the activations as the lattice basis, GPTQ performs an *orthogonal walk* through a nested sequence of affine subspaces. At step j it projects the current residual onto the hyperplane orthogonal to the j-th Gram-Schmidt vector, while the familiar error propagation update is exactly this orthogonal projection.

Theorem 1 *GPTQ and Babai's algorithm have the same results if we align the dimensional order of these two algorithms, e.g., running GPTQ from the last to the first dimension.*

This is our main technical contribution. The full proof is presented in Section F.

4.3. Quantization Error Bound

Having established the correspondence between GPTQ and Babai's nearest plane algorithm, we can now import Babai's approximation guarantee to obtain an upper bound on the layer-wise quantization error in the no-clipping cases.

Theorem 2 *Assume there is no clipping ($\mathbb{Z}^\dagger = \mathbb{Z}$). Let $\boldsymbol{D}$ be the diagonal matrix in the LDL decomposition of the Hessian matrix $\boldsymbol{X}^\top \boldsymbol{X}$. For every output channel i ($1 \leq i \leq r$) produced by Babai's algorithm, or equivalently GPTQ executed back-to-front, the quantization error has a tight error upper bound: $\|\boldsymbol{X} \operatorname{diag}(\boldsymbol{s}_i)\, \boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2 \leq \frac{1}{4}\boldsymbol{s}_i^\top \boldsymbol{D}\boldsymbol{s}_i$.*

The full proof is presented in Section D. We also empirically evaluated GPTQ in the no-clipping scenarios. The results and analysis are presented in Section A.

The quadratic form on the right-hand side of the error bound in Theorem 2 is sensitive to the pivot order of the LDL decomposition of the Hessian matrix, aka the quantization order. GPTQ introduces the so-called "act-order", the descending order of the Hessian diagonal. This translates to the ascending order of the Hessian diagonal when applied to Babai's algorithm. Section E further discusses on this topic.

5. Conclusion

We have shown that GPTQ, when executed back-to-front, is mathematically identical to Babai's nearest plane algorithm applied to the lattice defined by a layer's Hessian without basis reduction. Looking ahead, extending the analysis to clipped grids and exploring scale-aware basis reductions are the immediate next steps. More broadly, the lattice perspective opens a two-way channel: decades of CVP heuristics can refine practical quantizers, while the behavior of massive neural networks may, in turn, inspire new questions for lattice theory.

References

László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, March 1986. ISSN 1439-6912. doi: 10.1007/BF02579403. URL https://doi.org/10.1007/BF02579403.

Johann Birnick. The lattice geometry of neural network quantization – a short equivalence proof of gptq and babai's algorithm, 2025. URL https://arxiv.org/abs/2508.01077.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 4396–4429. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/0df38cd13520747e1e64e5b123a78ef8-Paper-Conference.pdf.

I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, apr 2003. ISSN 1439-6912. doi: 10.1007/s00493-003-0019-y. URL https://doi.org/10.1007/s00493-003-0019-y.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 4475–4488. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/1caf09c9f4e6b0150b06a07e77f2710c-Paper-Conference.pdf.

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

Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993. doi: 10.1109/ICNN.1993.298572.

Eldar Kurtic, Alexandre Marques, Shubhra Pandit, Mark Kurtz, and Dan Alistarh. " give me bf16 or give me death"? accuracy-performance trade-offs in llm quantization. *arXiv preprint arXiv:2411.02355*, 2024.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, dec 1982. ISSN 1432-1807. doi: 10.1007/BF01457454. URL https://doi.org/10.1007/BF01457454.

Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*, volume 671 of *The Springer International Series in Engineering and Computer Science*. Springer, New York, NY, 1 edition, 2002. ISBN 978-0-7923-7688-0. doi: 10.1007/978-1-4615-0897-7. URL https://doi.org/10.1007/978-1-4615-0897-7.

Donald J. Rose, Robert E. Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. doi: 10.1137/0205021.

P. van Emde Boas. Another np-complete problem and the complexity of computing short vectors in a lattice. Technical Report 8104, University of Amsterdam, Department of Mathematics, Netherlands, 1981.

Appendix A. Experimental Results

A.1. Setup

In this section, we provide a detailed description of our experimental setup and procedures for comparing the quantization accuracy of different methods.

We work with the Qwen3 family of models, which come in a range of sizes. We focus on the Qwen3-8B model for detailed head-to-head comparisons, while the other variants, Qwen3-0.6B, Qwen3-1.7B, Qwen3-4B, and Qwen3-14B, help us assess how our method performs across different model scales.

We construct the calibration dataset for the GPTQ algorithm using the FineWeb-Edu dataset (HuggingFaceFW/fineweb-edu, subset sample-10BT). The dataset is streamed and shuffled with a fixed seed for reproducibility. After tokenizing the text samples, our 256 sequences are accumulated into non-overlapping sequences of length 2048.

We use WikiText-2 and C4 for perplexity evaluations. For WikiText-2, the entire test split is first concatenated using two line breaks as separators and then tokenized with the default HuggingFace tokenizer for each model. For C4, we sample individual documents from the selected shard, tokenize them, and randomly extract sequences of the desired length. In both cases, sequences shorter than the target length (2048 tokens) are discarded, and sequences longer than the target length are cropped to the specified window.

A.2. Results

We compare the round-to-nearest (RTN) and GPTQ quantization methods under two settings: even-bitwidth MSE clipped integers (RTN-MSE, GPTQ-MSE), and Huffman-encoded uneven-bitwidth unclipped integers (RTN-Huffman, GPTQ-Huffman).

In GPTQ-MSE, we use the standard GPTQ method, which quantizes with group size 128 and uses the mean squared error (MSE) for scale selection. RTN-MSE is just a round-to-nearest quantization method without Hessian guidance, which also uses group size 128 and MSE scale selection. In GPTQ-Huffman and RTN-Huffman, we find a unique scalar scale for a whole weight matrix via an entropy-based binary search. We start with a range proportional to the weight standard deviation and iteratively test candidate scales. At each step, the weights are quantized (using GPTQ or RTN) to integers without clipping, and the Huffman coding cost (which is close to the entropy) of the quantized values is measured.

The scale is adjusted until the resulting Huffman encoding bits match the target bitwidth. This allows optimizing compression efficiency while maintaining accuracy.

As shown in Figure 2, the left subplot compares different quantization methods on Qwen3-8B, highlighting that GPTQ-Huffman maintains low perplexity even at reduced bitwidths. The right subplot demonstrates the scaling behavior of GPTQ-Huffman across multiple model sizes, illustrating how effective model size in gigabytes impacts perplexity for different quantization bitwidths, and showing 3.125-bit as the Pareto optimal bitwidth.

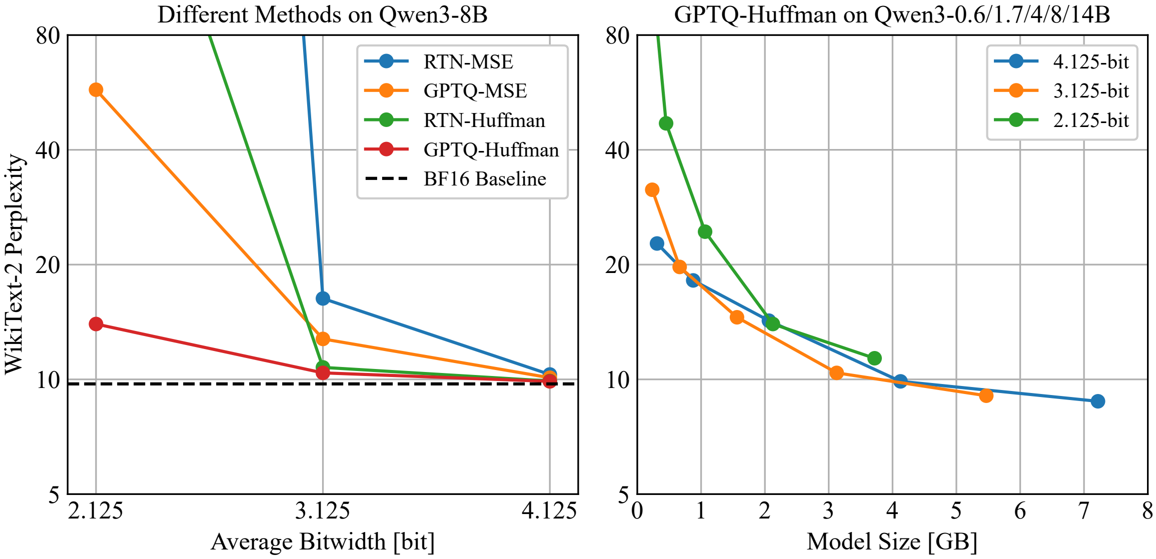


Figure 2: Perplexity compression trade-offs for Qwen3 models under different quantization schemes. **Left:** Comparison of quantization methods (RTN-MSE, GPTQ-MSE, RTN-Huffman, and GPTQ-Huffman) on Qwen3-8B evaluated on WikiText-2. Perplexity is plotted against the average effective bitwidth per weight, with the BF16 baseline shown as a dashed line. GPTQ-Huffman has the best (lowest) perplexity. **Right:** Scaling behavior of GPTQ-Huffman across multiple model sizes (0.6B, 1.7B, 4B, 8B, 14B) and bitwidths (4.125, 3.125, 2.125). The x-axis denotes the effective model size after quantization, and the y-axis shows perplexity on WikiText-2. Each curve corresponds to a fixed bitwidth, while points along a curve represent different model scales. Using our GPTQ-Huffman method, 3.125-bit stands out as the Pareto optimal bitwidth.

We now compare the benchmark results between RTN-MSE, GPTQ-MSE, RTN-Huffman, and GPTQ-Huffman using the Qwen3-8B model (Table 1). In addition, the results for other variants of Qwen3 with GPTQ-Huffman are shown in Table 2.

Table 3 shows additional results for the zero-shot and five-shot tasks. We use TruthfulQA for the zero-shot and WinoGrande for the five-shot section. Also, the additional results for GPTQ-Huffman of other Qwen3 models are in the Tables 4 and 5.

Table 1: Perplexity of Qwen3-8B model under GPTQ-Huffman, GPTQ-MSE, RTN-Huffman, and RTN-MSE with different bitwidths.

Method	Avg Bitwidth	Perplexity	
		WikiText-2	C4
RTN-MSE	4.125	10.3	15.2
	3.125	16.3	21.08
	2.125	2e10	2e10
GPTQ-MSE	4.125	10.1	13.92
	3.125	12.77	15.61
	2.125	57.51	36.14
RTN-Huffman	4.125	9.9	13.8
	3.125	10.75	14.63
	2.125	593.05	503
GPTQ-Huffman	4.125	9.88	13.14
	3.125	10.4	13.6
	2.125	13.97	16.89

Table 2: Perplexity of Qwen3 models under GPTQ-Huffman for different bitwidths.

Model	Avg Bitwidth	Perplexity	
		WikiText-2	C4
0.6B	16	20.96	26.37
	4.125	22.72	28.35
	3.125	31.43	37.92
	2.125	156.45	171.38
1.7B	16	21.03	25.11
	4.125	18.18	20.99
	3.125	19.72	23.15
	2.125	46.94	51.96
4B	16	13.66	17.07
	4.125	14.26	17.39
	3.125	14.55	18.17
	2.125	24.4	26.46
8B	16	9.73	13.55
	4.125	9.88	13.14
	3.125	10.4	13.6
	2.125	13.97	16.89
14B	16	8.65	12.23
	4.125	8.76	12.12
	3.125	9.06	13.97
	2.125	11.36	15.5

Table 3: Zero-shot and five-shot results for Qwen3-8B model.

Method	Avg Bitwidth	WinoGrande (%)	TruthfulQA (%)	
			mc-1	mc-2
BF16 Baseline	16	70.56	36.35	54.50
GPTQ-Huffman	4.125	70.09	35.01	53.36
	3.125	69.46	36.11	54.73
	2.125	62.43	31.09	49.01
GPTQ-MSE	4.125	70.8	36.35	54.55
	3.125	68.51	36.11	55.21
	2.125	55.64	28.4	46.91
RTN-Huffman	4.125	68.9	36.47	56.46
	3.125	67.96	35.13	53.68
	2.125	52.64	30.48	51.78
RTN-MSE	4.125	69.46	36.84	55.77
	3.125	57.62	34.03	52.76
	2.125	55.64	24.11	47.33

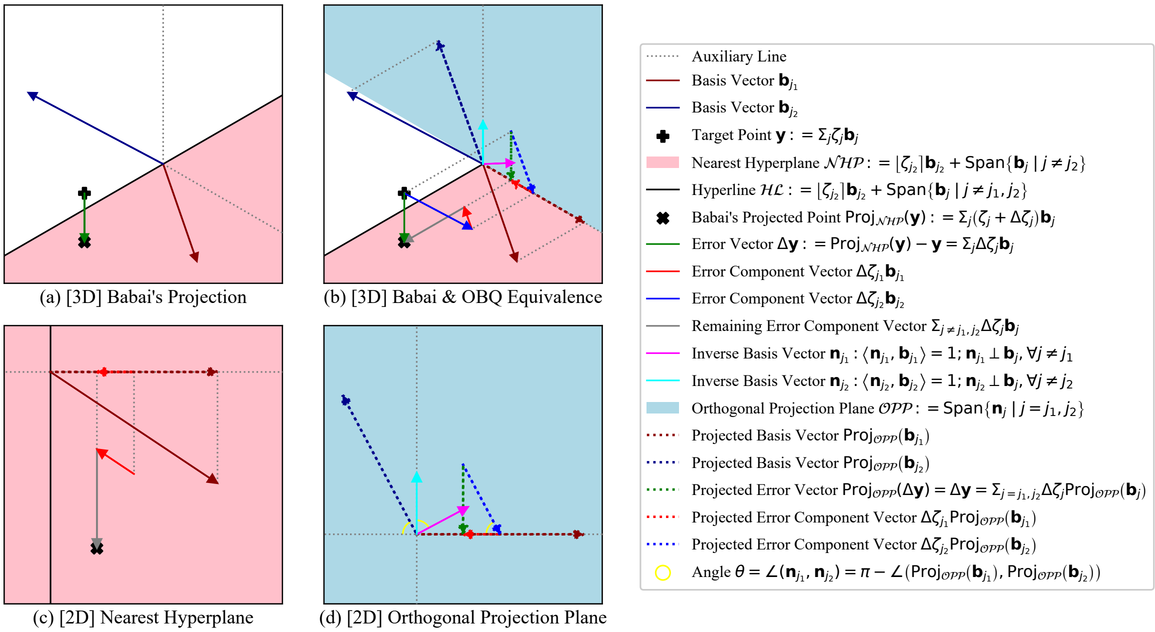
Table 4: TruthfullQA (%) zero-shot mc-1/mc-2 results for Qwen3 models quantized with GPTQ-Huffman.

Avg Bitwidth	0.6B	1.7B	4B	8B	14B
16	27.17/42.80	29.5/45.88	37.33/54.83	36.35/54.50	40.76/58.62
4.125	26.19/41.56	28.76/45.17	36.72/54.46	35.01/53.36	40.51/58.28
3.125	25.34/41.95	29.62/46.13	35.25/53.83	36.11/54.73	39.90/58.33
2.125	23.99/46.39	28.15/48.25	31.70/50.67	31.09/49.01	36.84/54.93

Table 5: WinoGrande (%) five-shot results for Qwen3 models quantized with GPTQ-Huffman.

Avg Bitwidth	0.6B	1.7B	4B	8B	14B
16	55.8	61.25	66.14	70.56	74.59
4.125	56.43	61.01	67.09	70.09	74.43
3.125	53.75	58.25	65.51	69.46	73.72
2.125	49.57	51.3	59.35	62.43	67.72

Appendix B. OBQ and Babai's Algorithm


Auxiliary Line
Basis Vector $\mathbf{b}_{j_1}$
Basis Vector $\mathbf{b}_{j_2}$
Target Point $\mathbf{y} := \Sigma_j \zeta_j \mathbf{b}_j$
Nearest Hyperplane $\mathcal{NHP} := \lfloor \zeta_{j_2} \rceil \mathbf{b}_{j_2} + \mathrm{Span}\{\mathbf{b}_j \mid j \neq j_2\}$
Hyperline $\mathcal{HL} := \lfloor \zeta_{j_2} \rceil \mathbf{b}_{j_2} + \mathrm{Span}\{\mathbf{b}_j \mid j \neq j_1, j_2\}$
Babai's Projected Point $\mathrm{Proj}_{\mathcal{NHP}}(\mathbf{y}) := \Sigma_j (\zeta_j + \Delta\zeta_j)\mathbf{b}_j$
Error Vector $\Delta\mathbf{y} := \mathrm{Proj}_{\mathcal{NHP}}(\mathbf{y}) - \mathbf{y} = \Sigma_j \Delta\zeta_j \mathbf{b}_j$
Error Component Vector $\Delta\zeta_{j_1}\mathbf{b}_{j_1}$
Error Component Vector $\Delta\zeta_{j_2}\mathbf{b}_{j_2}$
Remaining Error Component Vector $\Sigma_{j \neq j_1, j_2} \Delta\zeta_j \mathbf{b}_j$
Inverse Basis Vector $\mathbf{n}_{j_1} : \langle \mathbf{n}_{j_1}, \mathbf{b}_{j_1} \rangle = 1; \mathbf{n}_{j_1} \perp \mathbf{b}_j, \forall j \neq j_1$
Inverse Basis Vector $\mathbf{n}_{j_2} : \langle \mathbf{n}_{j_2}, \mathbf{b}_{j_2} \rangle = 1; \mathbf{n}_{j_2} \perp \mathbf{b}_j, \forall j \neq j_2$
Orthogonal Projection Plane $\mathcal{OPP} := \mathrm{Span}\{\mathbf{n}_j \mid j = j_1, j_2\}$
Projected Basis Vector $\mathrm{Proj}_{\mathcal{OPP}}(\mathbf{b}_{j_1})$
Projected Basis Vector $\mathrm{Proj}_{\mathcal{OPP}}(\mathbf{b}_{j_2})$
Projected Error Vector $\mathrm{Proj}_{\mathcal{OPP}}(\Delta\mathbf{y}) = \Delta\mathbf{y} = \Sigma_{j=j_1,j_2} \Delta\zeta_j \mathrm{Proj}_{\mathcal{OPP}}(\mathbf{b}_j)$
Projected Error Component Vector $\Delta\zeta_{j_1}\mathrm{Proj}_{\mathcal{OPP}}(\mathbf{b}_{j_1})$
Projected Error Component Vector $\Delta\zeta_{j_2}\mathrm{Proj}_{\mathcal{OPP}}(\mathbf{b}_{j_2})$
Angle $\theta = \angle(\mathbf{n}_{j_1}, \mathbf{n}_{j_2}) = \pi - \angle(\mathrm{Proj}_{\mathcal{OPP}}(\mathbf{b}_{j_1}), \mathrm{Proj}_{\mathcal{OPP}}(\mathbf{b}_{j_2}))$

(a) [3D] Babai's Projection

(b) [3D] Babai & OBQ Equivalence

(c) [2D] Nearest Hyperplane

(d) [2D] Orthogonal Projection Plane

Figure 3: Equivalence of OBQ's error propagation and Babai's projection. (a) 3D plot showing the target point being projected onto the nearest plane. (b) 3D plot showing how the projection error is propagated. (c) 2D plot showing the vectors on the nearest hyperplane in (b). (d) 2D plot showing the vectors on the orthogonal projection plane in (b).

Theorem 3 *Babai's nearest plane algorithm iteratively projects the target vector onto the nearest hyperplane and rounds the coefficient. The OBQ update step in GPTQ is exactly this projection.*

Proof Let $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_c]$ be the basis with $\boldsymbol{b}_j$ being a basis vector. Let $\boldsymbol{y} = \sum_{j \in J} \zeta_j \boldsymbol{b}_j$ be the current residual target point where J is the set of unprojected indices, and $\zeta_j \in \mathbb{R}$ is an unquantized floating-point number to be quantized to integers. Let $\mathcal{NHP} := \lfloor \zeta_{j_2} \rceil \mathbf{b}_{j_2} + \mathrm{Span}\{\mathbf{b}_j \mid j \neq j_2\}$ be the nearest hyperplane that is orthogonal to the Gram-Schmidt vector of basis $\boldsymbol{b}_{j_2}$. Figure 3 (a) is a 3D plot showing the projection error vector $\Delta\boldsymbol{y} = \mathrm{Proj}_{\mathcal{NHP}}(\boldsymbol{y}) - \boldsymbol{y}$. We focus on analyzing the error propagation in the direction of basis $\boldsymbol{b}_{j_1}$ induced by the projection of basis $\boldsymbol{b}_{j_2}$ and collapse the span of other basis vectors to a single dimension as illustrated by the hyperline $\mathcal{HL} := \lfloor \zeta_{j_2} \rceil \mathbf{b}_{j_2} + \mathrm{Span}\{\mathbf{b}_j | j \neq j_1, j_2\}$. Figure 3 (b) is a 3D plot showing the decomposition of the error $\Delta\boldsymbol{y} = \sum_{j \in J} \Delta\zeta_j \boldsymbol{b}_j$ as the error component vectors in the basis directions. Figure 3 (c) is a 2D plot showing the vectors on plane $\mathcal{NHP}$. The number ζ_j will be updated to $\zeta_j + \Delta\zeta_j$ such that $\mathrm{Proj}_{\mathcal{NHP}}(\boldsymbol{y}) = \sum_{j \in J} (\zeta_j + \Delta\zeta_j) \boldsymbol{b}_j$. Next, let $\boldsymbol{N} = \boldsymbol{B}^{-\top} = [\boldsymbol{n}_1, \ldots, \boldsymbol{n}_c]$ be the inverse basis. Then, we have $\langle \boldsymbol{n}_j, \boldsymbol{b}_j \rangle = 1$ and $\boldsymbol{n}_j \perp \boldsymbol{b}_{j'}, \forall j \neq j'$. We project all the vectors in Figure 3 (b) onto the orthogonal projection plane $\mathcal{OPP} := \text{Span}\left\{\boldsymbol{n}_j | j = j_1, j_2\right\}$ that is orthogonal to the hyperline $\mathcal{HL}$, and continue the proof in the 2D geometry in Figure 3 (d). Denote the angle $\theta = \angle\left(\boldsymbol{n}_{j_1}, \boldsymbol{n}_{j_2}\right) = \pi - \angle\left(\text{Proj}_{\mathcal{OPP}}\left(\boldsymbol{b}_{j_1}\right), \text{Proj}_{\mathcal{OPP}}\left(\boldsymbol{b}_{j_2}\right)\right)$. Then, $\frac{\Delta\zeta_{j_1}\left\|\text{Proj}_{\mathcal{OPP}}\left(\boldsymbol{b}_{j_1}\right)\right\|}{\Delta\zeta_{j_2}\left\|\text{Proj}_{\mathcal{OPP}}\left(\boldsymbol{b}_{j_2}\right)\right\|} = \cos\theta = \frac{\langle \boldsymbol{n}_{j_1}, \boldsymbol{n}_{j_2} \rangle}{\|\boldsymbol{n}_{j_1}\|\|\boldsymbol{n}_{j_2}\|} = \frac{\|\boldsymbol{n}_{j_2}\|}{\|\boldsymbol{n}_{j_1}\|}\frac{\langle \boldsymbol{n}_{j_1}, \boldsymbol{n}_{j_2} \rangle}{\langle \boldsymbol{n}_{j_2}, \boldsymbol{n}_{j_2} \rangle}$. For $j = j_1, j_2$, $\left\|\text{Proj}_{\mathcal{OPP}}\left(\boldsymbol{b}_j\right)\right\| \|\boldsymbol{n}_j\| = \frac{\langle \text{Proj}_{\mathcal{OPP}}(\boldsymbol{b}_j), \boldsymbol{n}_j \rangle}{\cos\left(\frac{\pi}{2} - \theta\right)} = \frac{\langle \boldsymbol{b}_j, \boldsymbol{n}_j \rangle}{\cos\left(\frac{\pi}{2} - \theta\right)} = \frac{1}{\cos\left(\frac{\pi}{2} - \theta\right)}$. For $j, j' \in \{j_1, j_2\}$, $\left\langle \boldsymbol{n}_j, \boldsymbol{n}_{j'} \right\rangle = \left(\boldsymbol{N}^\top \boldsymbol{N}\right)[j, j'] = \left(\boldsymbol{B}^\top \boldsymbol{B}\right)^{-1}[j, j']$. Combining the above equations, $\Delta\zeta_{j_1} = \frac{\left\|\text{Proj}_{\mathcal{OPP}}\left(\boldsymbol{b}_{j_2}\right)\right\|\|\boldsymbol{n}_{j_2}\|}{\left\|\text{Proj}_{\mathcal{OPP}}\left(\boldsymbol{b}_{j_1}\right)\right\|\|\boldsymbol{n}_{j_1}\|}\frac{\langle \boldsymbol{n}_{j_1}, \boldsymbol{n}_{j_2} \rangle}{\langle \boldsymbol{n}_{j_1}, \boldsymbol{n}_{j_2} \rangle}\Delta\zeta_{j_2} = \frac{\langle \boldsymbol{n}_{j_1}, \boldsymbol{n}_{j_2} \rangle}{\langle \boldsymbol{n}_{j_2}, \boldsymbol{n}_{j_2} \rangle}\Delta\zeta_{j_2} = \frac{\left(\boldsymbol{B}^\top \boldsymbol{B}\right)^{-1}[j_1, j_2]}{\left(\boldsymbol{B}^\top \boldsymbol{B}\right)^{-1}[j_2, j_2]}\Delta\zeta_{j_2}$. Finally, substituting $\boldsymbol{B} = \left(\boldsymbol{X}\ \text{diag}\left(\boldsymbol{s}_i\right)\right)[:, J]$ and $\zeta_j = \frac{\boldsymbol{w}_i[j]}{\boldsymbol{s}_i[j]}$ completes the proof. ■

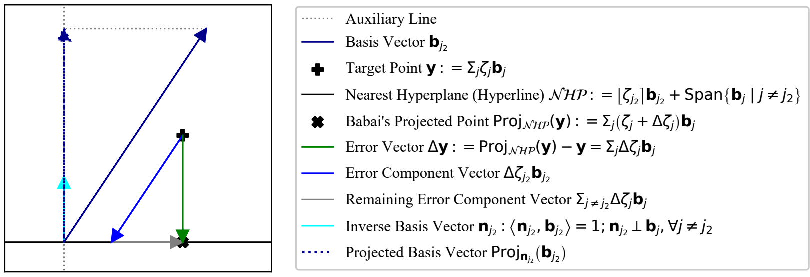

Auxiliary Line
Basis Vector $\mathbf{b}_{j_2}$
Target Point $\mathbf{y} := \Sigma_j \zeta_j \mathbf{b}_j$
Nearest Hyperplane (Hyperline) $\mathcal{NHP} := \lfloor \zeta_{j_2} \rceil \mathbf{b}_{j_2} + \text{Span}\{\mathbf{b}_j \mid j \neq j_2\}$
Babai's Projected Point $\text{Proj}_{\mathcal{NHP}}(\mathbf{y}) := \Sigma_j(\zeta_j + \Delta\zeta_j)\mathbf{b}_j$
Error Vector $\Delta\mathbf{y} := \text{Proj}_{\mathcal{NHP}}(\mathbf{y}) - \mathbf{y} = \Sigma_j \Delta\zeta_j \mathbf{b}_j$
Error Component Vector $\Delta\zeta_{j_2}\mathbf{b}_{j_2}$
Remaining Error Component Vector $\Sigma_{j \neq j_2}\Delta\zeta_j \mathbf{b}_j$
Inverse Basis Vector $\mathbf{n}_{j_2} : \langle \mathbf{n}_{j_2}, \mathbf{b}_{j_2} \rangle = 1; \mathbf{n}_{j_2} \perp \mathbf{b}_j, \forall j \neq j_2$
Projected Basis Vector $\text{Proj}_{\mathbf{n}_{j_2}}(\mathbf{b}_{j_2})$

Figure 4: Geometric interpretation of OBQ's quantization order. This 2D plot shows the target point being projected onto the nearest plane.

Corollary 4 *At each step, OBQ quantizes the unquantized dimension j such that the nearest hyperplane of dimension j is the closest to the target residual vector.*

Proof We use the same symbols defined in Theorem 3. Figure 4 is a 2D plot showing the distance (projection/quantization error) between the target residual vector $\boldsymbol{y}$ and the nearest hyperplane $\mathcal{NHP}$ orthogonal to the Gram-Schmidt vector of basis $\boldsymbol{b}_{j_2}$. For better illustration, we collapse $\mathcal{NHP}$ to a single dimension. The distance $\|\Delta\boldsymbol{y}\|$ can be written as $\|\Delta\boldsymbol{y}\| = \left\|\text{Proj}_{\boldsymbol{n}_{j_2}}\left(\Delta\boldsymbol{y}\right)\right\| = |\Delta\zeta_{j_2}| \left\|\text{Proj}_{\boldsymbol{n}_{j_2}}\left(\boldsymbol{b}_{j_2}\right)\right\| = \frac{|\Delta\zeta_{j_2}|}{\|\boldsymbol{n}_{j_2}\|}$. For each $\boldsymbol{w}_i$, OBQ independently selects $j = \text{argmin}_{j \in J} \frac{\left(\boldsymbol{q}_i[j] - \boldsymbol{w}_i[j]\right)^2}{\left(\boldsymbol{X}[:, J]^\top \boldsymbol{X}[:, J]\right)^{-1}[j, j]} = \text{argmin}_{j \in J} \frac{\left(\Delta\zeta_j\right)^2}{\langle \boldsymbol{n}_j, \boldsymbol{n}_j \rangle} = \text{argmin}_{j \in J} \frac{|\Delta\zeta_j|}{\|\boldsymbol{n}_j\|}$ as the next dimension to quantize, which is exactly minimizing this distance. ■

Appendix C. Algorithms

C.1. Pseudocodes of GPTQ and Babai's Algorithm

GPTQ algorithm. Algorithm 1 is the GPTQ algorithm for linear-layer quantization. The algorithm is identical to the original GPTQ paper (Frantar et al., 2023) except for missing the blocking mechanism that only affects the memory access pattern and computational speed, but not the numerical results.

Additional notations are as follows. $\boldsymbol{T} \in [0,1]^{c\times c}$ is a permutation matrix that modifies the dimensional order of GPTQ quantization. The default order is front-to-back (from the first to last dimension), i.e., $\boldsymbol{T} = \mathbf{I}$. $\lambda \in \mathbb{R}_+$ is a small damping factor for computing the Hessian matrix. Function LDL returns the lower triangular matrix in LDL decomposition. Function $\textsc{Round}\left(\cdot, \mathbb{Z}^\dagger\right) = \min\left(\max\left(\lfloor\cdot\rceil, \mathbb{Z}^\dagger_{\text{min}}\right), \mathbb{Z}^\dagger_{\text{max}}\right)$ returns the element-wise closest values in $\mathbb{Z}^\dagger$. We use Python-style indexing inside square brackets to select sub-matrices, e.g., $[j,:]$ selects the j-th row vector, $[:,j]$ selects the j-th column vector, and $[j:,j]$ selects the sub-column consisting of rows after j-th (included) row in j-th column, etc.

Babai's nearest plane algorithm. Algorithm 2 is Babai's nearest plane algorithm (Babai, 1986) to solve CVP, which iteratively projects a target vector onto the nearest hyperplane and rounds the coefficient.

Additional notations are as follows. Function LLL returns the transformation matrix of the LLL reduction with parameter delta defaulting to $\frac{3}{4}$. Function QR returns the orthogonal matrix in QR decomposition, the same as the normalized Gram-Schmidt orthogonalization process. Function Round and the indexing inside square brackets are defined as in the GPTQ algorithm.

Algorithm 1: GPTQ

```
Input: $\boldsymbol{W}, \boldsymbol{S}, \boldsymbol{X}, \boldsymbol{T}, \lambda, \mathbb{Z}^\dagger$
Output: $\boldsymbol{Z}, \boldsymbol{Q}$
$\boldsymbol{H} \leftarrow \boldsymbol{T}^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}\right) \boldsymbol{T}$
$\boldsymbol{L} \leftarrow \text{LDL}\left(\boldsymbol{H}^{-1}\right)$
$\boldsymbol{W}, \boldsymbol{S} \leftarrow \boldsymbol{T}^{-1}\boldsymbol{W}, \boldsymbol{T}^{-1}\boldsymbol{S}$
$\boldsymbol{Q}, \boldsymbol{Z} \leftarrow \boldsymbol{W}, \mathbf{0}$
for $j \leftarrow 1$ to $c$ do
    $\boldsymbol{\zeta} \leftarrow \boldsymbol{W}[j,:] / \boldsymbol{S}[j,:]$
    $\boldsymbol{Z}[j,:] \leftarrow \textsc{Round}\left(\boldsymbol{\zeta}, \mathbb{Z}^\dagger\right)$
    $\boldsymbol{Q}[j,:] \leftarrow \boldsymbol{Z}[j,:] * \boldsymbol{S}[j,:]$
    $\boldsymbol{\varepsilon} \leftarrow \boldsymbol{Q}[j,:] - \boldsymbol{W}[j,:]$
    $\boldsymbol{W}[j:,:] \leftarrow \boldsymbol{W}[j:,:] + \boldsymbol{L}[j:,j]\boldsymbol{\varepsilon}$
end
$\boldsymbol{Z}, \boldsymbol{Q} \leftarrow \boldsymbol{TZ}, \boldsymbol{TQ}$
```

Algorithm 2: Babai's Nearest Plane

```
Input: $\boldsymbol{B}, \boldsymbol{y}$
Output: $\boldsymbol{z}$
$\boldsymbol{T} \leftarrow \text{LLL}(\boldsymbol{B})$ // transformation
$\boldsymbol{A} \leftarrow \boldsymbol{BT}$ // basis reduction
$\boldsymbol{\Phi} \leftarrow \text{QR}(\boldsymbol{A})$ // orthogonalize
$\boldsymbol{y}', \boldsymbol{z} \leftarrow \boldsymbol{y}, \mathbf{0}$
for $j \leftarrow c$ to 1 do
    $\zeta \leftarrow \langle \boldsymbol{\Phi}[:,j], \boldsymbol{y}' \rangle / \langle \boldsymbol{\Phi}[:,j], \boldsymbol{A}[:,j] \rangle$
    $\boldsymbol{z}[j] \leftarrow \textsc{Round}(\zeta, \mathbb{Z})$
    $\boldsymbol{y}' \leftarrow \boldsymbol{y}' - \boldsymbol{A}[:,j]\boldsymbol{z}[j]$
end
$\boldsymbol{z} \leftarrow \boldsymbol{Tz}$
```

C.2. Applying Babai's Algorithm to Batched Quantization

We can introduce a factor of the Hessian matrix, $\boldsymbol{\mathcal{X}} = [\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_c]$ with $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{\mathcal{X}}^\top \boldsymbol{\mathcal{X}}$. The loss can then be reformulated as $\left\| \boldsymbol{\mathcal{X}} \operatorname{diag}(\boldsymbol{s}_i) \boldsymbol{z}_i - \boldsymbol{\mathcal{X}} \boldsymbol{w}_i \right\|^2$.

Theorem 5 *The CVPs using any possible factors $\boldsymbol{\mathcal{X}}$ of the Hessian matrix $\boldsymbol{X}^\top \boldsymbol{X}$ are equivalent under an orthogonal transformation (rotation and sign changes) of the lattice and external target vectors.*

Proof Let $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{X}}'$ be two possible factors of the Hessian matrix with $\boldsymbol{\mathcal{X}}^\top \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}'^\top \boldsymbol{\mathcal{X}}'$. The inner products $\boldsymbol{\chi}_{j_1}^\top \boldsymbol{\chi}_{j_2}$ and $\boldsymbol{\chi}_{j_1}'^\top \boldsymbol{\chi}_{j_2}'$ must be equal for all $1 \leq j_1, j_2 \leq c$. In other words, the lengths of $\boldsymbol{\chi}_{j_1}$ and $\boldsymbol{\chi}_{j_1}'$ must be the same, and the angle between $\boldsymbol{\chi}_{j_1}$ and $\boldsymbol{\chi}_{j_2}$ and the angle between $\boldsymbol{\chi}_{j_1}'$ and $\boldsymbol{\chi}_{j_2}'$ must be the same, for all $1 \leq j_1, j_2 \leq c$. ■

According to Theorem 5, any decomposition factor $\boldsymbol{\mathcal{X}}$ of the Hessian matrix $\boldsymbol{X}^\top \boldsymbol{X}$ can be used instead of $\boldsymbol{X}$ without changing the geometric properties of the CVP and its associated quantization problem. This is useful to reduce the computational cost, e.g., we can use a square matrix $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{c \times c}$ instead of the rectangular matrix $\boldsymbol{X} \in \mathbb{R}^{n \times c}$.

Given the equivalence we have shown in Section 4.1, the quantization problem can be converted to CVP, allowing us to apply Babai's nearest plane algorithm in the context of quantization. A naive way is to compute $\boldsymbol{B}_{(i)} = \boldsymbol{\mathcal{X}} \operatorname{diag}(\boldsymbol{s}_i)$ and $\boldsymbol{y}_{(i)} = \boldsymbol{\mathcal{X}} \boldsymbol{w}_i$ and run Babai's algorithm independently for all $1 \leq i \leq r$. However, this is computationally inefficient, as we will need to compute the expensive $(O(c^4))$ LLL basis reduction transformation $\boldsymbol{T}_{(i)}$ for the basis $\boldsymbol{B}_{(i)}$ and the expensive $(O(c^3))$ QR decomposition of $\boldsymbol{A}_{(i)} = \boldsymbol{B}_{(i)} \boldsymbol{T}_{(i)}$ for r times. However, a few adjustments can be made to simplify the computation and enable batched processing.

Disabling basis reduction. The LLL basis reduction is unfortunately scale-sensitive, generating different transformations $\boldsymbol{T}_{(i)}$ for different scales $\boldsymbol{s}_i$ (unless all the $\boldsymbol{s}_i$ vectors are parallel), which prohibits the reuse of QR decomposition results. Furthermore, LLL basis reduction is incompatible with clipping, as the roundings are performed in another basis, and there is no easy way to do the clipping for the original basis.

Changing quantization order. Quantization order is a feature in GPTQ that controls the rounding and clipping order of the dimensions. This order influences the quantization error, as we will discuss later in Section 4.3. In the context of Babai's algorithm, this corresponds to the order of the basis in the Gram-Schmidt orthogonalization and the hyperplane projections, as shown in Figure 1 (g-h). To do so, we can replace the LLL basis reduction in Babai's algorithm with a permutation by setting the transformation matrix $\boldsymbol{T}$ to a permutation matrix that is independent of i.

Theorem 6 *If $\boldsymbol{T}$ is a permutation matrix that does not depend on i, the orthogonal matrix $\boldsymbol{\Phi}$ can be reused without recomputing the QR decomposition for each i.*

Proof The permutation matrix $\boldsymbol{T} \in [0,1]^{c \times c}$ has exactly one non-zero element in each row and column. Scaling the rows of $\boldsymbol{T}$ can also be interpreted as scaling the columns of $\boldsymbol{T}$, therefore its multiplication with a diagonal matrix has property: $\operatorname{diag}(\boldsymbol{s}_i) \boldsymbol{T} = \boldsymbol{T} \operatorname{diag}(\boldsymbol{T}^{-1} \boldsymbol{s}_i)$. Let $\boldsymbol{A} = \boldsymbol{\mathcal{X}} \boldsymbol{T}$, $\boldsymbol{A}_{(i)} = \boldsymbol{\mathcal{X}} \operatorname{diag}(\boldsymbol{s}_i) \boldsymbol{T}$. Denote the QR decomposition of $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{\Phi} \boldsymbol{R}$ with $\boldsymbol{\Phi}$ being an orthogonal matrix and $\boldsymbol{R}$ being an upper triangular matrix. Then, the QR decomposition of $\boldsymbol{A}_{(i)}$ becomes $\boldsymbol{A}_{(i)} = \boldsymbol{\mathcal{X}} \operatorname{diag}(\boldsymbol{s}_i) \boldsymbol{T} = \boldsymbol{\mathcal{X}} \boldsymbol{T} \operatorname{diag}(\boldsymbol{T}^{-1} \boldsymbol{s}_i) = \boldsymbol{A} \operatorname{diag}(\boldsymbol{T}^{-1} \boldsymbol{s}_i) = \boldsymbol{\Phi} (\boldsymbol{R} \operatorname{diag}(\boldsymbol{T}^{-1} \boldsymbol{s}_i))$. Therefore, the QR decompositions of $\boldsymbol{A}_{(i)}$ share the same orthogonal matrix $\boldsymbol{\Phi}$ for all $1 \leq i \leq r$. ■

As shown in Theorem 6, changing quantization order does not require repeated computation of the QR decomposition. Note that, we also need to permute the scale $\boldsymbol{S}$ accordingly to $\boldsymbol{T}^{-1} \boldsymbol{S}$.

Selecting basis. Putting things together, we are interested in $\boldsymbol{A} = \boldsymbol{\mathcal{X}}\boldsymbol{T}$ and its QR decomposition $\boldsymbol{\Phi}$. Theorem 5 allows us to choose any Hessian factor $\boldsymbol{\mathcal{X}}$ while keeping the result intact. Without loss of generality, we can choose a $\boldsymbol{\mathcal{X}}$ such that $\boldsymbol{A}$ is an upper triangular matrix and the QR decomposition becomes trivial: $\boldsymbol{\Phi} = \mathbf{I}$, which simplifies the computation. The upper triangular matrix $\boldsymbol{A}$ can be directly computed from the Cholesky decomposition of the permuted Hessian matrix $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{T}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{T}$.

Applying all the considerations in this subsection, we construct Algorithm 3 for batched quantization using Babai's algorithm.

```
Algorithm 3: Babai's Quantize
Input: $\boldsymbol{W}, \boldsymbol{S}, \boldsymbol{X}, \boldsymbol{T}, \lambda, \mathbb{Z}^\dagger$
Output: $\boldsymbol{Z}, \boldsymbol{Q}$
$\boldsymbol{H} \leftarrow \boldsymbol{T}^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}\right) \boldsymbol{T}$
$\boldsymbol{A} \leftarrow \textsc{Cholesky}\left(\boldsymbol{H}\right)^\top$
$\boldsymbol{W}, \boldsymbol{S} \leftarrow \boldsymbol{T}^{-1}\boldsymbol{W}, \boldsymbol{T}^{-1}\boldsymbol{S}$
$\boldsymbol{Y}, \boldsymbol{Q}, \boldsymbol{Z} \leftarrow \boldsymbol{A}\boldsymbol{W}, \boldsymbol{W}, \boldsymbol{0}$
for $j \leftarrow c$ to 1 do
    $\boldsymbol{\omega} \leftarrow \boldsymbol{Y}[j,:]/\boldsymbol{A}[j,j]$
    $\boldsymbol{\zeta} \leftarrow \boldsymbol{\omega}/\boldsymbol{S}[j,:]$
    $\boldsymbol{Z}[j,:] \leftarrow \textsc{Round}\left(\boldsymbol{\zeta}, \mathbb{Z}^\dagger\right)$
    $\boldsymbol{Q}[j,:] \leftarrow \boldsymbol{Z}[j,:] * \boldsymbol{S}[j,:]$
    $\boldsymbol{Y} \leftarrow \boldsymbol{Y} - \boldsymbol{A}[:,j]\boldsymbol{Q}[j,:]$
end
$\boldsymbol{Z}, \boldsymbol{Q} \leftarrow \boldsymbol{T}\boldsymbol{Z}, \boldsymbol{T}\boldsymbol{Q}$
```

Ineffectiveness of additional GPTQ refinement on Babai's algorithm. A seemingly appealing idea is to take the solution returned by each Babai's iteration and then perform one further GPTQ-style error propagation step on the weights in the space projected by $\boldsymbol{A}$, as a further update on $\boldsymbol{Y}$, hoping to push the approximation even closer to the optimum. However, as proved in Section F.4, such an extra update vanishes: the intermediate quantity $\boldsymbol{\omega}$ and therefore the final results of $\boldsymbol{Z}$ and $\boldsymbol{Q}$ remain unchanged. In other words, once Babai's projection has been executed, any subsequent GPTQ-style correction is algebraically redundant. This result confirms that the equivalence established in Theorem 1 is already tight and that neither algorithm can be strengthened by naively composing it with the other.

Appendix D. Error Bound

D.1. Babai's Error Bound

Formally, let $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_c]$ be the set of normalized Gram-Schmidt vectors of the basis $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_c]$. Let $\tilde{\boldsymbol{A}} = [\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_c]$ denote the unnormalized Gram-Schmidt vectors with $\tilde{\boldsymbol{a}}_j = \langle \boldsymbol{\phi}_j, \boldsymbol{a}_j \rangle \boldsymbol{\phi}_j$. At iteration j, the algorithm replaces the exact coefficient ζ by the closest integer, so the deviation satisfies $|\zeta - \boldsymbol{z}[j]| \leq \frac{1}{2}$. Hence the error component along $\tilde{\boldsymbol{b}}_j$ has norm at most $\frac{1}{2}\|\tilde{\boldsymbol{a}}_j\|$. Because the $\tilde{\boldsymbol{A}}$ is orthogonal, these error components add in Euclidean norm, giving a bound on the residual vector $\boldsymbol{y}'$: $\|\boldsymbol{y}'\|^2 \le \frac{1}{4}\sum_{j=1}^{c}\|\tilde{\boldsymbol{a}}_j\|^2 = \frac{1}{4}\sum_{j=1}^{c}\langle\boldsymbol{\phi}_j, \boldsymbol{a}_j\rangle^2$. Babai's algorithm guarantees to return the center vector of the hyper-cuboid (Figure 1 (g)) constructed by the unnormalized Gram-Schmidt vectors $\tilde{\boldsymbol{A}}$ where the target $\boldsymbol{y}$ is located. Equality is attained when the target $\boldsymbol{y}$ lies at the corner of the hyper-cuboid, so the bound is tight.

Babai (1986) additionally proved the relative error bound for γ with $\|\boldsymbol{B}\boldsymbol{z} - \boldsymbol{y}\| \le \gamma \cdot \min_{\boldsymbol{z}'\in\mathbb{Z}^c}\|\boldsymbol{B}\boldsymbol{z}' - \boldsymbol{y}\|$. The bound is $1 \le \gamma \le \sqrt{1 + \max_{1\le j\le c}\frac{\sum_{j'=1}^{j}\|\tilde{\boldsymbol{a}}_{j'}\|^2}{\|\tilde{\boldsymbol{a}}_j\|^2}} \le \sqrt{c+1}\cdot \max_{1\le j'\le j\le c}\frac{\|\tilde{\boldsymbol{a}}_{j'}\|}{\|\tilde{\boldsymbol{a}}_j\|}$.

D.2. Quantization Error Bound

Theorem 7 (Theorem 2 with permutation matrix $\boldsymbol{T}$) *Assume there is no clipping ($\mathbb{Z}^\dagger = \mathbb{Z}$). Let $\boldsymbol{D}$ be the diagonal matrix in the LDL decomposition of the permuted Hessian matrix $\boldsymbol{T}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{T}$. For every output channel i ($1 \le i \le r$) produced by Babai's algorithm (Algorithm 3), or equivalently GPTQ executed back-to-front, the quantization error has a tight error upper bound: $\|\boldsymbol{X}\ \mathrm{diag}(\boldsymbol{s}_i)\,\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2 \le \frac{1}{4}\boldsymbol{s}_i^\top\boldsymbol{T}^{-\top}\boldsymbol{D}\boldsymbol{T}^{-1}\boldsymbol{s}_i$.*

Proof Denote $\boldsymbol{B}_{(i)} = \boldsymbol{\mathcal{X}}\ \mathrm{diag}(\boldsymbol{s}_i)$, $\boldsymbol{y}_{(i)} = \boldsymbol{\mathcal{X}}\boldsymbol{w}_i$ as in Section 4.1 so that the quantization problem is the CVP minimizing $\left\|\boldsymbol{B}_{(i)}\boldsymbol{z}_i - \boldsymbol{y}_{(i)}\right\|^2$. Applying Babai's algorithm with the permutation $\boldsymbol{T}$ gives the permuted basis $\boldsymbol{A}_{(i)} = \boldsymbol{B}_{(i)}\boldsymbol{T} = \boldsymbol{\mathcal{X}}\ \mathrm{diag}(\boldsymbol{s}_i)\,\boldsymbol{T} = \boldsymbol{\mathcal{X}}\boldsymbol{T}\ \mathrm{diag}\left(\boldsymbol{T}^{-1}\boldsymbol{s}_i\right)$. Write the unnormalized Gram-Schmidt vectors of $\boldsymbol{A}_{(i)}$ as $\tilde{\boldsymbol{A}}_{(i)} = \left[\tilde{\boldsymbol{a}}_{(i)1}, \ldots, \tilde{\boldsymbol{a}}_{(i)c}\right]$. Babai's guarantee therefore yields the tight bound $\left\|\boldsymbol{B}_{(i)}\boldsymbol{z}_i - \boldsymbol{y}_{(i)}\right\|^2 = \left\|\boldsymbol{A}_{(i)}\left(\boldsymbol{T}^{-1}\boldsymbol{z}_i\right) - \boldsymbol{y}_{(i)}\right\|^2 \le \frac{1}{4}\sum_{j=1}^{c}\left\|\tilde{\boldsymbol{a}}_{(i)j}\right\|^2$.

We may, without loss of generality, use Theorem 5 to rotate $\boldsymbol{\mathcal{X}}$ so that $\boldsymbol{A}_{(i)}$ is upper triangular. In that case, the norm $\left\|\tilde{\boldsymbol{a}}_{(i)j}\right\|$ simplifies to $\left|\boldsymbol{A}_{(i)}[j,j]\right|$. The summation on the right-hand side can then be expressed as $\mathrm{tr}\left(\boldsymbol{D}_{(i)}\right)$ with $\boldsymbol{D}_{(i)}$ denoting the diagonal matrix of the LDL decomposition of $\boldsymbol{A}_{(i)}^\top\boldsymbol{A}_{(i)}$. Let $\boldsymbol{\mathcal{L}}$ be the lower triangular matrix in the LDL decomposition of $\boldsymbol{T}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{T}$, so that $\boldsymbol{A}_{(i)}^\top\boldsymbol{A}_{(i)} = \mathrm{diag}\left(\boldsymbol{T}^{-1}\boldsymbol{s}_i\right)\boldsymbol{T}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{T}\ \mathrm{diag}\left(\boldsymbol{T}^{-1}\boldsymbol{s}_i\right) = \boldsymbol{\mathcal{L}}_{(i)}, \boldsymbol{D}_{(i)}, \boldsymbol{\mathcal{L}}_{(i)}^\top$ with $\boldsymbol{D}_{(i)} = \mathrm{diag}\left(\boldsymbol{T}^{-1}\boldsymbol{s}_i\right)\boldsymbol{D}\ \mathrm{diag}\left(\boldsymbol{T}^{-1}\boldsymbol{s}_i\right)$ and $\boldsymbol{\mathcal{L}}_{(i)} = \mathrm{diag}\left(\boldsymbol{T}^{-1}\boldsymbol{s}_i\right)\boldsymbol{\mathcal{L}}\ \mathrm{diag}\left(\boldsymbol{T}^{-1}\boldsymbol{s}_i\right)^{-1}$. The trace $\mathrm{tr}\left(\boldsymbol{D}_{(i)}\right) = \boldsymbol{s}_i^\top\boldsymbol{T}^{-\top}\boldsymbol{D}\boldsymbol{T}^{-1}\boldsymbol{s}_i$. Dividing by 4 completes the bound. ■

For no-clipping GPTQ with the default front-to-back order (Algorithm 1) and the permutation $\boldsymbol{T}$, the error bound is $\|\boldsymbol{X}\ \mathrm{diag}(\boldsymbol{s}_i)\,\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2 \le \frac{1}{4}\boldsymbol{s}_i^\top\boldsymbol{T}^{-\top}\mathbf{P}\boldsymbol{D}_{\mathrm{P}}\mathbf{P}\boldsymbol{T}^{-1}\boldsymbol{s}_i$ with $\boldsymbol{D}_{\mathrm{P}}$ being the diagonal matrix in the LDL decomposition of $\mathbf{P}\boldsymbol{T}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{T}\mathbf{P}$.

For the relative no-clipping quantization error bound, we have $1 \le \gamma \le \sqrt{1 + \max_{1\le j\le c}\frac{\sum_{j'=1}^{j}d_{j'}^2}{d_j^2}} \le \sqrt{c+1}\cdot\max_{1\le j'\le j\le c}\frac{\|d_{j'}\|}{\|d_j\|}$ where $d_j = \sqrt{\left(\mathrm{diag}(\boldsymbol{s}_i)\,\boldsymbol{T}^{-\top}\boldsymbol{D}\boldsymbol{T}^{-1}\mathrm{diag}(\boldsymbol{s}_i)\right)[j,j]}$ for Babai's algorithm and $d_j = \sqrt{\left(\mathrm{diag}(\boldsymbol{s}_i)\,\boldsymbol{T}^{-\top}\mathrm{P}\boldsymbol{D}_{\mathrm{P}}\mathrm{P}\boldsymbol{T}^{-1}\mathrm{diag}(\boldsymbol{s}_i)\right)[j,j]}$ for the GPTQ algorithm.

D.3. Expected Quantization Error over a Uniform Hyper-Cuboid

We have shown that, when clipping is disabled, Babai's nearest-plane (hence back-to-front GPTQ) ensures the tight worst-case bound

$$\|\boldsymbol{X}\ \mathrm{diag}\left(\boldsymbol{s}_i\right)\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2 \leq \frac{1}{4}\sum_{j=1}^{c}\|\tilde{\boldsymbol{a}}_j\|^2, \quad \tilde{\boldsymbol{A}} = [\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_c] \tag{1}$$

where $\tilde{\boldsymbol{a}}_j$ are the unnormalized Gram-Schmidt vectors of the permuted lattice basis $\boldsymbol{A}$.

Introduce the half-edge lengths

$$a_j = \frac{1}{2}\|\tilde{\boldsymbol{a}}_j\|, \quad j = 1, \ldots, c, \tag{2}$$

so that the Babai residual always lies in the axis-aligned hyper-cuboid $\prod_{j=1}^{c}[-a_j, a_j]$ and Eq. 1 is rewritten as

$$\epsilon_{\text{worst}} = \sum_{j=1}^{c} a_j^2. \tag{3}$$

Uniform prior on the unknown weight vector. Assume now that the continuous unquantized weight offset $\boldsymbol{u} = \boldsymbol{X}\left(\boldsymbol{w}_i - \ \mathrm{diag}(\boldsymbol{s}_i)\boldsymbol{z}_i\right)$ is uniformly distributed inside this hyper-cuboid, i.e., each coordinate $u_j \sim \mathrm{Uniform}\left(-a_j, a_j\right)$ and the coordinates are independent. The squared error becomes the random variable

$$\epsilon = \sum_{j=1}^{c} u_j^2. \tag{4}$$

Lemma 8 *For a scalar $u \sim \mathrm{Uniform}\left(-a, a\right)$ one has $\mathbb{E}[u^2] = \frac{a^2}{3}$.*

Proof

$$\mathbb{E}[u^2] = \frac{1}{2a}\int_{-a}^{a} u^2 \mathrm{d}u = \frac{1}{2a}\left[\frac{1}{3}x^3\right]_{-a}^{a} = \frac{a^2}{3}. \tag{5}$$

■

Expected residual norm. Using independence,

$$\mathbb{E}[\epsilon] = \sum_{j=1}^{c}\mathbb{E}\left[u_j^2\right] = \frac{1}{3}\sum_{j=1}^{c} a_j^2. \tag{6}$$

Ratio to the worst-case bound. Comparing Eq. 6 with Eq. 3 gives

$$\boxed{\mathbb{E}[\epsilon] = \frac{1}{3}\epsilon_{\text{worst}}} \quad \Longrightarrow \quad \mathbb{E}\left[\|\boldsymbol{X}\ \mathrm{diag}\left(\boldsymbol{s}_i\right)\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{w}_i\|^2\right] = \frac{1}{12}\sum_{j=1}^{c}\|\tilde{\boldsymbol{a}}_j\|^2. \tag{7}$$

Hence, under a uniform prior on the weights inside Babai's orthogonal hyper-cuboid, the average layer-wise quantization error is exactly $\frac{1}{3}$ of the worst-case guarantee stated in Theorem 2.

Appendix E. Discussion on Quantization Order

The quadratic form on the right-hand side of the error bound in Theorem 2 depends on the permutation matrix $\boldsymbol{T}$. Re-ordering the dimensions changes the entries of the diagonal matrix $\boldsymbol{D}$ before the scale $\boldsymbol{s}_i$ is "weighted" by them. A poor order may place large $\boldsymbol{D}$ entries against large $\boldsymbol{s}_i$ entries and hence inflate the bound.

For a batched quantization algorithm like GPTQ or our proposed Babai's algorithm, $\boldsymbol{T}$ should be independent of i. To develop a good heuristic order, a reasonable approximation to make, especially for large quantization group sizes, is that the elements of $\boldsymbol{s}_i[j]$ are equal for all $1 \leq j \leq c$. Then we can focus on finding the optimal pivot order for the LDL decomposition of the Hessian matrix $\boldsymbol{X}^\top \boldsymbol{X}$ to minimize the trace of $\boldsymbol{D}$.

Finding the optimal order is NP-hard, e.g. Rose et al. (1976). However, heuristics often effectively reduce the trace term in practice. Even in the clipping cases, the heuristics still can often reduce the error. GPTQ introduces the so-called "act-order", the descending order of the Hessian diagonal. This translates to the ascending order of the Hessian diagonal when applied to Babai's algorithm. This "act-order" is a good heuristic, but it only considers the information from the Hessian diagonal instead of the full matrix.

To improve the "act-order", we propose the "min-pivot" order, which is essentially taking the minimum diagonal entry at each LDL (or Cholesky) decomposition step. This order can be calculated by Algorithm 4, which has cubic time complexity and does not increase the overall time complexity of the whole quantization process. This order also has a geometric interpretation as the order of the Gram-Schmidt orthogonalization process of the basis: always taking the shortest residual vector as the next one to orthogonalize, agreeing with Babai's relative error bound.

Algorithm 4: Min-Pivot

```
Input: $\boldsymbol{H}$
Output: $\boldsymbol{T}$
$J \leftarrow \{1, \ldots, c\}$
$\boldsymbol{T} \leftarrow \boldsymbol{0}$
for $j \leftarrow 1$ to $c$ do
    $j' \leftarrow \mathrm{argmin}_{j' \in J} \boldsymbol{H}[j', j']$
    $\boldsymbol{H} \leftarrow \boldsymbol{H} - \boldsymbol{H}[:, j']\boldsymbol{H}[j', :]/\boldsymbol{H}[j', j']$
    $\boldsymbol{T}[j', j] \leftarrow 1$
    $J \leftarrow J \setminus \{j'\}$
end
```

Appendix F. Equivalence Proof of GPTQ and Babai's Algorithm

In this section, we prove Theorem 1 that GPTQ (Algorithm 1) and Babai's algorithm (Algorithm 3) are equivalent if the dimensional orders are opposite.

Because a permutation matrix $\boldsymbol{T}$ acts only as re-ordering coordinates, we may apply $\boldsymbol{T}$ once at the beginning (to $\boldsymbol{W}$, $\boldsymbol{S}$, and $\boldsymbol{X}$) and once at the end (to $\boldsymbol{Z}$ and $\boldsymbol{Q}$) without affecting any intermediate arithmetic. Hence, all algebra performed inside the two algorithms can be analyzed in the permuted basis where $\boldsymbol{T}$ is the identity. On that basis, the sole distinction between GPTQ and Babai's algorithm lies in the direction of the iterations. Proving that GPTQ running back-to-front ($j \leftarrow c$ to 1) reproduces Babai's updates in Babai's default iteration direction would complete the equivalence proof.

We follow a three-step proof scheme.

- **Step 1.** Proving that the original GPTQ algorithm (Algorithm 5) that uses relative quantization error row vector $\boldsymbol{\varepsilon} \in \mathbb{R}^{1\times r}$ is equivalent to a new algorithm (Algorithm 6) using the absolute quantization error matrix $\boldsymbol{\Delta} \in \mathbb{R}^{c\times r}$.
- **Step 2.** Reversing the iteration in Algorithm 6 and writing the reversed-iteration algorithm as Algorithm 7.
- **Step 3.** Proving that the reversed-iteration algorithm Algorithm 7 is equivalent to Babai's algorithm Algorithm 8.

Algorithms 5 to 8 are intentionally written in the linear algebra form. $\mathbf{e}_j \in \mathbb{R}^c$ is the standard basis vector whose elements are 0 except the j-th element being 1, which is used as the row or column selector of a matrix. The superscripts in parentheses denote the versions of the variables during the iterations. $\boldsymbol{\omega}, \boldsymbol{\zeta} \in \mathbb{R}^{1\times r}$ are intermediate row vectors. Additionally, $\boldsymbol{L}$ is the LDL decomposition of the Hessian inverse $\boldsymbol{H}^{-1} = \boldsymbol{L}\boldsymbol{D}_{\mathrm{L}}^{\frac{1}{2}}\boldsymbol{D}_{\mathrm{L}}^{\frac{1}{2}}\boldsymbol{L}^\top$ where $\boldsymbol{L}$ is a lower triangular matrix with all diagonal elements being 1, and $\boldsymbol{D}_{\mathrm{L}}^{\frac{1}{2}}$ is a non-negative diagonal matrix. Similarly, $\boldsymbol{U}$ is the "UDU" decomposition of the Hessian inverse $\boldsymbol{H}^{-1} = \boldsymbol{U}\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{U}^\top$ where $\boldsymbol{U}$ is an upper triangular matrix with all diagonal elements being 1, and $\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}$ is a non-negative diagonal matrix.

Note: the symbols are overloaded in Algorithms 5 to 8, and the variables using the same symbols may carry different values, even if the inputs to the algorithms are the same.

F.1. Step 1

To distinguish the variables using the same symbol in Algorithms 5 and 6, we use symbols without ˆ to denote the symbols in Algorithm 5, and use the symbols with ˆ for Algorithm 6.

Claim

$$\boldsymbol{\omega}_j = \hat{\boldsymbol{\omega}}_j, \quad 1 \le j \le c, \tag{8}$$

and consequently,

$$\boldsymbol{Z}^{(j)} = \hat{\boldsymbol{Z}}^{(j)}, \quad 0 \le j \le c, \tag{9}$$

and

$$\boldsymbol{Q}^{(j)} = \hat{\boldsymbol{Q}}^{(j)}, \quad 0 \le j \le c. \tag{10}$$

Proof Eq. 8 by Induction

The following equalities are held by the design of Algorithms 5 and 6:

$$\boldsymbol{Q}^{(0)} = \hat{\boldsymbol{Q}}^{(0)} = \boldsymbol{W}^{(0)} = \hat{\boldsymbol{W}}^{(0)}. \tag{11}$$

$$\boldsymbol{\omega}^{(j)} = \mathbf{e}_j^\top \boldsymbol{W}^{(j-1)}, \quad 1 \le j \le c. \tag{12}$$

Algorithm 5: GPTQ Original (Front-to-Back)

```
Input: $\boldsymbol{W}, \boldsymbol{S}, \boldsymbol{X}, \lambda, \mathbb{Z}^\dagger$
Output: $\boldsymbol{Z}, \boldsymbol{Q}$
$\boldsymbol{H} \leftarrow \boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}$
$\boldsymbol{L} \leftarrow \mathrm{LDL}\left(\boldsymbol{H}^{-1}\right)$
$\boldsymbol{W}^{(0)} \leftarrow \boldsymbol{W}$
$\boldsymbol{Q}^{(0)}, \boldsymbol{Z}^{(0)} \leftarrow \boldsymbol{W}^{(0)}, \mathbf{0}$
for $j \leftarrow 1$ to $c$ do
    $\boldsymbol{\omega}^{(j)} \leftarrow \mathbf{e}_j^\top \boldsymbol{W}^{(j-1)}$
    $\boldsymbol{\zeta}^{(j)} \leftarrow \boldsymbol{\omega}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right)^{-1}$
    $\boldsymbol{Z}^{(j)} \leftarrow \boldsymbol{Z}^{(j-1)} + \mathbf{e}_j \left( \textsc{Round}\left(\boldsymbol{\zeta}^{(j)}, \mathbb{Z}^\dagger\right) - \mathbf{e}_j^\top \boldsymbol{Z}^{(j-1)}\right)$
    $\boldsymbol{Q}^{(j)} \leftarrow \boldsymbol{Q}^{(j-1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j-1)}\right)$
    $\boldsymbol{\varepsilon}^{(j)} \leftarrow \mathbf{e}_j^\top \boldsymbol{Q}^{(j)} - \boldsymbol{\omega}^{(j)}$
    $\boldsymbol{W}^{(j)} \leftarrow \boldsymbol{W}^{(j-1)} + \boldsymbol{L} \mathbf{e}_j \boldsymbol{\varepsilon}^{(j)}$
end
$\boldsymbol{Z}, \boldsymbol{Q} \leftarrow \boldsymbol{Z}^{(c)}, \boldsymbol{Q}^{(c)}$
```

Algorithm 6: GPTQ Type-2 (Front-to-Back)

```
Input: $\boldsymbol{W}, \boldsymbol{S}, \boldsymbol{X}, \lambda, \mathbb{Z}^\dagger$
Output: $\boldsymbol{Z}, \boldsymbol{Q}$
$\boldsymbol{H} \leftarrow \boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}$
$\boldsymbol{L} \leftarrow \mathrm{LDL}\left(\boldsymbol{H}^{-1}\right)$
$\boldsymbol{W}^{(0)} \leftarrow \boldsymbol{W}$
$\boldsymbol{Q}^{(0)}, \boldsymbol{Z}^{(0)} \leftarrow \boldsymbol{W}^{(0)}, \mathbf{0}$
for $j \leftarrow 1$ to $c$ do
    $\boldsymbol{\omega}^{(j)} \leftarrow \mathbf{e}_j^\top \boldsymbol{W}^{(j-1)}$
    $\boldsymbol{\zeta}^{(j)} \leftarrow \boldsymbol{\omega}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right)^{-1}$
    $\boldsymbol{Z}^{(j)} \leftarrow \boldsymbol{Z}^{(j-1)} + \mathbf{e}_j \left( \textsc{Round}\left(\boldsymbol{\zeta}^{(j)}, \mathbb{Z}^\dagger\right) - \mathbf{e}_j^\top \boldsymbol{Z}^{(j-1)}\right)$
    $\boldsymbol{Q}^{(j)} \leftarrow \boldsymbol{Q}^{(j-1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j-1)}\right)$
    $\boldsymbol{\Delta}^{(j)} \leftarrow \boldsymbol{Q}^{(j)} - \boldsymbol{W}^{(0)}$ // new
    $\boldsymbol{W}^{(j)} \leftarrow \boldsymbol{W}^{(0)} - \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j)}$ // new
end
$\boldsymbol{Z}, \boldsymbol{Q} \leftarrow \boldsymbol{Z}^{(c)}, \boldsymbol{Q}^{(c)}$
```

Algorithm 7: GPTQ Type-2 (Back-to-Front)

```
Input: $\boldsymbol{W}, \boldsymbol{S}, \boldsymbol{X}, \lambda, \mathbb{Z}^\dagger$
Output: $\boldsymbol{Z}, \boldsymbol{Q}$
$\boldsymbol{H} \leftarrow \boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}$
$\boldsymbol{U} \leftarrow \mathrm{UDU}\left(\boldsymbol{H}^{-1}\right)$ // new
$\boldsymbol{W}^{(c+1)} \leftarrow \boldsymbol{W}$
$\boldsymbol{Q}^{(c+1)}, \boldsymbol{Z}^{(c+1)} \leftarrow \boldsymbol{W}^{(c+1)}, \mathbf{0}$
for $j \leftarrow c$ to 1 do
    $\boldsymbol{\omega}^{(j)} \leftarrow \mathbf{e}_j^\top \boldsymbol{W}^{(j+1)}$
    $\boldsymbol{\zeta}^{(j)} \leftarrow \boldsymbol{\omega}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right)^{-1}$
    $\boldsymbol{Z}^{(j)} \leftarrow \boldsymbol{Z}^{(j+1)} + \mathbf{e}_j \left( \textsc{Round}\left(\boldsymbol{\zeta}^{(j)}, \mathbb{Z}^\dagger\right) - \mathbf{e}_j^\top \boldsymbol{Z}^{(j+1)}\right)$
    $\boldsymbol{Q}^{(j)} \leftarrow \boldsymbol{Q}^{(j+1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j+1)}\right)$
    $\boldsymbol{\Delta}^{(j)} \leftarrow \boldsymbol{Q}^{(j)} - \boldsymbol{W}^{(c+1)}$
    $\boldsymbol{W}^{(j)} \leftarrow \boldsymbol{W}^{(c+1)} - \boldsymbol{U}^{-1} \boldsymbol{\Delta}^{(j)}$ // new
end
$\boldsymbol{Z}, \boldsymbol{Q} \leftarrow \boldsymbol{Z}^{(1)}, \boldsymbol{Q}^{(1)}$
```

Algorithm 8: Babai-Quantize (Default Order)

```
Input: $\boldsymbol{W}, \boldsymbol{S}, \boldsymbol{X}, \lambda, \mathbb{Z}^\dagger$
Output: $\boldsymbol{Z}, \boldsymbol{Q}$
$\boldsymbol{H} \leftarrow \boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}$
$\boldsymbol{A} \leftarrow \textsc{Cholesky}\left(\boldsymbol{H}\right)^\top$
$\boldsymbol{Y}^{(c+1)}, \boldsymbol{Q}^{(c+1)}, \boldsymbol{Z}^{(c+1)} \leftarrow \boldsymbol{A}\boldsymbol{W}, \boldsymbol{W}, \mathbf{0}$
for $j \leftarrow c$ to 1 do
    $\boldsymbol{\omega}^{(j)} \leftarrow \frac{\mathbf{e}_j^\top \boldsymbol{Y}^{(j+1)}}{\mathbf{e}_j^\top \boldsymbol{A} \mathbf{e}_j}$
    $\boldsymbol{\zeta}^{(j)} \leftarrow \boldsymbol{\omega}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right)^{-1}$
    $\boldsymbol{Z}^{(j)} \leftarrow \boldsymbol{Z}^{(j+1)} + \mathbf{e}_j \left(\textsc{Round}\left(\boldsymbol{\zeta}^{(j)}, \mathbb{Z}^\dagger\right) - \mathbf{e}_j^\top \boldsymbol{Z}^{(j+1)}\right)$
    $\boldsymbol{Q}^{(j)} \leftarrow \boldsymbol{Q}^{(j+1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j+1)}\right)$
    $\boldsymbol{Y}^{(j)} \leftarrow \boldsymbol{Y}^{(j+1)} - \boldsymbol{A}\mathbf{e}_j \mathbf{e}_j^\top \boldsymbol{Q}^{(j)}$
end
$\boldsymbol{Z}, \boldsymbol{Q} \leftarrow \boldsymbol{Z}^{(1)}, \boldsymbol{Q}^{(1)}$
```

$$\hat{\boldsymbol{\omega}}^{(j)} = \mathbf{e}_j^\top \hat{\boldsymbol{W}}^{(j-1)}, \quad 1 \le j \le c. \tag{13}$$

$$\boldsymbol{Q}^{(j)} = \boldsymbol{Q}^{(j-1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j \right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j-1)} \right), \quad 1 \le j \le c. \tag{14}$$

$$\hat{\boldsymbol{Q}}^{(j)} = \hat{\boldsymbol{Q}}^{(j-1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \hat{\boldsymbol{Z}}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j \right) - \mathbf{e}_j^\top \hat{\boldsymbol{Q}}^{(j-1)} \right), \quad 1 \le j \le c. \tag{15}$$

$$\boldsymbol{\varepsilon}^{(j)} = \mathbf{e}_j^\top \boldsymbol{Q}^{(j)} - \boldsymbol{\omega}^{(j)}, \quad 1 \le j \le c. \tag{16}$$

$$\boldsymbol{\Delta}^{(j)} = \hat{\boldsymbol{Q}}^{(j)} - \hat{\boldsymbol{W}}^{(0)}, \quad 1 \le j \le c. \tag{17}$$

$$\boldsymbol{W}^{(j)} = \boldsymbol{W}^{(j-1)} + \boldsymbol{L}\mathbf{e}_j \boldsymbol{\varepsilon}^{(j)}, \quad 1 \le j \le c. \tag{18}$$

$$\hat{\boldsymbol{W}}^{(j)} = \hat{\boldsymbol{W}}^{(0)} - \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j)}, \quad 1 \le j \le c. \tag{19}$$

Extend the definition of $\boldsymbol{\Delta}^{(j)}$ (Eq. 17) for $j = 0$,

$$\boldsymbol{\Delta}^{(j)} = \hat{\boldsymbol{Q}}^{(j)} - \hat{\boldsymbol{W}}^{(0)}, \quad 0 \le j \le c. \tag{20}$$

Then we have $\boldsymbol{\Delta}^{(0)} = \hat{\boldsymbol{Q}}^{(0)} - \hat{\boldsymbol{W}}^{(0)} = \hat{\boldsymbol{W}}^{(0)} - \hat{\boldsymbol{W}}^{(0)} = \mathbf{0}$, so that Eq. 19 can also be extended for $j = 0$,

$$\hat{\boldsymbol{W}}^{(j)} = \hat{\boldsymbol{W}}^{(0)} - \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j)}, \quad 0 \le j \le c. \tag{21}$$

(1) Eq. 8 holds for $j = 1$:

Using Eqs. 11, 12, 13,

$$\boldsymbol{\omega}^{(1)} = \mathbf{e}_1^\top \boldsymbol{W}^{(0)} = \mathbf{e}_1^\top \hat{\boldsymbol{W}}^{(0)} = \hat{\boldsymbol{\omega}}^{(1)}. \tag{22}$$

(2) Assume Eq. 8 holds for all $j \le j_*$, $1 \le j_* < c$.

Because $\boldsymbol{L}$ is a lower triangular matrix with all diagonal elements being 1, $\boldsymbol{L}^{-1}$ is also a lower triangular matrix with all diagonal elements being 1.

For $1 \le j < k \le c$,

$$\mathbf{e}_j^\top \boldsymbol{L} \mathbf{e}_k = \mathbf{e}_j^\top \boldsymbol{L}^{-1} \mathbf{e}_k = 0. \tag{23}$$

For $1 \le j \le c$,

$$\mathbf{e}_j^\top \boldsymbol{L} \mathbf{e}_j = \mathbf{e}_j^\top \boldsymbol{L}^{-1} \mathbf{e}_j = 1. \tag{24}$$

For $1 \leq j < c$,

$$\begin{aligned}
&\mathbf{e}_{j+1}^\top \boldsymbol{L} \left(\sum_{k=1}^{j} \mathbf{e}_k \mathbf{e}_k^\top \right) \\
=&\mathbf{e}_{j+1}^\top \boldsymbol{L} \left(\left(\sum_{k=1}^{c} \mathbf{e}_k \mathbf{e}_k^\top \right) - \mathbf{e}_{j+1}\mathbf{e}_{j+1}^\top - \left(\sum_{k=j+2}^{c} \mathbf{e}_k \mathbf{e}_k^\top \right) \right) \\
=&\mathbf{e}_{j+1}^\top \boldsymbol{L} \left(\sum_{k=1}^{j+1} \mathbf{e}_k \mathbf{e}_k^\top \right) - \mathbf{e}_c^\top \boldsymbol{L} \mathbf{e}_{j+1}\mathbf{e}_{j+1}^\top - \mathbf{e}_{j+1}^\top \boldsymbol{L} \left(\sum_{k=j+2}^{c} \mathbf{e}_k \mathbf{e}_k^\top \right) \\
=&\mathbf{e}_{j+1}^\top \boldsymbol{L}\mathbf{I} - \mathbf{e}_{j+1}^\top - \left(\sum_{k=j+2}^{c} \mathbf{e}_{j+1}^\top \boldsymbol{L} \mathbf{e}_k \mathbf{e}_k^\top \right) && \text{(Eq. 24)} \\
=&\mathbf{e}_{j+1}^\top \boldsymbol{L} - \mathbf{e}_{j+1}^\top - \left(\sum_{k=j+2}^{c} 0 \mathbf{e}_k^\top \right) && \text{(Eq. 23)} \\
=&\mathbf{e}_{j+1}^\top \left(\boldsymbol{L} - \mathbf{I} \right).
\end{aligned} \tag{25}$$

With Eq. 14, for $1 \leq j \leq c, 1 \leq k \leq c$ and $j \neq k$,

$$\begin{aligned}
\mathbf{e}_k^\top \boldsymbol{Q}^{(j)} =&\mathbf{e}_k^\top \left(\boldsymbol{Q}^{(j-1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j \right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j-1)} \right) \right) \\
=&\mathbf{e}_k^\top \boldsymbol{Q}^{(j-1)} + \mathbf{e}_k^\top \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j \right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j-1)} \right) \\
=&\mathbf{e}_k^\top \boldsymbol{Q}^{(j-1)} + 0 \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j \right) - \mathbf{e}_j^\top \boldsymbol{Q}^{(j-1)} \right) \\
=&\mathbf{e}_k^\top \boldsymbol{Q}^{(j-1)}.
\end{aligned} \tag{26}$$

Recursively applying Eq. 26, for $1 \leq j \leq c, 1 \leq k \leq c$,

$$\mathbf{e}_k^\top \boldsymbol{Q}^{(j)} = \begin{cases} \mathbf{e}_k^\top \boldsymbol{Q}^{(k)} & \text{if } 1 \leq k \leq j \leq c, \\ \mathbf{e}_k^\top \boldsymbol{Q}^{(0)} = \mathbf{e}_k^\top \boldsymbol{W}^{(0)} & \text{if } 1 \leq j < k \leq c. \end{cases} \tag{27}$$

Similar to Eq. 27, with Eq. 15, for $1 \leq j \leq c, 1 \leq k \leq c$,

$$\mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(j)} = \begin{cases} \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(k)} & \text{if } 1 \leq k \leq j \leq c, \\ \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(0)} = \mathbf{e}_k^\top \hat{\boldsymbol{W}}^{(0)} & \text{if } 1 \leq j < k \leq c. \end{cases} \tag{28}$$

With Eq. 28, for $1 \leq j \leq c, 1 \leq k \leq c$,

$$\begin{aligned}
\mathbf{e}_k^\top \boldsymbol{\Delta}^{(j)} =&\mathbf{e}_k^\top \left(\hat{\boldsymbol{Q}}^{(j)} - \hat{\boldsymbol{W}}^{(0)} \right) && \text{(Eq. 20)} \\
=&\mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(j)} - \mathbf{e}_k^\top \hat{\boldsymbol{W}}^{(0)} \\
=&\begin{cases} \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(k)} - \mathbf{e}_k^\top \hat{\boldsymbol{W}}^{(0)} = \mathbf{e}_k^\top \boldsymbol{\Delta}^{(k)} & \text{if } 1 \leq k \leq j \leq c, \\ \mathbf{e}_k^\top \hat{\boldsymbol{W}}^{(0)} - \mathbf{e}_k^\top \hat{\boldsymbol{W}}^{(0)} = \mathbf{e}_k^\top \boldsymbol{\Delta}^{(0)} = \mathbf{0} & \text{if } 1 \leq j < k \leq c. \end{cases}
\end{aligned} \tag{29}$$

For $1 \leq k \leq j \leq c$,

$$\begin{aligned}
&\mathbf{e}_k^\top \boldsymbol{L}\boldsymbol{\Delta}^{(j)} \\
=&\mathbf{e}_k^\top \boldsymbol{L}\mathbf{I}\boldsymbol{\Delta}^{(j)} \\
=&\mathbf{e}_k^\top \boldsymbol{L}\left(\sum_{k'=1}^{c} \mathbf{e}_{k'}\mathbf{e}_{k'}^\top\right)\boldsymbol{\Delta}^{(j)} \\
=&\sum_{k'=1}^{c} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(j)} \\
=&\left(\sum_{k'=1}^{k} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(j)}\right)+\left(\sum_{k'=k+1}^{c} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(j)}\right) \\
=&\left(\sum_{k'=1}^{k} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(k')}\right)+\left(\sum_{k'=k+1}^{c} 0\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(j)}\right) && \text{(Eqs. 23, 29)} \\
=&\left(\sum_{k'=1}^{k} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(k)}\right)+\left(\sum_{k'=k+1}^{c} 0\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(k)}\right) && \text{(Eq. 29)} \\
=&\left(\sum_{k'=1}^{k} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(k)}\right)+\left(\sum_{k'=k+1}^{c} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(k)}\right) && \text{(Eq. 23)} \\
=&\sum_{k'=1}^{c} \mathbf{e}_k^\top \boldsymbol{L}\mathbf{e}_{k'}\mathbf{e}_{k'}^\top\boldsymbol{\Delta}^{(k)} \\
=&\mathbf{e}_k^\top \boldsymbol{L}\left(\sum_{k'=1}^{c} \mathbf{e}_{k'}\mathbf{e}_{k'}^\top\right)\boldsymbol{\Delta}^{(k)} \\
=&\mathbf{e}_k^\top \boldsymbol{L}\mathbf{I}\boldsymbol{\Delta}^{(k)} \\
=&\mathbf{e}_k^\top \boldsymbol{L}\boldsymbol{\Delta}^{(k)}.
\end{aligned} \tag{30}$$

For $1 \leq j \leq c$,

$$\begin{aligned}
&\mathbf{e}_j^\top \boldsymbol{L}^{-1}\boldsymbol{\Delta}^{(j-1)} \\
=&\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{I}\boldsymbol{\Delta}^{(j-1)} \\
=&\mathbf{e}_j^\top \boldsymbol{L}^{-1}\left(\sum_{k=1}^{c}\mathbf{e}_k\mathbf{e}_k^\top\right)\boldsymbol{\Delta}^{(j-1)} \\
=&\sum_{k=1}^{c}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j-1)} \\
=&\left(\sum_{k=1}^{j-1}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j-1)}\right)+\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_j\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j-1)}+\left(\sum_{k=j+1}^{c}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j-1)}\right) \\
=&\left(\sum_{k=1}^{j-1}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j-1)}\right)+\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_j\mathbf{0}+\left(\sum_{k=j+1}^{c}0\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j-1)}\right) && \text{(Eqs. 23, 29)} \\
=&\left(\sum_{k=1}^{j-1}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j-1)}\right)+\left(\sum_{k=j+1}^{c}0\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j-1)}\right)+\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j)}-\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j)} \\
=&\left(\sum_{k=1}^{j-1}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j)}\right)+\left(\sum_{k=j+1}^{c}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j)}\right)+\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_j\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j)}-\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j)} && \text{(Eqs. 24, 29)} \\
=&\left(\sum_{k=1}^{c}\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{e}_k\mathbf{e}_k^\top\boldsymbol{\Delta}^{(j)}\right)-\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j)} \\
=&\mathbf{e}_j^\top \boldsymbol{L}^{-1}\left(\sum_{k=1}^{c}\mathbf{e}_k\mathbf{e}_k^\top\right)\boldsymbol{\Delta}^{(j)}-\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j)} \\
=&\mathbf{e}_j^\top \boldsymbol{L}^{-1}\mathbf{I}\boldsymbol{\Delta}^{(j)}-\mathbf{e}_j^\top\boldsymbol{\Delta}^{(j)} \\
=&\mathbf{e}_j^\top\left(\boldsymbol{L}^{-1}-\mathbf{I}\right)\boldsymbol{\Delta}^{(j)}.
\end{aligned} \tag{31}$$

According to the assumption, for $1 \leq k \leq j_* < c$, we have

$$\mathbf{e}_k^\top \boldsymbol{W}^{(k-1)} = \boldsymbol{\omega}^{(k)} = \hat{\boldsymbol{\omega}}^{(k)} = \mathbf{e}_k^\top \hat{\boldsymbol{W}}^{(k-1)} \tag{32}$$

and

$$\boldsymbol{Q}^{(k)} = \hat{\boldsymbol{Q}}^{(k)}. \tag{33}$$

For $1 \leq k \leq j_*$,

$$\begin{aligned}
\boldsymbol{\varepsilon}^{(k)} =& \mathbf{e}_k^\top \boldsymbol{Q}^{(k)} - \boldsymbol{\omega}^{(k)} && \text{(Eq. 16)} \\
=& \mathbf{e}_k^\top \boldsymbol{Q}^{(k)} - \mathbf{e}_k^\top \boldsymbol{W}^{(k-1)} \\
=& \mathbf{e}_k^\top \left(\boldsymbol{Q}^{(k)} - \boldsymbol{W}^{(k-1)} \right) \\
=& \mathbf{e}_k^\top \left(\hat{\boldsymbol{Q}}^{(k)} - \hat{\boldsymbol{W}}^{(k-1)} \right) && \text{(Eqs. 32, 33)} \\
=& \mathbf{e}_k^\top \left(\hat{\boldsymbol{Q}}^{(k)} - \left(\hat{\boldsymbol{W}}^{(0)} - \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(k-1)} \right) \right) && \text{(Eq. 21)} \\
=& \mathbf{e}_k^\top \left(\left(\hat{\boldsymbol{Q}}^{(k)} - \hat{\boldsymbol{W}}^{(0)} \right) + \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(k-1)} \right) \\
=& \mathbf{e}_k^\top \left(\boldsymbol{\Delta}^{(k)} + \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(k-1)} \right) && \text{(Eq. 20)} \\
=& \mathbf{e}_k^\top \left(\boldsymbol{\Delta}^{(k)} + \left(\boldsymbol{L}^{-1} - \mathbf{I} \right) \boldsymbol{\Delta}^{(k)} \right) && \text{(Eq. 31)} \\
=& \mathbf{e}_k^\top \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(k)} \\
=& \mathbf{e}_k^\top \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j_*)} && \text{(Eq. 30).}
\end{aligned} \tag{34}$$

$$\begin{aligned}
\boldsymbol{\omega}^{(j_*+1)} =& \mathbf{e}_{j_*+1}^\top \boldsymbol{W}^{(j_*)} && \text{(Eq. 12)} \\
=& \mathbf{e}_{j_*+1}^\top \left(\boldsymbol{W}^{(j_*-1)} + \boldsymbol{L} \mathbf{e}_{j_*} \boldsymbol{\varepsilon}^{(j_*)} \right) && \text{(Eq. 18)} \\
=& \mathbf{e}_{j_*+1}^\top \left(\boldsymbol{W}^{(0)} + \left(\sum_{k=1}^{j_*} \boldsymbol{L} \mathbf{e}_k \boldsymbol{\varepsilon}^{(k)} \right) \right) && \text{(Eq. 18)} \\
=& \mathbf{e}_{j_*+1}^\top \left(\hat{\boldsymbol{W}}^{(0)} + \left(\sum_{k=1}^{j_*} \boldsymbol{L} \mathbf{e}_k \mathbf{e}_k^\top \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j_*)} \right) \right) && \text{(Eq. 34)} \\
=& \mathbf{e}_{j_*+1}^\top \left(\hat{\boldsymbol{W}}^{(0)} + \boldsymbol{L} \left(\sum_{k=1}^{j_*} \mathbf{e}_k \mathbf{e}_k^\top \right) \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j_*)} \right) \\
=& \mathbf{e}_{j_*+1}^\top \left(\hat{\boldsymbol{W}}^{(0)} + \left(\boldsymbol{L} - \mathbf{I} \right) \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j_*)} \right) && \text{(Eq. 25)} \\
=& \mathbf{e}_{j_*+1}^\top \left(\hat{\boldsymbol{W}}^{(0)} - \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j_*)} + \boldsymbol{\Delta}^{(j_*)} \right) \\
=& \mathbf{e}_{j_*+1}^\top \left(\hat{\boldsymbol{W}}^{(0)} - \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j_*)} + \mathbf{0} \right) && \text{(Eq. 29)} \\
=& \mathbf{e}_{j_*+1}^\top \left(\hat{\boldsymbol{W}}^{(0)} - \boldsymbol{L}^{-1} \boldsymbol{\Delta}^{(j_*)} \right) \\
=& \mathbf{e}_{j_*+1}^\top \hat{\boldsymbol{W}}^{(j_*)} && \text{(Eq. 21)} \\
=& \hat{\boldsymbol{\omega}}^{(j_*+1)} && \text{(Eq. 13).}
\end{aligned} \tag{35}$$

Eq. 8 holds for $j = j_* + 1$. ■

F.2. Step 2

Algorithm 7 (back-to-front order) is generated by reversing the iteration direction of Algorithm 6. Besides changing the direction of the index j, we also need to change the LDL decomposition to a so-called "UDU" decomposition so that the error propagation is correctly applied to the not-yet-quantized weights in the front dimensions.

Justification

Let $\mathbf{P}$ be the anti-diagonal permutation matrix with $\mathbf{P} = \mathbf{P}^\top = \mathbf{P}^{-1}$. Let $\hat{\boldsymbol{L}}$ be the LDL decomposition of the permuted Hessian inverse $\mathbf{P}\boldsymbol{H}^{-1}\mathbf{P} = \hat{\boldsymbol{L}}\hat{\boldsymbol{D}}_{\mathrm{L}}^{\frac{1}{2}}\hat{\boldsymbol{D}}_{\mathrm{L}}^{\frac{1}{2}}\hat{\boldsymbol{L}}^\top$ where $\hat{\boldsymbol{L}}$ is a lower triangular matrix with all diagonal elements being 1, and $\hat{\boldsymbol{D}}_{\mathrm{L}}^{\frac{1}{2}}$ is a non-negative diagonal matrix.

Since we are changing the iteration direction instead of applying the permutation, we permute the matrix $\hat{\boldsymbol{L}}$ back, yielding $\boldsymbol{U} = \mathbf{P}\hat{\boldsymbol{L}}\mathbf{P}$. Alternatively, $\boldsymbol{U}$ can be calculated using the decomposition $\boldsymbol{H}^{-1} = \mathbf{P}\hat{\boldsymbol{L}}\mathbf{P}\mathbf{P}\hat{\boldsymbol{D}}_{\mathrm{L}}^{\frac{1}{2}}\mathbf{P}\mathbf{P}\hat{\boldsymbol{D}}_{\mathrm{L}}^{\frac{1}{2}}\mathbf{P}\mathbf{P}\hat{\boldsymbol{L}}^\top\mathbf{P} = \boldsymbol{U}\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{U}^\top$ where $\boldsymbol{U}$ is an upper triangular matrix with all diagonal elements being 1, and $\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}} = \mathbf{P}\hat{\boldsymbol{D}}_{\mathrm{L}}^{\frac{1}{2}}\mathbf{P}$ is a non-negative diagonal matrix.

The decomposition to calculate $\boldsymbol{U}$ from $\boldsymbol{H}^{-1}$ is what we call "UDU" decomposition, which can be considered as a variant of the LDL decomposition.

F.3. Step 3

To distinguish the variables using the same symbol in Algorithms 7 and 8, we use symbols with ˆ to denote the symbols in Algorithm 7, and use the symbols with ˜ for Algorithm 8.

We have the Cholesky decomposition of $\boldsymbol{H}$: $\boldsymbol{H} = \left(\boldsymbol{H}^{-1}\right)^{-1} = \left(\boldsymbol{U}\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{U}^\top\right)^{-1} = \left(\boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}\right)^\top \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}$, so that $\boldsymbol{A} = \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}$.

Claim

$$\hat{\boldsymbol{\omega}}_j = \tilde{\boldsymbol{\omega}}_j, \quad 1 \le j \le c, \tag{36}$$

and consequently,

$$\hat{\boldsymbol{Z}}^{(j)} = \tilde{\boldsymbol{Z}}^{(j)}, \quad 1 \le j \le c+1, \tag{37}$$

and

$$\hat{\boldsymbol{Q}}^{(j)} = \tilde{\boldsymbol{Q}}^{(j)}, \quad 1 \le j \le c+1. \tag{38}$$

Proof Eq. 36 by Induction

For $1 \le j \le c$,

$$\begin{aligned}
\tilde{\boldsymbol{\omega}}^{(j)} =& \frac{\mathbf{e}_j^\top \boldsymbol{Y}^{(j+1)}}{\mathbf{e}_j^\top \boldsymbol{A}\mathbf{e}_j} \\
=& \frac{\mathbf{e}_j^\top \boldsymbol{Y}^{(j+1)}}{\mathbf{e}_j^\top \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}\mathbf{e}_j} \\
=& \frac{\mathbf{e}_j^\top \boldsymbol{Y}^{(j+1)}}{\boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}[j,j]} \\
=& \boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}[j,j]\mathbf{e}_j^\top \boldsymbol{Y}^{(j+1)} \\
=& \mathbf{e}_j^\top \boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{Y}^{(j+1)}.
\end{aligned} \tag{39}$$

The following equalities are held by the design of Algorithms 6 and 8:

$$\hat{\boldsymbol{Q}}^{(c+1)} = \tilde{\boldsymbol{Q}}^{(c+1)} = \hat{\boldsymbol{W}}^{(c+1)} = \tilde{\boldsymbol{W}}. \tag{40}$$

$$\boldsymbol{Y}^{(c+1)} = \boldsymbol{A}\tilde{\boldsymbol{W}} = \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}\tilde{\boldsymbol{W}}. \tag{41}$$

$$\hat{\boldsymbol{\omega}}^{(j)} = \mathbf{e}_j^\top \hat{\boldsymbol{W}}^{(j+1)}, \quad 1 \le j \le c. \tag{42}$$

$$\hat{\boldsymbol{Q}}^{(j)} = \hat{\boldsymbol{Q}}^{(j+1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \tilde{\boldsymbol{Z}}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \hat{\boldsymbol{Q}}^{(j+1)}\right), \quad 1 \le j \le c. \tag{43}$$

$$\tilde{\boldsymbol{Q}}^{(j)} = \tilde{\boldsymbol{Q}}^{(j+1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \tilde{\boldsymbol{Z}}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \tilde{\boldsymbol{Q}}^{(j+1)}\right), \quad 1 \le j \le c. \tag{44}$$

$$\boldsymbol{\Delta}^{(j)} = \hat{\boldsymbol{Q}}^{(j)} - \hat{\boldsymbol{W}}^{(c+1)}, \quad 1 \le j \le c. \tag{45}$$

$$\hat{\boldsymbol{W}}^{(j)} = \hat{\boldsymbol{W}}^{(c+1)} - \boldsymbol{U}^{-1}\boldsymbol{\Delta}^{(j)}, \quad 1 \le j \le c. \tag{46}$$

$$\boldsymbol{Y}^{(j)} = \boldsymbol{Y}^{(j+1)} - \boldsymbol{A}\mathbf{e}_j\mathbf{e}_j^\top \tilde{\boldsymbol{Q}}^{(j)} = \boldsymbol{Y}^{(j+1)} - \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}\mathbf{e}_j\mathbf{e}_j^\top \tilde{\boldsymbol{Q}}^{(j)}, \quad 1 \le j \le c. \tag{47}$$

Because $\boldsymbol{U}$ is an upper triangular matrix with all diagonal elements being 1, $\boldsymbol{U}^{-1}$ is also an upper triangular matrix with all diagonal elements being 1.

For $1 \le k < j \le c$,

$$\mathbf{e}_j^\top \boldsymbol{U}\mathbf{e}_k = \mathbf{e}_j^\top \boldsymbol{U}^{-1}\mathbf{e}_k = 0. \tag{48}$$

$$\mathbf{e}_c^\top \boldsymbol{U} = \mathbf{e}_c^\top. \tag{49}$$

For $1 \le j \le c$,

$$\mathbf{e}_j^\top \boldsymbol{U}\mathbf{e}_j = \mathbf{e}_j^\top \boldsymbol{U}^{-1}\mathbf{e}_j = 1. \tag{50}$$

(1) Eq. 36 holds for $j = c$:

Using Eqs. 39, 40, 41, 42, 49,

$$\tilde{\boldsymbol{\omega}}^{(c)} = \mathbf{e}_c^\top \boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{Y}^{(c+1)} = \mathbf{e}_c^\top \boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}\tilde{\boldsymbol{W}} = \mathbf{e}_c^\top \boldsymbol{U}^{-1}\tilde{\boldsymbol{W}} = \mathbf{e}_c^\top \tilde{\boldsymbol{W}} = \mathbf{e}_c^\top \hat{\boldsymbol{W}}^{(c+1)} = \hat{\boldsymbol{\omega}}^{(c)}. \tag{51}$$

(2) Assume Eq. 36 holds for all $j \ge j_*$, $1 < j_* \le c$.

With Eq. 43, for $1 \le j \le c, 1 \le k \le c$ and $j \ne k$,

$$\begin{aligned}
\mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(j)} =& \mathbf{e}_k^\top \left(\hat{\boldsymbol{Q}}^{(j+1)} + \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \hat{\boldsymbol{Q}}^{(j+1)}\right)\right) \\
=& \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(j+1)} + \mathbf{e}_k^\top \mathbf{e}_j \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \hat{\boldsymbol{Q}}^{(j+1)}\right) \\
=& \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(j+1)} + 0 \left(\mathbf{e}_j^\top \boldsymbol{Z}^{(j)} \operatorname{diag}\left(\boldsymbol{S}^\top \mathbf{e}_j\right) - \mathbf{e}_j^\top \hat{\boldsymbol{Q}}^{(j+1)}\right) \\
=& \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(j+1)}.
\end{aligned} \tag{52}$$

Recursively applying Eq. 52, for $1 \le j \le c, 1 \le k \le c$,

$$\mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(j)} = \begin{cases} \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(k)} & \text{if } 1 \le j \le k \le c, \\ \mathbf{e}_k^\top \hat{\boldsymbol{Q}}^{(c+1)} = \mathbf{e}_k^\top \hat{\boldsymbol{W}}^{(c+1)} & \text{if } 1 \le k < j \le c. \end{cases} \tag{53}$$

Similar to Eq. 53, with Eq. 44, for $1 \leq j \leq c, 1 \leq k \leq c$,

$$\mathbf{e}_k^\top \tilde{\boldsymbol{Q}}^{(j)} = \begin{cases} \mathbf{e}_k^\top \tilde{\boldsymbol{Q}}^{(k)} & \text{if } 1 \leq j \leq k \leq c, \\ \mathbf{e}_k^\top \tilde{\boldsymbol{Q}}^{(c+1)} = \mathbf{e}_k^\top \tilde{\boldsymbol{W}} & \text{if } 1 \leq k < j \leq c. \end{cases} \tag{54}$$

For $1 \leq j \leq c$,

$$\begin{aligned} \boldsymbol{Y}^{(j)} =& \boldsymbol{Y}^{(j+1)} - \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}} \boldsymbol{U}^{-1} \mathbf{e}_j \mathbf{e}_j^\top \tilde{\boldsymbol{Q}}^{(j)} && \text{(Eq. 47)} \\ =& \boldsymbol{Y}^{(c+1)} - \left(\sum_{k=j}^{c} \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}} \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \tilde{\boldsymbol{Q}}^{(k)} \right) && \text{(Eq. 47)} \\ =& \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}} \boldsymbol{U}^{-1} \tilde{\boldsymbol{W}} - \left(\sum_{k=j}^{c} \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}} \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \tilde{\boldsymbol{Q}}^{(j)} \right) && \text{(Eq. 41)} \\ =& \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}} \boldsymbol{U}^{-1} \left(\tilde{\boldsymbol{W}} - \left(\sum_{k=j}^{c} \mathbf{e}_k \mathbf{e}_k^\top \right) \tilde{\boldsymbol{Q}}^{(j)} \right) \end{aligned} \tag{55}$$

For $1 \leq j < c$,

$$
\begin{aligned}
\tilde{\boldsymbol{\omega}}^{(j)} =& \mathbf{e}_j^\top \boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}} \boldsymbol{Y}^{(j+1)} && \text{(Eq. 39)} \\
=& \mathbf{e}_j^\top \boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}} \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}} \boldsymbol{U}^{-1} \left(\tilde{\boldsymbol{W}} - \left(\sum_{k=j+1}^{c} \mathbf{e}_k \mathbf{e}_k^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} \right) && \text{(Eq. 55)} \\
=& \mathbf{e}_j^\top \boldsymbol{U}^{-1} \left(\tilde{\boldsymbol{W}} - \left(\sum_{k=j+1}^{c} \mathbf{e}_k \mathbf{e}_k^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} \right) \\
=& \mathbf{e}_j^\top \boldsymbol{U}^{-1} \tilde{\boldsymbol{W}} - \left(\sum_{k=j+1}^{c} \mathbf{e}_j^\top \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} \\
=& \mathbf{e}_j^\top \boldsymbol{U}^{-1} \tilde{\boldsymbol{W}} - \left(\left(\sum_{k=1}^{c} \mathbf{e}_j^\top \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \right) - \left(\sum_{k=1}^{j-1} \mathbf{e}_j^\top \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \right) - \mathbf{e}_j^\top \boldsymbol{U}^{-1} \mathbf{e}_j \mathbf{e}_j^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} \\
=& \mathbf{e}_j^\top \boldsymbol{U}^{-1} \tilde{\boldsymbol{W}} - \left(\left(\sum_{k=1}^{c} \mathbf{e}_j^\top \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \right) - \left(\sum_{k=1}^{j-1} 0 \mathbf{e}_k^\top \right) - 1 \mathbf{e}_j^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} && \text{(Eqs. 48, 50)} \\
=& \mathbf{e}_j^\top \boldsymbol{U}^{-1} \tilde{\boldsymbol{W}} - \left(\sum_{k=1}^{c} \mathbf{e}_j^\top \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} + \mathbf{e}_j^\top \tilde{\boldsymbol{Q}}^{(j+1)} \\
=& \mathbf{e}_j^\top \boldsymbol{U}^{-1} \tilde{\boldsymbol{W}} - \left(\sum_{k=1}^{c} \mathbf{e}_j^\top \boldsymbol{U}^{-1} \mathbf{e}_k \mathbf{e}_k^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} + \mathbf{e}_j^\top \tilde{\boldsymbol{W}} && \text{(Eq. 54)} \\
=& \mathbf{e}_j^\top \left(\tilde{\boldsymbol{W}} - \boldsymbol{U}^{-1} \left(\left(\sum_{k=1}^{c} \mathbf{e}_k \mathbf{e}_k^\top \right) \tilde{\boldsymbol{Q}}^{(j+1)} - \tilde{\boldsymbol{W}} \right) \right) \\
=& \mathbf{e}_j^\top \left(\tilde{\boldsymbol{W}} - \boldsymbol{U}^{-1} \left(\mathbf{I} \tilde{\boldsymbol{Q}}^{(j+1)} - \tilde{\boldsymbol{W}} \right) \right) \\
=& \mathbf{e}_j^\top \left(\tilde{\boldsymbol{W}} - \boldsymbol{U}^{-1} \left(\tilde{\boldsymbol{Q}}^{(j+1)} - \tilde{\boldsymbol{W}} \right) \right).
\end{aligned} \tag{56}
$$

Because $\mathbf{e}_c^\top \left(\tilde{\boldsymbol{W}} - \boldsymbol{U}^{-1} \left(\tilde{\boldsymbol{Q}}^{(c+1)} - \tilde{\boldsymbol{W}} \right) \right) = \mathbf{e}_c^\top \tilde{\boldsymbol{W}} = \tilde{\boldsymbol{\omega}}^{(c)}$, Eq. 56 can be extended for $j = c$,

$$
\tilde{\boldsymbol{\omega}}^{(j)} = \mathbf{e}_j^\top \left(\tilde{\boldsymbol{W}} - \boldsymbol{U}^{-1} \left(\tilde{\boldsymbol{Q}}^{(j+1)} - \tilde{\boldsymbol{W}} \right) \right), \quad 1 \leq j \leq c. \tag{57}
$$

According to the assumption, for $1 < j_* \leq k \leq c$, we have

$$
\hat{\boldsymbol{Q}}^{(k)} = \tilde{\boldsymbol{Q}}^{(k)}. \tag{58}
$$

$$\begin{aligned}\tilde{\boldsymbol{\omega}}^{(j_*-1)} =&\mathbf{e}_{j_*-1}^\top\left(\tilde{\boldsymbol{W}}-\boldsymbol{U}^{-1}\left(\tilde{\boldsymbol{Q}}^{(j_*)}-\tilde{\boldsymbol{W}}\right)\right) && \text{(Eq. 57)}\\ =&\mathbf{e}_{j_*-1}^\top\left(\hat{\boldsymbol{W}}^{(c+1)}-\boldsymbol{U}^{-1}\left(\hat{\boldsymbol{Q}}^{(j_*)}-\hat{\boldsymbol{W}}^{(c+1)}\right)\right) && \text{(Eq. 58)}\\ =&\mathbf{e}_{j_*-1}^\top\left(\hat{\boldsymbol{W}}^{(c+1)}-\boldsymbol{U}^{-1}\boldsymbol{\Delta}^{(j_*)}\right) && \text{(Eq. 45)}\\ =&\mathbf{e}_{j_*-1}^\top\hat{\boldsymbol{W}}^{(j_*)} && \text{(Eq. 46)}\\ =&\hat{\boldsymbol{\omega}}^{(j_*-1)} && \text{(Eq. 42).}\end{aligned} \tag{59}$$

Eq. 36 holds for $j = j_* - 1$. ■

F.4. Proof of ineffectiveness of additional GPTQ refinement on Babai's algorithm

We may try to apply further GPTQ updates in Babai's algorithm by changing Line 9 in Algorithm 8 to

$$\boldsymbol{Y'}^{(j)} \leftarrow \boldsymbol{Y}^{(j)} + \boldsymbol{AU}\mathbf{e}_j\boldsymbol{\varepsilon}^{(j)} = \boldsymbol{Y}^{(j+1)} - \boldsymbol{A}\mathbf{e}_j\mathbf{e}_j^\top\tilde{\boldsymbol{Q}}^{(j)} + \boldsymbol{AU}\mathbf{e}_j\boldsymbol{\varepsilon}^{(j)} \tag{60}$$

However, as $\boldsymbol{A} = \boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}$, the $\tilde{\boldsymbol{\omega}}^{(j-1)}$ remains the same:

$$\begin{aligned}\tilde{\boldsymbol{\omega}'}^{(j-1)} =&\mathbf{e}_{j-1}^\top\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{Y'}^{(j)} && \text{(Eq. 39)}\\ =&\mathbf{e}_{j-1}^\top\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\left(\boldsymbol{Y}^{(j)}+\boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}\boldsymbol{U}\mathbf{e}_j\boldsymbol{\varepsilon}^{(j)}\right)\\ =&\mathbf{e}_{j-1}^\top\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{Y}^{(j)}+\mathbf{e}_{j-1}^\top\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{D}_{\mathrm{U}}^{-\frac{1}{2}}\boldsymbol{U}^{-1}\boldsymbol{U}\mathbf{e}_j\boldsymbol{\varepsilon}^{(j)}\\ =&\mathbf{e}_{j-1}^\top\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{Y}^{(j)}+\mathbf{e}_{j-1}^\top\mathbf{e}_j\boldsymbol{\varepsilon}^{(j)}\\ =&\mathbf{e}_{j-1}^\top\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{Y}^{(j)}+0\boldsymbol{\varepsilon}^{(j)}\\ =&\mathbf{e}_{j-1}^\top\boldsymbol{D}_{\mathrm{U}}^{\frac{1}{2}}\boldsymbol{Y}^{(j)}\\ =&\tilde{\boldsymbol{\omega}}^{(j-1)} && \text{(Eq. 39).}\end{aligned} \tag{61}$$

■