Water-Filling RoPE: Rate-Distortion-Optimal Frequency Allocation for Position Encoding

Method. Water-Filling RoPE (WF-RoPE)

Motivation

RoPE (Rotary Position Embedding) is how most modern language models tell tokens apart by position: it rotates each token's query and key vectors by an angle proportional to the token's position, using a fixed list of rotation speeds called frequencies. Those frequencies are laid out as a single geometric sequence – each one a fixed fraction of the last – and almost every fix for long-context failure only stretches or shrinks this one sequence by a single knob. Position interpolation shrinks all angles equally; NTK and YaRN scaling shrink fast ones less than slow ones with a hand-tuned rule; and the recent MrRoPE (Mixed-radix RoPE) reinterprets position as a mixed-radix number and unifies these fixes, but still hands the model two pre-chosen schedules rather than computing the best one. So the only thing anyone ever tunes is the overall scale, never the shape of the frequency list itself.

This matters now because three recent results line up. MrRoPE (January 2026) explicitly opened up the per-dimension freedom – letting each frequency be set independently – but never said which setting is best. A phase-modulation analysis (February 2026) proved that the standard geometric frequencies break a Nyquist/aliasing limit at long context (the fast frequencies wrap around into angles the model never saw in training), but only recommended a bigger overall scale. And a 2025 diagnostic showed the slow frequencies are so spread out that some dimensions are effectively dead for long-range lookups. Together they say: the budget of 'rotation resolution' is badly spent, the freedom to respend it exists, and there is a hard limit each frequency must respect – but nobody has written down what 'best' means and solved for it.

Prior work stopped short for concrete reasons. MrRoPE had no objective function over the per-dimension assignment, so it picked schedules by hand. The phase-modulation paper kept the geometric shape fixed and only constrained one scalar. The dimension-inefficiency paper diagnosed the waste but offered no recipe to redistribute it. And DPE (Dimension-Wise Position Encoding manipulation) remapped dimensions with a heuristic detector rather than an optimization, so it could not certify its choice was best.

When the gap closes, extending a model's context stops being trial-and-error over one knob and becomes a solved calculation: given a target length and a corpus, you compute the provably-best per-dimension frequencies in seconds and swap them in without any training. The best grid turns out to be non-monotone – some middle frequencies get denser, some fast ones are retired – which the geometric sequence can never represent, and it comes with a certificate proving exactly how much better it is.

Method

Distortion objective

Turn 'which per-dimension frequency is best' into a measurable, corpus-grounded distortion the grid can be optimized against.

1. On your target corpus at the chosen maximum length L_{max}, run the model once and record, for every query token, how far back it actually looks — weighting each looked-at distance by how much attention it received — and collect this into a histogram of relative distances δ from 0 to L_{max}, normalized so it sums to 1. This histogram $P(\delta)$ tells you which token-to-token distances the model genuinely needs to distinguish. If you have no corpus to measure on, just assume every distance from 0 to L_{max} is equally likely.

   Why: The best frequencies depend on which distances the model actually has to tell apart, so you measure that first.

2. For each rotation frequency (each 2-D sub-space i), write down a formula D_i that says how much positional error this frequency contributes: take the ideal relative-position signal cos(distance × frequency) folded into one clean cycle, subtract what the frequency actually produces when distances are large, square the difference, and average it using the distance histogram $P(\delta)$ from the previous step. Also compute its curvature (the second derivative). A frequency scores low when it stays within one clean cycle for the distances that matter, and high when it wraps around into angles never seen in training. Since these are exact sums over the histogram, no random sampling is needed. *Per-dimension distortion D_i : the expected squared error in the relative-position term contributed by sub-space i over the corpus relative-distance distribution $P(\delta)$, with $curv_i$ its second derivative.*

$$D_i(\theta_i) = \mathbb{E}_{\delta\sim P}\big[(\cos(\delta\,\theta_i) - \widehat{\cos})^2\big], \qquad \mathrm{curv}_i = D_i''(\theta_i) \tag{1}$$

Why: This turns 'good frequency' into a number you can minimize instead of an intuition.

Constrained water-filling solver

Solve the convex allocation in closed form under per-dimension Nyquist boxes and a fixed total budget, yielding the non-monotone optimal grid.

3. Give each frequency a hard ceiling — its Nyquist limit — so it can never rotate fast enough to wrap past the angles the model saw in training: by default set every ceiling to π divided by the full target length L_{max}, or tighten each one to the distance that captures all but a tiny fraction (about 0.1%) of the histogram mass. Then set the total 'resolution budget' R to exactly what the model's original geometric frequencies already spend, so that when you re-solve for better frequencies you are only moving budget around, never adding more.

   Why: The cap enforces the aliasing limit per dimension, and fixing the budget means any improvement comes from respending, not from spending more.

4. Find the best frequencies by reverse water-filling: there is a single tuning number λ; for each frequency set its value by the clipped square-root rule and cap it at that frequency's ceiling, then adjust λ up or down by binary search until the frequencies together spend exactly the budget R. Bracket λ between 'too small, budget unspent' and 'too large, all frequencies at their ceilings', and bisect until the budget is matched to high precision (~50 steps). You must decide how to weight each frequency's error w_i. 【author decision: weight every frequency equally (the simplest training-free default), or weight by how much attention energy each carries, or use per-attention-head weights for a head-specific grid — each gives a different optimum.】 *The allocation program: minimize budget-weighted total distortion subject to a per-dimension Nyquist box and a total angular-resolution budget R that makes the geometric grid one feasible point.*

$$\min_{\theta} \sum_i w_i D_i(\theta_i) \quad \text{s.t.} \quad \theta_i \le \frac{\pi}{L_{\max}\text{-eff}_i}, \quad \sum_i \log \tfrac{1}{\theta_i} \le R \tag{2}$$

   Reverse-water-filling closed form: each sub-space's optimal frequency is set by a single Lagrange multiplier λ (solved by 1-D bisection to meet budget R), clipped to its Nyquist box.

$$\theta_i^* = \mathrm{clip}\left(\sqrt{\frac{\lambda}{w_i\,\mathrm{curv}_i}},\, 0,\, \frac{\pi}{L_{\max}\text{-eff}_i}\right) \tag{3}$$

   Why: Classical rate-distortion theory says this clipped square-root rule is the exact optimum for splitting a budget across coordinates.

Train-free swap and certificate

Substitute the optimal grid with no fine-tuning and report the optimality-gap certificate Δ and non-monotonicity profile.

5. Replace the model's stored rotation-frequency list with your solved best frequencies $\theta*$. This is a one-line change — every attention layer automatically uses the new list — and you touch no weights and do no training. In a standard library this means swapping the precomputed inverse-frequency buffer (and any cached cosine/sine tables) for $\theta*$. Because only the frequencies changed, the model's relative-position behavior is preserved; it just scores long-distance pairs using the better frequencies. *RoPE's relative-position identity: the attention inner product depends on positions only through m - n via the frequency vector θ, and stays intact when θ is replaced – only the buffer changes.*

$$\langle q_m, k_n \rangle = \mathrm{Re}\big(x_m^* W_q^* W_k \, x_n \, e^{\,i\,\theta\,(m-n)}\big) \tag{4}$$

Why: Only the frequencies change, so the model's relative-position behavior is preserved and the swap is free.

6. Report the certificate Δ, which is the geometric grid's total weighted error minus the water-filling grid's total weighted error, and list the full change $\theta* - \theta_{geom}$ frequency by frequency. To make the pattern readable, tag each frequency as 'retired' if it ended up pinned at its ceiling, 'denser' if it grew, or 'thinned' if it shrank without hitting the ceiling — these tags are just for description. $A\Delta$ greater than zero is a proof on paper that the original geometric frequencies were not the best choice; Δ is zero only if they already were. *Optimality-gap certificate Δ : the budget-weighted distortion of the geometric grid minus that of the water-filling grid, non-negative and zero only when the geometric allocation is already optimal.*

$$\Delta = \sum_i w_i D_i(\theta_{\mathrm{geom}}) - \sum_i w_i D_i(\theta^*) \;\geq\; 0 \tag{5}$$

Why: $\Delta \geq 0$ proves on paper exactly how much better the new grid is and shows it is non-monotone, which a geometric sequence cannot be.

Falsification and controls

Test the long-context payoff against geometric baselines with a permutation control isolating the allocation and a box-only control isolating the bound.

7. Without any fine-tuning, test the model with the new frequencies on three long-context tasks at lengths 32K, 64K, and 128K: Needle-in-a-Haystack (can it find a planted fact), RULER (a long-context accuracy suite), and perplexity on long documents. Compare against the very same model using the standard RoPE frequencies and the other popular grids (NTK-aware, YaRN, and both MrRoPE schedules), each installed by the same one-line swap so only the frequency list differs. Plot recall, accuracy, and perplexity across the lengths and methods.

   Why: This checks the real payoff – better long-context behavior – against the grids people use today.

8. Make a scrambled version θ_{perm} by randomly shuffling the solved frequencies across dimensions: this keeps the exact same set of frequency values, the same total budget, and the same certificate number, but puts each frequency on the wrong dimension. Install it the same way and re-run the long-context tests. Do this for several different shuffles (about 5–10) and report the average and the range. If putting frequencies on the right dimensions is what matters, every shuffle should drop performance back to — or below — the standard geometric baseline.

   Why: If the shuffle erases the gain, the win comes from the assignment itself, not just from having those frequency values.

9. Make a 'ceiling-only' version θ_{box} – *only* by taking the original geometric frequencies and simply capping any that exceed their Nyquist ceiling, with no redistribution of budget. Install it and re-run the long-context tests. Whatever extra improvement the full water-filling frequencies show over this ceiling-only version is the credit due to actually redistributing the budget, as opposed to merely obeying the aliasing limit.

   Why: This separates how much help comes from obeying the aliasing limit versus from actually redistributing the budget.