# 1. DartQuant Remaining Issues Analysis

In the previous PDF, we discussed a fundamental assumption of DartQuant: **activation distributions are all Laplacian distributed**. Then, we introduced the SWD loss to automatically measure the *distance between an arbitrary distribution with a mean of 0 and a uniform distribution*.

However, some core issues remain unresolved:

1. Why target a uniform distribution? Why not target a Gaussian distribution?
2. Why use random Hadamard matrices for $R_3$ and $R_4$?

## 1.1 The Uniformity Assumption

In the last PDF, I answered (or rather, cleverly bypassed) Question 1. My response at the time was: "Because we want to maximize information entropy by making activation values uniformly distributed, thereby minimizing quantization error." Theoretically, there's nothing wrong with this. However, we can see that in LLMs, there are many "dead connections" (activation values being zero). In other words, reality contradicts the theory. Although a uniform distribution is theoretically optimal, existing LLM activation distributions cannot satisfy this ideal state. **So why can't we assume, as QLoRA does, that the activation distribution is Gaussian? Theoretically, converting Laplacian-distributed activation values to a Gaussian distribution is much easier than converting them to a uniform distribution.**

## 1.2 Random Hadamard Matrix

Regarding Question 2, the author's(DartQuant) answer is: $R_3$ acts after $RoPE$, which is a Rotary Position Embedding. It is position-sensitive and is not a standard linear transformation. We cannot pass $R_3$ through $RoPE$ to fuse it into the preceding linear layer weights. $R_4$ acts after the $SiLU/SwiGLU$ activation functions. Non-linear activation functions prevent $R_4$ from acting on the preceding weights as $R_{1,2}$ do. Therefore, $R_{3,4}$ must compute matrix multiplication online during the *inference stage*.

Hadamard matrices possess the theoretically minimum *incoherence*:

$$\mu(H) = \max_{i,j} |H_{ij}| = 1/\sqrt{d}$$

This is considered from the perspective of matrix elements. A more intuitive understanding comes from the perspective of vector dot products (basis transformation):

$$\mu(H) = \max_{i,j} |\langle h_i, e_j \rangle| = \frac{1}{\sqrt{d}}$$

This means that a Hadamard matrix can uniformly disperse values originally in the basis direction across all dimensions. However, in practice, *incoherence* only works optimally when the data's covariance matrix is close to $\sigma^2 I$ (isotropic).

But $RoPE$ is at $R_3$. After the input data passes through RoPE, block-diagonal correlation is introduced.

**Definition 1 (RoPE):**

$$RoPE(\mathbf{x}, m) = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ sin(m\theta) & \cos(m\theta) \end{pmatrix}$$

Random Hadamard is essentially a random addition and subtraction combination of dimensions. If there is such strong correlation between block diagonals, random addition and subtraction might lead to the superposition of outliers in correlated dimensions, actually increasing outliers.

**Question 2:** Why use a random Hadamard matrix? Are there no other methods for online inference?

> Because you need to calculate matrix multiplication online in real-time during the inference stage, the time complexity of your $R_{3,4}$ must be as fast as possible. Otherwise, it defeats the essence and original intention of quantization.

# 2. Improvement Schemes

Here, we introduce *learnable rotation matrices* and *SWD Loss specifically for Gaussian distributions*.

## 2.1 Learnable Structured Rotation

To address the online computation efficiency issues caused by the inability to fuse $R_3$ and $R_4$, and to overcome the sub-optimality of random matrices, we propose using learnable rotation matrices for optimization.

> **Symbol Convention:** Let the hidden layer dimension be $d$. The butterfly rotation $R_3 \in \mathbb{R}^{d\times d}$ acts directly on the complete $d$-dimensional activation vector. Define $K = \log_2 d$ layers of butterfly operations, with $d/2$ Givens pairings per layer, totaling $\frac{d}{2}\log_2 d$ learnable angle parameters.

**Definition 3 (Butterfly Operation):** A Butterfly Operation refers to the process of splitting an input sequence and combining it through specific additions, subtractions, and multiplications with rotation factors.

**Example 4:** This algorithm is commonly used in Fast Fourier Transforms (FFT). In the matrix representation of FFT, a large discrete Fourier transform matrix can be decomposed into the product of multiple *sparse*

*matrices*. Through this decomposition, the original $O(N^2)$ multiplication operations for the discrete Fourier transform are simplified to $O(N \log N)$ operations.

**Definition 5 (Givens Rotation):**

$$G(i, j, \theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Below is a complete analysis from the RoPE definition to the solution:

## 2.1.1 RoPE Analysis

Let $A \in \mathbb{R}^{L \times d}$ be the input activation matrix, where $L$ is the sequence length and $d$ is the hidden layer dimension.
We view $A$ as a set of $L$ row vectors: $A = [\mathbf{a}_1, \ldots, \mathbf{a}_L]^T$.

For the $m$-th position token $\mathbf{a}_m$, according to Definition 1, RoPE is defined as:

$$\mathrm{RoPE}(\mathbf{a}_m) = \mathcal{R}_m \mathbf{a}_m$$

where $\mathcal{R}_m \in \mathbb{R}^{d \times d}$ is a Block Diagonal Matrix:

$$\mathcal{R}_m = \mathrm{diag}\left(R_m^{(0)}, R_m^{(1)}, \ldots, R_m^{(d/2-1)}\right)$$

Each $2 \times 2$ sub-block $R_m^{(k)}$ corresponds to the $k$-th frequency band:

$$R_m^{(k)} = \begin{bmatrix} \cos(m\theta_k) & -\sin(m\theta_k) \\ \sin(m\theta_k) & \cos(m\theta_k) \end{bmatrix}, \quad \theta_k = 10000^{-2k/d}$$

**Remark 5:** The $m$-th row of $\mathrm{RoPE}(A)$ is $\mathbf{a}_m^T \mathcal{R}_m^T$. It preserves the norm of each $2 \times 2$ subspace (orthogonal transformation) but changes its direction.

## 2.1.2 Issues Caused by RoPE

Because we are concerned with quantization difficulty, which depends on the overall distribution range of feature channels on a given dataset, we must examine the **Empirical Covariance Matrix** of $\mathrm{RoPE}(A)$ over the entire sequence $L$.

Let $\tilde{\mathbf{a}}$ be the random variable after RoPE. Its covariance matrix $\Sigma_{\mathrm{rope}} \in \mathbb{R}^{d \times d}$ is:

$$\Sigma_{\mathrm{rope}} = \frac{1}{L} \sum_{m=1}^{L} (\mathcal{R}_m \mathbf{a}_m)(\mathcal{R}_m \mathbf{a}_m)^T = \frac{1}{L} \sum_{m=1}^{L} \mathcal{R}_m (\mathbf{a}_m \mathbf{a}_m^T) \mathcal{R}_m^T$$

Fixing a frequency index $k$ (similar for other frequency bands). **Assume** the local covariance of the input at different positions is approximately constant (stationarity assumption), i.e., $\mathbb{E}[\mathbf{a}_m^{(k)} {\mathbf{a}_m^{(k)}}^T] \approx \Sigma_{\text{in}}^{(k)}$ holds for all $m$, where $\Sigma_{\text{in}}^{(k)} = \begin{bmatrix} \sigma_1^2 & \rho \\ \rho & \sigma_2^2 \end{bmatrix}$.

**Proposition 6 (Diagonal Elements of RoPE's $k$-th Sub-block Covariance Matrix):** The diagonal elements (i.e., channel variances) of the $k$-th sub-block covariance $\Sigma_{\text{rope}}^{(k)}$ are:

$$\text{Var}(\tilde{a}_{2k}) \approx \underbrace{\frac{\sigma_1^2 + \sigma_2^2}{2}}_{\text{Mean Term}} + \underbrace{\frac{\sigma_1^2 - \sigma_2^2}{2} \cdot \frac{1}{L} \sum_{m=1}^{L} \cos(2m\theta_k)}_{\text{Oscillation Term}} \tag{*}$$

*Proof*: For conciseness, let $c = \cos(m\theta)$, $s = \sin(m\theta)$ in the following steps.

1. Calculate $\mathcal{R}_m \Sigma_{\text{in}}$:

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \sigma_1^2 & \rho \\ \rho & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} c\sigma_1^2 - s\rho & c\rho - s\sigma_2^2 \\ s\sigma_1^2 + c\rho & s\rho + c\sigma_2^2 \end{bmatrix}$$

2. Multiply by the transpose matrix $\mathcal{R}_m^T$:
   Expanding each term in the matrix:

- Element (1,1):

$$(c\sigma_1^2 - s\rho)c + (c\rho - s\sigma_2^2)(-s) = c^2\sigma_1^2 - sc\rho - sc\rho + s^2\sigma_2^2 = \sigma_1^2c^2 + \sigma_2^2s^2 - 2\rho sc$$

- Element (1,2):

$$(c\sigma_1^2 - s\rho)s + (c\rho - s\sigma_2^2)c = sc\sigma_1^2 - s^2\rho + c^2\rho - sc\sigma_2^2 = (\sigma_1^2 - \sigma_2^2)sc + \rho(c^2 - s^2)$$

- Element (2,1): (Due to symmetry, results same as (1,2))

$$= (\sigma_1^2 - \sigma_2^2)sc + \rho(c^2 - s^2)$$

- Element (2,2):

$$(s\sigma_1^2 + c\rho)s + (s\rho + c\sigma_2^2)c = s^2\sigma_1^2 + sc\rho + sc\rho + c^2\sigma_2^2 = \sigma_1^2s^2 + \sigma_2^2c^2 + 2\rho sc$$

3. Covariance contribution $\Sigma_m$ at the $m$-th position (Overall empirical covariance is $\frac{1}{L}\sum_m \Sigma_m$):

$$\Sigma_m = \begin{bmatrix} \sigma_1^2\cos^2(m\theta) + \sigma_2^2\sin^2(m\theta) - \rho\sin(2m\theta) & \frac{1}{2}(\sigma_1^2 - \sigma_2^2)\sin(2m\theta) + \rho\cos(2m\theta) \\ \frac{1}{2}(\sigma_1^2 - \sigma_2^2)\sin(2m\theta) + \rho\cos(2m\theta) & \sigma_1^2\sin^2(m\theta) + \sigma_2^2\cos^2(m\theta) + \rho\sin(2m\theta) \end{bmatrix}$$

4. Variance:
   The first and second variances are symmetric; we choose one to analyze.

$$\text{Var}_1 = \sigma_1^2\cos^2(m\theta) + \sigma_2^2\sin^2(m\theta) - \rho\sin(2m\theta)$$

5. Using trigonometric formulas:

- $\cos^2\alpha = \frac{1+\cos 2\alpha}{2}$
- $\sin^2\alpha = \frac{1-\cos 2\alpha}{2}$

$$\mathrm{Var}(\tilde{a}_{2k}) = \sigma_1^2\left(\frac{1+\cos 2m\theta}{2}\right) + \sigma_2^2\left(\frac{1-\cos 2m\theta}{2}\right) - \rho\sin 2m\theta$$

Grouping terms with $\sigma$ into constant and $\cos$ terms:

$$\mathrm{Var}(\tilde{a}_{2k}) = \underbrace{\frac{\sigma_1^2+\sigma_2^2}{2}}_{\text{Mean Term}} + \underbrace{\frac{\sigma_1^2-\sigma_2^2}{2}\cos(2m\theta)}_{\text{Oscillation Term}} - \underbrace{\rho\sin(2m\theta)}_{\text{Correlation Term}}$$

**Remark 7:** Since we are concerned with the overall variance rather than just the variance of a single block, we need to take the average.

When averaging over the entire sequence $L$:

- Mean Term: As it is a constant, the average remains the same.
- Oscillation Term: $\cos(2m\theta) \to \frac{1}{L}\sum^{L}\cos(2m\theta)$
- Correlation Term: Similar to the oscillation term's $\cos(2m\theta_k)$, $\sin(2m\theta_k)$ also oscillates over the sequence. When $L$ is large enough, $\frac{1}{L}\sum_{m=1}^{L}\sin(2m\theta_k) \approx 0$. Thus, this term naturally disappears after averaging, without requiring an additional assumption of $\rho = 0$.

Finally, we obtain:

$$\mathrm{Var}(\tilde{a}_{2k}) \approx \underbrace{\frac{\sigma_1^2+\sigma_2^2}{2}}_{\text{Mean Term}} + \underbrace{\frac{\sigma_1^2-\sigma_2^2}{2}\cdot\frac{1}{L}\sum_{m=1}^{L}\cos(2m\theta_k)}_{\text{Oscillation Term}}$$

□

Returning to our analysis, we now perform extreme value analysis on formula (*):

1. Small $k$, $\theta_k \approx 1$:
   $\cos(2m\theta_k)$ oscillates rapidly between $[-1, 1]$. When $L$ is large enough, $\frac{1}{L}\sum\cos \to 0$.

$$\mathrm{Var}_{\text{high}} \approx \frac{\sigma_1^2+\sigma_2^2}{2}$$

2. $k \to d/2$, $\theta_k \to 0 \implies \cos(2m\theta_k) \approx 1$:

$$\mathrm{Var}_{\text{low}} \approx \sigma_1^2$$

In other words, the diagonal elements (Variance) of $\Sigma_{\text{rope}}$ transition from a "uniform mean" to "original extreme values" as index $k$ increases (frequency decreases).

## 2.1.3 New Objective

Since "incoherence only works optimally when the data's covariance matrix is close to $\sigma^2 I$ (isotropic)", we need to find an orthogonal matrix $R$ such that:

$$\text{diag}(R\Sigma_{\text{rope}}R^T) \approx \lambda \mathbf{1}$$

## 2.1.4 Construction of Butterfly Topology

**Remark (Connection from Subspace Analysis to Dimensional Operations):** The analysis in Section 2.1.2 reveals variance non-uniformity at the granularity of $d/2$ RoPE subspaces. Since each RoPE subspace corresponds to two adjacent dimensions $(2k,\, 2k+1)$, the variance of subspace $k$ is approximately equal to the average variance of these two dimensions. Thus, variance non-uniformity at the subspace level directly maps to variance non-uniformity at the dimension level—specifically, dimensions $2k$ and $2k+1$ share approximately the same variance $\text{Var}^{(k)}$, which varies monotonically with $k$. A standard $d$-dimensional Butterfly Givens topology naturally pairs these "equi-variance" adjacent dimensions in initial layers (small strides) and achieves variance migration across frequency bands in higher layers (large strides), which is exactly what we need.

**Lemma 8:** For any two channels with different variances $\lambda_i, \lambda_j$ and correlation coefficient $\rho_{i,j}$, the variance of the $i$-th channel after a Givens rotation $G(\theta)$ is:

$$\lambda_i' = \lambda_i \cos^2\theta + \lambda_j \sin^2\theta + 2\rho_{ij}\sqrt{\lambda_i\lambda_j}\cos\theta\sin\theta$$

*proof*: Setting $\lambda_i' = \lambda_j'$ and solving for the optimal angle:

$$\theta^* = \frac{1}{2}\arctan\left(\frac{2\rho_{ij}\sqrt{\lambda_i\lambda_j}}{\lambda_i - \lambda_j}\right)$$

Corollary: When $\rho_{i,j} \neq 0$, the optimal angle is no longer $\pi/4$ and must be learned to find $\theta^*$ for each pair.

Lemma 8 tells us that Givens rotations can equalize any pair of channels. However, we have $d$ activation dimensions, and the variance varies monotonically along the dimension index. We need a **systematic pairing strategy** so that all channels are equalized within finite steps.

**Definition 9 (Butterfly Givens Layer):** Let the total number of channels be $N = d$ (assume $d = 2^K$). Define the $\ell$-th layer butterfly operation $B_\ell$ ($\ell = 0, 1, \ldots, K-1$) as: pairing all $d$ dimensions with stride $s = 2^\ell$, and applying an independent Givens rotation to each pair. Specifically, the pairing indices for layer $\ell$ are:

$$(i,\ i+2^{\ell}), \quad i=0,1,\ldots,N-1, \quad i \bmod 2^{\ell+1} < 2^{\ell}$$

A learnable Givens rotation $G(\theta_{ij}^{(\ell)})$ is applied to each pair $(i, j)$.

**Lemma 10 (Full Mixing of Butterfly Topology):** After $K = \log_2 N$ layers of Butterfly Givens rotations, assuming each channel's correlation coefficient $\rho_{i,j} = 0$. If all angles in each layer are taken as $\theta = \pi/4$, all channel variances converge to the global mean:

$$\lambda_{\text{eq}} = \frac{1}{N}\sum_{k=0}^{N-1}\lambda_k$$

*Proof*: By induction. Let $\{\lambda_i^{(\ell)}\}$ be the set of channel variances after the $\ell$-th layer operation.
**Base case** ($\ell = 0$): Pairing $(2k, 2k+1)$, by Lemma 8, after rotation:

$$\lambda_{2k}^{(1)} = \lambda_{2k+1}^{(1)} = \frac{\lambda_{2k}^{(0)} + \lambda_{2k+1}^{(0)}}{2}$$

At this point, there are at most $N/2$ distinct variance values (equalized within each pair).
**Inductive step**: Assume that after $\ell$ layers, every $2^{\ell}$ consecutive channels share the same variance (i.e., there are $N/2^{\ell}$ distinct values). The $(\ell+1)$-th layer pairs them with stride $2^{\ell}$, averaging these distinct values in pairs. After the operation, every $2^{\ell+1}$ consecutive channels share the same variance, and the number of distinct values halves to $N/2^{\ell+1}$.
After $K = \log_2 N$ layers, the number of distinct values is $N/2^K = 1$, meaning all channel variances are equal. Since each operation preserves the total variance (Givens rotation is an orthogonal transformation), the final value must be $\frac{1}{N}\sum_k \lambda_k$. □

**Example 11:** For $N = 8$ (i.e., $d = 8$, $K = 3$), the three-layer butterfly pairing pattern is:

- $B_0$ **(stride 1)**: $(\sigma_0^2, \sigma_1^2), (\sigma_2^2, \sigma_3^2), (\sigma_4^2, \sigma_5^2), (\sigma_6^2, \sigma_7^2)$ — Equalization within adjacent frequency bands.
  - Now, the 8 original independent variances become 4 sets of "local means".
    - New variance for channels 0, 1: $V_{01}^{(1)} = \frac{\sigma_0^2+\sigma_1^2}{2}$
    - New variance for channels 2, 3: $V_{23}^{(1)} = \frac{\sigma_2^2+\sigma_3^2}{2}$
    - New variance for channels 4, 5: $V_{45}^{(1)} = \frac{\sigma_4^2+\sigma_5^2}{2}$
    - New variance for channels 6, 7: $V_{67}^{(1)} = \frac{\sigma_6^2+\sigma_7^2}{2}$
- $B_1$ **(stride 2)**: $(0, 2), (1, 3), (4, 6), (5, 7)$ — Equalization across adjacent frequency groups.
  - Pairing logic: Pair by skipping 1 position. Note that inputs are now $V^{(1)}$ from the previous layer.
    - Pair channel 0 and 2 $\rightarrow$ mixing $V_{01}^{(1)}$ and $V_{23}^{(1)}$
    - Pair channel 1 and 3 $\rightarrow$ mixing $V_{01}^{(1)}$ and $V_{23}^{(1)}$ (same result as above)
    - Pair channel 4 and 6 $\rightarrow$ mixing $V_{45}^{(1)}$ and $V_{67}^{(1)}$
    - Pair channel 5 and 7 $\rightarrow$ mixing $V_{45}^{(1)}$ and $V_{67}^{(1)}$ (same result as above)

- New variance for channels 0, 1, 2, 3:

$$V_{0-3}^{(2)} = \frac{V_{01}^{(1)} + V_{23}^{(1)}}{2} = \frac{\sigma_0^2 + \sigma_1^2 + \sigma_2^2 + \sigma_3^2}{4}$$

- New variance for channels 4, 5, 6, 7:

$$V_{4-7}^{(2)} = \frac{V_{45}^{(1)} + V_{67}^{(1)}}{2} = \frac{\sigma_4^2 + \sigma_5^2 + \sigma_6^2 + \sigma_7^2}{4}$$

- $B_2$ **(stride 4)**: $(0,4),(1,5),(2,6),(3,7)$ — **Direct pairing of highest and lowest frequencies**. All channel variances are now unified:

$$V_{\text{final}}^{(3)} = \frac{V_{0-3}^{(2)} + V_{4-7}^{(2)}}{2} = \frac{\sum_{k=0}^{7} \sigma_k^2}{8}$$

**Remark 12:** Lemma 10 proves the sufficiency of $\theta = \pi/4$ under the ideal condition of $\rho_{i,j} = 0$. However, when the correlation between channels is non-zero, the optimal angle for each pair deviates from $\pi/4$ based on the local correlation structure. While a fixed $\pi/4$ might still approximately equalize variances, it is no longer mathematically optimal. This is the core motivation for ButterflyQuant, which treats all angles $\{\theta_{i,j}^{(\ell)}\}$ as learnable parameters: through data-driven optimization, each Givens automatically finds its corresponding $\theta^*$, while simultaneously compensating for offsets caused by correlations.

## 2.1.6 Construction and Complexity of Complete Rotation Matrix

**Definition 13 (Learnable Butterfly Rotation):** Define $R_3$ (or $R_4$) as the product of $K$ layers of Butterfly Givens rotations:

$$R = B_{K-1} \cdot B_{K-2} \cdots B_1 \cdot B_0$$

Each layer $B_\ell$ consists of $d/2$ parallel $2 \times 2$ Givens rotations, resulting in a total of

$$\frac{d}{2} \times K = \frac{d}{2} \log_2 d$$

learnable angle parameters.

**Proposition 14 (Complexity Analysis):**

| | Random Hadamard (DartQuant) | Butterfly Givens (Ours) |
|---|---|---|
| **Time Complexity** | $O(d \log d)$ | $O(d \log d)$ |
| **Parameter Count** | $0$ (fixed random) | $O(d \log d)$, specifically $\frac{d}{2} \log_2 d$ |

| | Random Hadamard (DartQuant) | Butterfly Givens (Ours) |
|---|---|---|
| **Orthogonality** | ✓ (by construction) | ✓ (product of Givens rotations) |
| **Learnable** | ✗ | ✓ |
| **Hardware Friendliness** | Requires random seed + online generation | Sparse structure, parallelizable |

*Proof of orthogonality*: Each $G(\theta)$ satisfies $G^T G = I$. Since Givens rotations in $B_\ell$ act on disjoint pairs of dimensions, $B_\ell$ is itself an orthogonal matrix. The product of orthogonal matrices remains orthogonal, so $R$ is orthogonal. □

**Remark 15:** The time complexity is identical to Hadamard, but the key advantages of the butterfly structure are:

1. **Learnability**: Angle parameters optimized via gradient descent adapt to specific models and data variance distributions.
2. **Structural Guarantee**: Regardless of parameter updates, $R$ always remains orthogonal without needing extra projection or regularization.
3. **Targeting**: The butterfly topology naturally fits RoPE's frequency band structure—lower layers handle internal subspaces, while higher layers handle energy migration across frequency bands.

## 2.1.7 Complete Algorithm Flow

**Stage 1: Define the Architecture**

First, we set up the skeleton of the "variance mixing pipeline".

1. Determine Dimensions: Let the hidden layer dimension be $d$ (e.g., $d = 4096$ in LLaMA-7B). Calculate the number of layers $K = \log_2 d$ (e.g., $\log_2 4096 = 12$).
2. Define Learnable Parameter Set $\Theta$: Install an adjustable Givens rotation angle at every "intersection" of this pipeline.
   - There are $K$ layers.
   - Each layer has $d/2$ pairings.
   - Therefore, we need to initialize a parameter tensor $\mathbf{\Theta}$ of size $[K, d/2]$.
   - *Initialization Strategy*: The ButterflyQuant paper recommends using Identity Initialization (i.e., all angles $\theta \approx 0$), which works better than random or Hadamard initialization because it allows the network to start learning how to rotate from a "no rotation" state.

**Stage 2: Construct Matrix $R_3$ (The Construction Algorithm)**

This is the core algorithmic logic. Given the current parameters $\mathbf{\Theta}$, how do we calculate the corresponding rotation matrix $R_3$? It is the product of $K$ sparse matrices:

$$R_3 = B_{K-1} \cdot B_{K-2} \cdots B_1 \cdot B_0$$

The specific construction loop is as follows (pseudo-code):
**Input:** Parameter set $\mathbf{\Theta}$
**Output:** Rotation matrix $R_3$

1. **Initialize Identity Matrix:** $R_{\text{total}} = I$.
2. **Loop over each layer** $\ell$ from $0$ to $K-1$:
   - Get Stride: $stride = 2^\ell$ (e.g., 1, 2, 4, 8...).
   - Construct sparse matrix $B_\ell$ for the current layer:
     - $B_\ell$ initialized to a zero matrix.
     - Traverse pairings: For each pair $(i, j)$ where $j = i + s$ (generated according to butterfly topology rules):
       - Retrieve the corresponding angle $\theta = \mathbf{\Theta}[\ell, \text{pair\_index}]$ from the parameter set $\mathbf{\Theta}$.
       - Calculate $c = \cos\theta, s = \sin\theta$.
       - Fill $c$ at positions $(i, i)$ and $(j, j)$ in $B_\ell$.
       - Fill $s$ at position $(i, j)$ in $B_\ell$.
       - Fill $-s$ at position $(j, i)$ in $B_\ell$.
   - Multiply: $R_{\text{total}} = B_\ell \cdot$R_{\text{total}}}$.
3. Return: $R_3 = R_{\text{total}}$.

(Note: During actual deployment for inference, we don't actually multiply out $R_3$ into a massive $d \times d$ dense matrix. Instead, we use the aforementioned loop to apply operations directly to the input vector $X$ using a CUDA Kernel, maintaining $O(d \log d)$ complexity instead of $O(d^2)$.)

**Stage 3: Training and Optimization**
Now that we have the skeleton and initial values for parameters $\theta$, how do we determine the exact values of $\theta$ to best mix the variances? We need to train it with data.
**Input:**

- A batch of Calibration Data, e.g., WikiText-2 with 128 samples.
- Post-RoPE activation input $X_{\text{in}}$.

**Algorithmic Flow:**

1. Forward Pass:
   - Construct (or implicitly apply) rotation $R_3$ using current $\mathbf{\Theta}$.
   - Calculate rotated activations: $X_{\text{out}} = R_3 \cdot X_{\text{in}}$.
   - Perform simulated quantization and dequantization on $X_{\text{out}}$ to obtain $\hat{X}_{\text{out}}$.
2. Loss Calculation: The ButterflyQuant paper uses two losses to guide optimization:
   - Reconstruction Loss ($\mathcal{L}_{\text{recon}}$): Minimizes error before and after quantization. $||X_{\text{in}} - R_3^T \cdot \hat{X}_{\text{out}}||^2$
   - Uniformity Regularization ($\mathcal{L}_{\text{uniform/Gaussian}}$): Forces the distribution of $X_{\text{out}}$ to be close to a uniform distribution (or Gaussian distribution, depending on settings). This directly corresponds to the "variance equalization" goal. $D_{KL}(P(X_{\text{out}})||\text{Uniform})$

3. Backward Pass:
    - Calculate gradient $\nabla_\theta \mathcal{L}$ of the Loss with respect to each angle $\theta$.
    - Since Givens rotations are continuously differentiable (sin/cos), gradients flow smoothly back to $\theta$.
4. Update Parameters: $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}$ (SGD update).

# 2.2 SWD Loss Targeting Gaussian Distribution

Returning to the issue in Section 1.1: Why target a uniform distribution? We argued that for LLMs with many dead connections, converting Laplacian activations to a Gaussian distribution is more natural than a uniform one.

**Proposition 16 (Theoretical Motivation for Gaussian Quantization):** Let the quantizer be a uniform quantizer with input distribution $p(x)$. For $b$-bit quantization, the high-resolution approximation of Mean Squared Quantization Error (MSQE) is:

$$D \approx \frac{\Delta^2}{12}, \quad \Delta = \frac{x_{\max} - x_{\min}}{2^b}$$

This is already optimal for a uniform distribution. However, in practical LLMs, the transformation required to push the distribution entirely toward uniform is often too drastic (ill-conditioned) due to dead connections. Pushing toward a Gaussian distribution is a milder target, which, when paired with non-uniform quantizers (like NF4), can still achieve low quantization error.

**Definition 17 (Gaussian SWD Loss):** Given a batch of activation values $\{x_i\}_{i=1}^n$ (zero mean), sorted as $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$. The target Gaussian quantiles are:

$$q_i = \Phi^{-1}\left(\frac{i - 0.5}{n}\right) \cdot \hat{\sigma}, \quad \hat{\sigma} = \sqrt{\frac{1}{n}\sum_i x_i^2}$$

where $\Phi^{-1}$ is the inverse CDF of the standard normal distribution. Gaussian SWD Loss is defined as:

$$\mathcal{L}_{\text{G-SWD}} = \frac{1}{n}\sum_{i=1}^{n} \left(x_{(i)} - q_i\right)^2$$

**Remark 18:** Compared to DartQuant's uniform SWD:

- Uniform SWD target quantiles are equally spaced: $q_i^{\text{unif}} = x_{\min} + (x_{\max} - x_{\min}) \cdot \frac{i-0.5}{n}$
- Gaussian SWD target quantiles are dense in the center and sparse at the tails, naturally adapting to the peaked distribution of LLM activations.
- The use of $\hat{\sigma}$ ensures the loss is scale-invariant: we only require the **shape** of the distribution to approach Gaussian, not to match a specific mean and variance.

**Remark 19 (Connection with QLoRA):** QLoRA's NF4 quantization scheme is a non-uniform quantization bin designed specifically based on Gaussian assumptions. Our Gaussian SWD Loss can be seen as actively shaping the distribution into a Gaussian form during the **rotation stage**, providing better input conditions for subsequent Gaussian-aware quantization (like NF4). These two are complementary: one optimizes the quantizer design, while the other optimizes the input distribution.