2. Randomized Matrix Multiplication (RMM)

2.1 Conceptual Overview

Randomized Matrix Multiplication (RMM) provides a fast, approximate method to compute the product of two matrices, significantly reducing computation when exact accuracy is not required. Given two matrices (A in R^{m×n}) and (B in R^{n×p}), instead of performing full dense multiplication costing O(mnp), RMM approximates AB by sampling a subset of influential outer products.

The key identity:

$$AB = \sum_{k=1}^{n} A_{:,k} B_{k,:}$$

Each term is a rank-1 outer product. RMM samples only a subset of these and rescales them to recover an unbiased estimate.

This mainly improves performance heavily in sparse matrices, and becomes increasingly powerful when matrices become either more sparse or more low-rank.

To study this effect rigorously, we systematically vary sparsity levels and intrinsic ranks and analyze how RMM's sample complexity, error, and runtime behave.

Randomized Matrix Multiplication (RMM) provides a fast, approximate method to compute the product of two matrices, significantly reducing computation when exact accuracy is not required. Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, instead of performing full dense multiplication costing $O(mnp)$, RMM approximates AB by sampling a subset of influential outer products.

The key identity:

$$ AB = \sum_{k=1}^n A_{:,k} B_{k,:} $$

Each term is a rank-1 outer product. RMM samples only a subset of these and rescales them to recover an unbiased estimate.

This mainly improves performance heavily in sparse matrices

2.2 The RMM Estimator

Define:

- $a_k = A_{:,k}$: k-th column of A
- $b_k^T = B_{k,:}$: k-th row of B

Choose probabilities $p_k > 0$ with $\sum p_k = 1$. Sample indices $k_1, \dots, k_s$ i.i.d. according to p_k.

Estimator:

$$ \tilde C = \frac{1}{s} \sum_{i=1}^s \frac{1}{p_{k_i}} a_{k_i} b_{k_i}^T $$

Two sampling choices:

- **Uniform**: $p_k = 1/n$
- **Importance sampling**: $p_k \propto \|a_k\| \, \|b_k\|$

2.3 Correctness and Proofs

Unbiasedness

$$ \mathbb{E}[\tilde C] = AB $$

Proof:

$$ \mathbb{E}\left[\frac{1}{p_K} a_K b_K^T\right] = \sum_k p_k \cdot \frac{1}{p_k} a_k b_k^T = AB $$

Thus RMM always targets the exact matrix product in expectation.

Error Bound

$$ \mathbb{E}|\tilde C - C|*F^2 = \frac{1}{s} \left(\sum* - |C|_F^2 \right) $$}^n \frac{|a_k|^2 |b_k|^2}{p_k

Errors decay as:

$$ \text{Error} \propto 1/\sqrt{s} $$

Optimal Sampling Distribution

Minimizing the variance term gives:

$$ p_k = \frac{|a_k| \, |b_k|}{\sum_j |a_j| \, |b_j|} $$

This is the **importance sampling rule**, guaranteed to minimize expected error.

2.4 Time and Space Complexity

Exact GEMM

$$ O(mnp) $$

RMM Preprocessing

- Compute column norms of A: $O(mn)$
- Compute row norms of B: $O(np)$ Total:

$$ O(mn + np) $$

RMM Multiplication

Each sampled outer product costs $O(mp)$. For s samples:

$$ O(smp) $$

RMM is faster when $s \ll n$.

Experimental Workflow

We evaluate RMM across a controlled suite of matrix families, systematically varying sparsity and intrinsic rank to understand how approximation quality depends on structure.

Matrix Families

1. Dense Gaussian matrices (worst case)

- No structure, nearly uniform column/row norms.
- Outer products contribute nearly equally, making sampling harder.
- Baseline for comparison.

2. Low-rank matrices

Constructed as: A = U_r Σ_r V_r^T with small rank r << min(m,n). - Energy is concentrated in first few dimensions. - Outer-product contributions decay sharply. - RMM succeeds with very few samples.

We systematically vary: - r = 5, 10, 20, 50, 100 - noise injection (0–10%) to simulate neural network weight patterns.

3. Sparse matrices

We generate matrices with sparsity levels: - 10% nonzero, - 5%, - 1%, - 0.1%.

This models graphs, NLP embeddings, recommender systems.

RMM is especially effective here: - Outer products are sparse → cheap to compute - Column norms vary drastically → sampling distribution becomes sharply peaked

4. Neural-network-like matrices

Real or synthetic fully-connected layer weights: - Heavy-tailed singular value spectrum - Semi-sparse structure - Low intrinsic dimensionality

These demonstrate RMM's practical relevance.

Experiments Across Sparsity and Rank

For each matrix family, and each configuration, we: - Compute exact $C = AB$ - Run RMM with $s \in \{0.5\%, 1\%, 2\%, 5\%, 10\%, 20\% \text{ of } n\}$ - Record: - relative Frobenius error - runtime - speedup - Repeat 10 trials for stability

Key additional experiment: Error vs Structure

We produce plots of: - Error vs samples for different ranks - Error vs samples for different sparsity levels - Samples required for <5% error as a function of rank r - Samples required for <5% error as a function of sparsity α

This reveals how fast sampling complexity drops as matrices become more structured.:

Synthetic data generate with gaussian matrix, very sparse matrix, use multiple samples of neural net that is very sparse, normal matrix, low rank matrix

---

2.7 Results (Figures)

Add figures for:

- Runtime vs n
- Error vs s
- Speedup vs s
- Performance across matrix types
- Error vs s across different sparsity levels
- Error vs s across different intrinsic ranks
- Samples required for a fixed error threshold (<5%) vs rank
- Samples required for <5% error vs sparsity level

These visualizations allow a clean comparison of how structure affects RMM's efficiency. (Figures)

Add figures for:

- Runtime vs n
- Error vs s
- Speedup vs s
- Performance across matrix types
- Comparision with other types like low rank approx, strassen, gemm, python mm function, across all types of matrices and figures

---

2.8 Observations on Figures

Key observations include:

- Importance sampling significantly reduces variance.
- Low-rank and sparse matrices give excellent accuracy with few samples — often achieving <5% error with s < 2% of columns.
- For extremely sparse matrices, even s < 1% is often sufficient.
- Dense Gaussian matrices require many samples due to uniform column norms.
- Error curves show clear 1/sqrt(s) decay predicted by theory.
- As rank decreases, the number of samples needed for good approximation decreases almost linearly.
- As sparsity increases, effective sampling distribution becomes more concentrated, dramatically reducing sample complexity.

Thus RMM's practical performance improves sharply as matrices become either more low-rank or more sparse, matching theoretical predictions.

Summaries you should include:

- Importance sampling significantly reduces variance.
- Low-rank and sparse matrices give excellent accuracy with few samples.
- Dense Gaussian matrices require many samples due to uniform column norms.
- Speedup grows as s/n decreases.
- Empirical 1/\sqrt{s} decay matches theory.

2.9 Why and where RMM Works Well in Practice

Reasons:

- Real matrices often have uneven column energies.
- Low-rank structure common in ML, NLP, and recommender systems.
- Outer products have heavy-tailed distributions.
- Approximate answers acceptable in many workloads.
- Reduced memory traffic improves practical speed.

2.10 Real-World Use Cases

- Approximate linear layers in neural networks
- Large recommender systems
- Power iteration for Markov chains
- Large-scale similarity search

2.11 Conclusion

RMM approximates dense multiplication using Monte Carlo sampling over outer products. It:

- Is unbiased
- Has error decaying as $1/\sqrt{s}$
- Achieves speedups when $s \ll n$
- Excels when data is sparse or low-rank
- Serves as a foundation for more advanced randomized linear algebra methods such as Randomized SVD.