Randomized and Quantum Approximate Matrix Multiplication

Simon Apers
Université Paris Cité, CNRS, IRIF, France

Arjan Cornelissen
Simons Institute, UC Berkeley, CA, USA

Samson Wang
Institute for Quantum Information and Matter, California Institute of Technology, Pasadena, CA, USA

Abstract

The complexity of matrix multiplication is a central topic in computer science. While the focus has traditionally been on exact algorithms, a long line of literature also considers randomized algorithms, which return an approximate solution in faster time. In this work, we adopt a unifying perspective that frames these randomized algorithms in terms of mean estimation. Using it, we first give refined analyses of classical algorithms based on random walks by Cohen-Lewis ('99), and based on sketching by Sarlós ('06) and Drineas-Kannan-Mahoney ('06). We then propose an improvement on Cohen-Lewis that yields a single classical algorithm that is faster than all the other approaches, if we assume no use of (exact) fast matrix multiplication as a subroutine. Second, we demonstrate a quantum speedup on top of these algorithms by using the recent quantum multivariate mean estimation algorithm by Cornelissen-Hamoudi-Jerbi ('22).

2012 ACM Subject Classification Theory of computation → Sketching and sampling; Theory of computation → Quantum computation theory

Keywords and phrases randomized algorithms, quantum algorithms, streaming, approximate matrix multiplication, mean estimation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.77

Related Version *Full Version*: https://arxiv.org/abs/2510.08509

1 Introduction

Over the last 6 decades, the complexity of multiplying two n-by-n matrices has gone down from the naïve $O(n^3)$ upper bound to $O(n^\omega)$ with $\omega < 2.372$ the matrix multiplication coefficient – see [2] for the latest improvement. Whether this coefficient can be brought down to its trivial optimum $\omega = 2$ is a major open question in computer science. In the meantime, however, the complexity can be reduced by considering algorithms for *approximate* matrix multiplication.

▶ **Definition 1** (Approximate matrix multiplication). *Given a description of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ via their matrix elements, return a matrix $\boldsymbol{C}$ which satisfies $\|\boldsymbol{C} - \boldsymbol{AB}\|_* \leq \varepsilon$ for some chosen norm $\|\cdot\|_*$ and target precision ε.*

In our work we specifically consider $\|\cdot\|_* = \|\cdot\|_{\max}$ (the "max-norm problem") and $\|\cdot\|_* = \|\cdot\|_F$ (the "Frobenius norm problem").

Approximate matrix multiplication is a setting that allows to exploit *randomized* algorithms, and this seems in contrast to exact matrix multiplication where the best known algorithms are deterministic. In particular, there are well-known Monte Carlo algorithms for approximate matrix multiplication that build on random walks (or "path integral Monte Carlo") [7] and sketching [20, 12]. These algorithms can ultimately be viewed from the

perspective of *multivariate mean estimation*, i.e., they cleverly define a random variable $\boldsymbol{X}$, taking values in $\mathbb{R}^d$ with $d = n^2$, whose mean $\boldsymbol{\mu}$ contains the entries of the matrix product $\boldsymbol{AB}$. Using e.g. the vector Bernstein inequality, one can argue that with high probability, the empirical mean $\tilde{\boldsymbol{\mu}}$ of L samples of $\boldsymbol{X}$, with covariance matrix $\boldsymbol{\Sigma} := \mathbb{E}[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^T]$, satisfies

$$\textit{multivariate mean estimation:} \quad \begin{cases} \|\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_\infty = \mathcal{O}\left(\sqrt{\max_i [\boldsymbol{\Sigma}]_{ii}/L}\right) \\ \|\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2 = \mathcal{O}\left(\sqrt{\mathrm{Tr}[\boldsymbol{\Sigma}]/L}\right) \end{cases} .$$

The fact that these randomized algorithms can be phrased as a mean estimation task means that they are amenable to a quantum speedup, since we can make use of *quantum multivariate mean estimation* from [10]: given quantum access to $\widetilde{O}(L)$ samples of $\boldsymbol{X}$, we can return an estimator $\tilde{\boldsymbol{\mu}}$ that with high probability satisfies

$$\textit{quantum multivariate mean estimation:} \quad \begin{cases} \|\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_\infty = \mathcal{O}\left(\sqrt{\mathrm{Tr}[\boldsymbol{\Sigma}]/L}\right) \\ \|\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2 = \mathcal{O}\left(\sqrt{d\,\mathrm{Tr}[\boldsymbol{\Sigma}]/L}\right) \end{cases} .$$

For the ℓ_2-norm, for instance, this yields a speedup over classical mean estimation when the number of quantum samples $L \gg d = n^2$.

In this work, we study classical and quantum algorithms for approximate matrix multiplication utilizing mean estimation. We make the following contributions:

- We refine the seminal algorithm of Cohen & Lewis [7]. While the original algorithm is incomparable to the main other approach based on matrix sketching [20, 12], our refinement yields a classical algorithm with better time complexity than prior classical art (if no fast matrix multiplication is allowed).
- We give a unified analysis of sketching algorithms for matrix multiplication [20, 12] based on mean estimation. Using this, we construct quantum analogues of all above algorithms. We find a quantum speedup in certain parameter regimes, and there is an unconditional speedup over prior quantum art.
- We extend our sketching-based analysis to give classical and quantum matrix multiplication algorithms for more than two matrices.

We display a simplified account of our results in Table 1 (more intricate statements of time complexities can be found in the Appendix). We express the runtimes using the element-wise matrix norm: for $\boldsymbol{M} \in \mathbb{R}^{n \times m}$ and $p, q \in [1, \infty]$, we write

$$\|\boldsymbol{M}\|_{p,q} = \left(\sum_{j=1}^{n} \left(\sum_{k=1}^{m} \left| [\boldsymbol{M}]_{jk} \right|^p \right)^{\frac{q}{p}} \right)^{\frac{1}{q}} .$$

Specifically, $\|\cdot\|_{\max} = \|\cdot\|_{\infty,\infty}$ and $\|\cdot\|_F = \|\cdot\|_{2,2}$ are the max-norm and Frobenius norm, respectively. In order to facilitate navigation of the table, we also present the partial ordering of all norms that appear in Figure 1.

To the authors' knowledge the best previous quantum algorithm is that of Shao [21], which operates via a quantum subroutine to perform inner product estimation on each of the n^2 inner products that determine a matrix product.[1] The data model Shao requires is efficient access to the unitaries which encode the rows and columns of $\boldsymbol{A}$ and $\boldsymbol{B}$ in the amplitudes

[1] See also the algorithm in [6] which can also be adapted to give an algorithm of similar complexity, though it is specialized to outputting a quantum state.

Table 1 Time complexities of approximate matrix multiplication (without use of fast matrix multiplication). For simplicity of presentation, we assume square matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$, and a matrix multiplication exponent of $\omega = 3$. We use an overline over any matrix to denote the matrix of element-wise absolute values. We present their partial ordering of the element-wise norms in Figure 1. If within a block (i.e., "Classical" or "Quantum") and a particular column there is an algorithm which is unconditionally faster than the others, we have shaded its complexity. Each of the algorithms utilizes $O(n^2)$ preprocessing time which we omit from the table for simplicity. We discuss the different (classical and quantum) data models in Appendix A.2 as well as more detailed accounts of the complexities. The cell left open would have a complexity that exceeds $O(n^3)$, and hence is not very useful.

		Runtime ($\widetilde{O}(\cdot)$)		Data model	Multiple matrices?
		$\|\cdot\|_{\max}$	$\|\cdot\|_F$		
Classical	Cohen-Lewis [7]	$\frac{\|\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}\|_{1,2}^2}{\varepsilon^2}$	$n\frac{\|\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}\|_{1,2}^2}{\varepsilon^2}$	RAM	✓
	Sarlós [20]	$n^2\frac{\|\boldsymbol{A}\|_{2,\infty}^2\|\boldsymbol{B}^T\|_{2,\infty}^2}{\varepsilon^2}$	$n^2\frac{\|\boldsymbol{A}\|_{2,2}^2\|\boldsymbol{B}\|_{2,2}^2}{\varepsilon^2}$	circuit	✓
	Drineas-Kannan-Mahoney [12]	$n^2\frac{\|\boldsymbol{A}^T\|_{\infty,2}^2\|\boldsymbol{B}\|_{\infty,2}^2}{\varepsilon^2}$	$n^2\frac{\|\boldsymbol{A}\|_{2,2}^2\|\boldsymbol{B}\|_{2,2}^2}{\varepsilon^2}$	circuit	
	This work (random walk)	$\frac{\|\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}\|_{\infty,\infty}\|\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}\|_{1,1}}{\varepsilon^2}$	$\frac{\|\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}\|_{1,1}^2}{\varepsilon^2}$	RAM	✓
Quantum	Shao [21]	$\frac{\|\boldsymbol{A}\|_{2,1}\|\boldsymbol{B}^T\|_{2,1}}{\varepsilon}$	$n\frac{\|\boldsymbol{A}\|_{2,1}\|\boldsymbol{B}^T\|_{2,1}}{\varepsilon}$	QROM	
	This work (quantum tug-of-war sketch)	$n^2\frac{\|\boldsymbol{A}\|_{2,2}\|\boldsymbol{B}\|_{2,2}}{\varepsilon}$		quantum circuit	✓
	This work (quantum random walk)	$\frac{\|\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}\|_{1,1}}{\varepsilon}$	$n\frac{\|\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}\|_{1,1}}{\varepsilon}$	QRAM	✓

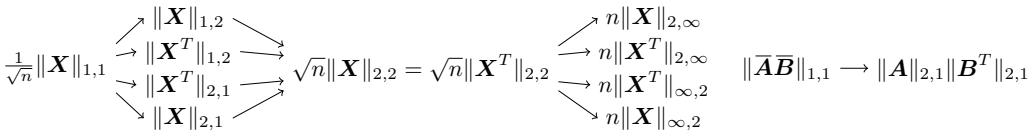

Figure 1 Partial ordering of matrix norms appearing in Table 1. Here, $\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$, and $\overline{\boldsymbol{A}}$ represents the matrix containing all element-wise absolute values of $\boldsymbol{A}$. "$\rightarrow$" denotes "$\leq$". If there is no arrow in between two quantities then they are incomparable.

of quantum states – this is achievable in $O(n^2)$ preprocessing time and polylog(n) time per query with a QROM [19] (we define and discuss quantum memory models in Appendix A.2). We note that Shao also presents other algorithms which depend on the condition number of the matrix which we do not display in our table. Whilst these approaches could be advantageous in certain settings, we focus on comparing algorithms for generic matrices which are efficient without additional assumptions.

We remark that our algorithms with the most refined time complexities inherit a particular property of the algorithm of Cohen *&* Lewis [7] – that the complexity depends on the norm of a matrix product, rather than a product of matrix norms. This can be especially advantageous for non-negative matrices or matrices with (approximate) sparsity patterns, and this advantage can be amplified in generalizations to multiple matrix products. As an example, when $\boldsymbol{A}$ and $\boldsymbol{B}$ are stochastic matrices, all Frobenius norm algorithms are slower than their counterparts based on [7] by a factor up to $\Omega(n)$.

2 Random walk algorithms

While the Cohen-Lewis algorithm [7] works for arbitrary matrices, its idea is best illustrated for the multiplication of stochastic matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k$, i.e., every matrix $\boldsymbol{A}_j \in \mathbb{R}^{n \times n}$ is entry-wise non-negative, and all row-sums equal 1. The matrix product $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k$ can then be interpreted as the probability transition matrix of a random walk on a directed graph, displayed in Figure 2. Simulting this random walk samples from the distribution $\boldsymbol{d}^T \boldsymbol{A}_1 \cdots \boldsymbol{A}_k$, for an arbitrary initial distribution $\boldsymbol{d} \in \mathbb{R}^n$. The original idea from Cohen & Lewis is to reconstruct the matrix product $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$ by learning it row-by-row, i.e., learning all the probability distributions by starting from $\boldsymbol{d} = \boldsymbol{e}_j$, with j running from 1 to n.

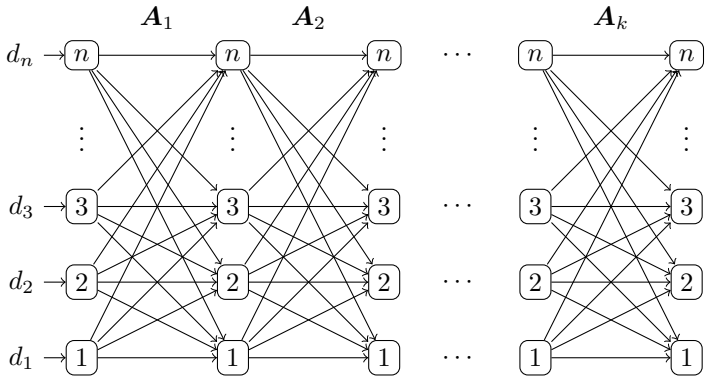

Figure 2 A pictorial representation of the matrix product $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$ of stochastic matrices as a random walk on a directed graph. The random walk samples a vertex on the left side of the graph with probability distribution $\boldsymbol{d}$, and then transitions through the graph, with the probability transition matrix of every layer equal to $\boldsymbol{A}_j$. After k steps, the probability distribution on the final layer of vertices is $\boldsymbol{d}^T \boldsymbol{A}_1 \cdots \boldsymbol{A}_k$.

We subtly modify this approach, by analyzing what happens when we start with an arbitrary initial distribution $\boldsymbol{d}$. We denote the resulting sampled path of vertices $(j_0, \ldots, j_k)$, and define the corresponding random variable $\boldsymbol{X}(j_0, \ldots, j_k) = \boldsymbol{e}_{j_0} \boldsymbol{e}_{j_k}^T$. We then prove in Lemma 7 that this can be used as an estimator for the matrix product, as $\mathbb{E}[\boldsymbol{X}] = \mathrm{diag}(\boldsymbol{d}) \cdot \boldsymbol{A}_1 \cdots \boldsymbol{A}_k$ where we denote $\mathrm{diag}(\boldsymbol{d})$ as the diagonal matrix who shares diagonal entries with $\boldsymbol{d}$.

Next, we adapt the above approach to work for matrices with negative entries as well, i.e., we explain how to compute $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$, where $\overline{\boldsymbol{A}}_1, \ldots, \overline{\boldsymbol{A}}_k$ are all stochastic matrices. The idea here, already observed by Cohen & Lewis, is to label all the edges of the directed graph with the signs of the matrix entries. For a sampled path $(j_0, \ldots, j_k)$, we then multiply all the signs together to obtain $s(j_0, \ldots, j_k) \in \{\pm 1\}$, and we modify the random variable to be $\boldsymbol{X}(j_0, \ldots, j_k) = s(j_0, \ldots, j_k) \boldsymbol{e}_{j_0} \boldsymbol{e}_{j_k}^T$. Crucially, this does not impact the upper bound on the trace of the covariance matrix, since we can still show that $\mathrm{Tr}[\boldsymbol{\Sigma}] \leq 1$.

Finally, we adapt the construction to any matrix product, by observing that for any sequence of matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k$, we can construct a *stochastic matrix product decomposition*. That is, we can find a vector $\boldsymbol{d}$ and a sequence of matrices $\boldsymbol{A}'_1, \ldots, \boldsymbol{A}'_k$, such that $\overline{\boldsymbol{A}}'_1, \ldots, \overline{\boldsymbol{A}}'_k$ are all stochastic, and such that $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k = \mathrm{diag}(\boldsymbol{d}) \boldsymbol{A}'_1 \cdots \boldsymbol{A}'_k$ (see Theorem 5). Moreover, we prove that we can perform this stochastic matrix product decomposition as a preprocessing routine in time linear in the size of the input. We can now interpret the vector $\boldsymbol{d}$ as the probability distribution $\boldsymbol{d}/\|\boldsymbol{d}\|_1$, and absorb the resulting factor $\|\boldsymbol{d}\|_1$ in the random variable, i.e., $\boldsymbol{X}(j_0, \ldots, j_k) = \|\boldsymbol{d}\|_1 \cdot s(j_0, \ldots, j_k) \cdot \boldsymbol{e}_{j_0} \boldsymbol{e}_{j_k}^T$. To actually implement these algorithms, we require efficiently addressable read/write-memory in both the classical and quantum settings. Putting all these ideas together gives the following result.

▶ **Theorem 2** (Simplified version of Theorems 8 & 9). *Consider a matrix product of k matrices $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$ and the problem in Definition 1. We use an overline over any matrix to denote the matrix of element-wise absolute values. We give classical algorithms in the RAM model, with run-times*

$$\widetilde{O}\left(k \cdot \|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{\infty,\infty} \cdot \|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{1,1}/\varepsilon^2\right), \qquad \text{and} \qquad \widetilde{O}\left(k \cdot \|\overline{\boldsymbol{A}_1} \cdots \overline{\boldsymbol{A}_k}\|_{1,1}^2/\varepsilon^2\right),$$

that solve the max*-norm and the Frobenius-norm problems, respectively, with high probability. Similarly, we give quantum algorithms in the QRAM-model, with run-times*

$$\widetilde{O}\left(k \cdot \|\overline{\boldsymbol{A}_1} \cdots \overline{\boldsymbol{A}_k}\|_{1,1}/\varepsilon\right), \qquad \text{and} \qquad \widetilde{O}\left(k \cdot \sqrt{nm}\|\overline{\boldsymbol{A}_1} \cdots \overline{\boldsymbol{A}_k}\|_{1,1}/\varepsilon\right),$$

that solve the max*-norm and the Frobenius-norm problems, respectively, with high probability. Here, $\boldsymbol{A}_1$ has n rows, and $\boldsymbol{A}_k$ has m columns. Both algorithms require classical preprocessing time linear in the size of the input up to polylogarithmic factors.*

The quantum complexity for Frobenius norm approximation stated above simply follows by considering standard norm conversion between the max-norm and the Frobenius norm, and rescaling the error parameter. We remark that aside from being unconditionally faster than other algorithms in certain norm regimes (and the absence of fast matrix multiplication), the algorithms based on random walks are particularly well-suited to stochastic matrices. In particular, their complexities grow linearly in the number of matrices, rather than geometrically as is the case for all other approaches.

3 Sketching algorithms

The key idea of sketching algorithms as used for matrix multiplication is to sample vectors $\boldsymbol{s} \in \mathbb{R}^n$ and to compute pairs of vectors $\boldsymbol{As}$ and $\boldsymbol{s}^T\boldsymbol{B}$, which together form "sketches" of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. With judicious choice of $\boldsymbol{s}$, we can ensure first that the outer product $\boldsymbol{Ass}^T\boldsymbol{B}$ forms an unbiased estimator of $\boldsymbol{AB}$, and second that the second-order moments of this estimator are controlled. Sarlós [20] and Drineas et al. [12] propose different constructions for $\boldsymbol{s}$, and we build on these for our generalization to multiple matrices, as well as for our quantum algorithms. We will refer to their approaches as the tug-of-war sketch and column-sample sketch, respectively.

The tug-of-war sketch [3] (also known as the AMS sketch) populates entries of $\boldsymbol{s}$ with a 4-wise independent hash function $h : [n] \to \{-1, 1\}$ the resulting sketched rows of $\boldsymbol{A}$ and sketched columns of $\boldsymbol{B}$ are "pulled" in either the positive or negative directions with equal probability, hence the name "tug-of-war sketch". The column-sample sketch samples a column of $\boldsymbol{A}$ (and correspondingly, a row of $\boldsymbol{B}$) according to a some probability distribution of choice. Thus, $\boldsymbol{s}$ takes the form of a 1-sparse vector weighted by the chosen probability distribution. Utilizing these sketches we give our results on sketching-based matrix multiplication algorithms.

▶ **Theorem 3** (Simplified version of Theorems 17 and 18). *Consider a matrix product of k square matrices $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k \in \mathbb{R}^{n\times n}$ and the problem in Definition 1. With classical preprocessing time linear in the size of the input up to polylogarithmic factors, we give a classical algorithm for the Frobenius norm problem with run-time*

$$\widetilde{O}\left(n^2 \cdot \left(3^k\|\boldsymbol{A}_1\|_{2,2}^2 \cdots \|\boldsymbol{A}_k\|_{2,2}^2/\varepsilon^2\right)^{\omega-2}\right),$$

and a quantum algorithm for the max-norm problem with run-time

$$\widetilde{O}\left(n^2 \cdot 3^k\|\boldsymbol{A}_1\|_{2,2} \cdots \|\boldsymbol{A}_k\|_{2,2}/\varepsilon\right),$$

both to arbitrary success probability probability, where ω is the matrix multiplication exponent. Both algorithms do not require efficiently-addressable memory, and run in the circuit model.

The complexity of our classical algorithm we state above reduces to that of [20] and [12] in the case of multiplication of two matrices. To the authors' knowledge this property is not shared by previous analyses of sketching algorithms for multiple matrices – they either only naturally extend to 3 matrices [12, Appendix A1] or incur large (super-quadratic) runtime in the inverse precision ε^{-1} [20, Appendix B2].

The classical algorithm can be sped up by use of fast matrix multiplication, as can all the sketching-based classical algorithms we consider in this manuscript. Interestingly, for any matrix multiplication exponent $\omega < 2.5$, this yields an algorithm with time complexity scaling sublinearly with ε^{-1}, which has better dependence on the precision even over quantum algorithms. The origin of the advantage using fast matrix multiplication can be viewed as follows. Conventionally, we might wish to estimate $\mathbb{E}[\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}] = \boldsymbol{A}\boldsymbol{B}$ by taking a sample mean of c independently drawn samples $\{\boldsymbol{s}_j\}_j$. Naturally, the time complexity is linear in c. However, due to the structure of the outer product, such a sample mean is also exactly equal to $\boldsymbol{A}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{B}$ where the rows of the matrix $\boldsymbol{S} \in \mathbb{R}^{c\times n}$ are given by $\{\boldsymbol{s}_j/\sqrt{c}\}_j$. The matrix $\boldsymbol{A}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{B}$ can now be evaluated with fast matrix multiplication with time complexity proportional to $c^{\omega-2}$. This provides a way to return the sample mean sublinearly (after preprocessing time proportional to nc to construct $\boldsymbol{S}$).

Our quantum algorithm as quoted above is based on the tug-of-war sketch utilized by Sarlós. We can also give a quantum algorithm with similar complexity based on the column-sample sketch of Drineas et al., however, unlike the algorithms based on the tug-of-war sketch both classical and quantum approaches can only accommodate multiplication of two matrices. We note that the quantum algorithm we state has unconditionally worse time complexity than the one we give in Theorem 2. However, we record it as it uses a weaker assumption on the type of data structure required, which we discuss in Appendix A.2 (and it can handle multiple matrices which the algorithm of [21] cannot).

4 Discussion

Our quantum algorithms for max-norm approximation depend on the trace of the covariance matrix, which is the same quantity that characterizes approximation in Frobenius norm. We note that in classical mean estimation, there is a different quantity that controls max-norm approximations: the maximum entry-wise variance – as clear examples, note the disparity in the complexity comparison of Table 1. It is an interesting open question if the quantum multivariate mean estimation of [10] in max-norm can be refined to depend on the same second-moment quantity that we see for classical mean estimation.

In the presentation of Table 1 we have made the simplifying assumption that the matrix multiplication exponent is $\omega = 3$. In reality, one can choose $\omega < 3$ by exploiting fast matrix multiplication, in which case there are interesting asymptotic conclusions about the runtime of sketching algorithms – they can outperform the quantum approaches we present in certain regimes. We remark that the most costly step of our quantum algorithms essentially is matrix-vector multiplication in coherent arithmetic (rather than rectangular matrix multiplication in classical sketching algorithms), and we leave it as an open question whether or not fast matrix multiplication can be exploited in our quantum algorithms. Meanwhile, in comparing our classical algorithm to those which can be sped up with fast matrix multiplication, we do stress that we may be more interested in practical considerations for immediate application. From this perspective, we recall that the Strassen algorithm [23] with $\omega \approx 2.8$ is commonly implemented. However, refinements based around the work of Coppersmith and Winograd beyond $\omega < 2.38$ [8] are considered "galactic", and not practically implementable [13].

Finally, our work considers generic unstructured matrices. However, other classical approaches to approximate matrix multiplication are particularly useful in exploiting sparse matrices [18]. We leave open the question of whether we can exploit sparsity in the quantum setting as well.

References

1 Jonathan Allcock, Jinge Bao, Aleksandrs Belovs, Troy Lee, and Miklos Santha. On the quantum time complexity of divide and conquer, 2023. doi:10.48550/arXiv.2311.16401.

2 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039. SIAM, 2025. doi:10.1137/1.9781611978322.63.

3 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 20–29, 1996. doi:10.1006/jcss.1997.1545.

4 Joran van Apeldoorn. *A Quantum View on Convex Optimization*. PhD thesis, Universiteit van Amsterdam, 2020. URL: https://hdl.handle.net/11245.1/cf4a77ff-835c-4169-bd2c-3e983cfa83a9.

5 Simon Apers and Sander Gribling. Quantum speedups for linear programming via interior point methods, 2023. arXiv:2311.03215.

6 Anna Bernasconi, Alessandro Berti, Gianna Maria Del Corso, and Alessandro Poggiali. Quantum subroutine for efficient matrix multiplication. *IEEE Access*, 2024. doi:10.1109/ACCESS.2024.3446176.

7 Edith Cohen and David D Lewis. Approximating matrix multiplication for pattern recognition tasks. *Journal of Algorithms*, 30(2):211–252, 1999. Earlier version in *SODA'97*. doi:10.1006/jagm.1998.0989.

8 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Computational algebraic complexity editorial. doi:10.1016/S0747-7171(08)80013-2.

9 Arjan Cornelissen. Quantum algorithms through graph composition, 2025. doi:10.48550/arXiv.2504.02115.

10 Arjan Cornelissen, Yassine Hamoudi, and Sofiene Jerbi. Near-optimal quantum algorithms for multivariate mean estimation. In *Proceedings of the 54th ACM Symposium on the Theory of Computing (STOC)*, pages 33–43, 2022. doi:10.1145/3519935.3520045.

11 Alexander M Dalzell, András Gilyén, Connor T Hann, Sam McArdle, Grant Salton, Quynh T Nguyen, Aleksander Kubica, and Fernando GSL Brandão. A distillation-teleportation protocol for fault-tolerant qram, 2025. arXiv:2505.20265.

12 Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006. Open access version on *M. W. Mahoney's webpage*. doi:10.1137/S0097539704442684.

13 Costas S Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the hermite and smith normal forms of an integer matrix. *SIAM Journal on Computing*, 18(4):658–669, 1989. doi:10.1137/0218045.

14 Samuel Jaques and Arthur G Rattew. QRAM: A survey and critique, 2023. arXiv:2305.10310.

15 Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In *8th Conference on the Theory of Quantum Computation, Communication and Cryptography*, page 20, 2013. doi:10.4230/LIPIcs.TQC.2013.20.

16 Ashley Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A*, 471(2181), 2015. doi:10.1098/rspa.2015.0301.

17 María Naya-Plasencia and André Schrottenloher. Optimal merging in quantum k-xor and k-sum algorithms. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 311–340. Springer, 2020. Cryptography ePrint Archive: 2019/501. doi:10.1007/978-3-030-45724-2_11.

18 Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013. doi:10.1145/2493252.2493254.

19 Anupam Prakash. *Quantum Algorithms for Linear Algebra and Machine Learning*. PhD thesis, University of California at Berkeley, 2014. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.html.

20 Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152. IEEE, 2006. Open access version on *CiteSeerX*. doi:10.1109/FOCS.2006.37.

21 Changpeng Shao. Quantum algorithms to matrix multiplication, 2018. arXiv:1803.01601.

22 Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 1996.

23 Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969. doi:10.1007/BF02165411.

24 Letian Tang. More-efficient quantum multivariate mean value estimator from generalized grover gate, 2025. doi:10.48550/arXiv.2504.06940.

A Technical background

A.1 Notation

We denote matrices and vectors in bold type $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ and $\boldsymbol{a} \in \mathbb{C}^N$. We denote their entries as $[\boldsymbol{A}]_{ij}$ and a_j respectively for all $i, j \in [N]$. Denote $[\boldsymbol{A}]_{k\boldsymbol{\cdot}} \in \mathbb{C}^{1\times N}$ as the k^{th} row matrix of $\boldsymbol{A}$ for all $k \in [N]$, and denote $[\boldsymbol{A}]_{\boldsymbol{\cdot}j} \in \mathbb{C}^{N\times 1}$ as the k^{th} column matrix of $\boldsymbol{A}$ for all $j \in [N]$. For any matrix, we write $\overline{\boldsymbol{A}}$ as the matrix with element-wise absolute values $[\overline{\boldsymbol{A}}]_{ij} = |[\boldsymbol{A}]_{ij}|$. We denote $\boldsymbol{I}$ as the identity matrix and we denote $\mathbf{1}_m$ as the all-ones vector of dimension m. The notation $\widetilde{O}(\cdot)$ hides polylogarithmic factors, that is we denote $\widetilde{O}(g(n)) = O\big(g(n)\log^k(g(n))\big)$ for some constant k. For vectors $\boldsymbol{a} \in \mathbb{R}^n$, we denote the usual ℓ_p-norm as $\|\boldsymbol{a}\|_p = (\sum_i a_i^p)^{1/p}$.

A.2 Quantum computational model and memory

In both our classical and quantum algorithms, we assume access to different levels of memory architecture (see "Data model" in Table 1). Specifically, some algorithms call an architecture that allows for efficient reading and writing operations – given a register containing a memory address, we assume to be able to swap the bit at that location in memory with one from another register. In the quantum setting, this means that we assume to have access to a quantum random-access gate (QRAM gate), that for all data $\boldsymbol{x} \in \{0,1\}^d$, $j \in [d]$ and $b \in \{0,1\}$ acts as

$$\text{QRAM} : |x_1,\dots,x_d\rangle\,|j\rangle\,|b\rangle \mapsto |x_1,\dots,x_{j-1},b,x_{j+1},\dots,x_d\rangle\,|j\rangle\,|x_j\rangle\,.$$

We remark that this is a stronger assumption than having access to a quantum read-only memory (QROM),[2] where one assumes to have access to a gate

$$\text{QROM} : |x_1,\dots,x_d\rangle\,|j\rangle\,|b\rangle \mapsto |x_1,\dots,x_d\rangle\,|j\rangle\,|b\oplus x_j\rangle\,.$$

[2] We also remark here that the many different nomenclatures for read-only and read-write operations on quantum memory. Instead of the acronyms QROM/QRAM [9], one can find QRAM/QRAG [1], QCRAM/full-QRAM [4], QACM/QAQM [17], QRACM/QRAQM [15] and more. We prefer the nomenclature QROM/QRAM, to mimic the nomenclature ROM/RAM for the corresponding classical counterparts.

Having access to efficiently-addressable write operations is standard in the classical literature, but its quantum counterpart has been somewhat contentious, particularly as implementing an $O(d)$-sized QRAM/QROM fault-tolerantly may require $O(d)$ (classical) effort [14, 11]. We stress here that we only require our quantum algorithm to be able to perform the same operations as its corresponding classical counterpart, i.e., if our original classical algorithms do not actually use efficiently-addressable writing operations, then neither do our corresponding quantum algorithms. In this work we focus on theoretical comparison between the classical and quantum computational models.

In other settings, we do not need to assume any efficient read or write operations. Instead, we operate in a *circuit model*. Here, our classical (or quantum) algorithm can be written wholly in terms of a fixed classical (or quantum) circuit acting on input data [22]. When we consider this setting, it is sufficient to consider the input data in the form of a binary encoding $|x_1\rangle \otimes \cdots \otimes |x_d\rangle$.

So far in the above exposition we have used a simplifying picture of binary data. In both the classical and quantum models, we make the assumption that we can represent real numbers exactly in memory, and that we can retrieve them and perform basic arithmetic operations (that is, addition, multiplication, subtraction and division) on them in constant time. This is a standard assumption in the classical literature and we make the same assumption quantumly.

A.3 Quantum subroutines

We will make use of a multivariate quantum mean estimation routine from [10] (and later refined in [24]), which we restate here for convenience.

▶ **Theorem 4** (Quantum mean estimation [10, Theorem 3.4]). *Let $d \in \mathbb{N}$, $\delta \in (0,1)$, $(\Omega, \mathbb{P})$ be a probability space, and $\boldsymbol{X} : \Omega \to \mathbb{R}^d$ a random variable. Suppose we have access to the following two operations:*

$$U_{\mathbb{P}} : |0\rangle \mapsto \sum_{\omega\in\Omega} \sqrt{\mathbb{P}(\omega)}\,|\omega\rangle\,, \qquad \text{and} \qquad O_{\boldsymbol{X}} : |\boldsymbol{Y}\rangle\,|\omega\rangle\,|0\rangle \mapsto |\boldsymbol{Y}\rangle\,|\omega\rangle\,|\boldsymbol{Y}^T\boldsymbol{X}(\omega)\rangle\,,$$

where $\omega \in \Omega$ and $\boldsymbol{Y} \in \mathbb{R}^d$. Let $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ be the expectation and covariance matrix of X, and let $N \geq \log(d/\delta)$. Then, we can produce a vector $\widetilde{\boldsymbol{\mu}} \in \mathbb{R}^d$, such that $\|\widetilde{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_\infty \leq \sqrt{\mathrm{Tr}[\boldsymbol{\Sigma}]}\log(d/\delta)/N$, with probability at least $(1-\delta)$, with $\widetilde{O}(N)$ queries to $U_{\mathbb{P}}$, $O_{\boldsymbol{X}}$ and an inner product routine between the random variable $X(\omega)$ and an arbitrary vector $\boldsymbol{Y} \in \mathbb{R}^d$, and $\widetilde{O}(d)$ additional elementary gates.

The time complexity is not explicitly analyzed in [10], so we discuss this now here. The three key steps of the algorithm consist of (1) preparation of a uniform superposition of a grid of points $\boldsymbol{Y} \in G \subset (-1/2, 1/2)^d$; (2) computing the inner product of $\boldsymbol{Y}$ with a data vector; (3) applying the inverse quantum Fourier transform.

Step (1) costs $\widetilde{O}(d)$ without any need of quantum memory assumptions, due to efficient state preparation algorithms for uniform superpositions. The inverse quantum Fourier transform is also efficient. We can thus focus on step (2) as the non-generic cost. Explicitly, we require an operation that computes the inner product, i.e., for all $\boldsymbol{Y} \in \mathbb{R}^d$, we take the operation $O'_{\boldsymbol{X}} : |\boldsymbol{Y}\rangle\,|\omega\rangle\,|0\rangle \mapsto |\boldsymbol{Y}\rangle\,|\omega\rangle\,|\boldsymbol{Y}^T\boldsymbol{X}(\omega)\rangle$. Since we assume to be able to do arithmetic operations in unit time, this leads to an additional $O(d)$ overhead per sample in the circuit model, bringing the total time complexity to $\widetilde{O}(dN)$. A similar observation was made in [5]. However, what we will use in this work is that if we can implement this inner product more efficiently, say with time complexity η, then the resulting total time complexity becomes $\widetilde{O}(\eta N + d)$.

B The Cohen-Lewis-inspired algorithm

In this section, we describe our first classical and quantum algorithms for implementing approximate matrix multiplication. The idea of this algorithm is based on the seminal work by Cohen and Lewis [7].

Suppose that we are given k matrices, $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k$, such that $\boldsymbol{A}_j \in \mathbb{R}^{n_{j-1}, n_j}$, for all $j \in [k]$. Then, $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$ is well-defined, and our goal is to output an approximation of this matrix product. We write $n = n_0$ and $m = n_k$, so that the resulting matrix product is of dimension $n \times m$. We assume that full descriptions of the matrices are available to us in memory.

B.1 Stochastic matrix product decomposition

We start by showing that we can assume all matrices have absolute row-sum 1, without loss of generality. To that end, we show that we can classically perform a stochastic matrix product decomposition as a preprocessing step, using time and memory that is similar to the size of the matrices themselves. In this section we omit proofs to streamline discussion and refer the reader to the full version of our paper for the full analyses. We start by formally introducing the decomposition.

▶ **Theorem 5** (Stochastic matrix product decomposition). *Let $k \in \mathbb{N}$, $n_j \in \mathbb{N}$ for all $\ell \in [k]_0$, and let $\boldsymbol{A}_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ for all $\ell \in [k]$. We recursively define diagonal matrices $\boldsymbol{D}_0, \ldots, \boldsymbol{D}_k$, such that*

$$\boldsymbol{D}_k = I\,, \quad \text{and} \quad \forall \ell \in [k], i \in [n_{\ell-1}]\,, \quad [\boldsymbol{D}_{\ell-1}]_{ii} = \sum_{j=1}^{n_\ell} [\overline{\boldsymbol{A}}_\ell]_{ij} \cdot [\boldsymbol{D}_\ell]_{jj} = \|[\overline{\boldsymbol{A}}_\ell \boldsymbol{D}_\ell]_{i\bullet}\|_1\,.$$

Next, for all $\ell \in [k]$, we let $\boldsymbol{A}'_\ell = \boldsymbol{D}_{\ell-1}^{-1} \boldsymbol{A}_\ell \boldsymbol{D}_\ell$. Then, $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k = \boldsymbol{D}_0 \boldsymbol{A}'_1 \cdots \boldsymbol{A}'_k$, all $(\boldsymbol{A}'_j)$'s have all row-sums equal to 1, and for all $\ell \in [k]$ and $i \in [n_{\ell-1}]$,

$$[\boldsymbol{D}_{\ell-1}]_{ii} = \sum_{j=1}^{n_\ell} [\overline{\boldsymbol{A}_\ell} \cdots \overline{\boldsymbol{A}_k}]_{ij} = [\overline{\boldsymbol{A}_\ell} \cdots \overline{\boldsymbol{A}_k} \mathbf{1}_{n_k}]_i\,.$$

Moreover, we can compute all $\boldsymbol{D}_j$'s and $(\boldsymbol{A}'_j)$'s classically in time linear in the input size.

Using this decomposition, we can also compute some element-wise norms of the matrix product efficiently. Specifically, for $p = 1$ and $q \in [1, \infty]$, the element-wise matrix norm can be computed in time linear in the input size – this follows from the characterization of $(\boldsymbol{D}_0)_{ii}$ in Theorem 5.

▶ **Lemma 6.** *Let $k \in \mathbb{N}$, $n_j \in \mathbb{N}$ for all $\ell \in [k]_0$, and let $\boldsymbol{A}_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ for all $\ell \in [k]$. Let $q \in [1, \infty]$. Let $\boldsymbol{D}_0$ be as in Theorem 5, and let $\boldsymbol{d}_0$ be the vector containing the diagonal of $\boldsymbol{D}_0$. Then, $\|\overline{\boldsymbol{A}_1} \cdots \overline{\boldsymbol{A}_k}\|_{1,q} = \|\boldsymbol{d}_0\|_q$, and as such it can be computed in linear time in the total size of the input.*

We now state the crucial observation that motivates the stochastic matrix product decomposition. It is a variant on path-integral Monte Carlo, and it mimics the approach taken by Cohen and Lewis [7], but differs in one crucial aspect: Cohen and Lewis estimate every row of the product matrix in a separate subroutine. Here, we sample over the rows as well, leading to a random variable with expectation exactly equal to the matrix product we are after. This is the core of the improvement we present in this work.

▶ **Lemma 7.** *Let $k \in \mathbb{N}$, $n_j \in \mathbb{N}$ for all $\ell \in [k]_0$, and let $\boldsymbol{A}_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ for all $\ell \in [k]$. Let $\boldsymbol{D}_0, \boldsymbol{A}'_1, \dots, \boldsymbol{A}'_k$ be the stochastic matrix product decomposition from Theorem 5. Let $\boldsymbol{d}_0$ be the vector containing the diagonal entries of $\boldsymbol{D}_0$. Then, we define the probability distribution p by sampling $j_0 \sim \boldsymbol{q}$, for a probability distribution $\boldsymbol{q}$, and then $j_\ell \sim (\overline{\boldsymbol{A}'_\ell})_{j_{\ell-1},\bullet}$, for all $\ell \in [k]$. Define the random variable $\boldsymbol{X}(j_0, \dots, j_k) = \frac{d_{j_0}}{q_{j_0}} \cdot \left(\prod_{\ell=1}^k \operatorname{sign}([\boldsymbol{A}'_\ell]_{j_{\ell-1},j_\ell})\right) \boldsymbol{e}_{j_0} \boldsymbol{e}_{j_k}^T$ where $\operatorname{sign}(x)$ is 1 if $x \geq 0$, and -1 otherwise. Then,*

$$\boldsymbol{A}_1 \cdots \boldsymbol{A}_k = \mathop{\mathbb{E}}_{(j_0,\dots,j_k)\sim p}[\boldsymbol{X}], \quad \text{and} \quad \mathop{\operatorname{Var}}_{(j_0,\dots,j_k)\sim p}[\boldsymbol{X}_{ij}] \leq \frac{(\boldsymbol{d}_0)_i}{\boldsymbol{q}_i} \cdot (\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{ij}.$$

Hence, if we take $\boldsymbol{q} = \boldsymbol{d}_0/\|\boldsymbol{d}_0\|_1$, then we obtain

$$\max_{i\in[n_0],j\in[n_k]} \operatorname{Var}[\boldsymbol{X}_{ij}] \leq \|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{\infty,\infty} \cdot \|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{1,1}, \quad \text{and} \quad \operatorname{Tr}[\boldsymbol{\Sigma}] \leq \|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{1,1}^2,$$

where $\boldsymbol{\Sigma} = \operatorname{Cov}(\boldsymbol{X})$. If we instead take $q_i \propto \|(\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{i,\bullet}\|_1 \cdot \|(\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{i,\bullet}\|_\infty$, then

$$\max_{i\in[n_0],j\in[n_k]} \operatorname{Var}[\boldsymbol{X}_{ij}] \leq \sum_{i=1}^{n_0} \|(\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{i,\bullet}\|_1 \cdot \|(\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{i,\bullet}\|_\infty.$$

Finally, if we take $\boldsymbol{q} = \boldsymbol{d}_0^2/\|\boldsymbol{d}_0\|_2^2$, then we obtain

$$\max_{i\in[n_0],j\in[n_k]} \operatorname{Var}[\boldsymbol{X}_{ij}] \leq \|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{1,2}^2.$$

From the above analysis, we observe that when we choose $\boldsymbol{q} \propto \boldsymbol{d}_0$, we minimize the trace of the covariance matrix, which means that this choice yields the best algorithm for the Frobenius-norm problem. On the other hand, by choosing $\boldsymbol{q} \propto \boldsymbol{d}_0^2$, we recover the complexity from [7] for the max-norm problem. Both choices for $\boldsymbol{q}$ yield incomparable complexities for the max-norm problem.

It is tempting to simply choose the optimal $\boldsymbol{q}$ for the max-norm problem, which is when $q_i \propto \|(\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{i,\bullet}\|_\infty \cdot \|(\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{i,\bullet}\|_1$. However, the problem is that it is not clear how to compute this choice for $\boldsymbol{q}$ efficiently in linear time as a preprocessing step. Hence, we are left with an algorithm with incomparable complexity to the max-norm algorithm given by Cohen and Lewis' approach, for instance if we take $\boldsymbol{q} = \boldsymbol{d}_0/\|\boldsymbol{d}_0\|_1$ (see next section for further discussion). If we additionally have good upper bounds on $\|(\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k)_{i,\bullet}\|_\infty$ a priori, for all $i \in [n]$, then we could indeed further improve the algorithm.

B.2 Classical algorithm

The idea for the classical algorithm is to compute the expectation from Lemma 7 using Monte Carlo sampling. It remains to figure out the number of samples required, and the cost to perform sampling, which we analyze in the following theorem.

▶ **Theorem 8** (Classical random walk algorithm)**.** *Let $k \in \mathbb{N}$, $n_j \in \mathbb{N}$ for all $\ell \in [k]_0$, and let $\boldsymbol{A}_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ for all $\ell \in [k]$. Let $\varepsilon > 0$ and $\delta \in (0,1)$, and let $n_0 = n$ and $n_k = m$. Then, with classical preprocessing time linear in the size of the input up to polylogarithmic factors, we give a classical algorithm in the RAM model with run-time*

$$\widetilde{O}\left(k \cdot \frac{\|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{\infty,\infty} \cdot \|\overline{\boldsymbol{A}}_1 \cdots \overline{\boldsymbol{A}}_k\|_{1,1}}{\varepsilon^2} \cdot \log\left(\frac{d}{\delta}\right)\right) \text{ and } \widetilde{O}\left(k \cdot \frac{\|\overline{\boldsymbol{A}_1} \cdots \overline{\boldsymbol{A}_k}\|_{1,1}^2}{\varepsilon^2} \cdot \log\left(\frac{1}{\delta}\right)\right),$$

that computes an approximation $\widetilde{\boldsymbol{A}}$ of $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$ with probability at least $(1-\delta)$, satisfying the precision bounds $\|\widetilde{\boldsymbol{A}} - \boldsymbol{A}_1 \cdots \boldsymbol{A}_k\|_{\max} \leq \varepsilon$ and $\|\widetilde{\boldsymbol{A}} - \boldsymbol{A}_1 \cdots \boldsymbol{A}_k\|_F \leq \varepsilon$, respectively.

Proof sketch. We take the average of N realizations of the random variable from Lemma 7. Using the bound on the trace of the covariance matrix, and the vector Bernstein inequality (see full version of paper for pedagogical introduction), we can obtain our stated bound on N such that the Frobenius error is at most ε. A similar analysis yields the corresponding complexity for the max-norm case.

Finally, one can check that each sample can be generated in $O(k)$ given preprocessing time linear in the input size to generate the stochastic matrix product decomposition. ◀

The resulting complexity for the Frobenius norm unconditionally improves over the result in [7]. Indeed, by comparing the expressions in Table 1, we can use Hölder's inequality to conclude that $\|\boldsymbol{A}_1 \cdots \boldsymbol{A}_k\|_{1,1}^2 \leq n\|\boldsymbol{A}_1 \cdots \boldsymbol{A}_k\|_{1,2}^2$. The situation is a bit more complicated for the max-norm problem, since the bound from [7] and our complexity are incomparable. Indeed, if the resulting matrix product $\boldsymbol{A}$ is the $n \times n$ all-ones matrix, then the Cohen-Lewis bound is $\|\boldsymbol{A}\|_{1,2}^2 = n^3$, whereas our bound is $\|\boldsymbol{A}\|_{\infty,\infty} \cdot \|\boldsymbol{A}\|_{1,1} = n^2$. On the other hand, if the resulting matrix product $\boldsymbol{A}$ is $\mathrm{diag}(\sqrt{n}, 1, \ldots, 1)$, then the Cohen-Lewis bound is $2n$, whereas our bound is $n^{3/2}$.

B.3 Quantum algorithm

In the quantum setting, we wish to speed-up the classical algorithm using the multivariate quantum mean estimation routine [10]. We describe our result in the following theorem, in the proof of which we compute the costs of implementing the input routines for this procedure.

▶ **Theorem 9** (Quantum random walk algorithm). *Let $k \in \mathbb{N}$, $n_j \in \mathbb{N}$ for all $\ell \in [k]_0$, and let $\boldsymbol{A}_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ for all $\ell \in [k]$. Let $\varepsilon > 0$ and $\delta \in (0,1)$, and let $n_0 = n$ and $n_k = m$. Then, with classical preprocessing in the RAM model with time linear in the size of the input up to polylogarithmic factors, there exists a quantum algorithm in the QRAM model with run-time*

$$\widetilde{O}\left(k \cdot \frac{\|\overline{\boldsymbol{A}_1} \cdots \overline{\boldsymbol{A}_k}\|_{1,1}}{\varepsilon} \cdot \log\left(\frac{nm}{\delta}\right)\right),$$

that computes an approximation $\widetilde{\boldsymbol{A}}$ satisfying $\|\widetilde{\boldsymbol{A}} - \boldsymbol{A}_1 \cdots \boldsymbol{A}_k\|_{\max} < \varepsilon$, with probability at least $(1-\delta)$.

Proof. We can bound the number of q-samples required by combining Lemma 7 and Theorem 4.

Observe that coherent sampling from p can be done by implementing the classical sampling procedure reversibly, as described in [16], with the same time complexity $\widetilde{O}(k)$. Thus, total cost per sample for implementing $U_{\mathbb{P}}$ is $\widetilde{O}(k)$.

Finally we compute the cost of implementing the oracle computing the random variable. To that end, observe that it must implement the following operation:

$$O_{\boldsymbol{X}} : |j_0, \ldots, j_k\rangle\,|0\rangle \mapsto |j_0, \ldots, j_k\rangle\,|\prod_{\ell=1}^{k} \mathrm{sign}([\boldsymbol{A}'_\ell]_{j_{\ell-1},j_\ell}) \cdot \boldsymbol{e}_{j_0}\boldsymbol{e}_{j_k}^T\rangle\,.$$

This essentially boils down to k look-ups of the signs of the entries in the input, and then $O(k)$ multiplications of signs. This can be implemented in $\widetilde{O}(k)$ time with the aid of QROM access to preprocessed data. We complete the proof by observing that we can also implement the inner product routine in the same way if we have access to QRAM, i.e., we can store the matrix $|\boldsymbol{Y}\rangle$ in QRAM and index into it efficiently to spend at most $\widetilde{O}(k)$ time to implement

$$O'_{\boldsymbol{X}} : |\boldsymbol{Y}\rangle\,|j_0, \ldots j_k\rangle\,|0\rangle \mapsto |\boldsymbol{Y}\rangle\,|j_0, \ldots, j_k\rangle\,|\prod_{\ell=1}^{k} \mathrm{sign}([\boldsymbol{A}'_\ell]_{j_{\ell-1},j_\ell}) Y_{j_0,j_k}\rangle\,. \qquad ◀$$

This approach gives us a quantum algorithm that estimates the matrix product in max-norm, whereas the corresponding classical algorithm provides an approximation in terms of Frobenius norm. We can convert between the two using norm conversion, since $\|\cdot\|_F \leq \sqrt{nm}\|\cdot\|_{\max}$.

C Sketching-based algorithms

C.1 Generic analysis

In this section we provide a unifying picture for matrix multiplication algorithms via sketching, which encapsulates the algorithms of [12] and [20], and also enables us to construct classical algorithms for multiple matrices as well as quantum algorithms. For simplicity, in this part of the exposition let us assume both $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$, though the discussion also generalizes to rectangular matrices. Our goal is to construct a matrix-valued random variable $\boldsymbol{S} \in \mathbb{R}^{c\times n}$ (with $c < n$) such that for all $\boldsymbol{A}, \boldsymbol{B}$ we have

$$\mathbb{E}[\boldsymbol{A}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{B}] = \boldsymbol{A}\boldsymbol{B}\,,$$

which in particular implies that $\mathbb{E}[\boldsymbol{S}^T\boldsymbol{S}] = \boldsymbol{I}$. The matrices $\boldsymbol{A}\boldsymbol{S}^T$ and $\boldsymbol{S}\boldsymbol{B}$ may be thought of as sketches of $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively. Specifically, judicious choices of $\boldsymbol{S}$ can enable efficient approximation of $\boldsymbol{A}\boldsymbol{B}$, as the complexity of evaluating $\boldsymbol{A}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{B}$ is $O(n^2c^{\omega-2})$, compared to $O(n^\omega)$ for the exact matrix product (here ω can either be $\omega = 3$, if we disallow fast matrix multiplication, or else it is the matrix multiplication exponent $\omega < 2.37$). Supposing the rows of $\boldsymbol{S}$ are chosen independently, and we denote them as the column vectors $\{\hat{\boldsymbol{s}}_j\}_{j=1}^c$, then we can also write

$$\mathbb{E}[\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}] = \boldsymbol{A}\boldsymbol{B}\,,$$

for a vector-valued random variable $\boldsymbol{s} := \hat{\boldsymbol{s}}\sqrt{c}$, and we may view the matrix multiplication as a mean over outer products, each costing $O(n^2)$ times to evaluate. In this perspective, we can view one instance of $\boldsymbol{A}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{B}$ as a *sample mean* of $\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}$ over c samples. For the sake of analysis, we use this picture and find it useful. However, we stress that the quantum and classical algorithm algorithmically adopt different approaches. The classical algorithms use the first formulation, explicitly computing $\boldsymbol{A}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{B}$, and thus can exploit fast matrix multiplication; whereas the quantum algorithms use the second formulation and explicitly implement a mean estimation of outer products.

Now that the problem is framed in terms of mean estimation, we can determine the convergence speed by investigating properties of the covariance matrix so that we may use the vector Bernstein inequality and Theorem 4 to analyze the classical and quantum algorithm respectively.

▶ **Lemma 10** (Covariance matrix for sketching algorithms)**.** *Consider the matrix-valued random variable* $\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}$*. The covariance matrix* $\boldsymbol{\Sigma}$ *satisfies*

$$\max_i[\boldsymbol{\Sigma}]_{ii} = \max_{ij}\operatorname{Var}\left[[\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}]_{ij}\right] = \max_{ij}\sum_{k\ell qr}[\boldsymbol{A}]_{ik}[\boldsymbol{B}]_{\ell j}[\boldsymbol{A}]_{iq}[\boldsymbol{B}]_{rj}\operatorname{Cov}\left[s_k s_\ell, s_q s_r\right],$$

$$\operatorname{Tr}[\boldsymbol{\Sigma}] = \mathbb{E}\left[\|\boldsymbol{A}\boldsymbol{B} - \boldsymbol{A}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{B}\|_F^2\right] = \sum_{ij}\sum_{k\ell qr}[\boldsymbol{A}]_{ik}[\boldsymbol{B}]_{\ell j}[\boldsymbol{A}]_{iq}[\boldsymbol{B}]_{rj}\operatorname{Cov}\left[s_k s_\ell, s_q s_r\right].$$

Lemma 10 shows that the convergence of both classical and quantum algorithms is controlled simply by covariances of elements of $\boldsymbol{s}$. In what follows we present two choices for $\boldsymbol{s}$, and their implications for both classical and quantum algorithms.

C.2 Classical algorithms

Let us now consider a general rectangular matrix product $\boldsymbol{AB}$ for $\boldsymbol{A} \in \mathbb{R}^{n\times q}$, $\boldsymbol{B} \in \mathbb{R}^{q\times m}$. In classical algorithms we will sample one matrix $\boldsymbol{S} \in \mathbb{R}^{c\times q}$ with rows constructed independently from some distribution, and explicitly compute $\boldsymbol{AS}^T\boldsymbol{SB}$ (recall that in order to analyze convergence, we will view this as taking a sample mean over c samples of $\boldsymbol{Ass}^T\boldsymbol{B}$). This costs $O((mn + nq + mq)c^{\omega-2})$ runtime. We recall that the approximation error in Frobenius norm and max-norm are controlled by $\mathrm{Tr}[\boldsymbol{\Sigma}]$ and $\max_i[\boldsymbol{\Sigma}]_{ii}$ respectively.

C.2.1 Tug-of-war sketch

In the tug-of-war sketch [3] (also known as AMS sketch), $\boldsymbol{S}$ is chosen to be the dense matrix with entries $[\boldsymbol{S}]_{ij} = h_i(j)/\sqrt{c}$ with each $h_i : [n] \to \{-1, 1\}$ drawn uniformly at random.[3]. In the mean estimation picture, our sketching vectors $\boldsymbol{s}$ thus satisfy $s_j = h(j)$.

▶ **Lemma 11** (Tug-of-war sketch covariance matrix). *For the tug-of-war sketch, the covariance matrix $\boldsymbol{\Sigma}$ satisfies*

$$\mathrm{Tr}[\boldsymbol{\Sigma}] = \|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2 + \|\boldsymbol{AB}\|_F^2 - 2\sum_{k=1}^{n}\|[\boldsymbol{A}]_{\bullet k}\|_2^2\|[\boldsymbol{B}]_{k\bullet}\|_2^2\,,$$

$$\max_i[\boldsymbol{\Sigma}]_{ii} = \|\boldsymbol{A}\|_{2,\infty}^2\|\boldsymbol{B}^T\|_{2,\infty}^2 + \|\boldsymbol{AB}\|_{\max}^2 - 2\min_{ij}\sum_k[\boldsymbol{A}]_{ik}^2[\boldsymbol{B}]_{kj}^2\,.$$

Proof sketch. We can first verify that $\mathbb{E}[\boldsymbol{ss}^T] = \boldsymbol{I}$. The variance can then be checked to be $\mathrm{Var}\,[s_is_j] = (1 - \delta_{ij})$. Likewise, the covariance (for $\{i, j\} \neq \{k, \ell\}$) can be evaluates to zero. This means that overall, we have $\mathrm{Cov}\big[s_is_j, s_ks_\ell\big] = (\delta_{ik}\delta_{j\ell} + \delta_{i\ell}\delta_{jk})(1 - \delta_{ij})$. We can substitute this into the expressions in Lemma 10 to give the stated results. ◀

With this, we can write down complexities for approximate matrix multiplication using the tug-of-war sketch. This was detailed for Frobenius norm approximation in [20], and we use the above to write the result for max-norm approximation also.

▶ **Theorem 12** (Classical tug-of-war algorithm, 2 matrices). *Let $\boldsymbol{A} \in \mathbb{R}^{n\times q}$, $\boldsymbol{B} \in \mathbb{R}^{q\times m}$. By sampling a single tug-of-war matrix of appropriate dimension, we can return a $\boldsymbol{C} \in \mathbb{R}^{n\times m}$ in the circuit model with success probability at least $(1 - \delta)$, which satisfies*

$$\|\boldsymbol{C} - \boldsymbol{AB}\|_{\max} \leq \varepsilon \quad \textit{in time} \quad O\left((mn + nq + qm)\left(\frac{\|\boldsymbol{A}\|_{2,\infty}^2\|\boldsymbol{B}^T\|_{2,\infty}^2}{\varepsilon^2}\right)^{\omega-2}\log\left(\frac{1}{\delta}\right)\right)\,,$$

$$\|\boldsymbol{C} - \boldsymbol{AB}\|_F \leq \varepsilon \quad \textit{in time} \quad O\left((mn + nq + qm)\left(\frac{\|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2}{\varepsilon^2}\right)^{\omega-2}\log\left(\frac{1}{\delta}\right)\right)\,.$$

Proof sketch. We can directly use matrix Bernstein inequalities (see full version of paper for pedagogical introduction), with expressions for covariance properties given by Lemma 11. The max-norm algorithm follows by choice of $c = L = 2\|\boldsymbol{A}\|_{2,\infty}^2\|\boldsymbol{B}^T\|_{2,\infty}^2\log(nm/\delta)/\varepsilon^2$ for the max-norm algorithm and $c = L = 2\|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2\log(1/\delta)/\varepsilon^2$ for the Frobenius norm algorithm. Finally, as discussed at the top of this section, the time complexity of computing $\boldsymbol{AS}^T\boldsymbol{SB}$ classically is $O((mn + nq + mq)c^{\omega-2})$. ◀

[3] In fact, drawing entries randomly from a 4-wise independent hash family suffices [20], though we will not exploit this.

C.2.2 Column-sample sketch

In the algorithm of Drineas, Kannan and Mahoney, for a matrix product of $\boldsymbol{A} \in \mathbb{R}^{n\times q}$ with $\boldsymbol{B} \in \mathbb{R}^{q\times m}$, $\boldsymbol{S} \in \mathbb{R}^{c\times q}$ is chosen to be a row 1-sparse matrix with entries populated as $[\boldsymbol{S}]_{i\bullet} = \boldsymbol{e}_j^T/\sqrt{cp_j}$ with probability p_j, where we denote $\boldsymbol{e}_j$ as the unit column vector with non-zero jth entry, and for some probability distribution $\{p_k\}_k$ [12]. This is their general framework, and in the end $\{p_k\}_k$ is judiciously chosen to minimize the convergence time for their algorithm in Frobenius norm approximation – note, however, that the minimizing distribution for the Frobenius norm does not optimize convergence in other norms, and indeed we find this not to be the case. In the mean estimation picture, we can write sketching vectors chosen as $\boldsymbol{s} = \boldsymbol{e}_j/\sqrt{p_j} \in \mathbb{R}^q$ with probability p_j.

▶ **Lemma 13** (Column-sample sketch covariance). *For the column-sample sketch as described above we have* $\mathrm{Cov}[s_i^2, s_j^2] = \frac{1}{p_i}\delta_{ij} - 1$ *and* $\mathrm{Cov}[s_i s_j, s_k u_\ell] = 0$ *for and* $i \neq j$ *or* $k \neq \ell$. *This implies that*

$$\mathrm{Tr}[\boldsymbol{\Sigma}] = \sum_{i=1}^{n} \frac{1}{p_i}\|[\boldsymbol{A}]_{\bullet,i}\|_2^2\|[\boldsymbol{B}]_{i,\bullet}\|_2^2 - \|\boldsymbol{AB}\|_F^2\,,\ \max_i[\boldsymbol{\Sigma}]_{ii} = \max_{ij}\left(\sum_k \frac{1}{p_k}[\boldsymbol{A}]_{ik}^2[\boldsymbol{B}]_{kj}^2 - (\boldsymbol{AB})_{ij}^2\right).$$

Similar to the tug-of-war sketch the above result is obtained by computing relevant covariances and substituting them into the bound in Lemma 10. We refer the reader to the full version of our paper for the full proof.

The probability distribution can now be optimized – note that the optimal probability distribution for the Frobenius norm (as found in [12]) may differ from the optimal distribution for the max-norm, and indeed we find this to be the case.

▶ **Theorem 14** (Classical column-sampling algorithm, 2 matrices). *Let* $\boldsymbol{A} \in \mathbb{R}^{n\times q}$, $\boldsymbol{B} \in \mathbb{R}^{q\times m}$. *By sampling a single column-sampling matrix of appropriate dimension, we can return a matrix* $\boldsymbol{C} \in \mathbb{R}^{n\times m}$, *with success probability at least* $(1-\delta)$, *which satisfies*

$$\|\boldsymbol{C} - \boldsymbol{AB}\|_{\max} \leq \varepsilon \quad \textit{in time} \quad O\left(nm\left(\frac{\|\boldsymbol{A}^T\|_{\infty,2}^2\|\boldsymbol{B}\|_{\infty,2}^2}{\varepsilon^2}\right)^{\omega-2}\log\left(\frac{1}{\delta}\right)\right),$$

$$\|\boldsymbol{C} - \boldsymbol{AB}\|_F \leq \varepsilon \quad \textit{in time} \quad O\left(nm\left(\frac{\|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2}{\varepsilon^2}\right)^{\omega-2}\log\left(\frac{1}{\delta}\right)\right),$$

where ω *is the matrix multiplication exponent.*

Proof sketch. Drineas et al. show that the minimizing probability distribution for $\mathrm{Tr}[\boldsymbol{\Sigma}]$ is the one with $p_k \propto \|[\boldsymbol{A}]_{\bullet,k}\|_2\|[\boldsymbol{B}]_{k,\bullet}\|_2$. The Frobenius norm algorithm follows by inspecting the matrix Bernstein inequalities and taking $c = L = \left(\sum_i \|[\boldsymbol{A}]_{\bullet,i}\|_2\|[\boldsymbol{B}]_{i,\bullet}\|_2\right)^2 \leq \|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2/\varepsilon^2$. For the max-norm algorithm we can write

$$\max_i[\boldsymbol{\Sigma}]_{ii} \leq \sum_k \max_{ij}[\boldsymbol{A}]_{ik}^2[\boldsymbol{B}]_{kj}^2/p_k - \min_{ij}(\boldsymbol{AB})_{ij}^2\,.$$

Here, one can check that the minimizing probability distribution for the upper bound is the one with probabilities $p_k \propto \max_{ij}[\boldsymbol{A}]_{ik}[\boldsymbol{B}]_{kj}$. The max-norm algorithm follows by choice of $c = L = \|\boldsymbol{A}\|_{\infty,2}^2\|\boldsymbol{B}^T\|_{\infty,2}^2/\varepsilon^2$ and inspecting the max-norm matrix Bernstein inequality.

Finally, we note that we can bring the complexity down with respect to dependencies on the matrix dimension, by observing that the computational cost of evaluating $\boldsymbol{AS}^T\boldsymbol{SB}$ is $O(nmc^{\omega-2})$ due to the sparsity of $\boldsymbol{S}$ (first evaluate $\boldsymbol{AS}^T$ and $\boldsymbol{SB}$, taking time $O(nc)$ and $O(mc)$ respectively, and then evaluate their product using fast matrix multiplication), and there is no dependence on the inner dimension. ◀

C.3 Quantum algorithm

Now we quantize the tug-of-war and column-sample matrix sketches. Here we coherently construct $\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}$ for given $\boldsymbol{s} \in \mathbb{R}^q$ from the tug-of-war sketch (see Appendix C.2.1) and use quantum multivariate mean estimation to estimate $\mathbb{E}[\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}]$. As the algorithm is effectively computing matrix-vector products, it is unclear how to exploit fast matrix multiplication in this setting, unlike in the classical algorithms discussed in the previous section which can bunch together samples. Both algorithms have the same complexity for square matrices, but the quantum algorithm based on the column-sample sketch uses a QROM as the sketching vectors are constructed from column and row norms of the matrices. When considering rectangular matrices, ability to use QROM yields an advantage for the column-sample sketch over the tug-of-war sketch for matrix products with large inner dimension ($\boldsymbol{A}$ "wide", $\boldsymbol{B}$ "tall"), as we can exploit the sparsity of the sketching vectors.

▶ **Theorem 15** (Quantum tug-of-war algorithm, 2 matrices)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times q}$, $\boldsymbol{B} \in \mathbb{R}^{q\times m}$. We have a quantum algorithm which works in the circuit model and returns a matrix $\boldsymbol{C} \in \mathbb{R}^{n\times m}$, with success probability at least $(1-\delta)$, which satisfies*

$$\|\boldsymbol{C} - \boldsymbol{A}\boldsymbol{B}\|_{\max} \le \varepsilon \quad \textit{in time} \quad \widetilde{O}\left((nm + mq + nq)\left(\frac{\|\boldsymbol{A}\|_{2,2}\|\boldsymbol{B}\|_{2,2}}{\varepsilon}\right)\log\left(\frac{1}{\delta}\right)\right),$$

Proof. In order to perform quantum multivariate mean estimation (as outlined in Section A.3), we use the following oracles. The action of the binary oracle may be written as $O_{\boldsymbol{X}} : |\boldsymbol{s}\rangle|0\rangle \mapsto |\boldsymbol{s}\rangle|\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}\rangle$, which can be obtained in $\widetilde{O}(nm + mq + qn)$ time, given QROM access to $\boldsymbol{A}$ and $\boldsymbol{B}$. Note that we will use the first register in superposition, but we do not need fast coherent read or write for the entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ – thus with a binary encoding of both we can also achieve the same runtime in the circuit model, without QROM. For the probability oracle, we relax the 4-wise independence assumption, and allow our event space to include all signed bitstrings $\Omega = \{-1,+1\}^n$. Here, the probability oracle has action $U_{\mathbb{P}} : |0\rangle \xrightarrow{\boldsymbol{H}^{\otimes n}} \frac{1}{\sqrt{2^n}}\sum_{2\boldsymbol{s}'-\boldsymbol{1}_n\in\Omega}|\boldsymbol{s}'\rangle$ which is instantiated simply by the Hadamard operation in constant time. Finally, we can write the action of the inner product oracle as $O_{\boldsymbol{Y},\boldsymbol{X}} : |\boldsymbol{Y}\rangle|\boldsymbol{s}\rangle|0\rangle \mapsto |\boldsymbol{Y}\rangle|\boldsymbol{s}\rangle|\mathrm{Tr}[\boldsymbol{Y}^T\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}]\rangle$, for chosen instances $\boldsymbol{Y} \in (-\frac{1}{2},\frac{1}{2})^{n\times m}$ from the mean estimation algorithm of Theorem 4. We remark that the Hilbert-Schmidt (trace) inner product we denote above is entirely equivalent to the vector inner product, should $\boldsymbol{Y}$ and $\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}$ be unfurled into column vectors. This inner product can be instantiated as

$$\begin{aligned}|\boldsymbol{Y}\rangle|\boldsymbol{s}\rangle|0\rangle|0\rangle &\xrightarrow{O_{\boldsymbol{X}}} |\boldsymbol{Y}\rangle|\boldsymbol{s}\rangle|\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}\rangle|0\rangle \\ &\longrightarrow |\boldsymbol{Y}\rangle|\boldsymbol{s}\rangle|\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}\rangle|\mathrm{Tr}[\boldsymbol{Y}^T\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}]\rangle \xrightarrow{O_{\boldsymbol{X}}^\dagger} |\boldsymbol{Y}\rangle|\boldsymbol{s}\rangle|0\rangle|\mathrm{Tr}[\boldsymbol{Y}^T\boldsymbol{A}\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{B}]\rangle\,,\end{aligned}$$

where the second step costs $O(nm)$. The quantum multivariate mean estimation routine of Theorem 4 can thus be instantiated with complexity $\widetilde{O}((nm+nq+mq)N)$, attaining max-norm error $\sqrt{\mathrm{Tr}[\boldsymbol{\Sigma}]}\log(nm/\delta)/N$ with success probability at least $(1-\delta)$, where $\mathrm{Tr}[\boldsymbol{\Sigma}]$ is given in Lemma 11. The theorem statement is satisfied by picking $N = \sqrt{2}\|\boldsymbol{A}\|_F\|\boldsymbol{B}\|_F\log(nm/\delta)/\varepsilon$. ◀

▶ **Theorem 16** (Quantum column-sample algorithm, 2 matrices)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times q}$, $\boldsymbol{B} \in \mathbb{R}^{q\times m}$. We have a quantum algorithm in the QROM model which returns a matrix $\boldsymbol{C} \in \mathbb{R}^{n\times m}$, with success probability at least $(1-\delta)$, which satisfies*

$$\|\boldsymbol{C} - \boldsymbol{A}\boldsymbol{B}\|_{\max} \le \varepsilon \quad \textit{in time} \quad \widetilde{O}\left(nm\left(\frac{\|\boldsymbol{A}\|_{2,2}\|\boldsymbol{B}\|_{2,2}}{\varepsilon}\right)\log\left(\frac{1}{\delta}\right)\right).$$

The analysis follows in the a similar way as in the proof of Theorem 15 and we refer the reader to the full version for the proof.

C.4 Multiple matrices

Now let's look at the general case of the product of k matrices which we label as $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$, with $\boldsymbol{A}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell+1}}$ for all $\ell \in [k]$. We approximate this by inserting $\boldsymbol{s}_\ell \boldsymbol{s}_\ell^T$ in between each matrix, that is we evaluate $\left(\Pi_{\ell=1}^{k-1} \boldsymbol{A}_\ell \boldsymbol{s}_\ell \boldsymbol{s}_\ell^T\right) \boldsymbol{A}_k$. This product costs $O(k \max_{i \neq j} n_i n_j)$ to evaluate classically (or quantumly, using coherent arithmetic). As for 2 matrices, we will estimate the mean of this quantity, and can repeat the analysis as before, with both quantum and classical algorithms analyzed through the lens of mean estimation, if we can bound the relevant covariances. We omit proofs here and leave them to the full version of our paper.

▶ **Theorem 17** (Sketching algorithms for multiple matrices). *Consider a matrix product of k matrices $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$, with $\boldsymbol{A}_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ for all $\ell \in [k]$. With classical preprocessing time linear in the size of the input up to polylogarithmic factors and success probability at least $(1-\delta)$, we give a classical algorithm which gives*

$$\|\boldsymbol{C} - \boldsymbol{AB}\|_F \leq \varepsilon \quad \text{in time} \quad \widetilde{O}\left(3^k \cdot \max_{i \neq j} n_i n_j \cdot \frac{\|\boldsymbol{A}_1\|_{2,2}^2 \cdots \|\boldsymbol{A}_k\|_{2,2}^2}{\varepsilon^2} \log\left(\frac{1}{\delta}\right)\right),$$

and a quantum algorithm which gives

$$\|\boldsymbol{C} - \boldsymbol{AB}\|_{\max} \leq \varepsilon \quad \text{in time} \quad \widetilde{O}\left(3^{k/2} \cdot \max_{i \neq j} n_i n_j \cdot \frac{\|\boldsymbol{A}_1\|_{2,2} \cdots \|\boldsymbol{A}_k\|_{2,2}}{\varepsilon} \log\left(\frac{1}{\delta}\right)\right).$$

So far we have not exploited fast matrix multiplication. We can instead insert $\boldsymbol{S}_\ell^T \boldsymbol{S}_\ell$ in between each matrix. That is, we approximate $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$ by evaluating $\left(\Pi_{\ell=1}^{k-1} \boldsymbol{A}_\ell \boldsymbol{S}_\ell^T \boldsymbol{S}_\ell\right) \boldsymbol{A}_k$. Note that unlike in the case of a product of 2 matrices, there is no longer a simple interpretation of this as a sample mean over choices of $\boldsymbol{s}_\ell \boldsymbol{s}_\ell^T$. Thus, now we need to analyze $\mathrm{Tr}[\boldsymbol{\Sigma}]$ for the matrix-valued random variable $\left(\Pi_{\ell=1}^{k-1} \boldsymbol{A}_\ell \boldsymbol{S}_\ell^T \boldsymbol{S}_\ell\right) \boldsymbol{A}_k$ explicitly.

▶ **Theorem 18** (Sketching algorithm for multiple matrices exploiting fast matrix multiplication). *Consider a matrix product of k matrices $\boldsymbol{A}_1 \cdots \boldsymbol{A}_k$, with $\boldsymbol{A}_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ for all $\ell \in [k]$. With classical preprocessing time linear in the size of the input up to polylogarithmic factors, we give a classical algorithm which returns $\boldsymbol{C}$ satisfying*

$$\|\boldsymbol{C} - \boldsymbol{AB}\|_F \leq \varepsilon \quad \text{in time} \quad \widetilde{O}\left(3^{k(\omega-2)} \cdot \max_{i \neq j} n_i n_j \left(\frac{\|\boldsymbol{A}_1\|_{2,2}^2 \cdots \|\boldsymbol{A}_k\|_{2,2}^2}{\varepsilon^2}\right)^{\omega-2} \log\left(\frac{1}{\delta}\right)\right),$$

with success probability at least $(1-\delta)$.