Quantum-inspired algorithms for solving low-rank linear equation systems with logarithmic dependence on the dimension

Quantum-Inspired Algorithms for Solving Low-Rank Linear Equation Systems with Logarithmic Dependence on the Dimension

Nai-Hui Chia
Department of Computer Science, University of Texas at Austin, TX, USA
Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, MD, USA
nchia@umd.edu

András Gilyén
QuSoft, CWI and University of Amsterdam, The Netherlands
agilyen@caltech.edu

Han-Hsuan Lin
Department of Computer Science, University of Texas at Austin, TX, USA
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
linhh@cs.nthu.edu.tw

Seth Lloyd
MIT, Departments of Mechanical Engineering and Physics, Cambridge, MA, USA
Xanadu, Toronto, Canada
slloyd@mit.edu

Ewin Tang
University of Washington, Seattle, WA, USA
ewint@cs.washington.edu

Chunhao Wang
Department of Computer Science, University of Texas at Austin, TX, USA
Department of Computer Science and Engineering, Pennsylvania State University, Philadelphia, PA, USA
cwang@psu.edu

Abstract

We present two efficient classical analogues of the quantum matrix inversion algorithm [16] for low-rank matrices. Inspired by recent work of Tang [27], assuming length-square sampling access to input data, we implement the pseudoinverse of a low-rank matrix allowing us to sample from the solution to the problem $Ax = b$ using fast sampling techniques. We construct implicit descriptions of the pseudo-inverse by finding approximate singular value decomposition of A via subsampling, then inverting the singular values. In principle, our approaches can also be used to apply any desired "smooth" function to the singular values. Since many quantum algorithms can be expressed as a singular value transformation problem [15], our results indicate that more low-rank quantum algorithms can be effectively "dequantised" into classical length-square sampling algorithms.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases sublinear algorithms, quantum-inspired, regression, importance sampling, quantum machine learning

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.47

Related Version This paper is a merge of two submitted papers, whose full versions are available at `https://arxiv.org/abs/1811.04852` and `https://arxiv.org/abs/1811.04909`.

Funding NHC, HHL, and CW were supported by Scott Aaronson's Vannevar Bush Faculty Fellowship. AG was supported by ERC Consolidator Grant QPROGRESS and partially supported by QuantERA project QuantAlgo 680-91-034. SL was supported by ARO and OSD under a Blue Sky Program.

Acknowledgements NHC, HHL, and CW thank Scott Aaronson for the valuable feedback on a draft of this paper. AG thanks Márió Szegedy for introduction to the problem and sharing insights, Ravi Kannan for helpful discussions and Ronald de Wolf for useful comments on the manuscript. We thank the anonymous reviewers for their valuable feedback on both submissions.

1 Introduction

Quantum computing provides potential exponential speed-ups over classical computers for a variety of linear algebraic tasks, including an operational version of matrix inversion [16]. Recently, inspired by the quantum algorithm for recommendation systems [20], Tang showed how to generalise the well-known FKV algorithm [13] to sample from the singular vectors of low-rank matrices [27] and to implement principal component analysis [26]. Intriguingly, Tang's work suggested that many of the quantum algorithms for low-rank matrix manipulation [25] can be extended to provide fast *classical* algorithms under suitable sampling assumptions, achieving logarithmic dependence on the dimension. In this work, we show that such exponential speed-ups are indeed possible in the case of low-rank matrix inversion. That is, we present proof-of-principle algorithms for approximately inverting low-rank matrices in runtime that is logarithmic in the dimensions. Our treatment is self-contained and improves some aspects of previous approaches [27], leading to smaller exponents in our runtime bounds.

The main motivation of this paper comes from the quantum algorithm of Harrow, Hassidim, and Lloyd [16] for matrix inversion. This algorithm is central to the current landscape of quantum machine learning algorithms [4]. HHL and its improved variants [1, 3, 5] have many applications in quantum machine learning, including least squares approximation [28], support vector machines [24], and kernel-based methods such as Gaussian process regression [30]. However, HHL requires strong input assumptions: it needs some efficient quantum way of accessing the input matrix A. If the matrix A is sparse [16] or has low Frobenius norm (thereby can be well approximated by a low-rank matrix) and is stored in quantum RAM using an efficient data structure [20, 29], then it is possible to run the HHL algorithm efficiently. It is well known that HHL for sparse input matrices is BQP-complete [16], so it arguably achieves exponential speedups in certain situations. However, the small-Frobenius-norm scenario is often more relevant for machine learning problems. By dequantising HHL in the low-rank regime, we indicate that this use case does not yield exponential speedups by default, contrary to earlier high hopes. The situation is analogous to the case of quantum recommendation systems [20] which was initially also believed to provide exponential quantum speedups. To summarise, the intuitive message of our work is that quantum matrix inversion does not give exponential quantum speedups for low-rank datasets (unless for some reason another high-rank, e.g. Fourier, transformation plays a crucial role in the problem).

In this paper, we give two algorithms (Algorithm 1 and Algorithm 2) for solving $Ax = b$ in time depending only logarithmically on input dimension, by assuming "length-square access" to input and applying sketching techniques exploiting this access.[1] We want to solve $Ax = b$, where we are given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, and wish to recover $x \in \mathbb{R}^n$. The equation might not have a solution, but we can always find an x minimizing $\|Ax - b\|^2$. Namely, $x = A^+b$ works, where A^+ is the pseudoinverse of A. If $A = \sum_{\ell=1}^{k} \sigma_\ell \, u^{(\ell)} {v^{(\ell)}}^T$ is a singular value decomposition of A, such that $\sigma_\ell > 0$, then the pseudoinverse is simply $A^+ = \sum_{\ell=1}^{k} v^{(\ell)} {u^{(\ell)}}^T/\sigma_\ell$, and $x = \sum_{\ell=1}^{k} v^{(\ell)} \langle u^{(\ell)}, b\rangle/\sigma_\ell$. The central tool to our result is

[1] This paper is a merge of two arXiv preprints [9, 14], containing Algorithm 1 and Algorithm 2, respectively.

working with a vector v via *length-square access*, allowing queries to the norm $\|v\|$ and individual coordinates v_i, as well as providing samples from the *length-square distribution* $\frac{|v_i|^2}{\|v\|^2}$. In our algorithms we assume length-square access to the input matrix A, requiring length-square access to the rows of A as well as to the vector of row-norms. The output of our algorithms is a description of an approximate solution $\tilde{x}$, providing efficient length-square access to $\tilde{x}$.

Applications of the classical stochastic regression algorithms presented here include a wide variety of data analysis problems. Consider, for example, the problem of finding the optimal investment portfolio amongst n stocks. As shown in [23], under typical assumptions used for classical portfolio management, finding and sampling from the optimal portfolio and mapping out the optimal risk-return curve is a low-rank matrix inversion problem which can be solved on a quantum computer in complexity that has logarithmic dependence on the dimensions; in contrast conventional classical portfolio optimization methods have polynomial dimension dependence. The results presented here show that our quantum-inspired classical algorithms can similarly allow one to map out the risk-return curve and sample from the optimal portfolio with classical complexity that depends logarithmically on the dimensions.

1.1 Discussion and related work

In a 2019 preprint, Arrazola et al. [2] implements and benchmarks the algorithms described in this work. They conclude that our algorithms perform well for very low-rank matrices, but that the key aspects of this work that make it comparable to quantum algorithms (e.g. inner product estimation and length-square sampling of the output vector) don't provide significant improvements over naive classical approaches. While such results suggest that quantum-inspired algorithms typically don't improve over existing classical techniques, our main conclusion – that quantum algorithms for low-rank linear systems likely don't yield exponential speedups – still stands. Notably, we are lacking complete end-to-end analyses of QML algorithms to compare these benchmarks to, which we need in order to make solid conclusions about whether quantum algorithms would suffer comparable slowdowns as these quantum-inspired ones, particularly with the high overhead of running quantum algorithms using current techniques.

Although in this paper we focus on implementing the pseudo-inverse of a matrix by inverting the singular values, one could in principle apply any desired function to the singular values, particularly for problems assuming close-to-low-rank input. A follow-up line of work [8, 10, 6, 11, 18] does precisely this to effectively dequantise other quantum machine learning results. The work of Chia et al. [7] unifies this line of research, showing that all of these applications can be studied in a common framework via singular value transformation, a problem known to generalize many quantum algorithms [15]. This supports Tang's suggestion [26] that many quantum algorithms can be effectively turned to randomised classical algorithms via length-square sampling techniques incurring only polynomial overheads. Exponential quantum speed-ups appear to be tightly related to problems where high-rank matrices play a crucial role, like in Hamiltonian simulation or the Fourier transform. However, more work remains to be done on understanding the class of problems for which exponential quantum speed-up can be achieved.

2 Our algorithms

We use the following notation. For $v \in \mathbb{C}^d$ we denote the Euclidean norm by $\|v\|$. For a matrix $A \in \mathbb{C}^{m\times n}$ we denote by $\|A\|$ the operator norm, and by $\|A\|_F$ the Frobenius norm. We use notation $A_{i\cdot}$ for the i-th row, $A_{\cdot j}$ for the j-th column, and $A^\dagger$ for the adjoint of A. We

use "bra-ket" notation: for $v \in \mathbb{C}^d$ we denote the corresponding column vector by $|v\rangle \in \mathbb{C}^{d\times 1}$, and by $\langle v| \in \mathbb{C}^{1\times d}$ its adjoint. Accordingly we denote the inner product $\langle v, w\rangle$ by $\langle v|w\rangle$. For a nonzero vector $x \in \mathbb{C}^n$, we denote by $\mathcal{D}_x$ the probability distribution on $\{1, \ldots, n\}$ where the probability that i is chosen is defined as $\mathcal{D}_x(i) = |x(i)|^2/\|x\|^2$ for all $i \in \{1, \ldots, n\}$.

2.1 Sampling

To have any hope for solving matrix inversion in sublinear time, we need additional assumptions on the input. In the case of quantum machine learning, the main assumption is typically that one can prepare certain quantum states related to the input matrix. We work with the quantum-inspired classical analogue of state preparation which is length-square sampling.

▶ **Definition 1** (Length-square distribution). *For a non-zero vector $v \in \mathbb{C}^n$, we define the* length-square probability distribution $q^{(v)}$ *on $[n]$ to satisfy $q_i^{(v)} := \frac{|v_i|^2}{\|v\|^2}$.*

This is a classical analogue of preparing quantum states $|v\rangle$: if v describes a normalised pure quantum state, the above distribution is exactly the distribution we get through measurement in the computational basis, as described by Born's rule. Moreover, the usual data structures that support fast quantum state preparation also support fast length-square sampling. We will describe a prominent example of such a data structure: it has been used to support previous quantum-inspired algorithms [27, 26], QML algorithms [22, 21, 20], and randomised linear algebra algorithms [13]. This data structure supports all the operations necessary to run our matrix inversion algorithms, and yields only logarithmic overheads in the runtime.

Specifically, to run Algorithm 1, we need *length-square access* to the input matrix A (and standard query access to b). We define length-square access below, first for a vector, then for a matrix.

▶ **Definition 2** (Length-square access to a vector). *We say that we have length-square access to the vector $v \in \mathbb{C}^n$ if we can request a sample from the distribution $q^{(v)}$ at cost $S(v)$. We also assume that we can query the elements of v with cost $Q(v)$, and that we can query the value of $\|v\|$ with cost $N(v)$.*[2] *We denote by $L(v) := S(v) + Q(v) + N(v)$ the overall access cost.*

▶ **Definition 3** (Length-square access to a matrix). *We say that we have (row) length-square access to the matrix $A \in \mathbb{C}^{m\times n}$ if we have length-square access to the rows $A_{i\cdot}$ of A for all $i \in [m]$ and length-square access to the vector of row norms $a \in \mathbb{R}^m$, where $a_i := \|A_{i\cdot}\|$. We denote by $L(A)$ the complexity of the length-square access to A.*

Note that length-square access to A implies the ability to determine $\|A\|_F$ in $N(A)$ time.

2.1.1 Data structures for length-square access

We can build a data structure for length-square access to a vector $v \in \mathbb{C}^n$ in the following way. For each $i \in [n]$, we store v_i as $|v_i|^2$ and $v_i/|v_i|$. Then, we build a binary tree, with $|v_i|^2$ as the leaves. In each interior node we store the sum of the left node and the right node. In particular, the root node stores $\|v\|^2$. Further, we can perform length-squared sampling on v in $\mathcal{O}(\log n)$ time by starting from the root node and recursing on a child with probability

[2] We assume for simplicity that $S(v), Q(v)$, and $N(v) \geq 1$.

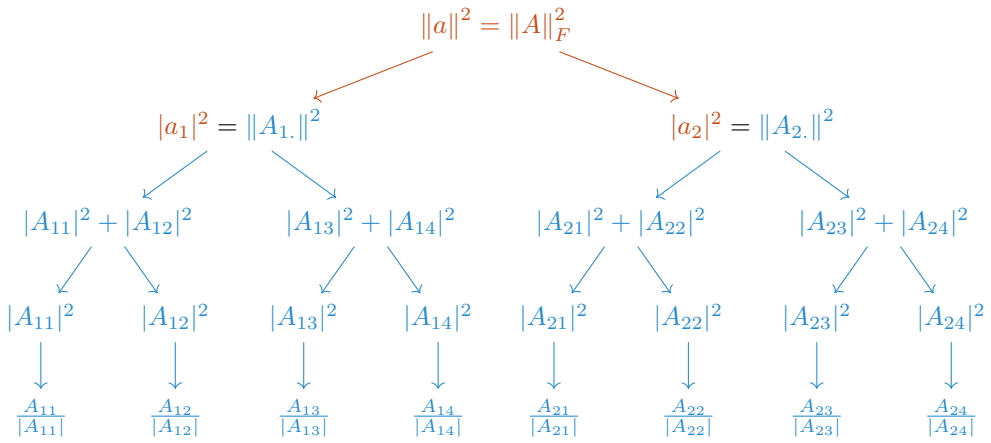

Figure 1 The length-square data structure for $A \in \mathbb{C}^{2\times 4}$. We compose the data structure for a in orange with the data structure for A's rows in blue.

proportional to its weight. Finally, this data structure supports finding and updating entries in $\mathcal{O}(\log n)$ time and can be modified to support sparsity, pruning down the tree to take space linear in the number of non-zero entries.

Now that we have length-square access data structures for a vector, we can get the same for a matrix $A \in \mathbb{C}^{m\times n}$ by simply creating the vector data structure for each row of A and a the vector of row norms (see Figure 1). We can view the data structure for a as simply placing another tree onto the root nodes of each row's tree. This view makes it evident that this data structure supports: $\mathcal{O}(\log mn)$ update time, $\mathcal{O}(\log n)$ time length-squared sampling to the rows of A and $\mathcal{O}(\log m)$ time length-squared sampling to a. In light of this data structure, for simplicity, some portions of our complexity analysis will assume $L(A) = \widetilde{\mathcal{O}}(1)$.

2.2 Main results

For simplicity, we treat the case when the matrix A has rank $k \ll m, n$, and does not have too small singular values.[3] In order to execute our algorithms, we use length-square sampling techniques, which have found great applications in randomised linear algebra [19] and the recent quantum-inspired classical algorithm for recommendation systems [27]. For simplicity we assign unit cost to arithmetic operations such as addition or multiplication of real or complex numbers, assuming that all numbers are represented with a small number of bits.

We present two algorithms, whose only essential difference is the parameters chosen. The time complexity of the first is better when $\mathrm{rank}(A) < \|A\|^2/\sigma_{\min}^2$, while the time complexity of the second is better when $\mathrm{rank}(A) > \|A\|^2/\sigma_{\min}^2$, where $\sigma_{\min}(A)$ denotes the minimum nonzero singular value of A.

In our first algorithm, by renormalizing, we can assume that $\|A\| \leq 1$ and $\|A^+\| \leq \kappa$. Our program is the following: we first show how to describe an approximate singular value decomposition $\sum_{\ell=1}^{k} \tilde{\sigma}_\ell |\tilde{u}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|$ using a succinct representation of the vectors. Then we show how to estimate the values $\langle\tilde{u}^{(\ell)}|b\rangle/\tilde{\sigma}_\ell$ via sampling, and how to sample from the

[3] This assumption can be relaxed by inverting A on the "well-conditioned" subspace, and dealing with small singular values similarly to the earlier works [13, 27], see follow-up work [7] for more details.

corresponding linear combination of the vectors $\tilde{v}^{(\ell)}$. The overall algorithm allows us to sample, query elements, and estimate the norm of an approximate solution $\tilde{x} \approx A^+ b$ to the equation $Ax = b$ in time poly-logarithmic in the size of the matrix. Essentially, we get fast length-square access to $\tilde{x}$.

The idea of the approximate singular value decomposition of A using length-square sampling comes from [13]. Consider using length-square sampling to sample some rows R. If we sample enough rows, then $\left\|A^T A - R^T R\right\|$ is small as shown by Theorem 6. This means that R's singular values and right singular vectors are close to A's singular values and right singular vectors. Further, as shown by Lemma 7, by applying the matrix A to R's right singular vectors, we can recover approximate left singular vectors of A. This is promising, but since R's rows are still length n, computing its singular value decomposition is prohibitively slow. However, we can apply the trick once more! We can sample columns of R to form the submatrix C. Again the singular values and left singular vectors of C are very close to the singular values and left singular vectors of R provided that $\left\|RR^T - CC^T\right\|$ is small. Since C will have logarithmic size in terms of the input dimensions, we can compute its singular value decomposition quickly, and using the above techniques, translate this to an approximate singular value decomposition of A.

Algorithm 1 Low-rank stochastic regression via length-square sampling.

Input: A vector $b \in \mathbb{C}^m$ and a matrix $A \in \mathbb{C}^{m \times n}$ s.t. $\|A\| \leq 1$, $\operatorname{rank}(A) = k$ and $\|A^+\| \leq \kappa$.

Goal 1: Query elements of a vector $\tilde{x}$ such that $\|\tilde{x} - x\| \leq \varepsilon \|x\|$ for $x = A^+ b$.

Goal 2: Sample from a distribution 2ε-close in total-variation distance to $\frac{|x_j|^2}{\|x\|^2}$.

Goal 3: Output a ν such that $|\nu - \|x\|| \leq 3\varepsilon \|x\|$.

1: **Init:** Set $r = 2^{10} \ln\left(\frac{8n}{\eta}\right) \frac{\kappa^4 k^2 \|A\|_F^2}{\varepsilon^2}$ and $c = 2^6 \cdot 3^4 \ln\left(\frac{8r}{\eta}\right) \frac{\kappa^8 k^2 \|A\|_F^2}{\varepsilon^2}$.

2: **Sample rows:** Sample r row indices $i_1, i_2, \ldots, i_r$ according to the row norms $q_i^{(a)} = \frac{\|A_{i.}\|^2}{\|A\|_F^2}$. Define R to be the matrix whose s-th row is $\frac{\|A\|_F}{\sqrt{r}} \frac{A_{i_s.}}{\|A_{i_s.}\|}$.

3: **Sample columns:** Sample $s \in [r]$ uniformly, then sample a column index j according to $q_j^{(R_{s.})} = q_j^{(A_{i_s.})} = \frac{|A_{i_s j}|^2}{\|A_{i_s.}\|^2}$. Sample a total number of c column indices $j_1, j_2, \ldots, j_c$ this way. Define the matrix C whose t-th column is $\frac{\|R\|_F}{\sqrt{c}} \frac{R_{.j_t}}{\left\|R_{.j_t}\right\|} = \frac{\|A\|_F}{\sqrt{c}} \frac{R_{.j_t}}{\left\|R_{.j_t}\right\|}$.

4: **SVD:** Query all elements of A corresponding to elements of C. Compute C's top k left singular vectors $w^{(1)}, \ldots, w^{(k)}$ and corresponding singular values $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_k$.

5: **Approximate right singular vectors of A:** Implicitly define $\tilde{v}^{(\ell)} := \frac{1}{\tilde{\sigma}_\ell} R^\dagger w_s^{(\ell)}$.

6: **Matrix elements:** For each $\ell \in [k]$ compute $\tilde{\lambda}_\ell$ such that $\left|\tilde{\lambda}_\ell - \langle \tilde{v}^{(\ell)} | A^\dagger | b \rangle\right| = \mathcal{O}\left(\frac{\varepsilon \tilde{\sigma}_\ell^2 \|b\|}{\sqrt{k}}\right)$.

7: **Output:** Row indices $i_1, i_2, \ldots, i_r$ and $w := \sum_{\ell=1}^k \frac{\tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^3} w^{(\ell)} \in \mathbb{C}^r$ s.t. $\|w\| = \mathcal{O}\left(\kappa^2 \sqrt{k} \|b\|\right)$.

This implicitly describes $\tilde{x} := R^\dagger w$. The goals follow from length-square access to $\tilde{x}$ described in Lemma 12; the routines are summarised below.

Queries to $\tilde{x}$: $\tilde{x}_j = \sum R_{j,s}^\dagger w_s$, so query $R_{s,j}$ for all $s \in [r]$ and compute naively.

Sampling from $|\tilde{x}_j|^2 / \|\tilde{x}\|^2$: Perform rejection sampling; each round uses one sample according to $q_j^{(R_{s.})} = \frac{|R_{s,j}|^2}{\|R_{s,.}\|^2}$ for some $s \in [r]$ and r queries to entries of R, in expectation, sampling takes $\|w\|^2 \|A\|_F^2 / \|\tilde{x}\|^2$ rounds.

Estimating $\|\tilde{x}\|$: Perform rejection sampling for a fixed number of rounds; the probability of success over those rounds approximates $\|\tilde{x}\|^2 / (\|w\|^2 \|A\|_F^2)$.

The correctness and complexity of Algorithm 1 is summarized in the following theorem, which is proven in Section 3. (We only work out the constant factors for the number of rows and columns to be sampled, because these parameters dominate the complexity.)

▶ **Theorem 4** (For Algorithm 1). *Assume that A has rank at most[4] k, $\|A\| \leq 1$, $\|A^+\| \leq \kappa$, and the projection of b to the column space of A has norm $\Omega(\|b\|)$. Also assume that we have $\widetilde{\mathcal{O}}(1)$-time[5] query access to b and $\widetilde{\mathcal{O}}(1)$-time length-square access to A. Then Algorithm 1 solves $Ax = b$ up to ε-multiplicative accuracy, such that $\|\tilde{x} - A^+b\| \leq \varepsilon\|A^+b\|$ with probability at least $1-\eta$ and we can execute Algorithm 1 in complexity $\widetilde{\mathcal{O}}\left(\kappa^{16}k^6\|A\|_F^6/\varepsilon^6\right)$, outputting an implicit description of $\tilde{x}$ that supports length-square access, with $Q(v) = \widetilde{\mathcal{O}}\left(\kappa^4k^2\|A\|_F^2/\varepsilon^2\right)$, $S(v) = \widetilde{\mathcal{O}}\left(\kappa^8k^3\|A\|_F^4/\varepsilon^2\right)$ (with high probability), and $N(v) = \widetilde{\mathcal{O}}\left(\kappa^8k^3\|A\|_F^4/\varepsilon^4\right)$ (outputting an estimate of the norm to $(1 \pm \varepsilon)$ multiplicative error).*

Our second algorithm has a similar flavor as Algorithm 1.

Algorithm 2 The algorithm for sampling from and querying to A^+b.

1: **Input:** ϵ, κ', $\|A\|$, $\|A\|_F$, and $A \in \mathbb{C}^{m\times n}$ with the length square access as defined in Definition 3.
2: Set $s = \Theta(\frac{\|A\|^6\|A\|_F^2\kappa'^8\ln(n/\delta)}{\epsilon^2})$, and $p = \Theta(\frac{\|A\|^{10}\|A\|_F^2\kappa'^{12}\ln(n/\delta)}{\epsilon^2})$.
3: Independently sample s row indices $i_1, \ldots, i_s$ according to the probability distribution $\{P_1, \ldots, P_m\}$ where $P_i = \frac{\|A_{i\cdot}\|^2}{\|A\|_F^2}$.
4: Let $S \in \mathbb{C}^{s\times n}$ be the matrix formed by the normalized rows $A(i_t, \cdot)/\sqrt{pP_{i_t}}$ for $t \in \{1, \ldots, s\}$.
5: Independently sample p column indices $j_1, \ldots, j_p$ by the following procedure: first sample a row index $t \in \{1, \ldots, s\}$ uniformly at random; then sample a column index j from the probability distribution $\mathcal{D}_{A_{i_t\cdot}}$.
6: Let $W \in \mathbb{C}^{s\times p}$ be the matrix formed by the normalized columns $S_{\cdot j_t}/\sqrt{pP'_{j_t}}$ for $t \in \{1, \ldots, p\}$, where $P'_j = \sum_{\ell=1}^{s}\mathcal{D}_{A_{i_\ell\cdot}}(j)/p$, and $i_1, \ldots, i_s$ are the indices sampled in step 2.
7: Compute the singular values $\sigma_1, \ldots, \sigma_k$ of W and their corresponding left singular vectors $u_1, \ldots, u_k$, where k is the rank of W. Define the matrix $V \in \mathbb{C}^{n\times k}$ as $V_{\cdot j} = S^\dagger\frac{u_j}{\sigma_j}$.
8: Construct the vector $w \in \mathbb{C}^s$ as $w_i = V_{\cdot i}^\dagger A^\dagger b$ by applying Lemma 15.
9: Compute w' as $w'_i = w_i/\sigma_i^2$.
To query the j-th entry of of the solution: output the inner product $V_{j\cdot}w'$.
To sample from the solution: use Lemma 13 to sample from Vw'.

The correctness and complexity of Algorithm 2 is summarized in the following theorem, which we prove in Section 4.

▶ **Theorem 5** (For Algorithm 2). *Let $A \in \mathbb{C}^{m\times n}$ be a matrix whose length square access defined in Definition 3 can be obtained in $\widetilde{O}(1)$ time, and $b \in \mathbb{C}^m$ be a vector. Assume $\|b\| = 1$ and the norm of the projection of b onto the column space of A is at least some constant.*

[4] While running the algorithm we can actually detect if A has lower rank and adapt the algorithm accordingly.

[5] In this paper by $\widetilde{\mathcal{O}}(T)$ we hide poly-logarithmic factors in T, the dimensions m, n and the failure probability η.

Let[6] $\kappa' := 1/\sigma_{\min}(A)$*, where* $\sigma_{\min}(A)$ *is the minimum nonzero singular value of* A*. Then, Algorithm 2 succeeds with probability* $1-\delta$*, can (1) approximate* $(A^+b)_i$ *for a given index* $i \in \{1,\dots,n\}$ *with error* $\epsilon\|A^+b\|$*, and (2) sample from a distribution that is* ϵ*-close to* $\mathcal{D}_{A^+b}$ *with time complexity*

$$\widetilde{O}\left(\frac{\|A\|^{6\omega+4}\|A\|_F^{2\omega}\kappa'^{8\omega+4}}{\epsilon^{2\omega}}\right), \tag{1}$$

where $\omega < 2.373$ *is the matrix multiplication exponent.*

Note that our complexity bound has smaller exponents than e.g. [27]. This partly comes from the fact that we only consider low-rank matrices, but we also get improvements by adapting and reanalysing the FKV algorithm [13].

We can achieve $\widetilde{\mathcal{O}}(1)$-time for the input accesses in Theorems 4 and 5 using, for example, the data structure in Section 2.1.1. If input access takes some different $\mathcal{O}(L)$ time instead, Algorithm 1's runtime increases by an additive term of $\widetilde{\mathcal{O}}\big(L\kappa^{12}k^4\|A\|_F^4/\varepsilon^4\big)$, which is L times the query complexity; the runtimes of $S(v)$ and $N(v)$ also increase by a factor of L. Similarly, Algorithm 2's runtime increases by an additive term at most $\widetilde{\mathcal{O}}\big(L\kappa'^{20}\|A\|^{16}\|A\|_F^4/\varepsilon^4\big)$.

For comparison with the quantum analogue, note that under the assumption that the data structure for A is stored in quantum memory, an ε-approximate quantum state $|\tilde{x}\rangle/\|\tilde{x}\|$ can be prepared in complexity $\widetilde{\mathcal{O}}(\kappa\|A\|_F\mathrm{polylog}(1/\varepsilon))$, as shown in [5, 15]. This directly enables length-square sampling, and its entries can be estimated with $\mathrm{poly}(\kappa/\varepsilon)$ overheads.

3 Proof of Theorem 4

In Algorithm 1, we first convert left singular vectors of C ($w^{(\ell)}$) to approximate right singular vectors of R ($\tilde{v}^{(\ell)}$), which also approximate right singular vectors of A. Then we "convert" these to left singular vectors of A in the form ($\langle\tilde{v}^{(\ell)}|A^\dagger/\tilde{\sigma}_\ell$). To clarify the formula for $\tilde{x}$, notice the following sequence of approximations:

$$A^+b = (A^\dagger A)^+A^\dagger b \approx (R^\dagger R)^+A^\dagger b \approx \Big(\sum_{\ell=1}^{k}\frac{1}{\tilde{\sigma}_\ell^2}|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|\Big)A^\dagger|b\rangle = R^\dagger\sum_{\ell=1}^{k}|w^{(\ell)}\rangle\frac{\langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle}{\tilde{\sigma}_\ell^3}$$
$$\approx R^\dagger\sum_{\ell=1}^{k}|w^{(\ell)}\rangle\frac{\tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^3} = \tilde{x}$$

The conversion step from right to left singular vectors of A magnifies previous inaccuracies. For this reason, unlike in earlier works [13, 27], it is beneficial to sample a higher number of columns than rows.

3.1 Correctness of Algorithm 1

To show correctness, we will show that $\|\tilde{x}-x\| \le \varepsilon\|x\|$ as desired. (If this holds, then the length-square distributions of x and $\tilde{x}$ are 2ε-close by Lemma 10 and a $(1\pm\varepsilon)$-approximation to $\|\tilde{x}\|$ is a $(1\pm 3\varepsilon)$-approximation to $\|x\|$.) The methods to query and sample from $\tilde{x}$ given the output of Algorithm 1 are exact, so this suffices for correctness.

[6] Note that κ' is different from κ in Algorithm 1. In order to compare the runtimes in Theorem 4 and Theorem 5 one should set $\kappa = \kappa'\|A\|$ and replace $\|A\|_F$ by $\|A\|_F/\|A\|$ in Theorem 4 while setting $\omega = 3$ in Theorem 5.

We break the correctness argument in two parts. In Section 3.1.1, we show that the approximate right singular vectors $\tilde{v}^{(\ell)}$ and approximate singular values $\tilde{\sigma}_\ell$ described in the algorithm satisfy

$$\left\| \sum_{\ell=1}^{k} \frac{|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|}{\tilde{\sigma}_\ell^2} A^\dagger A - \Pi_{\mathrm{rows}(A)} \right\| \leq \frac{\varepsilon}{2}. \tag{2}$$

If we could compute exact versions of our $\tilde{\lambda}_\ell$'s from Algorithm 1, $\lambda_\ell := \langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle$, then the corresponding output of the algorithm,

$$|x'\rangle := \sum_{\ell=1}^{k} \frac{|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|}{\tilde{\sigma}_\ell^2} A^\dagger|b\rangle = \sum_{\ell=1}^{k} \frac{\langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle}{\tilde{\sigma}_\ell^2}|\tilde{v}^{(\ell)}\rangle = \sum_{\ell=1}^{k} \frac{\lambda_\ell}{\tilde{\sigma}_\ell^2}|\tilde{v}^{(\ell)}\rangle, \tag{3}$$

would be sufficiently close to x due to Equation (2):

$$\|x' - x\| = \left\| \sum_{\ell=1}^{k} \frac{|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|}{\tilde{\sigma}_\ell^2} A^\dagger|b\rangle - |x\rangle \right\| = \left\| \left(\sum_{\ell=1}^{k} \frac{|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|}{\tilde{\sigma}_\ell^2} A^\dagger A - \Pi_{\mathrm{rows}(A)} \right) |x\rangle \right\| \leq \frac{\varepsilon}{2}\|x\|.$$

Remember that we assumed that the projection of b to the column space of A has norm $\Omega(\|b\|)$. Since $\|A\| \leq 1$ we also have that $\|x\| = \Omega(\|b\|)$. Therefore it suffices to find $\tilde{x}$ such that $\|\tilde{x} - x'\| = \mathcal{O}(\varepsilon\|b\|)$ in order to ensure $\|\tilde{x} - x'\| \leq \frac{\varepsilon}{2}\|x\|$.

In Section 3.1.2, we show that using the $\tilde{\lambda}_\ell$ from Algorithm 1, which satisfies $\left|\lambda_\ell - \tilde{\lambda}_\ell\right| = \mathcal{O}\left(\frac{\varepsilon\tilde{\sigma}_\ell^2\|b\|}{\sqrt{k}}\right)$, does not perturb the solution too much. Namely, $\|\tilde{x} - x'\|$ can be bounded by $\mathcal{O}(\varepsilon\|b\|)$ and $\|w\| = \mathcal{O}\big(\kappa^2 k\|b\|\big)$.

3.1.1 Finding approximate singular values and right singular vectors

First we invoke some improved bounds on length-square sampling from [19, Theorem 4.4].[7]

Length-square row sampling of a matrix $A \in \mathbb{C}^{m\times n}$ is as follows: pick a row index $i \in [m]$ with probability $p_i = \frac{\|A_i\|^2}{\|A\|_F^2}$, and upon picking index i set the random output $Y = \frac{A_i}{\sqrt{p_i}}$. Notice that in Algorithm 1 both R and C can be characterised as length-square (row) sampled matrices (the latter holding because every row of R has the same norm).

▶ **Theorem 6.** *Let $A \in \mathbb{C}^{m\times n}$ be a matrix and let $R \in \mathbb{C}^{s\times n}$ be the sample matrix obtained by length-squared sampling and scaling to have $\mathbb{E}[R^\dagger R] = A^\dagger A$. ($R$ consists of rows $Y_1, Y_2, \ldots, Y_s$, which are i.i.d. copies of $Y/\sqrt{s}$, as defined above.) Then, for all $\varepsilon \in [0, \|A\|/\|A\|_F]$,[8] we have*

$$\mathbb{P}\left[\left\|R^\dagger R - A^\dagger A\right\| \geq \varepsilon\|A\|\|A\|_F\right] \leq 2ne^{-\frac{\varepsilon^2 s}{4}}.$$

Hence, for $s \geq \frac{4\ln(2n/\eta)}{\varepsilon^2}$, with probability at least $(1-\eta)$ we have

$$\left\|R^\dagger R - A^\dagger A\right\| \leq \varepsilon\|A\|\|A\|_F.$$

In the following lemma $\|M\|$ denotes the operator norm, but the proof would also work for the Frobenius norm. Note that the following lemmas are independent of the dimensions of the matrices, which is the reason why we do not specify the dimensions. We use δ_{ij} to denote the Kronecker delta function, which is defined to be 1 if $i = j$ and 0 otherwise.

[7] In [19] the theorem is stated for real matrices, but the proof works for complex matrices as well.
[8] If $\varepsilon \geq \|A\|/\|A\|_F$, then the zero matrix is a good enough approximation to $AA^\dagger$.

▶ **Lemma 7** (Converting approximate left and right singular vectors). *Suppose that $w^{(\ell)}$ is a system of orthonormal vectors spanning the column space of C such that*

$$\sum_{\ell=1}^{k} |w^{(\ell)}\rangle\langle w^{(\ell)}| = \Pi_{\mathrm{cols}(C)} \qquad \text{and} \qquad \langle w^{(i)}|CC^\dagger|w^{(j)}\rangle = \delta_{ij}\tilde{\sigma}_i^2.$$

Suppose that $\mathrm{rank}(R) = \mathrm{rank}(C) = k$ *and* $\left\|RR^\dagger - CC^\dagger\right\| \leq \gamma$. *Let* $\tilde{v}^{(\ell)} := \frac{R^\dagger w^{(\ell)}}{\tilde{\sigma}_\ell}$, *then*

$$|\langle \tilde{v}^{(i)}|\tilde{v}^{(j)}\rangle - \delta_{ij}| \leq \frac{\gamma}{\tilde{\sigma}_i\tilde{\sigma}_j}, \qquad \text{and} \qquad \left|\langle \tilde{v}^{(i)}|R^\dagger R|\tilde{v}^{(j)}\rangle - \delta_{ij}\tilde{\sigma}_i^2\right| \leq \frac{\gamma\left(2\|R\|^2 + \gamma\right)}{\tilde{\sigma}_i\tilde{\sigma}_j}.$$

Proof. Let V be the matrix whose ℓ-th column is the vector $\tilde{v}^{(\ell)}$ and let us define the Gram matrix $G = V^\dagger V$. We have that

$$G_{ij} = |\langle \tilde{v}^{(i)}, \tilde{v}^{(j)}\rangle - \delta_{ij}| = \left|\frac{\langle w^{(i)}|RR^\dagger|w^{(j)}\rangle}{\tilde{\sigma}_i\tilde{\sigma}_j} - \delta_{ij}\right| \leq \left|\frac{\langle w^{(i)}|CC^\dagger|w^{(j)}\rangle}{\tilde{\sigma}_i\tilde{\sigma}_j} - \delta_{ij}\right| + \frac{\gamma}{\tilde{\sigma}_i\tilde{\sigma}_j} = \frac{\gamma}{\tilde{\sigma}_i\tilde{\sigma}_j}.$$

Now observe that

$$\left\|RR^\dagger RR^\dagger - CC^\dagger CC^\dagger\right\| \leq \left\|RR^\dagger(RR^\dagger - CC^\dagger)\right\| + \left\|(RR^\dagger - CC^\dagger)CC^\dagger\right\| \leq \gamma\left(2\|R\|^2 + \gamma\right).$$

Let $i, j \in [k]$, then

$$\langle w^{(i)}|CC^\dagger CC^\dagger|w^{(j)}\rangle = \langle w^{(i)}|CC^\dagger\left(\sum_{\ell=1}^{k}|w^{(\ell)}\rangle\langle w^{(\ell)}|\right)CC^\dagger|w^{(j)}\rangle = \delta_{ij}\sigma_i^4.$$

Finally we get that

$$\begin{aligned}\left|\langle \tilde{v}^{(i)}|R^\dagger R|\tilde{v}^{(j)}\rangle - \delta_{ij}\sigma_i^2\right| &= \left|\frac{\langle w^{(i)}|RR^\dagger RR^\dagger|w^{(j)}\rangle}{\tilde{\sigma}_i\tilde{\sigma}_j} - \delta_{ij}\sigma_i^2\right| \\ &\leq \left|\frac{\langle w^{(i)}|CC^\dagger CC^\dagger|w^{(j)}\rangle}{\tilde{\sigma}_i\tilde{\sigma}_j} - \delta_{ij}\sigma_i^2\right| + \frac{\gamma\left(2\|R\|^2 + \gamma\right)}{\tilde{\sigma}_i\tilde{\sigma}_j} \\ &= \frac{\gamma\left(2\|R\|^2 + \gamma\right)}{\tilde{\sigma}_i\tilde{\sigma}_j}.\end{aligned}$$

◀

▶ **Lemma 8.** *Let B be a matrix of rank at most k, and suppose that V has k columns that span the row and column spaces of B. Then*

$$\|B\| \leq \left\|(V^\dagger V)^{-1}\right\|\left\|V^\dagger BV\right\|.$$

Proof. Let $G := V^\dagger V$ be the Gram matrix of V and let $\tilde{V} := VG^{-\frac{1}{2}}$. It is easy to see that $\tilde{V}$ is an isometry and its columns still span the the row and column spaces of B. Since $\tilde{V}$ is an isometry we get that

$$\|B\| = \left\|\tilde{V}^\dagger B\tilde{V}\right\| = \left\|G^{-\frac{1}{2}}V^\dagger BVG^{-\frac{1}{2}}\right\| \leq \left\|G^{-1}\right\|\left\|V^\dagger BV\right\| = \left\|(V^\dagger V)^{-1}\right\|\left\|V^\dagger BV\right\|.$$

◀

▶ **Lemma 9** (Approximate left and right singular vectors). *Suppose that $\tilde{v}^{(i)}$ is a system of approximately orthonormal vectors spanning the row space of A such that*

$$|\langle \tilde{v}^{(i)}|\tilde{v}^{(j)}\rangle - \delta_{ij}| \leq \alpha \leq \frac{1}{4k} \qquad \text{and} \qquad \left|\langle \tilde{v}^{(i)}|R^\dagger R|\tilde{v}^{(j)}\rangle - \delta_{ij}\tilde{\sigma}_i^2\right| \leq \beta, \tag{4}$$

where $\tilde{\sigma}_i^2 \geq \frac{4}{5\kappa^2}$. *Suppose that* $\mathrm{rank}(A) = \mathrm{rank}(R) = k$ *and* $\left\|A^\dagger A - R^\dagger R\right\| \leq \theta$, *then*

$$\left\|\Pi_{\mathrm{rows}(A)} - \sum_{\ell=1}^{k}\frac{|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|}{\tilde{\sigma}_\ell^2}A^\dagger A\right\| \leq \frac{8k}{3}(\beta\kappa^2 + \theta\kappa^2 + \alpha). \tag{5}$$

Proof. Let $B := \sum_{\ell=1}^{k} \frac{|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|}{\tilde{\sigma}_\ell^2} A^\dagger A - \Pi_{\mathrm{rows}(A)}$, we will apply Lemma 8. For this observe

$$\begin{aligned}
\left|\langle\tilde{v}^{(i)}|B|\tilde{v}^{(j)}\rangle\right| &= \left|\sum_{\ell=1}^{k} \frac{\langle\tilde{v}^{(i)}|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|A^\dagger A|\tilde{v}^{(j)}\rangle}{\tilde{\sigma}_\ell^2} - \langle\tilde{v}^{(i)}|\tilde{v}^{(j)}\rangle\right| \\
&\leq \left|\sum_{\ell=1}^{k} \frac{\langle\tilde{v}^{(i)}|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|R^\dagger R|\tilde{v}^{(j)}\rangle}{\tilde{\sigma}_\ell^2} - \delta_{ij}\right| + \sum_{\ell=1}^{k} \frac{|\langle\tilde{v}^{(i)}|\tilde{v}^{(\ell)}\rangle|\theta\left\|\tilde{v}^{(\ell)}\right\|\left\|\tilde{v}^{(j)}\right\|}{\tilde{\sigma}_\ell^2} + \alpha \\
&\leq \left|\sum_{\ell=1}^{k} \frac{\langle\tilde{v}^{(i)}|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|R^\dagger R|\tilde{v}^{(j)}\rangle}{\tilde{\sigma}_\ell^2} - \delta_{ij}\right| + 2\theta\kappa^2 + \alpha \\
&\leq \left|\sum_{\ell\neq j}^{k} \frac{\langle\tilde{v}^{(i)}|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|R^\dagger R|\tilde{v}^{(j)}\rangle}{\tilde{\sigma}_\ell^2}\right| + \left|\langle\tilde{v}^{(i)}|\tilde{v}^{(j)}\rangle\frac{\langle\tilde{v}^{(j)}|R^\dagger R|\tilde{v}^{(j)}\rangle}{\tilde{\sigma}_j^2} - \delta_{ij}\right| + 2\theta\kappa^2 + \alpha \\
&\leq \left|\sum_{\ell\neq j}^{k} \frac{\langle\tilde{v}^{(i)}|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|R^\dagger R|\tilde{v}^{(j)}\rangle}{\tilde{\sigma}_\ell^2}\right| + \alpha(1+\beta/\sigma_j^2) + \delta_{ij}\beta/\sigma_j^2 + 2\theta\kappa^2 + \alpha \\
&\leq (1+k\alpha)\beta\frac{5}{4}\kappa^2 + 2\theta\kappa^2 + 2\alpha \leq 2(\beta\kappa^2 + \theta\kappa^2 + \alpha).
\end{aligned}$$

Let $e_\ell \in \mathbb{C}^k$ denote the ℓ-th standard basis vector and let us define $V := \sum_{\ell=1}^{k} |\tilde{v}^{(\ell)}\rangle\langle e_\ell|$. It follows that $\left\|V^\dagger B V\right\| \leq 2k(\beta\kappa^2 + \theta\kappa^2 + \alpha)$. By (4) we have that $\left\|V^\dagger V - I\right\| \leq k\alpha \leq 1/4$, and thus $\left\|(V^\dagger V)^{-1}\right\| \leq 4/3$. By Lemma 8 we get that $\|B\| \leq 8k(\beta\kappa^2 + \theta\kappa^2 + \alpha)/3$. ◀

If $\gamma \leq \frac{1}{10\kappa^2}$ and $\theta \leq \frac{1}{10\kappa^2}$, we get $\tilde{\sigma}_{\min}^2 \geq \frac{4}{5\kappa^2}$. Then by Lemma 7 we get that $\alpha \leq \frac{5}{4}\kappa^2\gamma$ and $\beta \leq 3\kappa^2\gamma$. Substituting this into Equation (5) we get the upper bound

$$\gamma\left(8k\kappa^4 + 10k\kappa^2/3\right) + \theta\frac{8k}{3}\kappa^2 \leq \gamma 12k\kappa^4 + \theta\frac{8k}{3}\kappa^2. \tag{6}$$

Choosing $\theta = \frac{1}{16}\frac{\varepsilon}{k\kappa^2}$ and $\gamma = \frac{1}{36}\frac{\varepsilon}{k\kappa^4}$, the above bound (6) becomes $\varepsilon/2$. Therefore to succeed with probability at least $1 - \eta/2$ it suffices to sample $r = 2^{10}\ln(8n/\eta)\kappa^4 k^2\|A\|_F^2/\varepsilon^2$ row indices, and then subsequently $c = 2^6 \cdot 3^4 \ln(8r/\eta)\kappa^8 k^2\|A\|_F^2/\varepsilon^2$ column indices as shown by Theorem 6.

3.1.2 The required precision for matrix element estimation

Recall from Equation (3) that $x' = \sum_{\ell=1}^{k} \frac{\lambda_\ell}{\tilde{\sigma}_\ell^2}\tilde{v}^{(\ell)}$, and $\tilde{x}$ is as above except we replace λ_ℓ with $\tilde{\lambda}_\ell$. As we argued in the beginning of the section, for the correctness of Algorithm 1 it suffices to ensure $\|\tilde{x} - x'\| = \mathcal{O}(\varepsilon)$, assuming that $\|b\| = 1$. Now we show that if we have $\left|\lambda_\ell - \tilde{\lambda}_\ell\right| = \mathcal{O}\left(\frac{\varepsilon\tilde{\sigma}_\ell^2\|b\|}{\sqrt{k}}\right)$, then the magnitude of perturbation can be bounded by $\mathcal{O}(\varepsilon)$, and we also get that $\|w\| = \mathcal{O}\left(\kappa^2\sqrt{k}\right)$. Let $e_\ell \in \mathbb{C}^k$ denote the ℓ-th standard basis vector; we rewrite $\|\tilde{x} - x'\|$ as

$$\left\|\sum_{\ell=1}^{k} \frac{\lambda_\ell - \tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^2}|\tilde{v}^{(\ell)}\rangle\right\| = \sqrt{\left\|\sum_{\ell=1}^{k} \frac{\lambda_\ell - \tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^2}|\tilde{v}^{(\ell)}\rangle\langle e_\ell|e_\ell\rangle\right\|^2} = \sqrt{\left\|\left(\sum_{\ell=1}^{k}|\tilde{v}^{(\ell)}\rangle\langle e_\ell|\right)\left(\sum_{\ell=1}^{k}\frac{\lambda_\ell - \tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^2}|e_\ell\rangle\right)\right\|^2}.$$

Let us define $V := \sum_{\ell=1}^{k}|\tilde{v}^{(\ell)}\rangle\langle e_\ell|$, and $|z\rangle := \sum_{\ell=1}^{k}\frac{\lambda_\ell - \tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^2}|e_\ell\rangle$, then we have that

$$\left\|\sum_{\ell=1}^{k} \frac{\lambda_\ell - \tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^2}|\tilde{v}^{(\ell)}\rangle\right\| = \sqrt{\langle z|V^\dagger V|z\rangle} \leq \sqrt{\|V^\dagger V\|}\|z\| = \mathcal{O}(\varepsilon),$$

where we used that $\left\|V^\dagger V\right\| \leq 1 + k\alpha \leq \frac{4}{3}$ as we have shown in the proof of Lemma 9.

Now we show that $\|w\| = \mathcal{O}\Big(\kappa^2\sqrt{k}\Big)$. Remember that $\tilde{v}^{(\ell)} = \frac{R^\dagger w^{(\ell)}}{\tilde{\sigma}_\ell}$, thus $\tilde{x} = R^\dagger_{,\ell}\sum_{\ell=1}^k \frac{\tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^3} w^{(\ell)}$. Let $w := \sum_{\ell=1}^k \frac{\tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^3} w^{(\ell)}$, then we get

$$\|w\| = \sqrt{\sum_{\ell=1}^k \frac{|\tilde{\lambda}_\ell|^2}{\tilde{\sigma}_\ell^2}} \leq \sqrt{\sum_{\ell=1}^k \frac{|\lambda_\ell|^2}{\tilde{\sigma}_\ell^6}} + \sqrt{\sum_{\ell=1}^k \frac{|\tilde{\lambda}_\ell - \lambda_\ell|^2}{\tilde{\sigma}_\ell^6}} \leq \mathcal{O}\big(\kappa^2\big)\sqrt{\sum_{\ell=1}^k \frac{|\lambda_\ell|^2}{\tilde{\sigma}_\ell^2}} + \mathcal{O}\big(\kappa^2\varepsilon\big).$$

Finally observe that

$$\begin{aligned}\sum_{\ell=1}^k \frac{|\lambda_\ell|^2}{\tilde{\sigma}_\ell^2} &= \sum_{\ell=1}^k \frac{\langle b|A|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle}{\tilde{\sigma}_\ell^2} \leq \operatorname{Tr}\left[\sum_{\ell=1}^k \frac{A|\tilde{v}^{(\ell)}\rangle\langle\tilde{v}^{(\ell)}|A^\dagger}{\tilde{\sigma}_\ell^2}\right] = \operatorname{Tr}\left[\sum_{\ell=1}^k \frac{\langle\tilde{v}^{(\ell)}|A^\dagger A|\tilde{v}^{(\ell)}\rangle}{\tilde{\sigma}_\ell^2}\right] \\ &\leq \sum_{\ell=1}^k \frac{\langle\tilde{v}^{(\ell)}|R^\dagger R|\tilde{v}^{(\ell)}\rangle}{\tilde{\sigma}_\ell^2} + \mathcal{O}\big(k\kappa^2\big)\big\|A^\dagger A - R^\dagger R\big\| \leq \mathcal{O}\big(k + k\beta\kappa^2 + k\theta\kappa^2\big) \leq \mathcal{O}(k+\varepsilon),\end{aligned}$$

where the last two inequalities follow from Lemma 9 and its follow-up discussion.

3.2 Complexity of Algorithm 1

The complexity is dominated by two parts of the algorithm: finding the left singular vectors of an r by c matrix, and estimating some matrix elements of A. If we use naive matrix multiplication, then computing the singular value decomposition of $CC^\dagger$ costs

$$\mathcal{O}\big(r^2c\big) = \widetilde{\mathcal{O}}\left(\kappa^{16}k^6\frac{\|A\|_F^6}{\varepsilon^6}\right).$$

In this section, we prove that this dominates the runtime of the algorithm. First, we use length-square sampling techniques similarly to Tang [27] to approximate the matrix elements $\lambda_\ell := \langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle$, which has complexity $\widetilde{\mathcal{O}}\Big(\kappa^8k^4\frac{\|A\|_F^4}{\varepsilon^4}\Big)$ as we show in Section 3.2.2. Second, we show how to efficiently length-square sample from $\tilde{x} := \sum_{\ell=1}^k \frac{\tilde{\lambda}_\ell}{\tilde{\sigma}_\ell^2}\tilde{v}^{(\ell)}$ using rejection sampling.

3.2.1 Length-square sampling techniques

Before we begin discussion of the two sampling techniques used, we note that the closeness of two vectors in Euclidean distance implies closeness of their corresponding distributions.

▶ **Lemma 10** (Bounding Total Variation distance by Euclidean distance [27, Lemma 4.1]). *For $v, w \in \mathbb{C}^n$, $\big\|q^{(v)}, q^{(w)}\big\|_{TV} \leq \frac{2\|v-w\|}{\max(\|v\|,\|w\|)}$.*

3.2.2 Estimating the matrix element $\langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle$

We use the inner product estimation method of Tang [27] for matrix element estimation. The proof can be found in one of the full versions [15].

▶ **Lemma 11** (Trace inner product estimation). *Suppose that we have length-square access to $A \in \mathbb{C}^{m\times n}$ and query access to the matrix $B \in \mathbb{C}^{m\times n}$ in complexity $Q(B)$. Then we can estimate $\operatorname{Tr}\big[A^\dagger B\big]$ to precision $\xi\|A\|_F\|B\|_F$ with probability at least $1-\eta$ in time*

$$\mathcal{O}\left(\frac{\log(1/\eta)}{\xi^2}(L(A)+Q(B))\right).$$

We can estimate $\lambda_\ell = \langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle = \mathrm{Tr}\big[\langle\tilde{v}^{(\ell)}|A^\dagger|b\rangle\big] = \mathrm{Tr}\big[A^\dagger|b\rangle\langle\tilde{v}^{(\ell)}|\big]$ using this lemma. Observe that $\big\||b\rangle\langle\tilde{v}^{(\ell)}|\big\|_F = \big\|\tilde{v}^{(\ell)}\big\|\|b\| \leq (1+\varepsilon)\|b\|$, and we can query the (i,j) matrix element of $|b\rangle\langle\tilde{v}^{(\ell)}|$ by querying b_i and $\tilde{v}_j^{(\ell)}$, which has $\widetilde{\mathcal{O}}(1)$ and $\widetilde{\mathcal{O}}(r)$ cost respectively. We desire to estimate λ_ℓ to additive precision $\mathcal{O}\Big(\frac{\varepsilon\sigma_\ell^2\|b\|}{\sqrt{k}}\Big)$ with success probability $\frac{\eta}{2k}$. By applying Lemma 11, we can compute such an estimate $\tilde{\lambda}_\ell$ with complexity

$$\widetilde{\mathcal{O}}\left(\log(2k/\eta)\frac{k\|A\|_F^2}{\varepsilon^2\sigma_\ell^4}r\right) = \widetilde{\mathcal{O}}\left(\kappa^4 k\frac{\|A\|_F^2}{\varepsilon^2}r\right) = \widetilde{\mathcal{O}}\left(\kappa^8 k^3\frac{\|A\|_F^4}{\varepsilon^4}\right).$$

3.2.3 Sampling from the approximate solution

Our goal is to sample from the length-square distribution of $\tilde{x} = R^\dagger w$. In order to tackle this problem we invoke a result from [27] about length-square sampling a vector that is a linear-combination of length-square accessible vectors. The proof can be found in one of the full versions [15].

▶ **Lemma 12** (Length-square sample a linear combination of vectors [27, Proposition 4.3]). *Suppose that we have length-square access to $R \in \mathbb{C}^{r\times n}$ having normalised rows, and we are given $w \in \mathbb{C}^r$ (as a list of numbers in memory). Then we can implement length-square access to $y := R^\dagger w \in \mathbb{C}^n$, so that we can*

1. *query for entries with complexity $Q(y) = \mathcal{O}(rQ(R))$;*
2. *sample from $q^{(y)}$ with complexity $S(y)$ satisfying*[9] $\mathbb{E}[S(y)] = \mathcal{O}\Big(\frac{r\|w\|^2}{\|y\|^2}(S(R)+rQ(R))\Big)$;
3. *estimate $\|y\|$ to $(1\pm\varepsilon)$ multiplicative error with success probability $\geq 1-\delta$ in complexity $\widetilde{N}(y) = \mathcal{O}\Big(\frac{r\|w\|^2}{\|y\|^2\varepsilon^2}(S(R)+rQ(R))\log\frac{1}{\delta}\Big)$.*

Since all rows of R have norm $\|A\|_F/\sqrt{r}$, and $\|\tilde{x}\| = \Omega(1)$ by Lemma 12 we can length-square sample from $\tilde{x}$ in expected complexity

$$\mathcal{O}\left(\frac{r\|w\|^2\|A\|_F^2/r}{\|\tilde{x}\|^2}r\right) = \mathcal{O}\left(\frac{\|w\|^2\|A\|_F^2}{\|\tilde{x}\|^2}r\right) = \mathcal{O}\Big(\kappa^4 k\|A\|_F^2 r\Big) = \widetilde{\mathcal{O}}\left(\frac{\kappa^8k^3\|A\|_F^4}{\varepsilon^2}\right).$$

Computing the complexity for the other routines follows similarly.

4 Proof of Theorem 5

We need some technique tools to prove Theorem 5. We first summarize how to sample the vector resulted from a matrix-vector multiplication.

▶ **Lemma 13** ([27]). *Let $M \in \mathbb{C}^{n\times k}$ and $v \in \mathbb{C}^k$. Given length square access to M as in Definition 3, one can output a sample from the vector Mv with probability $9/10$ in $O(k^2C(M,v))$ query and time complexity, where $C(M,v) := \sum_{j=1}^k \| v_jM_{\cdot j} \|^2 / \| Mv \|^2$.*

▶ **Lemma 14** ([27]). *Let $M \in \mathbb{C}^{n\times k}$ and $v \in \mathbb{C}^k$. If there exists an isometry $U \in \mathbb{C}^{n\times k}$ whose column vectors span the column space of M such that $\|M-U\|_F \leq \alpha$, then one can sample from a distribution which is $(\alpha+O(\alpha^2))$-close to $\mathcal{D}_{Mv}$ in $O(k^2(1+O(\alpha)))$ expected query and time complexity.*

[9] We also show the high probability form of this expected complexity bound: with probability $\geq 1-\delta$, $S(y) = \mathcal{O}\Big(\frac{r\|w\|^2}{\|y\|^2}(S(R)+rQ(R))\log\frac{1}{\delta}\Big)$.

We need the following lemma for estimating $x^\dagger Ay$,[10] whose proof can be found in one of the full versions [9].

▶ **Lemma 15.** *Let $x \in \mathbb{C}^m$, $y \in \mathbb{C}^n$, and $A \in \mathbb{C}^{m\times n}$. Given the sampling and query access to A, the query access to x and y, and the knowledge of $\|x\|$ and $\|y\|$, one can approximate $x^\dagger Ay$ to additive error ϵ with at least $1-\delta$ success probability using $O(\frac{\|x\|^2\|y\|^2\|A\|_F^2}{\epsilon^2}\log\frac{1}{\delta})$ queries and samples and the same time complexity.*

We also need the following result by Farforovskaya and Nikolskaya [12] to bound the error of inverting matrices.

▶ **Lemma 16** ([12])**.** *Let $f : [0, x_{\max}] \to \mathbb{C}$ be L-Lipschitz continuous and A, B be Hermitian matrices with $\|A\|, \|B\| \le x_{\max}$, and $\|A-B\| \le \epsilon$. Then it holds that*

$$\|f(A) - f(B)\| \le 4L\epsilon(\log(1 + 2x_{\max}/\epsilon) + 1)^2.$$

Now, we are ready to prove the main theorem.

Proof of Theorem 5. First note that the last two steps of Algorithm 2 are dealing with the vector Vw', which, by the previous steps, can be written as $Vw' = VD^{-2}V^\dagger A^\dagger b$, where $D \in \mathbb{R}^{k\times k}$ is the diagonal matrix with diagonal entries $\sigma_1, \ldots, \sigma_k$. This vector can also be written as

$$VD^{-2}V^\dagger A^\dagger b = \sum_{j=1}^{k} \frac{1}{\sigma_j^2} S^\dagger \frac{u_j u_j^\dagger}{\sigma_j^2} SA^\dagger b = S^\dagger (WW^\dagger)^{-2} SA^\dagger b,$$

where W is the matrix obtained in Algorithm 2 of Algorithm 2. To analyze the distance between the above vector to $(A^\dagger A)^+ A^\dagger b$, we define functions $f, g : [0,1] \to \mathbb{R}$ as

$$f(x) = \begin{cases} \frac{1}{x} & \text{when } x \in \left[\frac{1}{2\kappa'^2}, 1\right] \\ 2\kappa'^2 & \text{when } x \in \left[0, \frac{1}{2\kappa'^2}\right) \end{cases} \quad \text{and} \quad g(x) = \begin{cases} \frac{1}{x^2} & \text{when } x \in \left[\frac{1}{2\kappa'^2}, 1\right] \\ 4\kappa'^4 & \text{when } x \in \left[0, \frac{1}{2\kappa'^2}\right). \end{cases} \tag{7}$$

We show that f is $4\kappa'^4$-Lipschitz continuous and g is $4\kappa'^6$-Lipschitz continuous. We first consider the case where $x, x' \in [\frac{1}{2\kappa'^2}, 1]$. Then, $\frac{|f(x)-f(x')|}{|x-x'|} \le \frac{1}{4\kappa'^4}$ and $\frac{|g(x)-g(x')|}{|x-x'|} \le \frac{1}{4\kappa'^6}$. For the case where $x, x' \in [0, 1/2\kappa'^2)$, it is trivial that $\frac{|f(x)-f(x')|}{|x-x'|} = \frac{|g(x)-g(x')|}{|x-x'|} = 0$. Finally, if $x \in [0, 1/2\kappa'^2)$ and $x' \in [1/2\kappa'^2, 1]$, then $\frac{|f(x)-f(x')|}{|x-x'|} \le \frac{1}{4\kappa'^4}$ and $\frac{|g(x)-g(x')|}{|x-x'|} \le \frac{1}{4\kappa'^6}$ still hold since $\frac{|f(x)-f(x')|}{|x-x'|}$ and $\frac{|g(x)-g(x')|}{|x-x'|}$ must be less than the slopes of f and g on the singular point $1/2\kappa'^2$.

By Theorem 6 and the choice of s in Algorithm 2, we have that, with high probability,

$$\left\|S^\dagger S - A^\dagger A\right\| \le \frac{\epsilon}{2\kappa'^4\|A\|^2}, \quad \text{and} \quad \left\|WW^\dagger - SS^\dagger\right\| \le \frac{\epsilon}{2\kappa'^6\|S\|^4}. \tag{8}$$

As a consequence of Weyl's inequalities (see, for example, [17]), we have that, with high probability,

$$|\sigma_k(S^\dagger S) - \sigma_k(A^\dagger A)| \le \left\|A^\dagger A - S^\dagger S\right\| \le \frac{\epsilon}{2\kappa'^4}, \text{ and}$$
$$|\sigma_k(SS^\dagger) - \sigma_k(WW^\dagger)| \le \left\|SS^\dagger - WW^\dagger\right\| \le \frac{\epsilon}{2\kappa'^6}.$$

[10] Note that this lemma can be viewed a special case of Lemma 11.

Assuming $\epsilon \leq 1/2$, this further implies that

$$\sigma_{\min}(S^\dagger S) \geq \sigma_{\min}(A^\dagger A) - \frac{\epsilon}{2\kappa'^4} \geq \frac{3}{4\kappa'^4}, \text{ and}$$
$$\sigma_{\min}(WW^\dagger) \geq \sigma_{\min}(SS^\dagger) - \frac{\epsilon}{2\kappa'^6} \geq \frac{1}{2\kappa'^6}.$$

According to Equation (8), $\|S\|^2 \leq \|A\|^2 + \frac{\epsilon}{2\kappa'^4}$ and $\|W\|^2 \leq \|S\|^2 + \frac{\epsilon}{2\kappa'^6}$ with high probability. Now, Equation (8) together with applying Lemma 16 on f and g defined in Equation (7) imply that

$$\left\|(S^\dagger S)^+ - (A^\dagger A)^+\right\| \leq \frac{\epsilon}{2\|A\|^2}, \quad \text{and} \quad \left\|(WW^\dagger)^{-2} - (SS^\dagger)^{-2}\right\| \leq \frac{\epsilon}{2\|S\|^4}. \tag{9}$$

Hence, we have that, with high probability,

$$\begin{aligned}\left\|S^\dagger(WW^\dagger)^{-2}S - (A^\dagger A)^+\right\| &\leq \left\|S^\dagger(WW^\dagger)^{-2}S - S^\dagger(SS^\dagger)^{-2}S + S^\dagger(SS^\dagger)^{-2}S - (A^\dagger A)^+\right\| \\ &\leq \left\|S^\dagger(WW^\dagger)^{-2}S - S^\dagger(SS^\dagger)^{-2}S\right\| + \left\|(S^\dagger S)^+ - (A^\dagger A)^+\right\| \\ &\leq \frac{\epsilon}{\|A\|^2}.\end{aligned}$$

Therefore, we have

$$\left\|S^\dagger(WW^\dagger)^{-2}SA^\dagger b - (A^\dagger A)^+ A^\dagger b\right\| \leq \frac{\epsilon}{\|A\|} = O(\epsilon\|A^+ b\|)$$

with high probability, where the second inequality follows from the assumption that the norm of the projection of b onto the columns space of A is at least some constant c, which implies that $c \leq \|AA^+ b\| \leq \|A\|\|A^+ b\|$. Note that this implies that $(\mathcal{D}_{S^\dagger(WW^\dagger)^{-2}SA^\dagger b} - \mathcal{D}_{(A^\dagger A)^+ A^\dagger b})_{TV} \leq 2\epsilon$ by Lemma 10 (with appropriate choices of constant factors in p and s).

Now, we argue that V is approximately orthogonal. To see this, we bound the distance between the inner product $V(i,\cdot)V(\cdot,j)$ and δ_{ij} as follows

$$|V(i,\cdot)V(\cdot,j) - \delta_{ij}| = \left|\frac{u_i^\dagger}{\sigma_i} SS^\dagger \frac{u_j}{\sigma_j} - \delta_{ij}\right| \leq \left|\frac{u_i^\dagger}{\sigma_i} WW^\dagger \frac{u_j}{\sigma_j} - \delta_{ij}\right| + \frac{\epsilon}{2\kappa'^6\|S\|^3\sigma_i\sigma_j} \leq \frac{\epsilon}{2\kappa'^4\|S\|^3}.$$

We use $s = \Theta(\frac{\|A\|^6\|A\|_F^2\kappa'^8\ln(n/\delta)}{\epsilon^2})$ samples of rows from A to build S. We use $p = \Theta(\frac{\|A\|^{10}\|A\|_F^2\kappa'^{12}\ln(n/\delta)}{\epsilon^2})$ samples of columns from S to build W. Finally, we can compute the matrix $WW^\dagger$ in $\Theta\left(\|A\|^4\kappa'^4 \cdot \left(\frac{\kappa'^8\|A\|^6\|A\|_F^2\ln\frac{n}{\delta}}{\epsilon^2}\right)^\omega\right)$ time by dealing with the $\lceil p/s \rceil$ blocks of W. We can also compute the spectral decomposition of $WW^\dagger$ in $\Theta\left(\left(\frac{\kappa'^8\|A\|^6\|A\|_F^2\ln\frac{n}{\delta}}{\epsilon^2}\right)^\omega\right)$ time.

Given query access to b, we can get query and sampling access to $S^\dagger(WW^\dagger)^{-2}SA^\dagger b$ as follows: first, we compute $W^\dagger W$, its spectral decomposition, and apply the function g on the eigenvalues, this step takes $\Theta\left(\|A\|^4\kappa'^4 \cdot \left(\frac{\kappa'^8\|A\|^6\|A\|_F^2\ln\frac{n}{\delta}}{\epsilon^2}\right)^\omega\right)$. Then, we obtain query access and sampling access (by Lemma 13) to V, where $V_{\cdot i} := S^\dagger\frac{u_i}{\sigma_i}$ for u_i and σ_i the ith eigenvalue and eigenvector of $W^\dagger W$. Note that with this definition, $S^\dagger(WW^\dagger)^{-2}SA^\dagger b = VV^\dagger A^\dagger b$. We compute $b' := V^\dagger A^\dagger b \in \mathbb{C}^s$ by using Lemma 15. Finally, the query access can be obtained directly by computing $(Vb')_j$ for any $j \in [m]$ and the sampling access to Vb' can be obtained by using Lemma 14. ◀

References

1 Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 636–647, 2012. arXiv:1010.4458 doi:10.4230/LIPIcs.STACS.2012.636.

2 Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. Quantum-inspired algorithms in practice. *Quantum*, 4:307, 2020. arXiv:1905.10415 doi:10.22331/q-2020-08-13-307.

3 Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse Hamiltonians. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC)*, pages 283–292, 2014. arXiv:1312.1414 doi:10.1145/2591796.2591854.

4 Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549:195–202, 2017. arXiv:1611.09347 doi:10.1038/nature23474.

5 Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 33:1–33:14, 2019. arXiv:1804.01973 doi:10.4230/LIPIcs.ICALP.2019.33.

6 Zhihuai Chen, Yinan Li, Xiaoming Sun, Pei Yuan, and Jialin Zhang. A quantum-inspired classical algorithm for separable non-negative matrix factorization. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4511–4517. AAAI Press, 2019. arXiv:1907.05568

7 Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd ACM Symposium on the Theory of Computing (STOC)*, page 387–400, 2020. arXiv:1910.06151 doi:10.1145/3357713.3384314.

8 Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired sublinear algorithm for solving low-rank semidefinite programming. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 23:1–23:15, 2020. arXiv:1901.03254 doi:10.4230/LIPIcs.MFCS.2020.23.

9 Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. arXiv:1811.04852, 2018.

10 Chen Ding, Tian-Yi Bao, and He-Liang Huang. Quantum-inspired support vector machine. arXiv:1906.08902, 2019.

11 Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. Quantum-inspired algorithm for general minimum conical hull problems. *Physical Review Research*, 2(3):033199, 2020. arXiv:1907.06814 doi:10.1103/PhysRevResearch.2.033199.

12 Yuliya B. Farforovskaya and Ludmila N. Nikolskaya. Modulus of continuity of operator functions. *St. Petersburg Math. J. – Algebra i Analiz*, 20(3):493–506, 2009. doi:10.1090/S1061-0022-09-01058-9.

13 Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6):1025–1041, 2004. doi:10.1145/1039488.1039494.

14 András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. arXiv:1811.04909, 2018.

15 András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*, pages 193–204, 2019. arXiv:1806.01838 doi:10.1145/3313276.3316366.

16 Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv:0811.3171 doi:10.1103/PhysRevLett.103.150502.

17 Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

18 Dhawal Jethwani, François Le Gall, and Sanjay K. Singh. Quantum-inspired classical algorithms for singular value transformation. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 53:1–53:14, 2020. arXiv:1910.05699 doi:10.4230/LIPIcs.MFCS.2020.53.

19 Ravindran Kannan and Santosh Vempala. Randomized algorithms in numerical linear algebra. *Acta Numerica*, 26:95–135, 2017. doi:10.1017/S0962492917000058.

20 Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 49:1–49:21, 2017. arXiv:1603.08675 doi:10.4230/LIPIcs.ITCS.2017.49.

21 Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020. arXiv:1704.04992 doi:10.1103/PhysRevA.101.022316.

22 Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10:631–633, 2014. arXiv:1307.0401 doi:10.1038/nphys3029.

23 Patrick Rebentrost and Seth Lloyd. Quantum computational finance: quantum algorithm for portfolio optimization. arXiv:1811.03975, 2018.

24 Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014. arXiv:1307.0471 doi:10.1103/PhysRevLett.113.130503.

25 Patrick Rebentrost, Adrian Steffens, Iman Marvian, and Seth Lloyd. Quantum singular-value decomposition of nonsparse low-rank matrices. *Physical Review A*, 97:012327, 2018. arXiv:1607.05404 doi:10.1103/PhysRevA.97.012327.

26 Ewin Tang. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. arXiv:1811.00414, 2018.

27 Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*, pages 217–228, 2019. arXiv:1807.04271 doi:10.1145/3313276.3316310.

28 Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Physical Review Letters*, 109(5):050505, 2012. arXiv:1204.5242 doi:10.1103/PhysRevLett.109.050505.

29 Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical Review Letters*, 120(5):050502, 2018. arXiv:1704.06174 doi:10.1103/PhysRevLett.120.050502.

30 Zhikuan Zhao, Jack K. Fitzsimons, and Joseph F. Fitzsimons. Quantum-assisted Gaussian process regression. *Physical Review A*, 99(5):052331, 2019. arXiv:1512.03929 doi:10.1103/PhysRevA.99.052331.