Quantum-Inspired Classical Algorithms for Singular Value Transformation

Dhawal Jethwani
Indian Institute of Technology (BHU), Varanasi, India
dhawal.jethwani.cse15@iitbhu.ac.in

François Le Gall
Nagoya University, Japan
legall@math.nagoya-u.ac.jp

Sanjay K. Singh
Indian Institute of Technology (BHU), Varanasi, India
sks.cse@iitbhu.ac.in

Abstract

A recent breakthrough by Tang (STOC 2019) showed how to "dequantize" the quantum algorithm for recommendation systems by Kerenidis and Prakash (ITCS 2017). The resulting algorithm, classical but "quantum-inspired", efficiently computes a low-rank approximation of the users' preference matrix. Subsequent works have shown how to construct efficient quantum-inspired algorithms for approximating the pseudo-inverse of a low-rank matrix as well, which can be used to (approximately) solve low-rank linear systems of equations. In the present paper, we pursue this line of research and develop quantum-inspired algorithms for a large class of matrix transformations that are defined via the singular value decomposition of the matrix. In particular, we obtain classical algorithms with complexity polynomially related (in most parameters) to the complexity of the best quantum algorithms for singular value transformation recently developed by Chakraborty, Gilyén and Jeffery (ICALP 2019) and Gilyén, Su, Low and Wiebe (STOC 2019).

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Sampling algorithms, quantum-inspired algorithms, linear algebra

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.53

Related Version A full version of the paper is available at `https://arxiv.org/abs/1910.05699`.

Funding *François Le Gall*: FLG was partially supported by JSPS KAKENHI grants Nos. JP15H01677, JP16H01705, JP16H05853, JP19H04066 and by the MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) grant No. JPMXS0118067394.

Acknowledgements The authors are grateful to András Gilyén for discussions and comments about the manuscript. Part of this work has been done when DJ was visiting Kyoto University.

1 Introduction

Background. One of the most celebrated quantum algorithms discovered so far is the HHL algorithm [13]. This quantum algorithm solves a system of linear equations of the form $Ax = b$, where A is an $n \times n$ matrix and b is an n-dimensional vector, in time polynomial in $\log n$ when the matrix A is sufficiently sparse and well-conditioned. This is exponentially better that the best known classical algorithms, which run in time polynomial in n (see also [1, 7, 8, 20] for improvements and relaxations of the assumptions). There are nevertheless two significant caveats. First, the input should be given in a way that allows very specific quantum access. In particular, the HHL algorithm requires the ability to efficiently create a quantum state proportional to b. The second, and main, caveat is that the output of the HHL algorithm is not the solution x of the linear system (which is an n-dimensional vector)

but only a $O(\log n)$-qubit quantum state proportional to this vector. While measuring this quantum state can give some meaningful statistics about the solution x, this naturally does not give enough information to obtain the whole vector x. In this perspective, the HHL algorithm does not explicitly solve the system of equations, but instead enables sampling from the solution, in a very efficient way.

There have been several proposals to apply the HHL algorithm (and one of its core components, phase estimation) to linear-algebra based machine learning tasks, leading for instance to the discovery of quantum algorithms for principal component analysis (PCA) [15] and quantum support vector machine [16]. We refer to [3] for a recent survey on this field called quantum machine learning. One of the most convincing applications of quantum algorithms to machine learning has been speeding up recommendation systems [14]. In machine learning, recommendations systems are used to predict the preferences of users. From a mathematical perspective, the core task in recommendation systems can be modeled as follows: given an $m \times n$ matrix A (representing the preferences of m users) and an index $i \in [m]$ (representing one specific user), sample from the i-th row of a low-rank approximation of A. Kerenidis and Prakash [14] showed how to adapt the HHL algorithm to solve this problem in time polynomial in $\log(mn)$, which was exponentially better than the best known classical algorithms for recommendation systems.

Similarly to the HHL algorithm, the quantum algorithm from [14] works only under the assumption that the input is stored in an appropriate structure (called "Quantum Random-Access Memory", or "QRAM") that allows specific quantum access. Very recently, Tang [18] has shown that assuming that the input is stored in a classical data structure that allows ℓ^2-norm sampling access (i.e., allows sampling rows with probability proportional to their ℓ^2-norm), polylog(mn)-time classical algorithms for recommendation systems can be designed as well. This results eliminates one of the best examples of quantum speedup for machine learning. The paper [18] also introduced the term "quantum-inspired algorithms" to refer to such classical algorithms obtained by "dequantizing" quantum algorithms.

More quantum-inspired algorithms have soon been developed: Tang [17] first showed how to construct classical algorithms for PCA that essentially match the complexity of the quantum algorithm for PCA from [15] mentioned above. Gilyén, Lloyd and Tang [11] and, independently, Chia, Lin and Wang [6] have shown how to obtain new classical algorithms for solving linear systems of equations, which also essentially match the complexity of the quantum algorithms when the input matrix has low-rank (see below for details). We also refer to [2] for a discussion of the performance of these quantum-inspired algorithms in practice.

Singular value transformation. The Singular Value Decomposition (SVD) of a matrix $M \in \mathbb{C}^{m\times n}$ is a factorization of the form $M = U\Sigma V^*$ where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices and Σ is a $m \times n$ diagonal matrix with $\min(m, n)$ non-negative real numbers on the diagonal, where V^* denotes the complex-conjugate transpose of V. A crucial property is that this decomposition exists for any complex matrix. Given a function $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, the singular value transformation associated with f, denoted Φ_f, is the function that maps the matrix $M = U\Sigma V^*$ to the matrix $\Phi_f(M) = U\ \Sigma_f V^*$ where Σ_f is the diagonal matrix obtained from Σ by replacing each diagonal entry σ by $f(\sigma)$. We refer to Definition 4 in Section 2 for more details.

An important example is obtained by taking the "pseudo-inverse" function $\mathrm{inv}\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $\mathrm{inv}(x) = 1/x$ if $x > 0$ and $\mathrm{inv}(0) = 0$. Solving a linear system of equations $Ax = b$ corresponds[1] to calculating (or approximating) the vector $\Phi_{\mathrm{inv}}(A^*)b$. If all the

[1] Indeed, one solution is given by $x = A^+b$, where A^+ represents the Moore-Penrose pseudo-inverse of the matrix A (or simply the inverse when A is invertible). It is easy to check that $A^+ = \Phi_{\mathrm{inv}}(A^*)$.

singular values of A are between $1/\kappa$ and 1, for some value κ, the quantum-inspired algorithms from [6, 11] solve this task in time poly $(k_A, \kappa, \|A\|_F, 1/\epsilon, \log(mn))$, where k_A denotes the rank of A, $\|A\|_F$ denotes the Frobenius norm of A and ϵ denotes the approximation error.[2] One crucial point here is that the dependence on the dimensions of the matrix is only poly-logarithmic. Another important point is that the best known quantum algorithms (see [4, 11]) enable ℓ^2-norm sampling from the output in time $O(\kappa\|A\|_F \text{polylog}(mn/\epsilon))$ in the QRAM input model. This means that, except for the dependence in ϵ, for low-rank matrices the classical running time is polynomially related to the quantum running time.

The core computational problem in recommendation systems can also be described as approximating the i-row of the matrix $\Phi_{\text{th}}(A)$ for the threshold function $\text{th}\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $\text{th}(x) = x$ if $x \geq \sigma$ and $\text{th}(x) = 0$ otherwise (for some appropriate threshold value σ). This corresponds to approximating the vector $\Phi_{\text{th}}(A^*)b$ where b is the vector with 1 in the i-th coordinate and zero elsewhere. Ref. [18] shows how to solve this problem in time poly $(\|A\|_F/\sigma, 1/\epsilon, \log(mn))$. (For the value σ chosen for recommendation systems, the term $\|A\|_F/\sigma$ becomes an upper bound on the rank of a low-rank approximation of A.)

Our results. In this paper we significantly extend the class of functions for which the singular value transformation can be efficiently computed by "quantum-inspired" classical algorithms. The formal and most general statements of our results are given in Section 3. For the sake of readability, in this introduction we only describe our results for a restricted (but still very general) class of "smooth" functions. Let $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$ denote the sets of non-negative numbers and positive numbers, respectively. We say below that a function $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is "smooth" if f is differentiable in $\mathbb{R}_{>0}$ and the following condition holds: for any $\alpha, \beta \geq 1$, over the interval $[1/\alpha, \beta]$ the maximum values of f and its derivative f' can be upper bounded by a polynomial function of α and β. We are mostly interested in functions such that $f(0) = 0$ since typically we do not want the transformation to increase the rank. Our main results are the following two theorems (refer to Section 3 for the formal versions).[3]

▶ **Theorem 1** (Informal Version). *Let $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be any smooth function such that $f(0) = 0$. For any sufficiently small $\epsilon > 0$, there exists a classical algorithm that has sampling access to a matrix $A \in \mathbb{C}^{m\times n}$ with singular values in $[1/\kappa, 1]$ and to a non-zero vector $b \in \mathbb{C}^m$, receives as input an index $i \in [n]$, outputs with high probability an approximation of the i-th coordinate of the vector $\Phi_f(A^*)b$ with additive error ϵ, and has* poly $(\kappa, \|A\|_F, 1/\epsilon, \log(mn))$ *time complexity.*

▶ **Theorem 2** (Informal Version). *Let $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be any smooth function such that $f(0) = 0$ and $f(x) > 0$ for all $x > 0$. For any sufficiently small $\epsilon > 0$, there exists a classical algorithm that has sampling access to a matrix $A \in \mathbb{C}^{m\times n}$ with singular values in $[1/\kappa, 1]$ and to a non-zero vector $b \in \mathbb{C}^m$, and ℓ^2-samples with high probability from a distribution ϵ-close in total variation distance to the distribution associated with the vector $\Phi_f(A^*)b$, and has* poly $(\kappa, \|A\|_F, 1/\epsilon, \log(mn))$ *time complexity.*

[2] The term $\log(mn)$ represents the time complexity of implementing sampling and query operations (see Proposition 6 in Section 2.3), which we also include in the complexity.

[3] These informal versions can be derived from the formal versions given in Section 3 by observing that $\kappa_2/\|A\|_2 \leq \kappa$ if all the singular values of A are between $1/\kappa$ and 1. The smoothness condition implies that both Ω and ϕ are upper bounded by a polynomial of κ and $\|A\|_F$. Note that for Theorem 2 we actually need an additional smoothness condition expressing that the minimum value of f cannot be too small as well (see the term ω in the formal version of Theorem 2).

Note that instead of stating our results for the transformation $\Phi_f(A)$ we state them for the transformation $\Phi_f(A^*) = (\Phi_f(A))^*$ in Theorems 1 and 2. The reason is that this simplifies the presentation of our algorithms and makes the comparison with prior works easier.

Theorems 1 and 2 show that under the same assumptions (namely, sampling access to the input) and similar requirements for the output (i.e., outputting one coordinate of $\Phi_f(A^*)b$ or sampling from the associated distribution) as the prior works on quantum-inspired algorithms, we can efficiently compute classically the singular value transformation for any smooth enough function. This extends the results from [6, 11, 18] and significantly broadens the applicability of quantum-inspired algorithms.

Fast quantum algorithms have been constructed in recent works [4, 12] for singular value transformations. For the class of smooth functions we consider, the quantum running time obtained would be $O\left(\text{poly}\left(\kappa, \|A\|_F, \log(mn/\epsilon)\right)\right)$ in the QRAM input model. Our results thus show that except possibly for the dependence on ϵ, we can again obtain classical algorithms with running time polynomially related to the quantum running time.

Overview of our approach. We use the same sampling methods as in [2, 6, 9, 11, 18]: we first sample r rows from the input matrix $A \in \mathbb{C}^{m\times n}$ according to probability proportional to the row norms, which gives (after normalization) a matrix $S \in \mathbb{C}^{r\times n}$. We then do the same with matrix S, this time sampling c columns, which gives (after normalization) a matrix $W \in \mathbb{C}^{r\times c}$. The analysis of this process, which has been done in the seminal work by Frieze, Kannan and Vempala [9], shows that with high probability we have $A^*A \approx S^*S$ and $SS^* \approx WW^*$ when r and c are large enough (but still much smaller than m and n). Since W is a small matrix, we can then afford to compute its SVD.

The main contribution of this paper is the next step (and its analysis). We show how to use the SVD of the matrix W in order to compute the singular value transformation Φ_f. Using the SVD of W, we first compute the matrices $\Phi_{\text{inv}}(W)$, $\Phi_{\text{inv}}(W^*)$ and $\Phi_f(W)$. We then compute the matrix $P' = \Phi_{\text{inv}}(W)\Phi_f(W^*)\Phi_{\text{inv}}(W)\Phi_{\text{inv}}(W^*) \in \mathbb{C}^{r\times r}$. This matrix P' is the output of Algorithm 1 presented in Section 3.2. Our central claim is the following:

$$S^*P'SA^* \approx \Phi_f(A^*). \tag{1}$$

Proving (1) and quantifying the quality of the approximation is our main technical contribution. This is done in Proposition 13 (which itself relies on several lemmas proved in Sections 3.1 and 3.2). Finally, using similar post-processing techniques as in prior works [6, 18], from the output P' of Algorithm 1 we can efficiently approximate coordinates of $\Phi_f(A^*)b$ and sample from $\Phi_f(A^*)b$. This post-processing is described in Algorithms 2 and 3 in Section 3.3.

We now give an outline of the main ideas used to establish (1). The basic strategy is to exploit the relations $A^*A \approx S^*S$ and $SS^* \approx WW^*$ mentioned above. Our first insight is to define the function $h\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $h(x) = f(\sqrt{x})/\sqrt{x}$ if $x > 0$ and $h(0) = 0$, and observe that $\Phi_f(A^*) = \Phi_h(A^*A)A^*$. We then prove, in Lemma 10, that $A^*A \approx S^*S$ implies $\Phi_h(A^*A) \approx \Phi_h(S^*S)$. The next natural step would be to relate $\Phi_h(S^*S)$ and $\Phi_h(W^*W)$, but this cannot be done directly since the only guarantee is $SS^* \approx WW^*$, and not $S^*S \approx W^*W$. Instead, we observe that $\Phi_h(S^*S) = S^*PS$, where $P = \Phi_{\text{inv}}(S)\Phi_f(S^*)\Phi_{\text{inv}}(S)\Phi_{\text{inv}}(S^*)$. Since $\Phi_{\text{inv}}(S)\Phi_f(S^*) = \Phi_h(SS^*)$ and $\Phi_{\text{inv}}(W)\Phi_f(W^*) = \Phi_h(WW^*)$, and since we can show that $\Phi_h(SS^*)$ is close to $\Phi_h(WW^*)$ using Lemma 10, we are able to prove that $P \approx P'$ (this is proved in Lemma 12). To summarize, we have $S^*P'SA^* \approx S^*PSA^* = \Phi_h(S^*S)A^* \approx \Phi_h(A^*A)A^* = \Phi_f(A^*)$, as needed.

Related independent work. Independently from our work, Chia, Gilyén, Li, Lin, Tang and Wang simultaneously derived similar results [5]. They additionally provide general matrix arithmetic primitives for adding and multiplying matrices having sample and query access, and recover known dequantized algorithms. They also show how to use these results on the singular value transformation to obtain new quantum-inspired algorithms for other applications, including Hamiltonian simulation and discriminant analysis.

2 Preliminaries

2.1 Notations and conventions

General notations. In this paper we use the notation $[n] = \{1, ..., n\}$ for any integer $n \geq 1$. For any set S we denote $\text{Conv}(S)$ the convex hull of S.

Given a matrix $M \in \mathbb{C}^{m\times n}$, we use $M_{(i,.)} \in \mathbb{C}^{1\times n}$, $M_{(.,j)} \in \mathbb{C}^{m\times 1}$ and $M_{(i,j)} \in \mathbb{C}$ to denote its i-th row, its j-th column and its (i,j)-th element, respectively. The complex-conjugate transpose or Hermitian transpose of a matrix $M \in \mathbb{C}^{m\times n}$ (or a vector $v \in \mathbb{C}^n$) is denoted as M^* (and v^*, respectively). The notations $\|M\|_F$ and $\|M\|_2$ represent the Frobenius and spectral norm, respectively. Note that $\|M\|_2 \leq \|M\|_F$ for any M. For a vector $v \in \mathbb{C}^n$, we denote $\|v\|$ the ℓ^2 norm of the vector. In this paper we will use several times the following standard inequalities that hold for any vector $v \in \mathbb{C}^n$ and any matrices $M \in \mathbb{C}^{n\times m}$ and $N \in \mathbb{C}^{m\times p}$:

$$\|Mv\| \leq \|M\|_2\|v\|, \qquad \|MN\|_F \leq \|M\|_2\|N\|_F, \qquad \|MN\|_F \leq \|M\|_F\|N\|_2. \tag{2}$$

For a non-zero vector $v \in \mathbb{C}^n$, let $\mathcal{P}_v$ denote the probability distribution on $[n]$ where the probability of choosing $i \in [n]$ is defined as $\mathcal{P}_v(i) = \frac{|v_i|^2}{\|v\|^2}$. For two vectors v and w, the total variation distance between distributions $\mathcal{P}_v$ and $\mathcal{P}_w$ is defined as $\|\mathcal{P}_v - \mathcal{P}_w\|_{TV} = \frac{1}{2}\sum_{i=1}^n |\mathcal{P}_v(i) - \mathcal{P}_w(i)|$.

We will use the following easy inequality (see for instance [6, 18] for a proof): for any two vectors $v, w \in \mathbb{C}^n$,

$$\|\mathcal{P}_v - \mathcal{P}_w\|_{TV} \leq \frac{2\|v - w\|}{\|v\|}. \tag{3}$$

Singular Value Decomposition. The Singular Value Decomposition (SVD) of a matrix $M \in \mathbb{C}^{m\times n}$ is a factorization of the form $M = U\Sigma V^*$ where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices and Σ is an $m \times n$ diagonal matrix with $\min(m, n)$ non-negative real numbers, in non-increasing order, down the diagonal. The columns of U and V represent the left and right singular vectors, respectively. Each entry of this diagonal matrix is a singular value of matrix M. A crucial property is that a SVD exists for any complex matrix.

We can also write the SVD of a matrix as

$$M = U\Sigma V^* = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^* \tag{4}$$

where $\{u_i\}_{i\in[m]}$ and $\{v_j\}_{j\in[n]}$ are columns of matrices U and V and thus the left and right singular vectors of matrix M, respectively, and σ_i denotes the i-th singular value (the i-th entry of the diagonal matrix Σ) for each $i \in [\min(m, n)]$.

For any matrix $M \in \mathbb{C}^{m\times n}$, we denote the set of all singular values of M as $s(M)$. We denote its i-th singular value (in non-increasing order) as $\sigma_i(M)$, i.e., the value σ_i in the decomposition of Equation (4). We write $\sigma_{\max}(M)$ the largest singular value (i.e.,

$\sigma_{\max}(M) = \sigma_1(M)$), and write $\sigma_{\min}(M)$ the smallest non-zero singular value. We define the ℓ^2 condition number of M as $\kappa_2(M) = \sigma_{\max}(M)/\sigma_{\min}(M) \geq 1$. Note that with this definition, κ_2 is well defined even for singular matrices.

In this paper, we will use the following inequality by Weyl [19] quite often.

▶ **Lemma 3** (Weyl's inequality [19]). *For two matrices $M \in \mathbb{C}^{m\times n}$, $N \in \mathbb{C}^{m\times n}$ and any $i \in [\min(m,n)]$, $|\sigma_i(M) - \sigma_i(N)| \leq \|M - N\|_2$.*

Singular Value Transformation. We are now ready to introduce the Singular Value Transformation.

▶ **Definition 4** (Singular Value Transformation). *For any function $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $f(0) = 0$, the Singular Value Transformation associated to f is the function denoted Φ_f that maps any matrix $M \in \mathbb{C}^{m\times n}$ to the matrix $\Phi_f(M) \in \mathbb{C}^{m\times n}$ defined as follows:*

$$\Phi_f(M) = \sum_{i=1}^{\min(m,n)} f(\sigma_i) u_i v_i^*,$$

where the σ_i's, the u_i's and the v_i's correspond to the SVD of M given in Eq. (4).

It is easy to check that the value $\Phi_f(M)$ does not depend on the SVD of M chosen in the definition (i.e., it does not depend on which U and which V are chosen). Also note that from our requirement on the function f, the rank (i.e., the number of nonzero singular values) of $\Phi_f(M)$ is never larger than the rank of M.

The Moore-Penrose pseudo-inverse of matrix M is the matrix $M^+ = \sum_{i=1}^{k} \sigma_i^{-1} v_i u_i^*$, where k is the rank of the matrix M. Note that we only consider non-trivial singular values of the matrix. As in the introduction, we define the inverse function $\mathrm{inv}\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $\mathrm{inv}(0) = 0$ and $\mathrm{inv}(x) = 1/x$ for $x > 0$. Then we have $\Phi_{\mathrm{inv}}(M^*) = M^+$. Note that $MM^+ = M\Phi_{\mathrm{inv}}(M^*) = \Pi_{\mathrm{col}(M)}$ and $M^+M = \Phi_{\mathrm{inv}}(M^*)M = \Pi_{\mathrm{row}(M)}$, where $\Pi_{\mathrm{col}(M)}$ denotes the orthogonal projector into the column space of M and $\Pi_{\mathrm{row}(M)}$ denotes the orthogonal projector into the row space of M.

2.2 ℓ^2-norm sampling

We now present the assumptions to sample from a matrix and then introduce the technique of ℓ^2-norm sampling that has been used in previous works [2, 6, 9, 11, 18].

Sample accesses to matrices. Let $M \in \mathbb{C}^{m\times n}$ be a matrix. We say that we have sample access to M if the following conditions hold:

1. We can sample from the probability distribution $\mathcal{R}_M\colon [m] \to [0,1]$ defined as $\mathcal{R}_M(i) = \frac{\|M_{(i,.)}\|^2}{\|M\|_F^2}$ for any $i \in [m]$.
2. For each $i \in [m]$, we can sample from the probability distribution $\mathcal{R}_M^i\colon [n] \to [0,1]$ defined as $\mathcal{R}_M^i(j) = \frac{|M_{(i,j)}|^2}{\|M_{(i,.)}\|^2}$ for any $j \in [n]$. (Note that $\mathcal{R}_M^i$ is precisely the distribution $\mathcal{P}_u$ introduced in Section 2, where u is the i-th row of M.)

We define sample access to a vector $v \in \mathbb{C}^m$ using the same definition, by taking the matrix $M \in \mathbb{C}^{m\times 1}$ that has v as unique row. Note that with this definition, the distribution $\mathcal{R}_M$ is precisely the distribution $\mathcal{P}_v$ introduced in Section 2.1.

For an algorithm handling matrices and vectors using sample accesses, the sample complexity of the algorithm is defined as the total number of samples used by the algorithm.

ℓ^2-norm sampling. Let $M \in \mathbb{C}^{m\times n}$ be a matrix for which we have sample access. Consider the following process. For some integer $q \geq 1$, sample q row indices $p_1, p_2, \ldots, p_q \in [m]$ using the probability distribution $\mathcal{R}_M$ and then form the matrix $N \in \mathbb{C}^{q\times n}$ by defining

$$N_{(i,.)} = \frac{M_{(p_i,.)}}{\|M_{(p_i,.)}\|} \frac{\|M\|_F}{\sqrt{q}}$$

for each $i \in [q]$. Note that this corresponds to selecting the rows with indices $p_1, \ldots, p_q$ of M and re-normalizing them. We will also use the following fact which is easy to observe using the definition of matrix N :

$$\|N\|_F = \|M\|_F \tag{5}$$

The central insight of the ℓ^2-norm sampling approach introduced in [9] is that the matrix N obtained by this process is in some sense close enough to M to be able to perform several interesting calculations. We will in particular use the following result that shows that when q is large enough, with high probability the matrix N^*N is close to the matrix M^*M.

▶ **Lemma 5** (Lemma 2 in [9]). *For any $\eta \in (0,1)$, any $\beta > 0$ and for $q \geq \frac{1}{\eta\beta^2}$, the inequality $\|M^*M - N^*N\|_F \leq \beta\|M\|_F^2$ holds with probability at least $1-\eta$.*

2.3 Data structures for storing matrices

The following proposition shows that there exist low over-head data structures that enable sampling access to matrices.

▶ **Proposition 6** ([18]). *There exists a tree-like data structure that stores a matrix $M \in \mathbb{C}^{m\times n}$ in $O(a\log^2(mn))$ space, where a denotes the number of non-zero entries of M, and supports the following operations:*

1) *Output $\|M\|_F^2$ in $O(1)$ time;*
2) *Read and update an entry $M_{(i,j)}$ in $O(\log^2(mn))$ time;*
3) *Output $\|M_{(i,.)}\|$ in $O(\log^2(m))$ time;*
4) *Sampling from $\mathcal{R}_M$ in $O(\log^2(mn))$ time;*
5) *For any $i \in [m]$, sampling from $\mathcal{R}_M^i$ in $O(\log^2(mn))$ time.*

The data structure of Proposition 6 can naturally be used to store vectors as well.

We will need the following two technical lemma in our main algorithms. Lemma 7 shows that a vector-matrix-vector product can be efficiently approximated given sampling access. Lemma 8 states that, given sampling access to k vectors represented by a $n \times k$ matrix, sampling from their linear combination is possible.

▶ **Lemma 7** ([6]). *Let $v \in \mathbb{C}^m$ and $w \in \mathbb{C}^n$ be two vectors and $M \in \mathbb{C}^{m\times n}$ be a matrix, all stored in the data structure specified in Proposition 6. Then for any $\epsilon' > 0$ and $\delta > 0$, the value v^*Mw can be approximated with additive error ϵ' with probability at least $1-\delta$ in sample complexity $O\left(\frac{\|v\|^2\|w\|^2\|M\|_F^2}{\epsilon'^2}\log\left(\frac{1}{\delta}\right)\right)$ and time complexity $O\left(\frac{\|v\|^2\|w\|^2\|M\|_F^2}{\epsilon'^2}\text{polylog}\left(\frac{mn}{\delta}\right)\right)$.*

▶ **Lemma 8** ([18]). *Let $M \in \mathbb{C}^{n\times k}$ be a matrix stored in the data structure specified in Proposition 6. Let $v \in \mathbb{C}^k$ be an input vector. Then a sample from Mv can be obtained in expected sample complexity $O\left(k^2C(M,v)\right)$ and expected time complexity $O\left(k^2C(M,v)\log^2(nk)\right)$, where $C(M,v) = \frac{\sum_{i=1}^k \|v_iM_{(.,i)}\|^2}{\|Mv\|^2}$.*

3 Formal Versions and Proofs of the Main Theorems

We now give the formal versions of Theorems 1 and 2 presented in the introduction. In this section, κ_2 will always denote the ℓ^2 condition number of the matrix A. We define the intervals L and Q (which depend on A) as follows:

$$L = \left[\frac{\|A\|_2}{\sqrt{2}\kappa_2}, \frac{\|A\|_2}{\sqrt{2}\kappa_2}\sqrt{(2\kappa_2^2+1)}\right] \quad \text{and} \quad Q = \left[\frac{\|A\|_2^2}{2\kappa_2^2}, \frac{\|A\|_2^2}{2\kappa_2^2}\left(2\kappa_2^2+1\right)\right]. \tag{6}$$

▶ **Theorem 1** (Formal Version). *Let $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be any function such that $f(0) = 0$. For any $\eta > 0$ and any sufficiently small $\epsilon_1 > 0$, there exists a classical algorithm that has sampling access as in Proposition 6 to a matrix $A \in \mathbb{C}^{m\times n}$ and to a non-zero vector $b \in \mathbb{C}^m$, receives as input an index $i \in [n]$ and has the following behavior: if f is differentiable on the set L, the algorithm outputs with probability at least $1-\eta$ a value λ such that $|(\Phi_f(A^*)b)_i - \lambda| \leq \epsilon_1$, using*

$$O\left(\frac{\|A\|_F^8\|b\|^4\kappa_2^4}{\epsilon_1^4\eta}\left(\frac{\kappa_2}{\|A\|_2}\right)^6\Omega^2\left\{\phi + 3\sqrt{2}\Omega\frac{\kappa_2}{\|A\|_2}\right\}^2 \operatorname{polylog}\left(\frac{mn}{\eta}\right)\right)$$

samples and

$$O\left(\frac{\|A\|_F^{12}\|b\|_2^6\kappa_2^{12}}{\epsilon_1^6\eta^3}\left\{\phi + 7\sqrt{2}\Omega\frac{\kappa_2}{\|A\|_2}\right\}^6 \operatorname{polylog}(mn)\right)$$

time complexity, where $\Omega = \max_{\sigma\in L}|f(\sigma)|$ and $\phi = \max_{\sigma\in L}|f'(\sigma)|$.

▶ **Theorem 2** (Formal Version). *Let $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be any function such that $f(0) = 0$ and $f(x) > 0$ for all $x > 0$. For any $\eta > 0$ and any sufficiently small $\epsilon_2 > 0$, there exists a classical algorithm that has sampling access as in Proposition 6 to a matrix $A \in \mathbb{C}^{m\times n}$ and to a non-zero vector $b \in \mathbb{C}^m$ and has the following behavior: if f is differentiable on the set L and the projection of b on the column space of $\Phi_f(A^*)$ has norm $\Omega(\|b\|)$, with probability at least $1-\eta$ the algorithm samples from a distribution which is ϵ_2-close in total variation distance to the distribution $\mathcal{P}_{\Phi_f(A^*)b}$, using*

$$O\left(\frac{\|A\|_F^{10}\kappa_2^{14}}{\epsilon_2^4\eta^2\|A\|_2^6}\left(\frac{\Omega}{\omega}\right)^2\left\{\frac{\phi}{\omega} + 3\sqrt{2}\frac{\Omega}{\omega}\frac{\kappa_2}{\|A\|_2}\right\}^4 \operatorname{polylog}(mn)\right)$$

samples and

$$O\left(\frac{\|A\|_F^{12}\kappa_2^{12}}{\epsilon_2^6\eta^3}\left\{\frac{\phi}{\omega} + 7\sqrt{2}\frac{\Omega}{\omega}\frac{\kappa_2}{\|A\|_2}\right\}^6 \operatorname{polylog}(mn)\right)$$

time complexity, where $\Omega = \max_{\sigma\in L}|f(\sigma)|$, $\phi = \max_{\sigma\in L}|f'(\sigma)|$ and $\omega = \min_{\sigma\in L}|f(\sigma)|$.

Theorems 1 and 2 are stated for a fixed function f and their correctness is guaranteed for matrices A such that f is differentiable on L (remember that L depends on A). Another way of interpreting these theorems is as follows: for a matrix A and vector b (given as inputs), the algorithms of Theorems 1 and 2 work for any function $f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $f(0) = 0$ (and $f(x) > 0\ \forall\ x > 0$ for Theorem 2) that is differentiable in the set L.

Section 3 is organized as follows. Section 3.1 presents a crucial lemma that gives an upper bound on $\|\Phi_g(X) - \Phi_g(Y)\|_F$ in terms of $\|X - Y\|_F$, the values of g and the values of its derivative g'. In Section 3.2 we present our central procedure, which performs row and column sampling to compute a matrix $P' \in \mathbb{C}^{r\times c}$, and analyze this procedure using the lemma proved in Section 3.1. Finally, in Section 3.3 we prove Theorems 1 and 2 by applying appropriate post-processing to the matrix P'.

3.1 Bound on the distance between two singular value transformations

The following lemma uses a result from [10] in order to derive an upper bound on the distance between two singular value transformations of positive semi-definite matrices. The proof of the lemma has been omitted.

▶ **Lemma 10.** *Let $X, Y \in \mathbb{C}^{m\times m}$ be two $m \times m$ positive semi-definite matrices, and write $S = \mathrm{Conv}\left((s(X) \cup s(Y)) \setminus \{0\}\right)$. For any function $g\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $g(0) = 0$ and g is differentiable in S, we have: $\|\Phi_g(X) - \Phi_g(Y)\|_F \leq \|X - Y\|_F \cdot \max_{\sigma\in S}\left\{|g'(\sigma)| + \left|\frac{g(\sigma)}{\sigma}\right|\right\}$.*

3.2 Core procedure

Let us consider Algorithm 1 below. The goal of this subsection is to analyze its behavior.

Algorithm 1 Computing the matrix P'.

Parameters: Three real numbers $\theta, \gamma \in \left(0, \frac{\|A\|_2^2}{4\kappa_2^2\|A\|_F^2}\right)$ and $\eta \in (0,1)$

Input: $A \in \mathbb{C}^{m\times n}$ stored in the data structure specified in Proposition 6

1: Set $r = \lceil 3/(\eta\theta^2)\rceil$.
2: Set $c = \lceil 3/(\eta\gamma^2)\rceil$.
3: Sample r row indices p_1,...., p_r using operation 4) of Proposition 6. Let $S \in \mathbb{C}^{r\times n}$ be the matrix whose s-th row is $S_{(s,.)} = \frac{A_{(p_s,.)}}{\|A_{(p_s,.)}\|}\frac{\|A\|_F}{\sqrt{r}}$, for each $s \in [r]$.
4: Sample c column indices q_1,...., q_c by repeating the following procedure c times: sample a row index $s \in [r]$ uniformly at random and then sample a column index $q \in [n]$ with probability $\frac{|S_{(s,q)}|^2}{\|S_{(s,.)}\|^2} = \frac{|A_{(p_s,q)}|^2}{\|A_{(p_s,.)}\|^2}$ using operation 5) of Proposition 6.
5: Define the matrix $W \in \mathbb{C}^{r\times c}$ such that $W_{(s,t)} = \frac{S_{(s,q_t)}}{\|S_{(.,q_t)}\|}\frac{\|S\|_F}{\sqrt{c}} = \frac{S_{(s,q_t)}}{\|S_{(.,q_t)}\|}\frac{\|A\|_F}{\sqrt{c}}$, for each $(s,t) \in [r]\times[c]$. Query all the entries of A corresponding to entries of W using operation 2) of Proposition 6.
6: Compute the singular value decomposition of matrix W.
7: Compute the matrix $P' = \Phi_{\mathrm{inv}}(W)\Phi_f(W^*)\Phi_{\mathrm{inv}}(W)\Phi_{\mathrm{inv}}(W^*)$ using the output of the SVD step.

The sampling process of Steps 3–5 is exactly the same as in prior works [2, 6, 9, 11, 18], but with different values for c and r. The following lemma analyzes the matrices S and W obtained by this process. The proof is similar as in these prior works (but with different values for c and r).

▶ **Lemma 11.** *For any input matrix A and any parameters (θ, γ, η) in the specified range, with probability at least $1 - 2\eta/3$ the following statements are simultaneously true for the matrices S and W computed by Algorithm 1:*

$$\|S\|_F = \|A\|_F \tag{7}$$

$$\|A^*A - S^*S\|_F \leq \theta\|A\|_F^2, \tag{8}$$

$$\|SS^* - WW^*\|_F \leq \gamma\|S\|_F^2, \tag{9}$$

$$\sigma_{\min}(S) > \frac{\|A\|_2}{\sqrt{2}\kappa_2}, \qquad \sigma_{\max}(S) < \frac{\|A\|_2}{\sqrt{2}\kappa_2}\sqrt{(2\kappa_2^2+1)}, \tag{10}$$

$$\sigma_{\min}(W) > \frac{\|A\|_2}{\sqrt{2}\kappa_2}, \qquad \sigma_{\max}(W) < \frac{\|A\|_2}{\sqrt{2}\kappa_2}\sqrt{(2\kappa_2^2+1)}. \tag{11}$$

Lemma 11 above guarantees in particular that with high probability all the nontrivial singular values of the matrix S and W are in the interval L defined in Equation (6).

The main originality of our approach is Step 7 of Algorithm 1, which we now analyze. Let us define the matrix $P = \Phi_{\text{inv}}(S)\Phi_f(S^*)\Phi_{\text{inv}}(S)\Phi_{\text{inv}}(S^*)$. The following lemma shows that the output P' of Algorithm 1 is close to the matrix P. Due to space constraints, here we only give a sketch of the proof (which omits some long calculations).

▶ **Lemma 12.** *Assume that Statements (7)-(11) of Lemma 11 all hold (which happens with probability at least $1 - 2\eta/3$). Assume that f is differentiable in L and $f(0) = 0$. Then the matrix $P' \in \mathbb{C}^{r\times r}$ obtained as the output of Algorithm 1 satisfies the following inequality, where $\Omega = \max_{\sigma\in L}|f(\sigma)|$ and $\phi = \max_{\sigma\in L}|f'(\sigma)|$.*

$$\|P' - P\|_F \leq 2\gamma\|A\|_F^2 \left(\frac{\kappa_2}{\|A\|_2}\right)^4 \left\{\phi + 7\sqrt{2}\Omega\frac{\kappa_2}{\|A\|_2}\right\}, \tag{12}$$

Sketch of the proof. Let us define a function $h\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as follows. For any $\sigma \in Q$ we define $h(\sigma) = f(\sqrt{\sigma})\text{inv}(\sqrt{\sigma}) = f(\sqrt{\sigma})/\sqrt{\sigma}$, we define $h(0) = f(0)\text{inv}(0) = 0$, and we define $h(\sigma)$ arbitrarily when $\sigma \notin Q\cup\{0\}$. Since f is differentiable in L, the function h is differentiable in Q. From Equations (10) and (11) we know that $\text{Conv}\left(s(SS^*) \cup s(WW^*) \setminus \{0\}\right) \subset Q$ and can write $\Phi_h(SS^*) = \Phi_{\text{inv}}(S)\Phi_f(S^*)$ and $\Phi_h(WW^*) = \Phi_{\text{inv}}(W)\Phi_f(W^*)$.

Using the definition of P and P', we now have

$$\begin{aligned}
&\|P' - P\|_F \\
&= \|\Phi_{\text{inv}}(W)\Phi_f(W^*)\Phi_{\text{inv}}(W)\Phi_{\text{inv}}(W^*) - \Phi_{\text{inv}}(S)\Phi_f(S^*)\Phi_{\text{inv}}(S)\Phi_{\text{inv}}(S^*)\|_F \\
&= \|\Phi_h(WW^*)\Phi_{\text{inv}}(WW^*) - \Phi_h(SS^*)\Phi_{\text{inv}}(SS^*)\|_F \\
&= \|\{\Phi_h(WW^*) - \Phi_h(SS^*)\}\,\Phi_{\text{inv}}(WW^*) + \Phi_h(SS^*)\,\{\Phi_{\text{inv}}(WW^*) - \Phi_{\text{inv}}(SS^*)\}\|_F \\
&\leq \|\{\Phi_h(WW^*) - \Phi_h(SS^*)\}\,\Phi_{\text{inv}}(WW^*)\|_F + \|\Phi_h(SS^*)\,\{\Phi_{\text{inv}}(WW^*) - \Phi_{\text{inv}}(SS^*)\}\|_F \\
&\leq \|\Phi_{\text{inv}}(WW^*)\|_2\|\Phi_h(WW^*) - \Phi_h(SS^*)\|_F + \|\Phi_h(SS^*)\|_2\|\Phi_{\text{inv}}(WW^*) - \Phi_{\text{inv}}(SS^*)\|_F.
\end{aligned}$$

Using Lemma 10 twice for Φ_h and Φ_{inv}, we obtain

$$\begin{aligned}
\|P' - P\|_F \leq\ &\|\Phi_{\text{inv}}(WW^*)\|_2\|WW^* - SS^*\|_F \left(\max_{\sigma\in Q}\left\{|h'(\sigma)| + \left|\frac{h(\sigma)}{\sigma}\right|\right\}\right) \\
&+ \|\Phi_h(SS^*)\|_2\|WW^* - SS^*\|_F \left(\max_{\sigma\in Q}\left\{|\text{inv}'(\sigma)| + \left|\frac{\text{inv}(\sigma)}{\sigma}\right|\right\}\right).
\end{aligned}$$

Now we use condition (9). Also, since the non-trivial singular values of SS^* and WW^* lie in the set Q, the non-trivial singular values of S and W lie in set L (i.e., if $\sigma \in Q$ then $\sigma^{1/2} \in L$). Using this observation, we can then derive the claimed upper bound by routine calculations (omitted here). ◀

The next proposition is the main result of this subsection. Due to space constraints, the proof has been omitted.

▶ **Proposition 13.** *Let $b \in \mathbb{C}^m$ be any non-zero vector and ϵ be any positive number such that*

$$\epsilon < \frac{1}{2}\|A\|_2\|b\|\left\{\phi + 3\sqrt{2}\Omega\frac{\kappa_2}{\|A\|_2}\right\}. \tag{13}$$

Let us fix the parameters of Algorithm 1 as follows:

$$\theta = \epsilon \left(2\|A\|_F^2 \frac{\kappa_2^2}{\|A\|_2} \left\{ \phi + 3\sqrt{2}\Omega \frac{\kappa_2}{\|A\|_2} \right\} \|b\| \right)^{-1}, \tag{14}$$

$$\gamma = \epsilon \left(2\|A\|_F^2 \frac{\kappa_2^2}{\|A\|_2} \left\{ \phi + 7\sqrt{2}\Omega \frac{\kappa_2}{\|A\|_2} \right\} \|b\| \right)^{-1}. \tag{15}$$

Then, under the assumptions of Lemma 12, the two vectors $x = S^* P' S A^* b$ and $\Phi_f(A^*)b$ satisfy the inequality $\|x - \Phi_f(A^*)b\| \leq \epsilon$.

3.3 Post-processing and proofs of Theorems 1 and 2

Proof of Theorem 1. Let us write

$$\epsilon' = \frac{\epsilon_1}{4\Omega\sqrt{r\,(2\kappa_2^2+1)}} \left(\frac{\|A\|_2}{\kappa_2} \right)^2 \quad \text{and} \quad \delta' = \eta/3r. \tag{16}$$

The algorithm we consider for estimating the value $(\Phi_f(A^*)b)_i$ is described below.

Algorithm 2 Estimating $(\Phi_f(A^*)b)_i$.

1: Apply Algorithm 1 with matrix A as input, using the values θ and γ given by Equations (14) and (15) with $\epsilon = \epsilon_1/2$, and using the desired η as parameters. This returns a matrix P' and a description of a matrix S.
2: Compute an estimation z of the vector $SA^*b \in \mathbb{C}^{r\times 1}$ by estimating, for each $j \in [r]$, the quantity $S_{(j,.)}A^*b$ using Lemma 7 with parameters ϵ' and δ' given by Equation (16).
3: Compute the row vector $S^*_{(i,.)} \in \mathbb{C}^{1\times r}$ by querying all the elements in the i-th row of S^* (i.e., the i-th column of S).
4: Output the complex number $S^*_{(i,.)}P'z$.

We now analyze Algorithm 2. Let us write $x' = S^*P'z \in \mathbb{C}^{n\times 1}$, where P' and z are the matrices and the vector computed at Steps 1 and 2 of the algorithm, respectively. Remember that $P' = \Phi_{\text{inv}}(W)\Phi_f(W^*)\Phi_{\text{inv}}(W)\Phi_{\text{inv}}(W^*)$, where W is the matrix computed in Algorithm 1. Note that the output of Algorithm 2 is the i-th coordinate of the vector x'.

Let us write $x = S^*P'SA^*b$. From the analysis of Section 3.2, and especially Lemma 11 and Proposition 13, we know that Statements (10) and (11) and the inequality $\|x - \Phi_f(A^*)b\| \leq \frac{\epsilon_1}{2}$ simultaneously hold with probability $1 - 2\eta/3$.

The vector x' then satisfies the inequality

$$\begin{aligned}
\|x' - x\| &\leq \|S^*P'z - S^*P'SA^*b\| \\
&\leq \|S^*\|_2\|P'\|_2\|z - SA^*b\| \\
&\leq \|S^*\|_2\|\Phi_{\text{inv}}(W)\|_2\|\Phi_f(W^*)\|_2\|\Phi_{\text{inv}}(W)\|_2\|\Phi_{\text{inv}}(W^*)\|_2\|z - SA^*b\| \\
&\leq \left\{ \frac{\|A\|_2}{\sqrt{2}\kappa_2} \left(2\kappa_2^2+1\right)^{1/2} \right\} \left\{ \Omega \left(\frac{\sqrt{2}\kappa_2}{\|A\|_2} \right)^3 \right\} \|z - SA^*b\|,
\end{aligned}$$

where we used Statements (10) and (11) and the bound $\|\Phi_f(W^*)\|_2 \leq \Omega$ to derive the last inequality.

Lemma 7 now guarantees that with probability at least $1-\eta/3$ we have $\|z - SA^*b\| \leq \epsilon'\sqrt{r}$, which implies:

$$\|x' - x\| \leq \left\{ \frac{\|A\|_2}{\sqrt{2}\kappa_2} \left(2\kappa_2^2+1\right)^{1/2} \right\} \left\{ \Omega \left(\frac{\sqrt{2}\kappa_2}{\|A\|_2} \right)^3 \right\} \left\{ \frac{\epsilon_1}{4\Omega\sqrt{(2\kappa_2^2+1)}} \left(\frac{\|A\|_2}{\kappa_2} \right)^2 \right\} = \frac{\epsilon_1}{2}.$$

In conclusion, the inequality

$$\|x' - \Phi_f(A^*)b\| \leq \|x' - x\| + \|x - \Phi_f(A^*)b\| \leq \epsilon_1 \tag{17}$$

holds with overall probability at least $1 - \eta$ for sufficiently small $\epsilon_1 > 0$ (a precise upper bound can be derived by using Proposition 13 with $\epsilon = \epsilon_1/2$).

This implies that Algorithm 2 outputs, with probability at least $1 - \eta$, the i-th coordinate of a vector x' that satisfies Equation (17). This proves the correctness of Algorithm 2.

Let us now analyze the complexity of Algorithm 2. Algorithm 1 (and thus Step 1 of Algorithm 2) has time complexity dominated by the computation of the SVD of the matrix W, i.e.,

$$O\left(\max\left\{r^2c, rc^2\right\}\operatorname{polylog}(mn)\right) = O\left(\frac{\|A\|_F^{12}\|b\|_2^6\kappa_2^{12}}{\epsilon_1^6\eta^3}\left\{\phi + 7\sqrt{2}\Omega\frac{\kappa_2}{\|A\|_2}\right\}^6\operatorname{polylog}(mn)\right).$$

Algorithm 1 uses $r + c$ samples.

Observe that $\|S_{(j,.)}\| = \frac{\|A\|_F}{\sqrt{r}}$ for any $j \in [r]$ (see Step 3 of Algorithm 1). Step 2 of Algorithm 2 thus uses

$$O\left(\frac{\|S_{(j,.)}\|^2\|b\|^2\|A^*\|_F^2}{\epsilon'^2}\operatorname{polylog}\left(\frac{mn}{\delta}\right)r\right) = O\left(\frac{\|A\|_F^8\|b\|^4\kappa_2^4}{\epsilon_1^4\eta}\left(\frac{\kappa_2}{\|A\|_2}\right)^6\Omega^2\left\{\phi + 3\sqrt{2}\Omega\frac{\kappa_2}{\|A\|_2}\right\}^2\operatorname{polylog}\left(\frac{mn}{\eta}\right)\right)$$

samples, and has the same time complexity.

Finally, Step 3 of Algorithm 2 has time complexity $O(r)$, while Step 4 has time complexity $O(r^2)$. These two steps do not use any sample.

In conclusion, the time complexity of Algorithm 2 is dominated by Step 1, while the sample complexity is dominated by Step 2. ◀

Proof sketch of Theorem 2. Let us write

$$\epsilon'' = \frac{\epsilon_2\omega\alpha\|b\|}{8\Omega\sqrt{r\left(2\kappa_2^2+1\right)}}\left(\frac{\|A\|_2}{\kappa_2}\right)^2 \quad \text{and} \quad \delta'' = \eta/3r, \tag{18}$$

where α is a constant such that the norm of the projection of b on the column space of $\Phi_f(A)$ is at least $\alpha\|b\|$. The algorithm we use to sample from a distribution ϵ_2-close to $\mathcal{P}_{\Phi_f(A^*)b}$ is described below.

Algorithm 3 Sample access to a distribution ϵ_2-close to $\mathcal{P}_{\Phi_f(A^*)b}$.

1: Apply Algorithm 1 with matrix A as input, using the values θ and γ given by Equations (14) and (15) with $\epsilon = \frac{\epsilon_2\omega\alpha}{4}\|b\|$, and using the desired η as parameters. This returns a matrix P' and a description of a matrix S.
2: Compute an estimation z of the vector $SA^*b \in \mathbb{C}^{r\times 1}$ by estimating, for each $j \in [r]$, the quantity $S_{(j,.)}A^*b$ using Lemma 7 with parameters ϵ'' and δ'' given by Equation (18).
3: Compute the vector $P'z$.
4: Use Lemma 8 to output a sample from $x' = S^*P'z$.

Note that Algorithm 3 is very similar to Algorithm 2: the main modification is Step 4. Also note that we can use Lemma 8 since we have sample access to the columns of S^*, from the information obtained at Step 1, and we can compute the vector $P'z$ from the information obtained at Steps 1 and 2.

The complete analyses of the correctness and the complexity of Algorithm 3, which are similar to the analyses done for Algorithm 2 in the proof of Theorem 2, are omitted due to space constraints. ◀

References

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

2 Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. Quantum-inspired algorithms in practice. *arXiv:1905.10415*, 2019.

3 Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549:195–202, 2017.

4 Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation. In *Proceeding of the 46th International Colloquium on Automata, Languages, and Programming*, pages 33:1–33:14, 2019.

5 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 Annual ACM Symposium on Theory of Computing*, to appear, 2020. arXiv:1910.06151.

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

7 Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

8 B. David Clader, Bryan C. Jacobs, and Chad R. Sprouse. Preconditioned quantum linear system algorithm. *Physical Review Letters*, 110:250504, 2013.

9 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.

10 Michael I. Gil. Perturbations of functions of diagonalizable matrices. *Electronic Journal of Linear Algebra*, 27(1):645, 2014.

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

12 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 Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204, 2019.

13 Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 15(103):150502, 2009.

14 Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, pages 49:1–49:21, 2017.

15 Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10:631–633, 2014.

16 Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

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

18 Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual Symposium on Theory of Computing*, pages 217–228, 2019.

19 Hermann Weyl. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Mathematische Annalen*, 71(4):441–479, 1912.

20 Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical Review Letters*, 120:050502, 2018.