Open access • Proceedings Article • DOI:10.1145/3357713.3384314

# Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing Quantum machine learning — Source link

Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin ...+2 more authors

**Institutions:** University of Texas at Austin, California Institute of Technology, University of Maryland, College Park, University of Washington

**Published on:** 22 Jun 2020 - Symposium on the Theory of Computing

**Topics:** Quantum machine learning, Linear algebra, Low-rank approximation, Matrix (mathematics) and Support vector machine

# Sampling-Based Sublinear Low-Rank Matrix Arithmetic Framework for Dequantizing Quantum Machine Learning

Nai-Hui Chia
University of Texas at Austin
Austin, Texas, USA
nai@cs.utexas.edu

András Gilyén
California Institute of Technology
Pasadena, California, USA
agilyen@caltech.edu

Tongyang Li
University of Maryland
College Park, Maryland, USA
tongyang@cs.umd.edu

Han-Hsuan Lin
University of Texas at Austin
Austin, Texas, USA
linhh@cs.utexas.edu

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

Chunhao Wang
University of Texas at Austin
Austin, Texas, USA
chunhao@cs.utexas.edu

## ABSTRACT

We present an algorithmic framework for quantum-inspired classical algorithms on close-to-low-rank matrices, generalizing the series of results started by Tang's breakthrough quantum-inspired algorithm for recommendation systems [STOC'19]. Motivated by quantum linear algebra algorithms and the quantum singular value transformation (SVT) framework of Gilyén et al. [STOC'19], we develop classical algorithms for SVT that run in time independent of input dimension, under suitable quantum-inspired sampling assumptions. Our results give compelling evidence that in the corresponding QRAM data structure input model, quantum SVT does not yield exponential quantum speedups. Since the quantum SVT framework generalizes essentially all known techniques for quantum linear algebra, our results, combined with sampling lemmas from previous work, suffices to generalize all recent results about dequantizing quantum machine learning algorithms. In particular, our classical SVT framework recovers and often improves the dequantization results on recommendation systems, principal component analysis, supervised clustering, support vector machines, low-rank regression, and semidefinite program solving. We also give additional dequantization results on low-rank Hamiltonian simulation and discriminant analysis. Our improvements come from identifying the key feature of the quantum-inspired input model that is at the core of all prior quantum-inspired results: $\ell^2$-norm sampling can approximate matrix products in time independent of their dimension. We reduce all our main results to this fact, making our exposition concise, self-contained, and intuitive.

## CCS CONCEPTS

• **Theory of computation → Quantum computation theory**; **Machine learning theory**; **Sketching and sampling**.


*STOC '20, June 22–26, 2020, Chicago, IL, USA*

ACM ISBN 978-1-4503-6979-4/20/06…$15.00
https://doi.org/10.1145/3357713.3384314

## KEYWORDS

quantum machine learning, low-rank approximation, sampling, quantum-inspired algorithms, quantum machine learning, dequantization

**ACM Reference Format:**
Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. 2020. Sampling-Based Sublinear Low-Rank Matrix Arithmetic Framework for Dequantizing Quantum Machine Learning. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20), June 22–26, 2020, Chicago, IL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3357713.3384314

## 1 INTRODUCTION

### 1.1 Motivation

Quantum machine learning (QML) is a relatively new field of study with a rapidly growing number of proposals for how quantum computers could significantly speed up machine learning tasks [10, 19]. If any of these proposals yield substantial practical speedups, it could be the killer application motivating the development of scalable quantum computers [38]. Many of the proposals are based on Harrow, Hassidim, and Lloyd's algorithm (HHL) for solving sparse linear equation systems in time poly-logarithmic in input size [25]. However, QML applications are less likely to admit exponential speedups in practice compared to, say, Shor's algorithm for factoring [42], because unlike their classical counterparts, QML algorithms must make strong input assumptions and learn relatively little from their output [1]. These caveats arise because both loading input data into a quantum computer and extracting amplitude data from an output quantum state are hard in their most generic forms.

A recent line of research analyzes the speedups of QML algorithms by developing classical counterparts that carefully exploit these restrictive input and output assumptions. This began with a breakthrough 2018 paper by Tang [45] showing that the quantum recommendation systems algorithm [29], previously believed to be one of the strongest candidates for a practical exponential speedup in QML, does not give an exponential speedup. Specifically, Tang described a "dequantized" algorithm that solves the same problem as the quantum algorithm and only suffers from a polynomial slowdown. Tang's algorithm crucially exploits the structure of the input assumed by the quantum algorithm, which is used for efficiently preparing states. Subsequent work relies on similar techniques to

dequantize a wide range of QML algorithms, including those for principal component analysis and supervised clustering [44], low-rank linear system solving [9, 21], low-rank semidefinite program solving [8], support vector machines [13], nonnegative matrix factorization [7], and minimal conical hull [18]. These results show that the advertised exponential speedups of many QML algorithms disappear if the corresponding classical algorithms can use input assumptions analogous to the state preparation assumptions of the quantum algorithms. Previous papers [9, 21, 44] have observed that these techniques can likely be used to dequantize all QML that operates on low-rank data. Apart from a few QML algorithms that assume sparse input data [25], much of QML depends on some low-rank assumption. As a consequence, these dequantization results have drastically changed our understanding of the landscape of potential QML algorithm speedups, by either providing strong barriers for or completely disproving the existence of exponential quantum speedups for the corresponding QML problems.

Recent works on quantum algorithms use the primitive of singular value transformation to unify many quantum algorithms ranging from quantum walks to QML, under a quantum linear algebra framework called quantum singular value transformation (QSVT) [6, 22, 34]. Since this framework effectively captures all known linear algebraic QML techniques, a natural question is what aspects of this framework can be dequantized. Understanding the quantum-inspired analogue of QSVT promises a unification of dequantization results and more intuition about potential quantum speedups, which helps to guide future quantum algorithms research.

## 1.2 Main Results

Our work gives a *simple* framework of quantum-inspired classical algorithms with *wide applicability*, grasping the capabilities and limitations of these techniques. We use this framework to dequantize many quantum linear algebra algorithms, including QSVT with certain input models. We give an overview of our results here, deferring proofs to the full version of this paper.

*Sampling and query access model.* Our framework assumes a specific input model called *sampling and query access*, which can be thought of as a classical analogue to quantum state preparation assumptions, i.e., the ability to prepare a state $|v\rangle$ proportional to some input vector $v$. If we have sampling and query access to a vector $v \in \mathbb{C}^n$, denoted $\mathrm{SQ}(v)$, we can efficiently make the following kinds of queries (Definition 2.5): (1) given an index $i \in [n]$, output the corresponding entry $v(i)$; (2) sample an index $j \in [n]$ with probability $|v(j)|^2/\|v\|^2$; and (3) output the vector's $\ell^2$-norm $\|v\|$. If we have sampling and query access to a matrix $A \in \mathbb{C}^{m\times n}$, denoted $\mathrm{SQ}(A)$, we have $\mathrm{SQ}(A(i,\cdot))$ for all rows $i$ and also $\mathrm{SQ}(a)$ for $a$ the vector of row norms (i.e., $a(i) := \|A(i,\cdot)\|$).

To motivate this definition, we make the following observations about this input model. First, this model naturally admits classical algorithms with similar properties to the corresponding QML algorithms. Second, as far as we know, if input data is given *classically*,[1] classical algorithms in the sampling and query model can be run whenever the corresponding algorithms in the quantum model can (Remark 2.13). For example, if input is loaded in the QRAM data structure, as commonly assumed in QML in order to satisfy state preparation assumptions [10, 37], then we have log-time sampling and query access to it. Consequently, a fast classical algorithm for a problem in this classical model implies lack of quantum speedup for the problem.

*Matrix arithmetic.* We make a conceptual contribution by defining the slightly more general notion of *oversampling and query access* to a vector or matrix (Definition 2.7), where we have sampling and query access to another vector/matrix that gives an entry-wise upper bound on the absolute values of the entries of the actual vector/matrix, which we can only query. With this definition comes the insight that *this input model is closed under arithmetic operations*. Though this closure property comes into play relatively little in applications to dequantizing QML, the essential power of quantum-inspired algorithms lies in its ability to use sampling and query access to input matrices to build oversampling and query access to increasingly complex arithmetic expressions on input, possibly with some approximation error, without paying the (at least) linear time necessary to compute such expressions in conventional ways. As a simple example, if we have oversampling and query access to matrices $A^{(1)}, \ldots, A^{(t)}$, we have oversampling and query access to linear combinations $\sum_{i=1}^t \lambda_i A^{(i)}$ as well (Lemma 2.12).

The "oversampling" input model is also closed under (approximate) matrix products — the key technique underlying our main results. Such results have been known in the classical literature for some time [14]; we now give an example illustrating the flavor of main ideas. Suppose we are given sampling and query access to two matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{m\times p}$, and desire (over)sampling and query access to $A^\dagger B$. $A^\dagger B$ is a sum of outer products $\sum A(i,\cdot)^\dagger B(i,\cdot)$, so we can randomly sample them to get a good estimator for $A^\dagger B$. We can use $\mathrm{SQ}(A)$ to pull samples $i_1, \ldots, i_s$ according to the row norms of $A$, a distribution we will denote $p$ (so $p(i) = \|A(i,\cdot)\|^2/\|A\|_F^2$). Consider $Z := \frac{1}{s}\sum_{k=1}^s \frac{1}{p(i_k)} A(i_k,\cdot)^\dagger B(i_k,\cdot)$. $Z$ is an unbiased estimator of $A^\dagger B$:

$$\mathbb{E}[Z(i,j)] = \frac{1}{s}\sum_{k=1}^{s}\sum_{\ell=1}^{n} p(\ell)\frac{A(\ell,i)^\dagger B(\ell,j)}{p(\ell)} = [A^\dagger B](i,j).$$

Further, the variance of this estimator is small. In the following computation, we consider $s = 1$, because the variance for general $s$ decreases as $1/s$.

$$\mathbb{E}[\|A^\dagger B - Z\|_F^2] \le \sum_{i,j}\mathbb{E}[|Z(i,j)|^2] = \sum_{i,j,\ell} p(\ell)\frac{|A(\ell,i)|^2|B(\ell,j)|^2}{p(\ell)^2}$$

$$= \sum_{\ell}\frac{1}{p(\ell)}\|A(\ell,\cdot)\|^2\|B(\ell,\cdot)\|^2 = \sum_{\ell}\|A\|_F^2\|B(\ell,\cdot)\|^2 = \|A\|_F^2\|B\|_F^2.$$

Due to Chebyshev's inequality, we can approximate $A^\dagger B$ by a relatively small linear combination of outer products of rows of $A$ and the corresponding rows of $B$ with high success probability. Moreover, if we have $\mathrm{SQ}(A)$ and $\mathrm{SQ}(B)$, then we also have (over)sampling and query access to the outer products (Lemma 2.11). Using that oversampling and query access is closed under taking linear combinations (Lemma 2.12), this also yields oversampling and query access to $Z \approx A^\dagger B$. In our applications we will keep $Z$ as an outer

[1] This assumption is important. When input data is quantum (say, it is gathered experimentally from a quantum system), a classical computer has little hope of performing linear algebra on it efficiently.

product $A'^{\dagger}B'$ for convenience. Nevertheless, our central tool will be an approximate matrix product protocol: see the key lemma in Section 1.4.

Note that the discussion so far suggests that for a matrix $A$ we need (over)sampling and query access to both $A$ and $A^{\dagger}$. In fact, we show in the full version that having either one of them suffices.

So far, we have shown that if we have (over)sampling and query access to our vectors and matrices, we can perform ordinary linear algebra operations, i.e. matrix arithmetic. We will leverage our approximate matrix product protocol to add matrix functions to our toolkit: given oversampling and query access to an input matrix, we can get oversampling and query access to an approximation of a (Lipschitz) function applied to that matrix. Therefore, one can think about oversampling and query access as a classical analogue to the quantum block-encodings in quantum singular value transformation [22], which support linear combinations, products, and low-degree polynomials (that is, approximations of Lipschitz functions) of input matrices.

*Even singular value transformation.* Our main result is that, given (over)sampling and query access to an input matrix $A \in \mathbb{C}^{m\times n}$, we can find a succinct an efficient description of an *even singular value transformation* of $A$. This primitive is based on the even SVT used by Gilyén et al. [22]: given a function $f\colon [0,\infty) \to \mathbb{C}$, the even SVT is[2] $f(\sqrt{A^{\dagger}A})$, applying $f$ to the singular values of $A$ and replacing left singular vectors with the corresponding right singular vectors (so if $A = \sum \sigma_i u_i v_i^{\dagger}$ is the singular value decomposition of $A$, then $f(\sqrt{A^{\dagger}A}) = \sum f(\sigma_i) v_i v_i^{\dagger}$). The primitive of singular value transformation has been shown to generalize a large portion of quantum machine learning algorithms [22]; we bring this observation into the quantum-inspired landscape.

**Main theorem** (informal version of Theorem 3.2). *Suppose we are given sampling and query access to a matrix $A \in \mathbb{C}^{m\times n}$ (that is, $\mathrm{SQ}(A)$) and a function $f\colon [0,\infty) \to \mathbb{C}$ such that $f$ and $\bar{f}(x) := (f(x) - f(0))/x$ are $L$-Lipschitz and $L'$-Lipschitz, respectively. Then, for sufficiently small $\varepsilon, \delta > 0$, we can find a subset of (normalized) rows of $A$, $R \in \mathbb{C}^{r\times n}$, and a subset of (normalized) columns of $R$, $C \in \mathbb{C}^{r\times c}$ such that*

$$\Pr\left[\|R^{\dagger}\bar{f}(CC^{\dagger})R + f(0)I - f(A^{\dagger}A)\| > \varepsilon\right] < \delta.$$

*Let $T$ be the time the sampling and query oracle takes to respond. Then finding $R$ and $C$ and computing $\bar{f}(CC^{\dagger})$ takes $O\big(r^2c + rcT\big)$ time, where*

$$r = \tilde{\Theta}\Big(\frac{L^2\|A\|^2\|A\|_{\mathrm{F}}^2}{\varepsilon^2}\log\frac{1}{\delta}\Big) \qquad c = \tilde{\Theta}\Big(\frac{L'^2\|A\|^6\|A\|_{\mathrm{F}}^2}{\varepsilon^2}\log\frac{1}{\delta}\Big).$$

We call $R^{\dagger}\bar{f}(CC^{\dagger})R$ an *RUR decomposition* because $R \in \mathbb{C}^{r\times n}$ is a subset of rows of the input matrix ($R$ corresponds to the 'R' of the RUR decomposition, and $\bar{f}(CC^{\dagger}) \in \mathbb{C}^{r\times r}$ corresponds to the 'U'). More precisely, an RUR decomposition expresses a desired matrix as a linear combination of outer products of rows of the input matrix.[3] The matrix $U$ encodes the coefficients in the linear combination.

We want our output in the form of an RUR decomposition, since we can describe such a decomposition implicitly just as a list of row indices and some additional coefficients, which avoids picking up a dependence on $m$ or $n$ in our runtimes. Further, having $\mathrm{SQ}(A)$ implies that we can exploit the RUR structure to gain oversampling and query access to the output matrix, enabling the evaluation of matrix-vector expressions. In particular, for an RUR decomposition, we can get oversampling and query access to approximations of $R^{\dagger}URb$ and $R^{\dagger}URMb$, for a matrix $M \in \mathbb{C}^{n\times n}$ and a vector $b \in \mathbb{C}^n$, in time independent of $n$.

More general results follow as corollaries of our main result on even SVT. For an arbitrary matrix $A$ with $\mathrm{SQ}(A)$ and $\mathrm{SQ}(A^{\dagger})$ access oracles,[4] we can perform generic (non-even) SVT (Theorem 3.4), where the output is given as an approximate *CUR decomposition* expressing the desired matrix as a linear combination of outer products of columns and rows of $A$. We can also perform eigenvalue transformation on Hermitian matrices (Theorem 3.5), where the output is given as an approximate RUR decomposition. Given an RUR (or CUR) decomposition, one can also approximately diagonalize the matrix $U$ in order to recover an approximate eigenvalue decomposition (or SVD) of the desired matrix, see e.g. Theorem 3.5.

However, using only our main theorem about even SVT, we can directly recover most existing quantum-inspired machine learning algorithms without using the more advanced Theorems 3.4 and 3.5 discussed above, yielding faster dequantization for QML algorithms. In Section 1.3, we outline our results recovering such applications.

For some intuition on error bounds and time complexity, we consider how the parameters in our main theorem behave in a restricted setting: suppose that $A$ has minimum singular value $\sigma$ and $\|A\|_{\mathrm{F}}/\sigma$ is dimension-independent.[5] This condition simultaneously bounds the rank and condition number of $A$. Further suppose[6] that $f$'s Lipschitz constant satisfies

$$L\|A\|^2 < C \max_{x,y\in[0,\|A\|^2]} |f(x) - f(y)|$$

for some dimension-independent $C$. Note that $C$ must be at least 1, therefore such an $f$ is at most $C$-times "steeper" compared to the least possible "steepness". Under these assumptions, we can get an RUR decomposition to additive error $(\varepsilon \max_{x,y\in[0,\|A\|^2]} |f(x) - f(y)|)$ in runtime independent of dimensions (i.e., $r, c$ are dimensionless). The precise runtime is

$$\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^6}{\|A\|^2\sigma^4}\frac{C^6}{\varepsilon^6}\log^3\frac{1}{\delta}\right).$$

Dependence on $\sigma$ arises because we bound $L' \le L/\sigma^2$: our algorithm's dependence on $L'$ implicitly enforces a low-rank constraint in this case. All of our analyses give qualitatively similar results to this, albeit in more general settings allowing approximately low-rank input.

[2] For a Hermitian matrix $H$ and a function $f\colon \mathbb{R} \mapsto \mathbb{C}$, $f(H)$ denotes applying $f$ to the eigenvalues of $H$. That is, $f(H) := \sum_{i=1}^{n} f(\lambda_i) v_i v_i^{\dagger}$, for $\lambda_i$ and $v_i$ the eigenvalues and eigenvectors of $H$.

[3] This is the relevant variant of the notion of a *CUR decomposition* from the randomized numerical linear algebra and theoretical computer science communities [17].

[4] Only one of $\mathrm{SQ}(A)$ or $\mathrm{SQ}(A^{\dagger})$ suffices, but it is more convenient to assume both.

[5] By a dimension-independent or dimensionless quantity, we mean a quantity that is both independent of the size of the input matrix and is scale-invariant, i.e., does not change under scaling $A \leftarrow \alpha A$.

[6] This criterion is fairly reasonable. For example, the polynomials used in QSVT satisfy it.

*Implications for quantum singular value transformation.* Gilyén et al.'s QSVT framework [22] assumes that the input matrix $A$ is given by a *block-encoding*, which is a quantum circuit implementing a unitary transformation whose top-left block contains (up to scaling) $A$ itself [34]. Given a block-encoding of $A$, one can apply certain kinds of degree-$d$ polynomials of $A$ to an input quantum state, incurring only about $d$ times the implementation cost of the input block-encoding. One can get a block-encoding of an input matrix $A$ through various methods. If $A$ is $s$-sparse with efficiently computable elements and $\|A\| \leq 1$, then one can directly get a block-encoding of $A/s$ [22, Lemma 48]. If $A$ is in the QRAM data structure (used for efficient state preparation for QML algorithms [37]), one can directly get a block-encoding of $A/\|A\|_{\mathrm{F}}$ [22, Lemma 50]. This latter normalization means that QRAM-based QSVT has an implicit dependence on the Frobenius norm $\|A\|_{\mathrm{F}}$. This dependence on $\|A\|_{\mathrm{F}}$ suggests lack of exponential speedup for QRAM-based QSVT, since $\|A\|_{\mathrm{F}}$ is the key parameter in the complexity of our corresponding classical algorithms. This is in contrast to sparsity-based QSVT, which instead has dependence on $\|A\|$ and the sparsity $s$, and generalizes algorithms like HHL that solve BQP-complete problems.

Our results give compelling evidence that there is indeed no exponential speedup for QRAM-based QSVT, and show that oversampling and query access can be thought of as a classical analogue to block-encodings in the bounded Frobenius norm regime. Indeed, if we are given matrices and vectors in the QRAM data structure, then by converting them to block-encodings, we can apply any function to the input that can be obtained by composing addition, scalar multiplication, matrix multiplication, and singular value transformation. Since this data structure gives us sampling and query access to input, we can classically approximately evaluate the same types of expressions.

In particular, we show that we can apply the singular value transform of a matrix $A \in \mathbb{C}^{m\times n}$ satisfying $\|A\|_{\mathrm{F}} = 1$ to $b \in \mathbb{C}^n$ in QRAM (Theorem 3.7). Our algorithm simulates sampling and query access to $v := p^{(\mathrm{QV})}(A)b$ up to $\varepsilon\|v\|$ error in $\mathrm{poly}(d, \frac{1}{\varepsilon}, \frac{\|b\|}{\|v\|}, \log mn)$ time, where $p(x)$ is a degree-$d$ polynomial of the kind QSVT can apply and $p^{(\mathrm{QV})}(A)$ is the type of SVT that QSVT performs on $A$ (Definition 3.6). This runtime is only polynomially slower than the corresponding quantum algorithm, except in the $\varepsilon$ parameter.[7] Theorem 3.7 also dequantizes QSVT for block-encodings derived from (purifications of) density operators [22, Lemma 45] that come from some well-structured classical data. The situation in this case is even nicer, since density operators are already normalized. This gives evidence that QSVT with these kinds of block-encodings do not give inherent exponential speedups (though, if input preparation/output analysis protocols have no classical analogues, they can play a part in an algorithm achieving an exponential speedup). QSVT using other types of block-encodings (with potentially large Frobenius norm) remains intact.

[7] The QML algorithms we discuss generally only incur $\mathrm{polylog}(\frac{1}{\varepsilon})$ terms, but need to eventually pay $\mathrm{poly}(1/\varepsilon)$ to extract information from output quantum states. So, we believe this exponential speedup is artificial. See the open questions section for more discussion of this error parameter.

## 1.3 Applications: Dequantizing QML & More

With our main results, we can recover existing quantum-inspired algorithms for recommendation systems [45], principal component analysis [44], supervised clustering [44], support vector machines [13], low-rank matrix inversion [9, 21], and semidefinite program solving [8]. We also propose new quantum-inspired algorithms for low-rank Hamiltonian simulation and discriminant analysis (dequantizing the quantum algorithm of Cong & Duan [11]). Our framework achieves these results with a conceptually simple analysis, and often admits faster and more general results.

For the following results, we assume our sampling and query access to the input takes $O(1)$ time. There are data structures that can support such queries (Remark 2.13), and if the input is in QRAM, the runtime only increases by at most a factor of log of input size. We note here that, though our outputs are often in the form of oversampling and query access $\mathrm{SQ}_\phi$ (Definition 2.7), via rejection sampling, one can think about this access as the same as sampling and query access, except one can only compute the norm up to some relative error (Lemma 2.8).

*Recommendation systems.* Our framework gives a simpler and faster variant of Tang's dequantization [45] of Kerenidis & Prakash's quantum recommendation systems [29]. This result is notable for being the first result in this line of work and for dequantizing what was previously believed to be the strongest candidate for practical exponential quantum speedups for a machine learning problem [38]. The task is as follows: given sampling and query access to a matrix $A \in \mathbb{R}^{m\times n}$, a row index $i \in [m]$, and a singular value threshold $\sigma$, sample from the $i^{\text{th}}$ row of some $\hat{A} \in \mathbb{R}^{m\times n}$, where $\hat{A}$ is a $\sigma$-thresholded low-rank approximation of $A$. Specifically, $\hat{A}$ is $\varepsilon\|A\|_{\mathrm{F}}$-close in additive Frobenius norm error to a singular value transform of $A$ that is smoothly thresholded to keep only singular vectors with value at least $\sigma$.

We can rewrite our target low-rank approximation as $A \cdot t(A^\dagger A)$, where $t$ is a step function that is zero for $x \leq \frac{5}{6}\sigma^2$, one for $x \geq \frac{7}{6}\sigma^2$, and a linear interpolation between the two for $x \in [\frac{5}{6}\sigma^2, \frac{7}{6}\sigma^2]$. In other words, our low-rank approximation is $A$ multiplied by a smoothened projector. We can use our main theorem Theorem 3.2 to approximate $t(A^\dagger A)$ by some $R^\dagger U R$. Then, the $i^{\text{th}}$ row of our low-rank approximation is $A(i,\cdot)R^\dagger U R$, which is a product of a vector with an RUR decomposition. Thus, using previously-discussed matrix arithmetic lemmas, we have $\mathrm{SQ}_\phi(A(i,\cdot)R^\dagger U R)$, so we can get the sample from this row as desired. The runtime is dominated by $\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^6\|A\|^{10}}{\sigma^{16}\varepsilon^6}\log^3\frac{1}{\delta}\right)$, an improvement on the previous runtime $\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^{24}}{\sigma^{24}\varepsilon^{12}}\log^3\frac{1}{\delta}\right)$ of [45].

*Supervised clustering.* Because dequantizing Lloyd, Mohseni, and Rebentrost's supervised clustering algorithm [32] only requires simple sampling subroutines (demonstrated by Tang [44]), our algorithm trivially recovers this result. Given a dataset of points $q_1, \ldots, q_{n-1} \in \mathbb{R}^d$, the goal is to estimate the distance between their centroid and a new point $p \in \mathbb{R}^d$, $\|p - \frac{1}{n-1}(q_1 + \cdots + q_{n-1})\|^2$. We can reduce this problem to estimating $wM(wM)^\dagger$ to $\varepsilon$ additive error, for a certain choice of vector $w \in \mathbb{R}^n$ and $M \in \mathbb{R}^{n\times d}$. This can

be done with a simple inner product estimation procedure in time $O\left(\frac{Z^2}{\varepsilon^2}\log\frac{1}{\delta}\right)$, where $Z = \|M\|_{\mathrm{F}}^2\|w\|^2 = 4(\|p\|^2 + \frac{1}{n-1}\sum_{i=1}^{n-1}\|q_i\|^2)$.

*Principal component analysis.* Our framework improves on Tang's dequantization [44] of the quantum principal component analysis (qPCA) algorithm [33]. Since the actual task being solved by the original quantum algorithm is underspecified, we describe the task as is performed in the dequantization. Given a matrix $\mathrm{SQ}(X) \in \mathbb{C}^{m\times n}$ such that $X^\dagger X$ has top $k$ eigenvalues $\{\lambda_i\}_{i=1}^k$ and eigenvectors $\{v_i\}_{i=1}^k$, the goal is to compute eigenvalue estimates $\{\hat{\lambda}_i\}_{i=1}^k$ such that $\sum|\hat{\lambda}_i - \lambda_i| \le \varepsilon\,\mathrm{Tr}(X^\dagger X)$ and eigenvector estimates $\{\mathrm{SQ}_\phi(\hat{v}_i)\}_{i=1}^k$ such that $\|\hat{v}_i - v_i\| \le \varepsilon$. To avoid degeneracy conditions, we must have a gap assumption granting $|\lambda_i - \lambda_{i+1}| \ge \eta\|X\|^2$ for all $i \in [k]$.

Then, we can approach the problem as follows. First, we use that an importance-sampled submatrix of $X$ has approximately the same singular values as $X$ itself to get our estimates $\{\hat{\lambda}_i\}_{i=1}^k$. With these estimates, we can define smoothened step functions $f_i$ for $i \in [k]$ such that $f_i(X^\dagger X) = v_i^\dagger v_i$. We can then use our main theorem to find an RUR decomposition for $f_i(X^\dagger X)$. We use additional properties of the RUR description to argue that it is indeed a rank-1 outer product $\hat{v}_i^\dagger\hat{v}_i$, which is our desired approximation for the eigenvector. We have sampling and query access to $\hat{v}_i$ because it is $R^\dagger x$ for some vector $x$. Altogether, this algorithm runs in time $\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^6}{\|A\|^2\sigma^4}\varepsilon^{-6}\eta^{-6}\log^3\frac{k}{\delta}\right)$, a major improvement over the original dequantization's runtime $\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^{36}}{\sigma^{24}\|A\|^{12}}\varepsilon^{-12}\eta^{-6}\log^3\frac{k}{\delta}\right)$.

*Matrix inversion.* Our framework can generalize a pair of results giving quantum-inspired versions of low-rank matrix inversion [9, 21]. Given a matrix $\mathrm{SQ}(A) \in \mathbb{C}^{m\times n}$ and a vector $\mathrm{SQ}(b) \in \mathbb{C}^m$, the goal is to obtain $\mathrm{SQ}_\phi(A^+_{\sigma,\eta}b)$ where $A^+_{\sigma,\eta}$ is a pseudo-inverse of $A$ smoothly thresholded to invert only the singular values that are at least $\sigma$.

We can rewrite $A^+_{\sigma,\eta}b = \iota(A^\dagger A)A^\dagger b$ for $\iota$ a function encoding a thresholded inverse. Namely, $\iota(x) = 1/x$ for $x \ge \sigma^2$, $\iota(x) = 0$ for $x \le (1-\eta)\sigma^2$, and is a linear interpolation between the endpoints for $x \in [(1-\eta)\sigma^2, \sigma^2]$. By our main theorem, we can find an RUR decomposition for $\iota(A^\dagger A)$, from which we can then get $\mathrm{SQ}(R^\dagger URA^\dagger b)$ via sampling techniques. Altogether, this algorithm takes $\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^6\|A\|^{22}}{\sigma^{28}\eta^6\varepsilon^6}\log^3\frac{1}{\delta}\right)$ time with no restriction on $A$, whereas the result of [21] applies to strictly rank-$k$ $A$ and gets the incomparable runtime $\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^6k^6\|A\|^{16}}{\sigma^{22}\eta^6\varepsilon^6}\log^3\frac{1}{\delta}\right)$.

*Support vector machines.* We use our framework to dequantize Rebentrost, Mohseni, and Lloyd's quantum support vector machine [39], which was previously noted to be possible by Ding, Bao, and Huang [13]. The idea is to find a hyperplane best explaining $m$ data points in a matrix $\mathrm{SQ}(X) \in \mathbb{R}^{m\times n}$ with labels $\mathrm{SQ}(y) \in \{\pm1\}^m$. With regularization, this reduces to approximately solving the linear system

$$\begin{bmatrix} 0 & \vec{1}^\dagger \\ \vec{1} & XX^\dagger + \gamma^{-1}I \end{bmatrix}\begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}.$$

Call the above matrix $F$, and let $\hat{F} := F/\mathrm{Tr}(F)$. The quantum algorithm approximately solves the linear system by applying $\hat{F}^+_{\lambda,\eta}$ to $y$. So, our goal is to output $\mathrm{SQ}_\phi(v)$ for $v \in \mathbb{R}^{m+1}$ satisfying $\|v - \hat{F}^+_{\lambda,\eta}\left[\begin{smallmatrix}0\\y\end{smallmatrix}\right]\| \le \varepsilon\|\hat{F}^+_{\lambda,\eta}\left[\begin{smallmatrix}0\\y\end{smallmatrix}\right]\|$. To do this, we use our matrix arithmetic techniques in order to get oversampling and query access to $\mathrm{SQ}_\varphi(\hat{F})$ from $\mathrm{SQ}(X)$. Then, using $\mathrm{SQ}_\varphi(\hat{F})$, we run the quantum-inspired matrix inversion algorithm discussed above, immediately giving us the desired $v$. This takes time $\widetilde{O}\left(\lambda^{-28}\eta^{-6}\varepsilon^{-6}\log^3\frac{1}{\delta}\right)$. We solve the problem in the same generality as the original quantum algorithm, unlike the prior dequantization result [13], which also lacks explicit error bounds or runtime bounds; the paper simply argues that the algorithm is polynomial time in the right parameters.

*Hamiltonian simulation.* Our framework can be used to give a Hamiltonian simulation algorithm for low-rank Hamiltonians. Given a Hermitian matrix $\mathrm{SQ}(H) \in \mathbb{C}^{n\times n}$ such that $\|H\| \le t$ and $\|H^+\| \le 1/\sigma$ along with a unit vector $\mathrm{SQ}(b) \in \mathbb{C}^n$, the goal is to obtain $\mathrm{SQ}_\phi(v)$ where $\|v - e^{iH}b\|_{\mathrm{F}} \le \varepsilon$.

In order to use our even SVT result, we split our desired transformation into even and odd parts: $e^{ix} = \cos(x) + i\sin(x) = \cos(x) + i\,\mathrm{sinc}(x)x$. We use even singular value transformation to apply the even functions cos and sinc; for an even function $g(x)$, let $f_g(x) := g(\sqrt{x})$, so that $g(H) = f_g(H^\dagger H)$ and we can rewrite

$$e^{iH}b = f_{\cos}(H^\dagger H)b + if_{\mathrm{sinc}}(H^\dagger H)H^\dagger b.$$

Then, using our main theorem, we can find RUR decompositions for both even SVTs, gaining sampling and query access to the matrix-vector products for the even and odd parts of the expression, from which sampling and query access to our estimate of $e^{iH}b$ follows. This takes $\widetilde{O}\left(\frac{\|H\|_{\mathrm{F}}^6t^{10}}{\sigma^{16}}t^6\varepsilon^{-6}\log^3\frac{1}{\delta}\right)$ time, which is dimension-independent if we think of the desired error as $t\varepsilon$, the natural choice for additive error. This algorithm also works if $H$ is not strictly low-rank, in which case the output will be a version of $e^{iH}$ where eigenvalues $\le \sigma$ are thresholded away. We also provide a version of this algorithm that works for all $H$ without a dimension-independent runtime. This version gets improved runtimes when $t = 1$.

*Semidefinite program (SDP) solving.* We solve the problem of SDP-feasibility, improving on prior work of Chia et al. [8] dequantizing some versions of quantum SDP solvers [2, 5]. Given $m \in \mathbb{N}$, $b_1, \ldots, b_m \in \mathbb{R}$, and Hermitian matrices $A^{(1)}, \ldots, A^{(m)}$ such that $-I \le A^{(i)} \le I$ for all $i \in [m]$, let $\mathcal{S}_\varepsilon$ be the set of all $X$ satisfying

$$\begin{aligned} \mathrm{Tr}[A^{(i)}X] &\le b_i + \varepsilon \quad \forall\, i \in [m]; \\ X &\ge 0; \\ \mathrm{Tr}[X] &= 1. \end{aligned}$$

The task is to differentiate whether $\mathcal{S}_0 \ne \emptyset$ (in which case the output should be an $X \in \mathcal{S}_\varepsilon$) or $\mathcal{S}_\varepsilon = \emptyset$ (in which case the output should be "infeasible"). Note that general SDPs can be reduced to this feasibility problem via a simple binary search.

By using the matrix multiplicative weights (MMW) method [3], SDP $\varepsilon$-feasibility reduces to estimating $\mathrm{Tr}[A^{(i)}X]$ up to $\varepsilon/4$ error given $\mathrm{SQ}(A^{(i)})$ for all $i \in [m]$ and $X$ implicitly defined as a Gibbs

state

$$X := \frac{\exp[-A]}{\mathrm{Tr}(\exp[-A])} \quad \text{where} \quad A := \frac{\varepsilon}{4} \sum_{\tau=1}^{\lesssim \ln(n)/\varepsilon^2} A^{(j_\tau)}.$$

To estimate $\mathrm{Tr}[A^{(i)}X]$, we first notice that we have $\mathrm{SQ}_\phi(A)$, since it is a linear combination of matrices that we have sampling and query access to (Lemma 2.12). Then, we can find approximations of the Gibbs state by applying eigenvalue transformation (Theorem 3.5) according to the exponential function to get $\exp[-A]$ as an RUR decomposition. Then the estimation of $\mathrm{Tr}[A^{(i)}X]$ can be performed by the usual SQ sampling techniques. This strategy solves the feasibility problem and when applicable outputs the $\varepsilon$-approximate solution of the SDP as an RUR decomposition in time[8] $\widetilde{O}\left(\frac{\|A^{(\cdot)}\|_{\mathrm{F}}^{22}}{\varepsilon^{46}} \ln^{23}(n) + m\frac{\|A^{(\cdot)}\|_{\mathrm{F}}^{14}}{\varepsilon^{28}} \ln^{13}(n)\right)$.

For the same feasibility problem, the previous quantum-inspired SDP solver [8] proved a complexity bound $\widetilde{O}\left(mr^{57}\varepsilon^{-92} \ln^{37}(n)\right)$, assuming that the constraint matrices have rank at most $r$. Since the rank constraint implies that $\left\|A^{(\cdot)}\right\|_{\mathrm{F}} \leq \sqrt{r}$, under this assumption our algorithm has complexity $\widetilde{O}\left(r^{11}\varepsilon^{-46} \ln^{23}(n) + mr^7\varepsilon^{-28} \ln^{13}(n)\right)$. So, our new algorithm both solves a more general problem and also greatly improves the runtime.

*Discriminant analysis.* We present a new dequantized algorithm, a classical analogue to Cong and Duan's quantum discriminant analysis algorithm [11]. The high-level idea is to find the vectors that best explain the way data points are classified. Cong and Duan reduces this idea to the following task: given matrices $\mathrm{SQ}(B)$ and $\mathrm{SQ}(W)$, find eigenvectors and eigenvalues of $\sqrt{W^\dagger W}(B^\dagger B)^{-1}\sqrt{W^\dagger W}$. They solve a version of this task where one only needs to output approximate eigenvectors and one can ignore the singular vectors of $B$ and $W$ that are smaller than a parameter $\sigma$.

We achieve this goal by using Theorem 3.2 to approximate $\sqrt{W^\dagger W} \approx R_W^\dagger U_W R_W$ and $(B^\dagger B)^{-1} \approx R_B^\dagger U_B R_B$ by RUR decompositions. Then, we use Lemma 3.1 to approximate $R_W R_B^\dagger$ by small matrices $R'_W R_B'^\dagger$ by small submatrices. This yields an approximate RUR decomposition of the matrix whose eigenvalues and vectors we want to find:

$$R_W^\dagger (U_W R'_W R_B'^\dagger U_B R'_B R_W'^\dagger U_W) R_W.$$

Finding eigenvectors from an RUR decomposition follows from an observation: for a matrix $C_W$ formed by sampling columns from $R_W$ (using $\mathrm{SQ}(W)$), and $[C_W]_k$ the rank-$k$ approximation to $C_W$ (which can be computed because $C_W$ has size independent of dimension), $(([C_W]_k)^+ R_W)^\dagger$ is an approximate projective isometry (that is, its singular values are close to one or zero). This roughly formalizes the intuition of $C_W$ preserving the left singular vectors and singular values of $R_W$. We can rewrite $R_W^\dagger U R_W = R_W^\dagger (C_k^+)^\dagger C_k^\dagger U C_k C_k^+ R_W$, which holds by choosing $k$ sufficiently large and choosing $C$ to be the same sketch used for $U$. Then, we can compute the SVD as $C_k^+ U (C_k^+)^\dagger = VDV^\dagger$ which gives us an approximate SVD for $R_W^\dagger U R_W$: the eigenvectors are $(C_k^+ R_W)^\dagger V$, and the eigenvalues are the diagonal entries of $D$. We show that this has the approximation properties analogous to the quantum algorithm. Our algorithm runs in $\widetilde{O}\left(\left(\frac{\|B\|^4\|B\|_{\mathrm{F}}^6}{\varepsilon^6\sigma^{10}} + \frac{\|W\|^{10}\|W\|_{\mathrm{F}}^6}{\varepsilon^6\sigma^{16}}\right)\log^3\frac{1}{\delta}\right)$ time.

*What else is there?* Though we have presented many dequantized versions of QML algorithms, the question remains of what QML algorithms don't have such versions. That is, what algorithms still have the potential to give exponential speedups?

Because QSVT generalizes essentially all known quantum linear algebra techniques, we restrict our focus to algorithms in that framework. As we noted previously, we only demonstrate lack of exponential speedup for QSVT with block-encodings coming from QRAM and density operators. Other kinds of block-encodings, such as those coming from sparsity assumptions, remain impervious to our techniques. The most well-known quantum linear algebra algorithms of this "dequantization-resistant" type are HHL [25] and its derivatives. Sparse matrix inversion is BQP-complete, which explains why our techniques leave these speedups untouched. Nevertheless HHL has serious caveats, as noted by Aaronson [1]. In particular, HHL only gives an exponential speedup when the condition number of the input matrix is poly-logarithmic in dimension, which doesn't happen in typical datasets. This constraint hamstrings most attempts to apply HHL to practical problems, especially when combined with the typical QML constraints that quantum algorithms need quantum states as input and often can only give quantum states as output. Work like Zhao et al. on Gaussian process regression [50] and Lloyd et al. on topological data analysis [31] attempt to address these issues to get a super-polynomial quantum speedup.

## 1.4 Techniques

*Placing sampling and query access in the sketching context.* As we will see below, the fundamental idea of quantum-inspired algorithms is to reduce dimensionality of input matrices to speed up linear algebra computations. So, using sketching techniques is natural here. Recall that the fundamental difference between quantum-inspired algorithms and traditional sketching algorithms is that we assume that we can perform measurements of states corresponding to input in time independent of input dimension (that is, we have efficient sampling and query access to input), and in exchange want algorithms that run in time independent of dimension. The kind of samples we get from sampling and query access is usually called *importance sampling* or *length-square sampling* in classical literature.

The quantum-inspired model is weaker than the standard sketching algorithm model (Remark 2.13): an algorithm taking $T$ time in the quantum-inspired model for an input matrix $A$ can be converted to a standard algorithm that runs in time $O(\mathrm{nnz}(A) + T)$, where $\mathrm{nnz}(A)$ is the number of nonzero entries of $A$. So, we can also think about an $O(T)$-time quantum-inspired algorithm as an $O(\mathrm{nnz}(A) + T)$-time sketching algorithm, where the $\mathrm{nnz}(A)$ portion of the runtime can only be used to facilitate importance sampling. This viewpoint could be advantageous in some cases, for example

[8] Here we use $\|A^{(\cdot)}\|_* := \max_{i\in[m]} \|A^{(i)}\|_*$. Note that this bound does not appear to be dimension-independent due to the normalizing assumption $\|A^{(\cdot)}\| \leq 1$. If we would relax this assumption, then we could get a dimension-independent bound corresponding to precision $\varepsilon\|A^{(\cdot)}\|$, by replacing $\|A^{(\cdot)}\|_{\mathrm{F}}$ with the "stable rank" $\|A^{(\cdot)}\|_{\mathrm{F}}/\|A^{(\cdot)}\|$. Then the resulting runtime bound is dimension-independent apart from the $\ln(n)$ factors, that come from MMW.

in some streaming scenario [29]. Nevertheless, our primary motivation here is not to develop better generic sketching algorithms, but to better understand the scope of problems facilitating large quantum speed-ups.

A natural question is whether more modern types of sketches can be used in our model. After all, importance sampling is only one of many sketching techniques studied in the large literature on sketching algorithms. Notably, though, *other types of sketches seem to fail in the input regimes where quantum machine learning succeeds*: assuming sampling and query access to input, importance sampling takes time independent of dimension, whereas other randomized linear algebra methods such as Count-Sketch, Johnson-Lindenstrauss, and leverage score sampling all still take time linear in input-sparsity.

Furthermore, importance sampling is highly compatible with quantum-like algorithms: given the ability to query entries and obtain importance samples of the input, we can query entries and obtain importance samples of the output, analogously to the way quantum machine learning algorithms move from an input quantum state to an output quantum state. This insight unlocks surprising power in importance sampling. For example, it reveals that Frieze, Kannan, and Vempala's low-rank approximation algorithm (FKV) [20], which, as stated, requires $O(kmn)$ time to output the desired matrix, actually can produce useful results (samples and entries) in time independent of input dimension. Our goal is to develop a framework that demonstrates what can be done with importance sampling and establishes a classical frontier for quantum algorithms to push past.

*Importance sampling to even singular value transformation.* The fundamental property of importance sampling is its ability to efficiently approximate matrix products (and by extension, vectors and higher-order tensors). This is our key lemma, which states that if we have sufficient access to two matrices, we can approximate their product by a product of matrices of smaller dimension:

**Key lemma** [14] (informal version of Lemma 3.1). *Suppose we are given* $\mathrm{SQ}(X) \in \mathbb{C}^{m\times n}$ *and* $\mathrm{SQ}(Y) \in \mathbb{C}^{m\times p}$. *Then we can find normalized submatrices of* $X$ *and* $Y$, $X' \in \mathbb{C}^{s\times n}$ *and* $Y' \in \mathbb{C}^{s\times p}$, *in* $O(s)$ *time for* $s = \Theta(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$, *such that*

$$\Pr\left[\|X'^{\dagger}Y' - X^{\dagger}Y\|_{\mathrm{F}} \le \varepsilon\|X\|_{\mathrm{F}}\|Y\|_{\mathrm{F}}\right] > 1-\delta.$$

*We subsequently have* $O(s)$*-time* $\mathrm{SQ}(X'), \mathrm{SQ}(X'^{\dagger}), \mathrm{SQ}(Y'), \mathrm{SQ}(Y'^{\dagger})$.

Prior quantum-inspired algorithms [8, 9, 44, 45] indirectly used this lemma by using FKV, which finds a low-rank approximation to the input matrix in the form of an approximate low-rank SVD and relies heavily on this lemma in the analysis. By using FKV once, one can gain access to singular values and right singular vectors; by using it twice, one can gain access to a full SVD. Then, by applying functions to the approximate singular values, one can argue that the resulting expression is close to the desired expression. One could theoretically use this procedure to give a classical algorithm for singular value transformation, but we prove our main results without going through the full analysis of the low-rank approximation.

Instead, we use the key lemma twice to get an RUR decomposition of an even singular value transformation of the input (Theorem 3.2). Notice that, because we wish to run in time independent of dimension, the best we can do is to express the output based on the given input, as an RUR decomposition does. The proof of our main theorem is straightforward. Recall that, given $\mathrm{SQ}(A) \in \mathbb{C}^{m\times n}$, we wish to approximate $f(A^{\dagger}A)$ for $f$ a function that, without loss of generality, satisfies $f(0) = 0$.

$$f(A^{\dagger}A) \approx f(R^{\dagger}R) = R^{\dagger}\bar{f}(RR^{\dagger})R \approx R^{\dagger}\bar{f}(CC^{\dagger})R,$$

where the first approximation follows from the key lemma with $R \in \mathbb{C}^{r\times n}$ normalized rows of $A$, the equality follows from $\bar{f}(x) = f(x)/x$, and the second approximation follows from the key lemma with $C \in \mathbb{C}^{r\times c}$ normalized columns of $R$. We then take $\bar{f}(CC^{\dagger})$ to be the "U" of our RUR decomposition, finding it by naively computing the SVD of $C$ in $O(r^2c)$ time. The analysis is straightforward: we use that $f$ and $\bar{f}$ are Lipschitz to argue that the error from approximating our matrix products propagates well. We also use a variant of the key lemma to give a spectral norm variant of the main theorem.

Though this analysis is much simpler than FKV, it gives improved results in our applications. Our approach has several advantages. The reduction first given by Tang to get an SVT-based low-rank approximation bound from the standard notion of low-rank approximation [45, Theorem 4.7] induces a quadratic loss in precision, which appears to be only an artifact of the analysis. Also, FKV gives Frobenius norm error bounds, though for applications we often only need spectral norm bounds; our main theorem can get improved runtimes by taking advantage of the weaker spectral norm bounds. Finally, we take a reduced number of rows compared to columns, whereas FKV approximates the input by taking the same number of rows and columns.

The flexibility of singular value transformation also leads to easy generalization of results. For example, another important technical difference from previous work [8, 9, 21] is that our results do not assume that the input is strictly low-rank. Instead, following [22, 45], our algorithms work on close-to-low-rank matrices by doing SVTs that smoothly threshold to only operate on large-enough singular values. That is, we implicitly take a low-rank approximation of the input before applying our singular value transformation.

*General transformation results.* We can bootstrap our algorithm for even SVT to get results for generic SVT (Theorem 3.4) and eigenvalue transformation (Theorem 3.5).

For generic SVT: consider a function $f : \mathbb{R} \to \mathbb{C}$ satisfying $f(0) = 0$ and a matrix $A \in \mathbb{C}^{m\times n}$. Given $\mathrm{SQ}(A)$ and $\mathrm{SQ}(A^{\dagger})$, we give an algorithm to output a CUR decomposition approximating $f^{(\mathrm{SV})}(A)$. Our strategy is to apply our main result Theorem 3.2 to $g(A^{\dagger}A)$, for $g(x) := f(\sqrt{x})/\sqrt{x}$, and subsequently approximate matrix products with Lemma 3.1 to get an approximation of the form $A'R'^{\dagger}UR + g(0)A$:

$$f^{(\mathrm{SV})}(A) = Ag(A^{\dagger}A) \approx AR^{\dagger}UR + A(g(0)I) \approx A'R'^{\dagger}UR + g(0)A.$$

Here, $A'R'^{\dagger}UR$ is a CUR decomposition as desired, since $A'$ is a normalized subset of columns of $A$. One could further approximate $g(0)A$ by a CUR decomposition if necessary (e.g. by adapting the eigenvalue transformation result below). Some QML applications of even SVT look similar to this (e.g., matrix inversion and Hamiltonian simulation), but we can use the additional structure in these problems to do this kind of approximation better.

As for eigenvalue transformation, consider a function $f : \mathbb{R} \to \mathbb{C}$ and a Hermitian matrix $H \in \mathbb{C}^{n\times n}$, given SQ($H$). We wish to compute the eigenvalue transform $f(H)$. If $f$ is even (so $f(x) = f(-x)$), then $f(H) = f(\sqrt{H^\dagger H})$, so the result follows from our main theorem for even SVT.

For non-even $f$, we use a different strategy, similar to the one used for quantum-inspired semidefinite programming [8]: first we find the eigenvectors and eigenvalues of $H$ and then apply $f$ to the eigenvalues. Let $\pi(x)$ be a (smoothened) step function that is a linear interpolation between 0 and 1 on $[0.5\varepsilon^2, \varepsilon^2]$. Then

$$H \approx \pi(HH^\dagger)H\pi(H^\dagger H) \approx R^\dagger\bar{\pi}(CC^\dagger)RHR^\dagger\bar{\pi}(CC^\dagger)R$$
$$\approx R^\dagger\bar{\pi}(CC^\dagger)M\bar{\pi}(CC^\dagger)R = R^\dagger(C_\sigma C_\sigma^+)^\dagger\bar{\pi}(CC^\dagger)M\bar{\pi}(CC^\dagger)C_\sigma C_\sigma^+R,$$

where the second approximation follows from Theorem 3.2, the third approximation follows from the key lemma with $M \approx RHR^\dagger$, and $C_\sigma$ is the low-rank approximation of $C$ formed by transforming $C$ according to the "filter" function on $x$ that is 0 for $x < \sigma(< \varepsilon)$ and $x$ otherwise. $\hat{U} := C_\sigma^+ R \in \mathbb{C}^{c\times n}$ is close to an isometry, which we argue by showing $(C_\sigma^+R)(C_\sigma^+R)^\dagger \approx I$. We are nearly done now: since the rest of the matrix expression, $C_\sigma^\dagger\bar{\pi}(CC^\dagger)M\bar{\pi}(CC^\dagger)C_\sigma \in \mathbb{C}^{c\times c}$, consists of submatrices of $H$ of size independent of $n$, we can directly compute its unitary eigendecomposition $UDU^\dagger$. This gives the approximate decomposition $H \approx (\hat{U}U)D(\hat{U}U)^\dagger$, with $\hat{U}U$ and $D$ acting as approximate eigenvectors and eigenvalues of $H$, respectively. Some simple analysis shows that $f(H) \approx (\hat{U}U)f(D)(\hat{U}U)^\dagger$ in the desired sense. Therefore, our output approximation of $f(H)$ comes in the form of an RUR decomposition that can be rewritten in the form of an approximate eigendecomposition.

## 1.5 Related Work

Our work bridges the fields of randomized algorithms and quantum algorithms for linear algebra. Thus, we interact with a diverse body of related work.

*Randomized numerical linear algebra.* Generally speaking, the techniques our framework uses belong to randomized linear algebra algorithms (see the surveys [35, 48]). Our core primitive is importance sampling: see the survey by Kannan and Vempala [28] for algorithms using this type of sampling. In addition to the low-rank approximation algorithms [20] used in the quantum-inspired literature, others have used importance sampling for, e.g., orthogonal tensor decomposition [16, 36, 43] (generalizing low-rank approximation [20]) and support vector machines [26].

*Classical algorithms for quantum problems.* We are aware of two important prior results from before Tang's first paper [45] that connect quantum algorithms to randomized numerical linear algebra. The first is Van den Nest's work on using probabilistic methods for quantum simulation [46], which defines a notion of "computationally tractable" (CT) state equivalent to our notion of sampling and query access and then uses it to simulate restricted classes of quantum circuits. We share some essential ideas with this work, such as the simple sampling lemma Lemma 2.9, but dequantized algorithms critically use low-rank assumptions on the input for "simulating" QML in a way that would not be possible were we only viewing such algorithms as large quantum circuits. The second is a paper by Rudi et al. [41] that uses the Nyström method to simulate a sparse Hamiltonian $H$ on a sparse input state in time poly-logarithmic in dimension and polynomial in $\|H\|_F$, assuming sampling and query access to $H$. Our Hamiltonian simulation results do not require a sparsity assumption and still achieve a dimension-independent runtime, but get slightly larger exponents in exchange.

*Practical implementation.* A work by Arrazola et al. [4] implements and tests quantum-inspired algorithms for regression and recommendation systems. This work makes various conclusions, and for example, suggests that the $\varepsilon^2$ scaling in the number of rows/columns taken in our recommendation systems algorithm is inherent. However, we are unsure of these results' implications for the broader question of whether QML algorithms can achieve practical speedups, for two reasons. First, our algorithms use a restricted model of computation in order to get a broad asymptotic result for generic applications of quantum machine learning. However, if we wish to compare QML to the best classical algorithm in practice, other sketching algorithms are more natural to run on a classical computer and are likely to be faster. For example, Dahiya, Konomis, and Woodruff [12] conducted an empirical study of sketching algorithms for low-rank approximation on both synthetic datasets and the movielens dataset, reporting that their implementation "finds a solution with cost at most 10 times the optimal one ... but does so 10 times faster." For comparison, Arrazola et al. [4] claim that the running times of quantum-inspired algorithms are worse than directly computing the singular value decomposition for medium-sized matrices (e.g. $10^4 \times 10^4$). Second, the authors implement the quantum-inspired algorithms in a simple, non-optimized way in Python and then compare it to the well-optimized LAPACK library C implementation of singular value decomposition. These caveats make it difficult to draw definitive conclusions about the practicality of quantum-inspired algorithms as a whole from these experimental results.

*Quantum machine learning.* As mentioned in Section 1.3, our work has major implications for the landscape of quantum machine learning. In particular, our work suggests that the most promising way to get exponential speedups for algorithms fitting in the framework of quantum singular value transformation [22] is via algorithms that use sparse matrices as input (as opposed to those with input in QRAM), such as HHL [25]. Such algorithms have other major caveats (mentioned by Aaronson [1]) that make it difficult to find applications with the potential for practical super-polynomial speedups. Proposals for such applications include Gaussian process regression [50] and topological data analysis [31].

*Related independent work.* Independently from our work, Jethwani, Le Gall, and Singh [27] simultaneously derived similar results. They implicitly derive a version of our even SVT result, and use it to achieve generic SVT (approximate SQ($b^\dagger f^{(SV)}(A)$) for a vector $b$) by writing $f^{(SV)}(A) = Ag(A^\dagger A)$ for $g(x) = f(\sqrt{x})/\sqrt{x}$ and then using sampling subroutines to get the solution from the resulting expression $b^\dagger AR^\dagger UR$. It is difficult to directly compare the main SVT results, because the parameters that appear in their runtime bounds are somewhat non-standard, but one can see that for typical

choices of $f$, their results require a strictly low-rank $A$. In comparison our results apply to general $A$, and we also demonstrate how to apply them to (re)derive dequantized algorithms.

### 1.6 Open Questions

Our framework recovers recent dequantization results, and we hope that it will be used for dequantizing more quantum algorithms. In the meantime, our work leaves several natural open questions.

First, in the quantum setting, linear algebra algorithms [22] can achieve logarithmic dependence on the precision $\varepsilon$. Can classical algorithms also achieve such exponentially improved dependence, when the goal is restricted to sampling from the output (i.e., without the requirement to query elements of the output)? If not, is there a mildly stronger classical model that can achieve this? Could this exponential advantage be exploited in a meaningful way?

Second, our algorithms still have significant slowdown as compared to their quantum counterparts. Can we shave condition number factors to get runtimes of the form $\widetilde{O}\left(\frac{\|A\|_{\mathrm{F}}^6}{\sigma^6\varepsilon^6}\log^3\frac{1}{\delta}\right)$ (for the recommendation systems application, for instance)? Can we get even better runtimes by somehow avoiding SVD computation?

Finally, is there an approach to QML that does not go through HHL (whose demanding assumptions make exponential speedups difficult to demonstrate even in theory) or a low-rank assumption (which, as we demonstrate, makes the tasks "easy" for classical computers)?

## 2 PRELIMINARIES

To begin with, we define notation to be used throughout this paper. For $n \in \mathbb{N}$, $[n] := \{1, \ldots, n\}$. For $z \in \mathbb{C}$, its absolute value is $|z| = \sqrt{z^* z}$, where $z^*$ is the complex conjugate of $z$. $f \lesssim g$ denotes the ordering $f = O(g)$ (and respectively for $\gtrsim$ and $\approx$). $\widetilde{O}(g)$ is shorthand for $O(g\,\mathrm{poly}(\log g))$. Finally, we assume that arithmetic operations (e.g addition and multiplication of real numbers) and function evaluation oracles (computing $f(x)$ from $x$) take unit time, and that queries to oracles (like the queries to input discussed in Section 2.2) are at least unit time cost.

### 2.1 Linear Algebra

In this paper, we consider complex matrices $A \in \mathbb{C}^{m\times n}$ for $m, n \in \mathbb{N}$. For $i \in [m], j \in [n]$, we let $A(i,\cdot)$ denote the $i$-th row of $A$, $A(\cdot, j)$ denote the $j$-th column of $A$, and $A(i,j)$ denote the $(i,j)$-th element of $A$. $(A|B)$ denotes the concatenation of matrices $A$ and $B$ and $\mathrm{vec}(A) \in \mathbb{C}^{mn}$ denotes the vector formed by concatenating the rows of $A$. For vectors $v \in \mathbb{C}^n$, $\|v\|$ denotes standard Euclidean norm (so $\|v\| := (\sum_{i=1}^n |v_i|^2)^{1/2}$). For a matrix $A \in \mathbb{C}^{m\times n}$, the *Frobenius norm* of $A$ is $\|A\|_{\mathrm{F}} := \|\mathrm{vec}(A)\| = (\sum_{i=1}^m \sum_{j=1}^n |A(i,j)|^2)^{1/2}$ and the *spectral norm* of $A$ is $\|A\| := \|A\|_{\mathrm{Op}} := \sup_{x\in\mathbb{C}^n, \|x\|=1} \|Ax\|$.

A *singular value decomposition* (SVD) of $A$ is a representation $A = UDV^\dagger$, where for $N := \min(m,n)$, $U \in \mathbb{C}^{m\times N}$ and $V \in \mathbb{C}^{n\times N}$ are isometries and $D \in \mathbb{R}^{N\times N}$ is diagonal with $\sigma_i := D(i,i)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_N \geq 0$. We can also write this decomposition as $A = \sum_{i=1}^N \sigma_i U(\cdot,i)V(\cdot,i)^\dagger$. We now formally define singular value transformation:

**Definition 2.1.** For a function $f : [0,\infty) \to \mathbb{C}$ such[9] that $f(0) = 0$ we define the singular value transform of $A \in \mathbb{C}^{m\times n}$ via a singular value decomposition $A = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^\dagger$ as follows

$$f^{(\mathrm{SV})}(A) := \sum_{i=1}^{\min(m,n)} f(\sigma_i) u_i v_i^\dagger. \tag{1}$$

**Definition 2.2.** For a function $f : \mathbb{R} \to \mathbb{C}$ and a Hermitian $A \in \mathbb{C}^{n\times n}$ we define the eigenvalue transform of $A$ via a unitary eigendecomposition $A = \sum_{i=1}^n \lambda_i v_i v_i^\dagger$ as follows

$$f^{(\mathrm{EV})}(A) := \sum_{i=1}^{n} f(\lambda_i) v_i v_i^\dagger. \tag{2}$$

Since we only consider eigenvalue transformations of Hermitian matrices, where singular vectors/values and eigenvectors/values (roughly) coincide, the key difference is that eigenvalue transformations can distinguish eigenvalue sign. As this is the standard notion of a matrix function, we will usually drop the superscript in notation: $f(A) := f^{(\mathrm{EV})}(A)$.

We will use the following standard definition of a Lipschitz function.

**Definition 2.3.** We say $f : \mathbb{R} \to \mathbb{C}$ is $L$-Lipschitz on $\mathfrak{F} \subseteq \mathbb{R}$ if for all $x, y \in \mathfrak{F}$, $|f(x) - f(y)| \leq L|x - y|$.

### 2.2 Sampling and Query Access Oracles

Since we want our algorithms to run in time sublinear in input size, we must be careful in defining the access model. Our input model is unconventional, being designed as a reasonable classical analogue for the input model of some quantum algorithms. The sampling and query oracle we present below can be thought of as a classical analogue to a quantum state, and will be used heavily to move between intermediate steps of these quantum-inspired algorithms. First, as a warmup, we define a simple query oracle:

**Definition 2.4** (Query access)**.** For a vector $v \in \mathbb{C}^n$, we have $\mathrm{Q}(v)$, *query access* to $v$ if for all $i \in [n]$, we can obtain $v(i)$. Likewise, for a matrix $A \in \mathbb{C}^{m\times n}$, we have $\mathrm{Q}(A)$ if for all $(i,j) \in [m] \times [n]$, we can obtain $A(i,j)$. Let $\mathbf{q}(v)$ (or $\mathbf{q}(A)$) denote the (time) cost of such a query.

For example, in the typical RAM access model, we are given our input $v \in \mathbb{C}^n$ as $\mathrm{Q}(v)$ with $\mathbf{q}(v) = 1$. For brevity, we will sometimes abuse this notation (and other access notations) and write, for example, "$\mathrm{Q}(A) \in \mathbb{C}^{m\times n}$" instead of "$\mathrm{Q}(A)$ for $A \in \mathbb{C}^{m\times n}$".

**Definition 2.5** (Sampling and query access to a vector)**.** For a vector $v \in \mathbb{C}^n$, we have $\mathrm{SQ}(v)$, *sampling and query access* to $v$, if we can:

(1) Query for entries of $v$ as in $\mathrm{Q}(v)$;
(2) Obtain independent samples $i \in [n]$ following the distribution $\mathcal{D}_v \in \mathbb{R}^n$, where $\mathcal{D}_v(i) := |v(i)|^2/\|v\|^2$;
(3) Query for $\|v\|$.

Let $\mathbf{q}(v)$, $\mathbf{s}(v)$, and $\mathbf{n}(v)$ denote the cost of querying entries, sampling indices, and querying the norm respectively. Further define $\mathbf{sq}(v) := \mathbf{q}(v) + \mathbf{s}(v) + \mathbf{n}(v)$.

[9] The $f(0) = 0$ requirement ensures that the definition is independent of the (not necessarily unique) choice of SVD.

We will refer to these samples as *importance samples from* $v$, though one can view them as measurements of the quantum state $|v\rangle := \frac{1}{\|v\|}\sum v_i|i\rangle$ in the computational basis.

Quantum-inspired algorithms typically don't give exact sampling and query access to the output vector. Instead, we get a more general version of sampling and query access, which assumes we can only access a sampling distribution that *oversamples* the correct distribution.[10]

**Definition 2.6.** For $v \in \mathbb{C}^n, p \in \mathbb{R}^n_{\geq 0}$ is a *$\phi$-oversampled importance sampling distribution* of $v$ (for $\phi \geq 1$) if $\sum_{i=1}^n p(i) = 1$ and, for all $i \in [n]$, $p(i) \geq \mathcal{D}_v(i)/\phi = \frac{|v(i)|^2}{\phi\|v\|^2}$.

If $p$ is a $\phi$-oversampled importance sampling distribution of $v$, any given output $i \in [n]$ is no more than $\phi$-times rarer in $p$ compared to the desired distribution $\mathcal{D}_v$. As a result, intuitively, estimators that use $\mathcal{D}_v$ can also use $p$, with a factor $\phi$ increase in the number of samples necessary. For example, we can convert a sample from $p$ to a sample from $\mathcal{D}_v$ with probability $1/\phi$ with rejection sampling: sample an $i$ distributed as $p$, then accept the sample with probability $(\mathcal{D}_v(i)/p(i))/\phi$.

**Definition 2.7** (Oversampling and query access)**.** For $v \in \mathbb{C}^n$ and $\phi \geq 1$, we have $\mathrm{SQ}_\phi(v)$, *$\phi$-oversampling and query access* to $v$, if we have $\mathrm{Q}(v)$ and $\mathrm{SQ}(\tilde{v})$ for $\tilde{v} \in \mathbb{C}^n$ a vector satisfying $\beta := \|\tilde{v}\|^2 = \phi\|v\|^2$ and $|\tilde{v}(i)|^2 \geq |v(i)|^2$ for all $i \in [n]$. Denote $p(i) := \mathcal{D}_{\tilde{v}}(i)$, $\mathbf{s}_\phi(v) := \mathbf{s}(\tilde{v})$, $\mathbf{q}_\phi(v) := \mathbf{q}(\tilde{v})$, $\mathbf{n}_\phi(v) := \mathbf{n}(\tilde{v})$, and $\mathbf{sq}_\phi(v) := \mathbf{s}_\phi(v) + \mathbf{q}_\phi(v) + \mathbf{q}(v) + \mathbf{n}_\phi(v)$.

$\mathrm{SQ}_1(v)$ is the same as $\mathrm{SQ}(v)$, if we take $\tilde{v} = v$. Note that our algorithms need to know $\beta$ (even if $\|v\|$ is known), as $\beta$ cannot be deduced from a small number of queries, samples, or probability computations. So, we will be choosing $\tilde{v}$ (and, correspondingly, $\phi$) such that $\|\tilde{v}\|^2$ remains computable, even if potentially some $c\tilde{v}$ satisfies all our other requirements for some $c < 1$ (giving a smaller value of $\phi$). Finally, note that oversampling access implies an approximate version of the usual sampling access:

**Lemma 2.8.** *Suppose we are given $\mathrm{SQ}_\phi(v)$ and some $\delta \in (0,1]$. Denote $\widetilde{\mathbf{sq}}(v) := \phi\mathbf{sq}_\phi(v)\log\frac{1}{\delta}$. We can sample from $\mathcal{D}_v$ with probability $\geq 1-\delta$ in $O(\widetilde{\mathbf{sq}}(v))$ time. We can also estimate $\|v\|$ to $\nu$ multiplicative error for $\nu \in (0,1]$ with probability $\geq 1-\delta$ in $O\left(\frac{1}{\nu^2}\widetilde{\mathbf{sq}}(v)\right)$ time.*

We will generally compare our algorithms, which output $\mathrm{SQ}_\phi(v)$, to a quantum algorithm that can output (and measure) $|v\rangle$. So, $\widetilde{\mathbf{sq}}(v)$ is the relevant complexity measure that we will analyze and bound: if we wish to mimic samples from the output of the quantum algorithm we dequantize, we will pay a one-time cost to run our quantum-inspired algorithm, and then pay $\widetilde{\mathbf{sq}}(v)$ cost per additional measurements.

**Lemma 2.9** (Linear combinations, Proposition 4.3 of [45])**.** *Given $\mathrm{SQ}(v_1), \ldots, \mathrm{SQ}(v_k) \in \mathbb{C}^n$ and $\lambda_1, \ldots, \lambda_k \in \mathbb{C}$, we have $\mathrm{SQ}_\phi(\sum \lambda_i v_i)$ for $\phi = k\frac{\sum\|\lambda_i v_i\|^2}{\|\sum\lambda_i v_i\|^2}$ and $\mathbf{sq}_\phi(\sum\lambda_i v_i) := \max_{i\in[k]}\mathbf{s}(v_i) + \sum_{i=1}^k \mathbf{q}(v_i)$ (after paying $O\left(\sum_{i=1}^k \mathbf{n}(v_i)\right)$ one-time pre-processing cost to query for norms).*

So, our general goal will be to express our output vector as a linear combination of a small number of input vectors that we have sampling and query access to. Then, we can get an approximate SQ access to our output using Lemma 2.8, where we pay an additional "cancellation constant" factor of $k\frac{\sum\|\lambda_i v_i\|^2}{\|\sum\lambda_i v_i\|^2}$. We introduce some notation to this: for $V \in \mathbb{C}^{n\times k}$ the matrix whose columns are $v_i$'s and $x \in \mathbb{C}^k$ the vector whose entries are $\lambda_i$'s,

$$\mathrm{C}_{V,x} := \frac{\sum_i |x(i)|^2\|V(\cdot,i)\|^2}{\|\sum_i x(i)V(\cdot,i)\|^2} \leq \frac{\|x\|^2\|V\|_{\mathrm{F}}^2}{\|Vx\|^2}.$$

As we can see, $k\mathrm{C}_{V,x}$ is only large when $Vx$ has significantly smaller norm than the components $v_i$ in the sum suggest. Usually, in our applications, we can intuitively think about this overhead being small when the desired output vector mostly lies in a subspace spanned by singular vectors with large singular values in our low-rank input. Quantum algorithms also have similar overheads. For example, the quantum recommendation systems algorithm [29] incurs such a cost factor when performing a swap test to project the input vector on the subspace spanned by the top singular vectors of the input matrix. Assuming this cancellation is not too large, other subroutines dominate the runtime in our applications.

We also define oversampling and query access for a matrix. Though the oversampling approximation is unusual, this model is also discussed in prior work [15, 20] and is the right notion for the sampling procedures we will use.

**Definition 2.10** (Oversampling and query access to a matrix)**.** For a matrix $A \in \mathbb{C}^{m\times n}$, we have $\mathrm{SQ}(A)$ if we have $\mathrm{SQ}(A(i,\cdot))$ for all $i \in [m]$ and $\mathrm{SQ}(a)$ for $a \in \mathbb{R}^m$ the vector of row norms $(a(i) := \|A(i,\cdot)\|)$.

We have $\mathrm{SQ}_\phi(A)$ if we have $\mathrm{Q}(A)$ and $\mathrm{SQ}(\tilde{A})$ for $\tilde{A} \in \mathbb{C}^{m\times n}$ satisfying $\beta := \|\tilde{A}\|_{\mathrm{F}}^2 = \phi\|A\|_{\mathrm{F}}^2$ and $|\tilde{A}(i,j)|^2 \geq |A(i,j)|^2$ for all $(i,j) \in [m]\times[n]$. Let $p$ and $p_i$ denote the distributions on $\tilde{a}$ and $\tilde{A}(i,\cdot)$, respectively. The (known upper bounds on the) complexity of (over)sampling and querying from the matrix $A$ is denoted by $\mathbf{s}_\phi(A) := \max(\mathbf{s}(\tilde{A}(i,\cdot)), \mathbf{s}(\tilde{a}))$, $\mathbf{q}_\phi(A) := \max(\mathbf{q}(\tilde{A}(i,\cdot)), \mathbf{q}(\tilde{a}))$, $\mathbf{q}(A) := \max(\mathbf{q}(A(i,\cdot)))$, and $\mathbf{n}_\phi(A) := \mathbf{n}(\tilde{a})$ respectively. We also use the notation $\mathbf{sq}_\phi(A) := \max(\mathbf{s}_\phi(A), \mathbf{q}_\phi(A), \mathbf{q}(A), \mathbf{n}_\phi(A))$ and $\mathbf{sq}_\phi(A^{(\dagger)}) := \mathbf{sq}_\phi(A) + \mathbf{sq}_\phi(A^\dagger)$ sometimes omitting the subscripts if $\phi = 1$.

Observe that $\mathrm{SQ}_\phi(A)$ implies $\mathrm{SQ}_\phi(\mathrm{vec}(A))$: we can take $\widetilde{\mathrm{vec}(A)} = \mathrm{vec}(\tilde{A})$, and the distribution for $\mathrm{vec}(\tilde{A})$ is sampled by sampling $i$ from $\mathcal{D}_{\tilde{a}}$, and then sampling $j$ from $\mathcal{D}_{\tilde{A}(i,\cdot)}$. This gives the output $(i,j)$ with probability $|\tilde{A}(i,j)|^2/\|\tilde{A}\|_{\mathrm{F}}^2$. Therefore, $\mathrm{SQ}_\phi(A)$ can be thought of as $\mathrm{SQ}_\phi(\mathrm{vec}(A))$, with the additional guarantees that we can compute marginals $\sum_{j=1}^n \mathcal{D}_{\mathrm{vec}(\tilde{A})}(i,j)$ and can sample from the resulting conditional distributions $\mathcal{D}_{\mathrm{vec}(\tilde{A})}(i,j)/\sum_{j=1}^n \mathcal{D}_{\mathrm{vec}(\tilde{A})}(i,j)$.

**Lemma 2.11.** *Given vectors $u \in \mathbb{C}^m, v \in \mathbb{C}^n$ with $\mathrm{SQ}_{\varphi_u}(u), \mathrm{SQ}_{\varphi_v}(v)$ access we have $\mathrm{SQ}_\phi(A)$ for their outer product $A := uv^\dagger$ with $\phi = \varphi_u\varphi_v$ and $\mathbf{s}_\phi(A) = \mathbf{s}_{\varphi_u}(u) + \mathbf{s}_{\varphi_v}(v)$, $\mathbf{q}_\phi(A) = \mathbf{q}_{\varphi_u}(u) + \mathbf{q}_{\varphi_v}(v)$, $\mathbf{q}(A) = \mathbf{q}(u) + \mathbf{q}(v)$, and $\mathbf{n}_\phi(A) = \mathbf{n}_{\varphi_u}(u) + \mathbf{n}_{\varphi_v}(v)$,*

The above shows that Definition 2.10 is a faithful generalization of Definition 2.7, i.e., for a vector $v$ we get back essentially the same

[10] Oversampling turns out to be the "natural" form of approximation in this setting; other forms of error do not propagate through quantum-inspired algorithms well, and the ones that do can usually be translated into oversampling of a desired distribution.

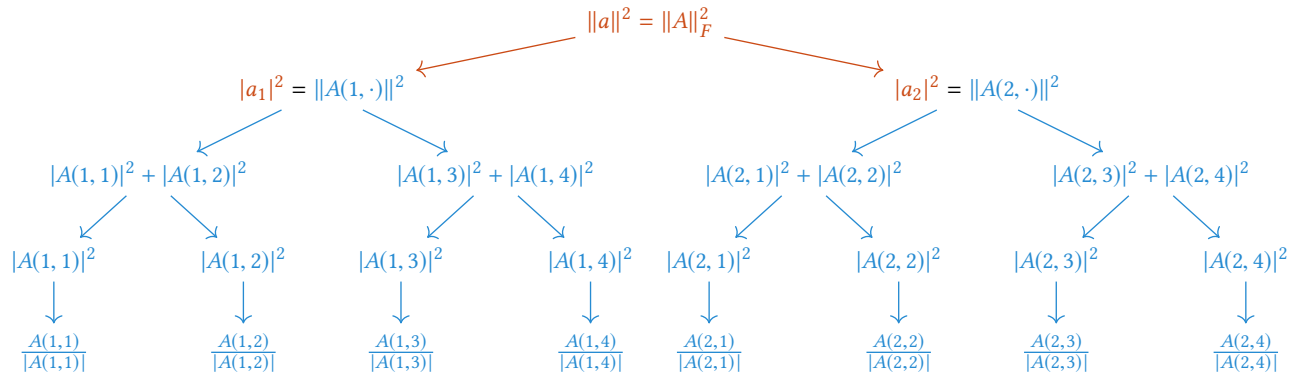


**Figure 1: Dynamic data structure for $A \in \mathbb{C}^{2\times 4}$. We compose the data structure for $a$ with the data structure for $A$'s rows.**

definition if we think about it as a row / column matrix. Using the same ideas as in Lemma 2.9, we can extend sampling and query access of input matrices to linear combinations of those matrices.

**Lemma 2.12.** *Given* $\mathrm{SQ}_{\varphi^{(1)}}(A^{(1)}), \ldots, \mathrm{SQ}_{\varphi^{(\tau)}}(A^{(\tau)}) \in \mathbb{C}^{m\times n}$, *we have* $\mathrm{SQ}_\phi(A)$ *for* $A := \sum_{t=1}^{\tau} \lambda_t A^{(t)}$ *with* $\phi = \frac{\tau \sum_{t=1}^{\tau} \varphi^{(t)} \|\lambda_t A^{(t)}\|_F^2}{\|A\|_F^2}$ *and* $\mathbf{s}_\phi(A) = \max_{t\in[\tau]} \mathbf{s}_{\varphi^{(t)}}(A^{(t)}) + \mathbf{q}_\phi(A)$, $\mathbf{q}_\phi(A) = \sum_{t=1}^{\tau} \mathbf{q}_{\varphi^{(t)}}(A^{(t)})$, $\mathbf{q}(A) = \sum_{t=1}^{\tau} \mathbf{q}(A^{(t)})$, *and* $\mathbf{n}_\phi(A) = 1$ *(after paying* $\sum_{t=1}^{\tau} \mathbf{n}_{\varphi^{(t)}}(A^{(t)})$ *one-time pre-processing cost).*

Quantum machine learning algorithms and their corresponding quantum-inspired algorithms have the potential to achieve exponential speedups when their state preparation procedures run in time polylog$(n)$. So, the most interesting regime for us is when our sampling and query oracles take polylogarithmic time. This assumption can be satisfied in various ways.

**Remark 2.13.** Below, we list various settings where we have sampling and query access to input matrices and vectors, and whenever relevant, we compare the resulting runtimes to the time to prepare analogous quantum states. Note that because we do not analyze classical algorithms in the bit model, their runtimes may be missing log factors that should be counted for a fair comparison between classical and quantum.

*Data structure.* Given $v \in \mathbb{C}^n$ in the standard RAM model, the alias method [47] takes $\Theta(n)$ pre-processing time to output a data structure that uses $\Theta(n)$ space and can sample from $v$ in $\Theta(1)$ time. In other words, we can get $\mathrm{SQ}(v)$ with $\mathbf{sq}(v) = \Theta(1)$ in $O(n)$ time, and by extension, for a matrix $A \in \mathbb{C}^{m\times n}$, $\mathrm{SQ}(A)$ with $\mathbf{sq}(A) = \Theta(1)$ in $O(mn)$ time.

More precisely, the pre-processing time is linear in the number of non-zero entries of the input vector/matrix (which we denote $\mathrm{nnz}(v)$/$\mathrm{nnz}(A)$). A direct consequence of this observation is that the quantum-inspired setting is more restrictive than the typical randomized numerical linear algebra algorithm setting. With this data structure, a fast quantum-inspired algorithm (say, one running in time $O(T\mathbf{sq}(A))$ for $T$ independent of input size) implies an algorithm in the standard computational model (running in $O(\mathrm{nnz}(A) + T)$ time).

*Dynamic data structure.* QML algorithms often assume that their input is in a QRAM data structure [6, 23, 30, 37, 40, 49], arguing that, with the right type of quantum access, this data structure allows for circuits preparing input states with linear gate count but polylog depth. Hardware might be able to parallelize these circuits enough so that they run in polylog time. In the interest of considering the best of all possible worlds for QML, we will treat circuit depth as runtime for QRAM and ignore technicalities.

This data structure (see Fig. 1) admits sampling and query access to the data it stores with just-as-good runtimes: specifically, for a matrix $A \in \mathbb{C}^{m\times n}$, we get $\mathrm{SQ}(A)$ with $\mathbf{q}(A) = O(1)$ and $\mathbf{s}(A) = O(\log mn)$. So, quantum-inspired algorithms can be used whenever QML algorithms assume this form of input.

Further, unlike the alias method stated above, this data structure supports updating entries in $O(\log mn)$ time, which can be useful for applications of QML where data can accumulate over time [29].

*Integrability assumption.* For $v \in \mathbb{C}^n$, suppose we can compute entries and sums $\sum_{i\in I(b)} |v_i|^2$ in time $T$, where $I(b) \subset [n]$ is the set of indices whose binary representation begins with the bitstring $b$. Then we have $\mathrm{SQ}(v)$ where $\mathbf{q}(v) = O(T)$, $\mathbf{s}(v) = O(T \log n)$, and $\mathbf{n}(v) = O(T)$. Analogously, a quantum state corresponding to $v$ can be prepared in time $O(T \log n)$ via Grover-Rudolph state preparation [24]. (One can think about the QRAM data structure as pre-computing all the necessary sums for this protocol.)

*Uniformity assumption.* Given $O(1)$-time $\mathrm{Q}(v) \in \mathbb{C}^n$ and a $\beta = \phi\|v\|^2$ such that $\max |v_i|^2 \le \beta/n$, we have $\mathrm{SQ}_\phi(v)$ with $\mathbf{sq}_\phi(v) = O(1)$, by using the all-1 vector times $\sqrt{\beta/n}$ as an upper bound. Assuming the ability to query entries of $v$ in superposition, a quantum state corresponding to $v$ can be prepared in time $O\left(\sqrt{\phi}\log n\right)$.

*Sparsity assumption.* If $A \in \mathbb{C}^{m\times n}$ has at most $s$ non-zero entries per row (with efficiently computable locations) and the matrix elements are $|A(i,j)| \le c$ (and efficiently computable), then we have $\mathrm{SQ}_\phi(A)$ for $\phi = c^2 \frac{sm}{\|A\|_F^2}$, simply by using the uniform distribution over non-zero entries for the oversampling and query oracles. For example, for $\mathrm{SQ}(\tilde{a})$ we can set $\tilde{a}(i) := c\sqrt{s}$, and for $\tilde{A}(i,\cdot)$ we use the vector with entries $c$ at the non-zeros of $A(i,\cdot)$ (potentially adding some "dummy" zero locations to have exactly $s$ non-zeroes).

If $A$ is not much smaller than we expect, $\phi$ is independent of dimension. For example, if $A$ has exactly $s$ non-zero entries per row and $|A(i,j)| \geq c'$ for non-zero entries, then $\phi \leq (c/c')^2$. This kind of sparsity assumption is used in some QML and Hamiltonian simulation problems [25].

*CT states.* In 2009, Van den Nest defined the notion of a "computationally tractable" (CT) state [46]. Using our notation, $|\psi\rangle \in \mathbb{C}^n$ is a CT state if we have $\mathrm{SQ}(\psi)$ with $\mathbf{sq}(\psi) = \mathrm{polylog}(n)$. Van den Nest's paper identifies several classes of CT states, including product states, quantum Fourier transforms of product states, matrix product states of polynomial bond dimension, stabilizer states, and states from matchgate circuits.

## 2.3 Matrix Sketches

**Definition 2.14.** For a distribution $p \in \mathbb{R}^m$, we say that a matrix $S \in \mathbb{R}^{s\times m}$ is *sampling according to p* if each row of $S$ is independently chosen to be $e_i/\sqrt{sp(i)}$ with probability $p_i$.

We call such $S$'s *importance sampling* sketches when $p$ comes from $\mathrm{SQ}(A)$ for some $A \in \mathbb{C}^{m\times n}$, and we call them $\phi$-oversampled importance sampling sketches if $p$ comes from $\mathrm{SQ}_\phi(A)$.

In the standard algorithm setting, sketching $A$ down to $SA$ with an importance sampling sketch requires reading all of $A$ to compute $\mathcal{D}_a$. If we have $\mathrm{SQ}_\phi(A)$, we can efficiently create a $\phi$-oversampling sketch $S$ by pulling samples from $p$, and $SA$ will be a (normalized) subset of rows of $A$. The core technique of our quantum-inspired algorithms is to use these kinds of sketches to approximate matrix expressions. Further, we can chain them with a simple observation.

**Lemma 2.15.** *Given $\mathrm{SQ}_\phi(A)$ and $S \in \mathbb{R}^{r\times m}$ (described in pairs $(i_1, p(i_1)), \ldots, (i_r, p(i_r))$) sampled according to $p$ with $r \geq 2\phi^2 \ln\frac{2}{\delta}$, then with probability $\geq 1-\delta$ we have $\mathrm{SQ}_{2\phi}((SA)^\dagger)$ with $\mathbf{q}((SA)^\dagger) = \mathbf{q}(A), \mathbf{s}_\phi((SA)^\dagger) = \mathbf{s}_\phi(A) + r\mathbf{q}_\phi(A), \mathbf{q}_\phi((SA)^\dagger) = r\mathbf{q}_\phi(A)$, and $\mathbf{n}_\phi((SA)^\dagger) = \mathbf{n}_\phi(A)$. If $\phi = 1$, then for all $r$, we have $\mathrm{SQ}((SA)^\dagger)$ with the runtimes specified above.*

When we refer to sketching $A$ down to $SAT$, we use the above observation for sampling $T$.

# 3 MAIN RESULTS

## 3.1 Singular Value Transformation

We begin with a fundamental observation: given sampling and query access to a matrix $A$, we can approximate the matrix product $A^\dagger B$ by a sum of rank-one outer products. This is the key lemma we use most in our applications.

**Lemma 3.1** (Approximating matrix multiplication to Frobenius norm error; corollary of [14, Theorem 1])**.** *Consider $X \in \mathbb{C}^{m\times n}$, $Y \in \mathbb{C}^{m\times p}$, and take $S \in \mathbb{R}^{s\times m}$ to be sampled according to $r := \frac{p+q}{2}$, where $p, q \in \mathbb{R}^m$ are $\phi_1, \phi_2$-oversampled importance sampling distributions from $X, Y$ respectively. Then,*

$$\Pr\left[\|X^\dagger S^\dagger SY - X^\dagger Y\|_{\mathrm{F}} < \sqrt{\frac{8\phi_1\phi_2\log 2/\delta}{s}}\|X\|_{\mathrm{F}}\|Y\|_{\mathrm{F}}\right] > 1-\delta.$$

*Moreover, $\|SX\|_{\mathrm{F}}^2 \leq 2\phi_1\|X\|_{\mathrm{F}}^2$ and $\|SY\|_{\mathrm{F}}^2 \leq 2\phi_2\|Y\|_{\mathrm{F}}^2$.*

We make a couple remarks. First, the bounds on $\|SX\|_{\mathrm{F}}^2$ can be improved to something like $\|X\|_{\mathrm{F}}^2$ for a sufficiently large sketch, but we will not need such bounds. Second, if $X = Y$, we can get an improved spectral norm bound: instead of depending on $\|X\|_F^2$, error depends on $\|X\|\|X\|_F$.

Theorem 3.2 (Even singular value transformation). *Let $A \in \mathbb{C}^{m\times n}$ and $f : \mathbb{R}^+ \to \mathbb{C}$ be such that, $f$ and $\bar{f}(x) := (f(x) - f(0))/x$ are $L$-Lipschitz and $L'$-Lipschitz, respectively, on $\cup_{i=1}^n [\sigma_i^2 - d, \sigma_i^2 + d]$ for some $d > 0$. Take parameters $\varepsilon$ and $\delta$ such that $0 < \varepsilon \lesssim L\|A\|_*^2$, $\delta \in (0,1]$, and $d > \bar{\varepsilon} := \|A\|\|A\|_{\mathrm{F}}\left(\frac{\phi^2\log(1/\delta)}{\min(r,c)}\right)^{1/2}$. Choose a norm $* \in \{\mathrm{F}, \mathrm{Op}\}$.*

*Given $\mathrm{SQ}_\phi(A)$, consider the sketch $S \in \mathbb{R}^{r\times m}$ sampled from $p$ and the sketch $T^\dagger \in \mathbb{R}^{c\times n}$ sampled from the distribution for $\mathrm{SQ}_{2\phi}((SA)^\dagger)$ (given by Lemma 2.15), where $r = \tilde{\Omega}\left(\phi^2L^2\|A\|_*^2\|A\|_{\mathrm{F}}^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ and $c = \tilde{\Omega}\left(\phi^2L'^2\|A\|^4\|A\|_*^2\|A\|_{\mathrm{F}}^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$. Then, for $R := SA$ and $C := SAT$, we can achieve the bound*

$$\Pr\left[\|R^\dagger\bar{f}(CC^\dagger)R + f(0)I - f(A^\dagger A)\|_* > \varepsilon\right] < \delta. \tag{3}$$

*Finding the sketches takes time $O\left((r+c)\mathbf{sq}_\phi(A)\right)$.*

We remark that no additional log terms are necessary (i.e., $\tilde{\Omega}$ becomes $\Omega$) when Frobenius norm is used. Later we will need some bounds on the norms of the matrices in our decomposition. The following lemma gives the bounds we need for our applications.

**Lemma 3.3** (Norm bounds for even singular value transformation)**.** *Suppose the assumptions from Theorem 3.2 hold and the event in Eq. (3) occurs (that is, $R^\dagger\bar{f}(CC^\dagger)R \approx f(A^\dagger A) - f(0)I$). Then we can additionally assume that the following bounds also hold:*

$$\|R\| = O(\|A\|) \quad \text{and} \quad \|R\|_{\mathrm{F}} = O(\|A\|_{\mathrm{F}}), \tag{4}$$

$$\|\bar{f}(CC^\dagger)\| \leq \max\left\{|\bar{f}(x)| \,\middle|\, x \in \bigcup_{i=1}^{\min(r,c)}[\sigma_i^2 - \bar{\varepsilon}, \sigma_i^2 + \bar{\varepsilon}]\right\}, \tag{5}$$

$$\text{when } * = \mathrm{Op}, \left\|R^\dagger\sqrt{\bar{f}(CC^\dagger)}\right\| \leq \sqrt{\|f(A^\dagger A) - f(0)I\| + \varepsilon}. \tag{6}$$

While we will primarily use the simple and fast primitive of even singular value transformation to recover "dequantized QML"-type results, we can also get generic singular value transformation and eigenvalue transformation results by bootstrapping Theorem 3.2.

Theorem 3.4 (Generic singular value transformation). *Let $A \in \mathbb{C}^{m\times n}$ be given with both $\mathrm{SQ}_\phi(A)$ and $\mathrm{SQ}_\phi(A^\dagger)$ and let $f : \mathbb{R} \to \mathbb{C}$ be a function such that $f(0) = 0$, $g(x) := f(\sqrt{x})/\sqrt{x}$ is $L$-Lipschitz, and $\bar{g}(x) := g(x)/x$ is $L'$-Lipschitz. Then, for $0 < \varepsilon \lesssim L\|A\|^3$, we can output sketches $R := SA \in \mathbb{C}^{r\times n}$ and $C := AT \in \mathbb{C}^{m\times c}$, along with $M \in \mathbb{C}^{r\times c}$, with $r = \tilde{\Omega}\left(\phi^2L^2\|A\|^2\|A\|_{\mathrm{F}}^4\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ and $c = \tilde{\Omega}\left(\phi^2L^2\|A\|^4\|A\|_{\mathrm{F}}^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$, such that*

$$\Pr\left[\|CMR + g(0)A - f^{(\mathrm{SV})}(A)\| > \varepsilon\right] < \delta.$$

*Finding $S, M$, and $T$ takes time*

$$\widetilde{O}\left((L^4L'^2\|A\|^{16}\|A\|_{\mathrm{F}}^6 + L^6\|A\|^{10}\|A\|_{\mathrm{F}}^8)\frac{\phi^6}{\varepsilon^6}\log^3\left(\frac{1}{\delta}\right)\mathbf{sq}_\phi\left(A^{(\dagger)}\right)\right).$$

Theorem 3.5 (Eigenvalue transformation). *Suppose we are given a Hermitian* $\mathrm{SQ}_\phi(A) \in \mathbb{C}^{n\times n}$*, a function* $f : \mathbb{R} \to \mathbb{C}$ *that is L-Lipschitz on* $\cup_{i=1}^n [\lambda_i - d, \lambda_i + d]$ *for some* $d > \frac{\varepsilon}{L}$*, and some*[11] $\varepsilon \lesssim L\|A\| \frac{\|A\|}{\|A\|_F}$*. Then we can output* $S \in \mathbb{C}^{s\times n}$*,* $N \in \mathbb{C}^{s'\times s}$*, and* $D \in \mathbb{C}^{s'\times s'}$ *with* $s = \widetilde{O}\left(\phi^2\|A\|^4\|A\|_{\mathrm{F}}^2 \frac{L^6}{\varepsilon^6} \log\frac{1}{\delta}\right)$ *and* $s' = O\left(\|A\|_{\mathrm{F}}^2 L^2/\varepsilon^2\right)$*, such that*

$$\Pr\left[\|(SA)^\dagger N^\dagger D N (SA) + f(0)I - f^{(\mathrm{EV})}(A)\| > \varepsilon\right] < \delta,$$

*in time* $\widetilde{O}\left(L^{22}\varepsilon^{-22}\|A\|^{16}\|A\|_{\mathrm{F}}^6 \left(\log^3\frac{1}{\delta}\right)\mathbf{sq}_\phi(A)\right)$*. Moreover, this decomposition satisfies the following further properties. First, NSA is an approximate isometry:* $\|(NSA)(NSA)^\dagger - I\| \le (\frac{\varepsilon}{L\|A\|})^3$*. Second, D is a diagonal matrix and its diagonal entries satisfy* $|D(i,i) + f(0) - f(\lambda_i)| \le \varepsilon$ *for all* $i \in [s']$ *(when eigenvalues* $\lambda_i$ *are appropriately ordered).*

## 3.2 Dequantizing QSVT

We can use the above results to dequantize the quantum singular value transformation described by Gilyén et al. [22] in the case of close-to-low-rank input.

**Definition 3.6.** For a matrix $A \in \mathbb{C}^{m\times n}$ and $p(x) \in \mathbb{C}[x]$ degree-$d$ polynomial of parity-$d$ (i.e., even if $d$ is even and odd if $d$ is odd), we define the notation $p^{(\mathrm{QV})}(A)$ in the following way:

(1) If $p$ is *even*, meaning that we can express $p(x) = q(x^2)$ for some polynomial $q(x)$, then
$$p^{(\mathrm{QV})}(A) := q(A^\dagger A) = p(\sqrt{A^\dagger A}).$$
(2) If $p$ is *odd*, meaning that we can express $p(x) = x \cdot q(x^2)$ for some polynomial $q(x)$, then
$$p^{(\mathrm{QV})}(A) := A \cdot q(A^\dagger A).$$

Theorem 3.7. *Suppose we are given a matrix* $A \in \mathbb{C}^{m\times n}$ *satisfying* $\|A\|_{\mathrm{F}} = 1$ *via the oracles for* $\mathrm{SQ}(A)$ *with* $\mathbf{sq}(A) = O(\log(mn))$*, a vector* $\mathrm{SQ}(b) \in \mathbb{C}^n$ *with* $\|b\| = 1$ *and* $\mathbf{sq}(b) = O(\log n)$*, and a degree-d polynomial* $p(x)$ *of parity-d such that* $|p(x)| \le 1$ *for all* $x \in [-1, 1]$*.*

*Then with probability* $\ge 1 - \delta$*, for* $\varepsilon$ *a sufficiently small constant, we can get* $\mathrm{SQ}_\phi(v) \in \mathbb{C}^n$ *such that* $\|v - p^{(\mathrm{QV})}(A)b\| \le \varepsilon\|p^{(\mathrm{QV})}(A)b\|$ *in* $\mathrm{poly}\left(d, \frac{1}{\|p^{(\mathrm{QV})}(A)b\|}, \frac{1}{\varepsilon}, \frac{1}{\delta}, \log mn\right)$ *time (with* $\widetilde{\mathbf{sq}}(v)$ *also having similar runtime bound).*

From this result it follows that QSVT, as described in [22, Theorem 17], has no exponential speedup when the block-encoding of $A$ comes from a quantum-accessible "QRAM" data structure as in [22, Lemma 50]. In the setting of QSVT, given $A$ and $b$ in QRAM, one can prepare $|b\rangle$ and construct a block-encoding for $A/\|A\|_{\mathrm{F}} = A$ in $\mathrm{polylog}(mn)$ time. Then one can apply (quantum) SVT by a degree-$d$ polynomial on $A$ and apply the resulting map to $|b\rangle$ with $d \cdot \mathrm{polylog}(mn)$ gates and finally project down to get the state $|p^{(\mathrm{QV})}(A)b\rangle$ with probability $\ge 1-\delta$ after $\Theta\left(\frac{1}{\|p^{(\mathrm{QV})}(A)b\|}\log\frac{1}{\delta}\right)$ iterations of the circuit. So, getting a sample from $|p^{(\mathrm{QV})}(A)b\rangle$ takes $\Theta\left(d\frac{1}{\|p^{(\mathrm{QV})}(A)b\|}\,\mathrm{polylog}(mn/\delta)\right)$ time. This circuit gives an exact outcome, possibly with some $\log(1/\varepsilon)$ factors representing the discretization error in truncating real numbers to finite precision (which we ignore, since we do not account for them in our classical algorithm runtimes).

Analogously, by Remark 2.13, having $A$ and $b$ in (Q)RAM implies having $\mathrm{SQ}(A)$ and $\mathrm{SQ}(b)$ with $\mathbf{sq}(A) = O(\log mn)$ and $\mathbf{sq}(b) = O(\log n)$. Since QSVT also needs to assume $\max_{x\in[-1,1]}|p(x)| \le 1$, the classical procedure matches the assumptions for QSVT. Our algorithm runs only polynomially slower than the quantum algorithm, since the quantum runtime clearly depends on $d$, $\frac{1}{\|p^{(\mathrm{QV})}(A)b\|}$, and $\log(mn)$. We are exponentially slower in $\varepsilon$ and $\delta$ (these errors are conflated for the quantum algorithm). However, this exponential advantage vanishes if the desired output isn't a quantum state but some fixed value (or an estimate of one), since then the quantum algorithm must also pay $\frac{1}{\varepsilon}$ during the sampling or tomography procedures (meanwhile the success probability 1-$\delta$ can be typically exponentially boosted on the classical side). Note that, unlike in the quantum output, we can query entries of the output, which a quantum algorithm cannot do without paying at least a $\frac{1}{\varepsilon}$ factor.

Theorem 3.7 also dequantizes QSVT for block-encodings of density operators when the density operator comes from some well-structured classical data. Indeed, [22, Lemma 45] assumes we can efficiently prepare a purification of the density operator $\rho$. The rough classical analogue is the assumption that we have sampling and query access to some $A \in \mathbb{C}^{m\times n}$ with $\rho = A^\dagger A$. Since $\mathrm{Tr}(\rho) = 1$, we have $\|A\|_{\mathrm{F}} = 1$. Then, $p^{(\mathrm{QV})}(\rho) = r^{(\mathrm{QV})}(A)$ for $r(x) = p(x^2)$ and $\|\rho\| = \|A\|^2$, so we can repeat the above argument to show the lack of exponential speedup for this input model too.

We can mimic the quantum algorithm with our techniques because low-degree polynomials are smooth. For example, a degree-$d$ polynomial bounded on $[-1, 1]$ is $d^2$-Lipschitz, by Markov's inequality. We use inequalities of this type to prove the statement.

Technically, QSVT can use $A^\dagger$ in QRAM instead of $A$ (cf. [22, Lemma 50]). This does not result in a discrepancy, because in the full version we describe a method to get $\mathrm{SQ}(B)$ and $\mathrm{SQ}(B^\dagger)$ for a matrix $B$ satisfying $\|B - A\| \le \varepsilon$, given only $\mathrm{SQ}(A)$.

[11] The correct way to think about $\varepsilon$ is as some constant fraction of $L\|A\|$. If $\varepsilon > L\|A\|$ then $f(0)I$ is a satisfactory approximation. The bound we give says that we want an at least $\|A\|_{\mathrm{F}}/\|A\|$ improvement over trivial, which is modest in the close-to-low-rank regime that we care about. Similar assumptions appear in applications.

## ACKNOWLEDGMENTS

ET thanks Craig Gidney for the reference to alias sampling. AG is grateful to Saeed Mehraban for insightful suggestions about proving perturbation bounds on partition functions. Part of this work was done while visiting the Simons Institute for the Theory of Computing. We gratefully acknowledge the Institute's hospitality.

NHC, HHL, and CW were supported by Scott Aaronson's Vannevar Bush Faculty Fellowship from the U.S. Department of Defense. AG acknowledges funding provided by Samsung Electronics Co., Ltd., for the project "The Computational Power of Sampling on Quantum Computers"; additional support was provided by the Institute for Quantum Information and Matter, an NSF Physics Frontiers Center (NSF Grant PHY-1733907). TL was supported by IBM PhD Fellowship, QISE-NET Triplet Award (NSF DMR-1747426), and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams program. ET was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1762114.

# REFERENCES


[1] Scott Aaronson. 2015. Read the fine print. *Nature Physics* 11, 4 (2015), 291.

[2] Joran van Apeldoorn and András Gilyén. 2019. Improvements in quantum SDP-solving with applications. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 99:1–99:15. arXiv: 1804.05058

[3] Sanjeev Arora and Satyen Kale. 2016. A Combinatorial, Primal-Dual Approach to Semidefinite Programs. *Journal of the ACM* 63, 2 (2016), 12:1–12:35. Earlier version in STOC'07.

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

[5] Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. 2019. Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. 27:1–27:14. arXiv: 1710.02581

[6] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. 2019. 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)*. 33:1–33:14. arXiv: 1804.01973

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

[8] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. 2019. Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches. (2019). arXiv: 1901.03254

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

[10] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. 2018. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474, 2209 (Jan. 2018), 20170551.

[11] Iris Cong and Luming Duan. 2016. Quantum discriminant analysis for dimensionality reduction and classification. *New Journal of Physics* 18, 7 (jul 2016), 073011. arXiv: 1510.00113

[12] Yogesh Dahiya, Dimitris Konomis, and David P Woodruff. 2018. An empirical evaluation of sketching for numerical linear algebra. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1292–1300.

[13] Chen Ding, Tian-Yi Bao, and He-Liang Huang. 2019. Quantum-Inspired Support Vector Machine. arXiv: 1906.08902

[14] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. 2006. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM J. Comput.* 36, 1 (2006), 132–157.

[15] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. 2002. Competitive recommendation systems. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC)*. 82–90.

[16] Petros Drineas and Michael W. Mahoney. 2007. A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear Algebra Appl.* 420, 2-3 (2007), 553–571.

[17] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. 2008. Relative-error CUR matrix decompositions. *SIAM J. Matrix Anal. Appl.* 30, 2 (Jan. 2008), 844–881.

[18] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. 2019. A quantum-inspired algorithm for general minimum conical hull problems. arXiv: 1907.06814

[19] Vedran Dunjko and Peter Wittek. 2020. A non-review of Quantum Machine Learning: trends and explorations. *Quantum Views* 4 (March 2020), 32.

[20] Alan Frieze, Ravi Kannan, and Santosh Vempala. 2004. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM* 51, 6 (2004), 1025–1041.

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

[22] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. 2019. 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)*. 193–204. arXiv: 1806.01838

[23] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Quantum random access memory. *Physical Review Letters* 100, 16 (2008), 160501. arXiv: 0708.1879

[24] Lov Grover and Terry Rudolph. 2002. Creating superpositions that correspond to efficiently integrable probability distributions. (2002). arXiv: quant-ph/0208112

[25] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum algorithm for linear systems of equations. *Physical Review Letters* 103, 15 (2009), 150502. arXiv: 0811.3171

[26] Elad Hazan, Tomer Koren, and Nati Srebro. 2011. Beating SGD: Learning SVMs in sublinear time. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.). 1233–1241.

[27] Dhawal Jethwani, François Le Gall, and Sanjay K. Singh. 2019. Quantum-inspired classical algorithms for singular value transformation. (2019). arXiv: 1910.05699

[28] Ravindran Kannan and Santosh Vempala. 2017. Randomized algorithms in numerical linear algebra. *Acta Numerica* 26 (2017), 95–135.

[29] Iordanis Kerenidis and Anupam Prakash. 2017. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. 49:1–49:21. arXiv: 1603.08675

[30] Iordanis Kerenidis and Anupam Prakash. 2020. Quantum gradient descent for linear systems and least squares. *Physical Review A* 101, 2 (2020), 022316. arXiv: 1704.04992

[31] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. 2016. Quantum algorithms for topological and geometric analysis of data. *Nature Communications* 7 (2016), 10138. arXiv: 1408.3106

[32] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. 2013. Quantum algorithms for supervised and unsupervised machine learning. arXiv: 1307.0411

[33] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. 2014. Quantum principal component analysis. *Nature Physics* 10 (2014), 631–633. arXiv: 1307.0401

[34] Guang Hao Low and Isaac L. Chuang. 2017. Optimal Hamiltonian simulation by quantum signal processing. *Physical Review Letters* 118, 1 (2017), 010501. arXiv: 1606.02685

[35] Michael W. Mahoney. 2011. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning* 3, 2 (2011), 123–224.

[36] Michael W. Mahoney, Mauro Maggioni, and Petros Drineas. 2008. Tensor-CUR decompositions for tensor-based data. *SIAM J. Matrix Anal. Appl.* 30, 3 (2008), 957–987.

[37] Anupam Prakash. 2014. *Quantum algorithms for linear algebra and machine learning*. Ph.D. Dissertation. UC Berkeley.

[38] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (2018), 79. arXiv: 1801.00862

[39] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. 2014. Quantum support vector machine for big data classification. *Physical Review Letters* 113, 13 (2014), 130503. arXiv: 1307.0471

[40] Patrick Rebentrost, Maria Schuld, Leonard Wossnig, Francesco Petruccione, and Seth Lloyd. 2019. Quantum gradient descent and Newton's method for constrained polynomial optimization. *New Journal of Physics* 21, 7 (2019), 073023. arXiv: 1612.01789

[41] Alessandro Rudi, Leonard Wossnig, Carlo Ciliberto, Andrea Rocchetto, Massimiliano Pontil, and Simone Severini. 2020. Approximating Hamiltonian dynamics with the Nyström method. *Quantum* 4 (2020), 234. arXiv: 1804.02484

[42] Peter W. Shor. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* 26, 5 (1997), 1484–1509. Earlier version in FOCS'94. arXiv: quant-ph/9508027

[43] Zhao Song, David Woodruff, and Huan Zhang. 2016. Sublinear time orthogonal tensor decomposition. In *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 793–801.

[44] Ewin Tang. 2018. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. (2018). arXiv: 1811.00414

[45] Ewin Tang. 2019. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*. 217–228. arXiv: 1807.04271

[46] Maarten Van den Nest. 2011. Simulating quantum computers with probabilistic methods. *Quantum Information and Computation* 11, 9&10 (2011), 784–812. arXiv: 0911.1624

[47] Michael D. Vose. 1991. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on Software Engineering* 17, 9 (1991), 972–975.

[48] David P. Woodruff. 2014. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science* 10, 1–2 (2014), 1–157.

[49] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. 2018. Quantum linear system algorithm for dense matrices. *Physical Review Letters* 120, 5 (2018), 050502. arXiv: 1704.06174

[50] Zhikuan Zhao, Jack K. Fitzsimons, and Joseph F. Fitzsimons. 2019. Quantum-assisted Gaussian process regression. *Physical Review A* 99 (May 2019), 052331. Issue 5. arXiv: 1512.03929