Quantum Machine Learning Without Any Quantum

Ewin Tang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2023

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Quantum Machine Learning Without Any Quantum

Ewin Tang

Chair of the Supervisory Committee:
James Lee
Paul G. Allen School of Computer Science & Engineering

Could quantum machine learning someday run faster than classical machine learning? Over the past decade, the field of QML has produced many proposals for attaining large quantum speedups for computationally intensive tasks in machine learning and data analysis. However, it was unclear whether these speedups could be realized in end-to-end applications, as there had been no rigorous way to analyze such speedups. We remedy this issue by presenting a framework of classical computation to serve as an analogue to quantum linear algebra: in particular, we give a classical version of the *quantum singular value transformation* (QSVT) framework of Gilyén, Su, Low, and Wiebe [GSLW19]. Within this framework, we observe that the space of QML algorithms splits into two classes: either input data is *sparse* or given in a *quantum-accessible data structure*, which implicitly requires such matrices to have low rank. The former class is BQP-complete, meaning that it must give exponential speedups; otherwise, exponential quantum speedups don't exist at all. On the other hand, the latter class can be "dequantized," meaning that our classical framework produces algorithms to perform computations in this class at most polynomially slower than QSVT.

We give two forms of evidence for this claim. First, we prove that our framework has extensibility properties, showing that we can compute the same type of matrix arithmetic expressions that QSVT can compute. Second, with our framework, we dequantize eight QML algorithms appearing in the literature, including recommendation systems [KP17] and low-rank semidefinite programming [BKLLSW19], which were previously believed to be among the best candidates for exponential quantum speedup [Pre18]. We can then conclude that these candidates do not give exponential speedups when run on classical data, radically limiting the space of settings where we could hope for exponential speedups from QML.

The classical algorithms presented here center around one key idea: data structures that support efficient *quantum* algorithms for preparing an input quantum state also admit efficient *classical* algorithms for measuring that quantum state in the computational basis. As observed in the classical sketching literature, these measurements, ℓ_2^2 *importance samples*, can be used to approximate a product of matrices by a product of "sketched" matrices of far lower dimension. This simple idea turns out to be extremely extensible when input matrices are sufficiently low-rank. Our work forms the beginning of a theory of quantum-inspired linear algebra, demonstrating that we can compute a large class of linear algebraic expressions in time independent of input dimension, provided that weak sampling assumptions on the input are satisfied.

Contents

Acknowledgments

Grad school has been rocky, to put it mildly. In many ways, it was a period of rapid growth, but with that growth came growing pains. I'd like to acknowledge some of the people who helped me leave UW stronger and happier than I entered it.

Thank you to my collaborators: James Lee, András Gilyén, Seth Lloyd, Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, Chunhao Wang, Zhao Song, Robin Kothari, Jeongwan Haah, Ryan O'Donnell, Ainesh Bakshi, Kevin Tian, Allen Liu, Sitan Chen, Jordan Cotler, Robert Huang, Jerry Li, Daogao Liu, and Ankur Moitra. Though most of the time I spent with you all was virtual, quantum computing taught me that some beautiful connections can form from interaction at a distance. I treasure our time stumbling through the unknown together.

Since this thesis covers my work on quantum-inspired algorithms, I especially want to acknowledge those who worked on this topic, for showing me that the observations I made as a fresh-faced undergrad actually had implications quite a bit beyond the recommendation system I originally used them for. A special thanks goes to Scott Aaronson, who first got me thinking about this topic. (The title of this thesis is inspired by Scott, who, when talking about my first algorithm [Tan19], quipped that it was a new near-term application of quantum computing, realizable by a quantum computer with just *zero* logical qubits!)

Thanks to Swati Padmanabhan, Alisa Liu, Alex Fang, Kentrell Owens, and Stephanie You, for giving me joy when joy was hard to find. You changed what friendship means to me. Zach Tatlock and the rest of Race Condition Running have been deeply kind and welcoming at a level I aspire to, and honestly didn't know was possible. Thank you for being my second home in the department; I feel so lucky to have met you all. As for my first home, the UW theory group has been a great community within which to learn and develop; I am proud to have been a small part of it.

Thanks to my committee—Andrea Coladangelo, Ludwig Schmidt, and Kai-Mei Fu—for reading (some of) this thesis.[1] Finally, thank you to my advisor, James Lee, for being my anchor while I explored Hilbert space.

Bibliographical note

This thesis covers my work on quantum-inspired linear algebra, which I initiated in my last year of undergrad and continued to explore over the course of my PhD. Specifically, this thesis draws primarily from the two papers [CGLLTW22; BT23], with an introduction adapted from the comment [Tan22]. This directly supersedes most of my other work on this topic [Tan19; Tan21; GLT18] and supersedes other work on quantum-inspired algorithms [CLW18; CLLW20; DBH22]. For the sake of exposition I treat the collective achievements of this line of work as a unified whole, but, as in most scientific endeavors, these ideas are the product of many researchers, only one of whom is me. Comparisons to prior work are done in Section 8. My work was performed jointly with András Gilyén, Seth Lloyd, Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, Chunhao Wang, Zhao Song, and Ainesh Bakshi.

[1] Oh, and you too, dear reader: thanks for taking a look!

1 Prelude

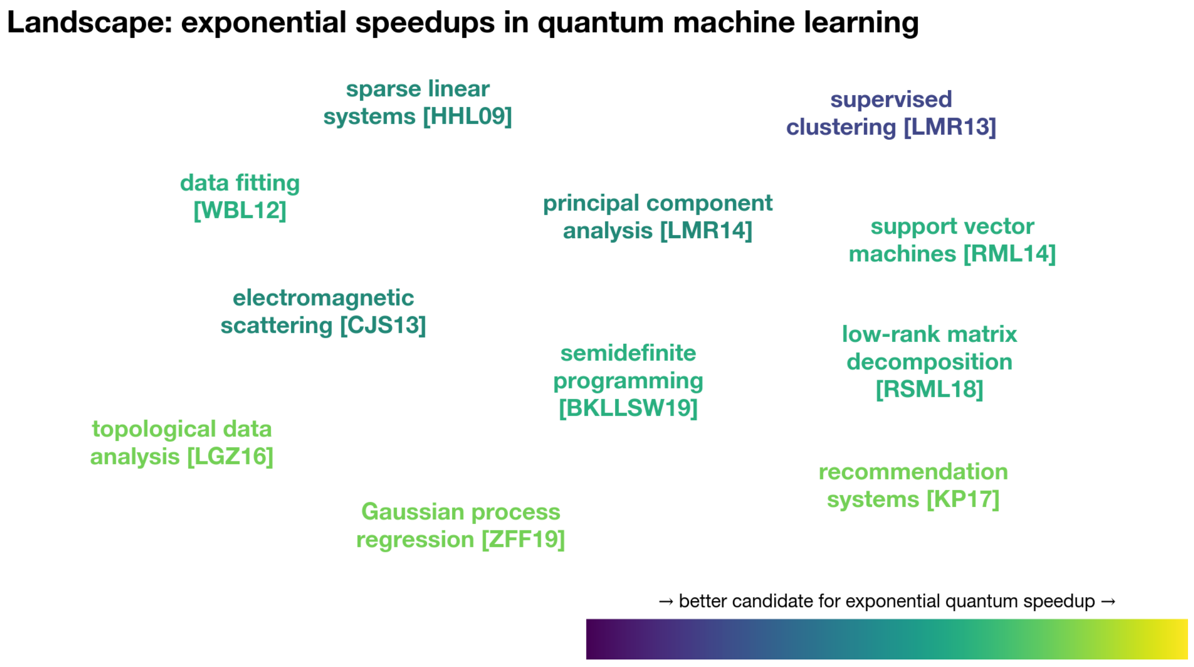
I still find it miraculous that the laws of quantum physics let us solve any classical problems exponentially faster than today's computers seem able to solve them. So maybe it shouldn't surprise us that, in machine learning like anywhere else, Nature will still make us work for those speedups. —Scott Aaronson [Aar15]

1.1 Overview

The most tantalizing goal of the quantum algorithms researcher is the *exponential speedup*: an algorithm, preferably relevant to practice, that quantum computers can perform exponentially faster than usual, classical computers can (when running the fastest known algorithms). This is the largest possible speedup a quantum computer can achieve, and algorithms that give exponential speedups are the most compelling reasons we have for the development of scalable quantum computers. Shor's algorithm [Sho97] gives an exponential speedup for factoring numbers, but despite fifty years of research, we still only have a couple of definitive examples of exponential speedup [Aar22]. However, we have hope for a third in the realm of machine learning and data analysis.

Quantum machine learning (QML) has been studied in various settings for decades [BJ99], but we will focus on QML algorithms using *quantum linear algebra*, since such algorithms are the ones aiming for exponential speedups on practical tasks [DW20; Cil+18]. The conceit of quantum linear algebra is as follows: quantum systems implicitly manipulate exponentially large matrices in polynomial time, so perhaps we can harness Nature's linear algebra processor to manipulate data exponentially faster than we can with classical computers. The key work in this line is Harrow, Hassidim, and Lloyd's quantum algorithm for sampling from the solution x to a sparse system of linear equations, $Ax = b$ [HHL09]. QML has since rapidly developed into an active field of study with numerous proposals for quantum speedups for machine learning tasks in domains ranging from recommendation systems [KP17] to topological data analysis [LGZ16].

At first glance, many applications of QML seem to admit exponential speedups. However,



Landscape: exponential speedups in quantum machine learning
sparse linear systems [HHL09]
supervised clustering [LMR13]
data fitting [WBL12]
principal component analysis [LMR14]
support vector machines [RML14]
electromagnetic scattering [CJS13]
semidefinite programming [BKLLSW19]
low-rank matrix decomposition [RSML18]
topological data analysis [LGZ16]
recommendation systems [KP17]
Gaussian process regression [ZFF19]
→ better candidate for exponential quantum speedup →

Figure 1: Pictured is a rough map of the landscape of quantum machine learning algorithms before this line of work. A collection of notable QML algorithms are listed, colored by a subjective judgment of how well they address input and output assumptions. Note that, even the best candidate presented here is not as good as, say, Shor's algorithm, since all require some form of quantum-accessible memory, which is a strong assumption on quantum hardware, particularly in the near-term.

these exponential speedups are less likely to manifest in practice compared to, say, Shor's algorithm, because unlike their classical counterparts, QML algorithms must make strong input assumptions and learn relatively little from their output [Aar15] (see Fig. 1). For example, consider the aforementioned quantum algorithm for solving $Ax = b$ when A is sparse and well-conditioned. On the input side, this algorithm requires the ability to prepare quantum states encoding b and apply quantum circuits encoding A, so it is limited to input that supports these operations efficiently. On the output side, this algorithm produces the solution x encoded in a quantum state, so we are limited to information about x that can be extracted from these states efficiently. Both input loading and output extraction tasks are hard in their most generic forms, leading to unique assumptions that make evaulating QML proposals against classical algorithms difficult. So, our understanding of speedups in this space is much murkier than it might appear at first glance. The question remains: will quantum computers someday give super-polynomial speedups for machine learning? Is there a way to rule this possibility out in some regimes?

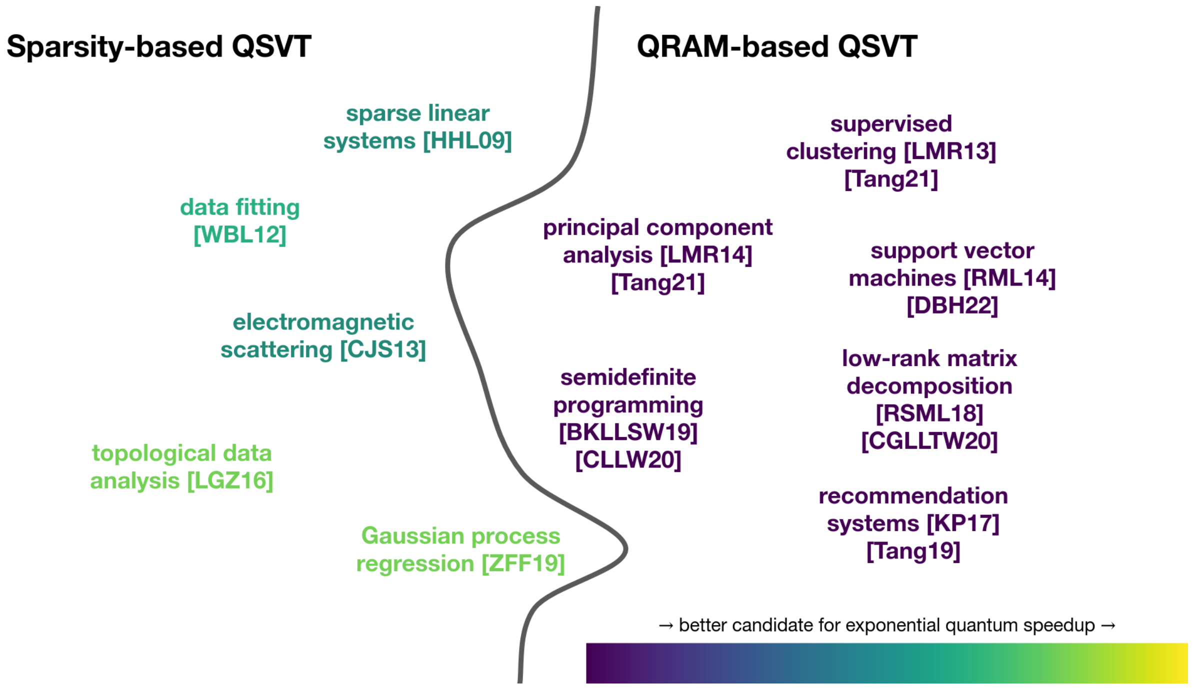

Sparsity-based QSVT
QRAM-based QSVT
sparse linear systems [HHL09]
data fitting [WBL12]
electromagnetic scattering [CJS13]
topological data analysis [LGZ16]
Gaussian process regression [ZFF19]
supervised clustering [LMR13] [Tang21]
principal component analysis [LMR14] [Tang21]
support vector machines [RML14] [DBH22]
semidefinite programming [BKLLSW19] [CLLW20]
low-rank matrix decomposition [RSML18] [CGLLTW20]
recommendation systems [KP17] [Tang19]
→ better candidate for exponential quantum speedup →

Figure 2: Pictured is the landscape of quantum machine learning algorithms from Fig. 1 after this line of work. In this thesis, we dequantize all of the algorithms on the right-hand side, showing that they do not give exponential speedups on classical data. For specifics, see Fig. 7. All of these algorithms can be placed in the QSVT framework, and in this setting, the dequantized algorithms are precisely the algorithms that do not rely on sparsity assumptions.

This thesis develops a framework of classical algorithms which can give formal evidence against an exponential quantum advantage. This framework produces "dequantized" versions of QML algorithms: fully classical algorithms that, on classical data, perform only polynomially slower than their quantum counterparts. The existence of a dequantized algorithm means that its quantum counterpart cannot give exponential speedups on classical data, illuminating the landscape of QML speedups.

Prior to this dequantizing framework, the primary formal evidence for or against quantum linear algebra speedups was that certain problems QML could solve are BQP-complete, meaning that if they cannot produce an exponential speedup, then no quantum algorithm can. BQP-completeness is positive evidence for the existence of a speedup, but has only been shown for a handful of QML problems, notably the aforementioned sparse linear systems sampling [HHL09]. Dequantized algorithms complement BQP-completeness arguments by giving negative evidence of a quantum speedup, provided that one is given input matrices and vectors as lists of entries in a data structure, referred to here as being given "classical data" (as opposed to quantum data, where matrices and vectors are given as, say, quantum states or classical descriptions of quantum circuits). We will show how to dequantize a large swathe of QML, and in fact, dequantize a general quantum linear algebra framework.

Quantum singular value transformation (QSVT), a framework introduced by Gilyén, Low, Su, and Wiebe, unifies many quantum algorithms ranging from quantum walks to Hamiltonian simulation [LC17; CGJ19; GSLW19]. Since this framework captures essentially all known linear algebraic QML techniques [MRTC21], including all prior dequantized QML algorithms (up to minor technical details), it is our natural target for dequantizing. We cannot hope to dequantize *all* of QSVT, because with sparse input data encoded appropriately, QSVT can simulate algorithms for BQP-complete problems [JW06; HHL09]. However, we show that we can dequantize the QSVT framework, provided that the input data comes in the state preparation data structure commonly used for quantum linear algebra. Such data structures only allow for efficient QML when the input is low-rank. Nevertheless, they are the only way we know how to run quantum linear algebra on classical data without strong restrictions on the structure of the data, so this setting covers all QML algorithms that do not rely on sparsity assumptions. We present a classical analogue of the QSVT framework that is only polynomially slower when the input is low rank, and apply it to dequantize QML algorithms. This framework inherits unique closure properties about QSVT, making it potentially of interest to classical sketching researchers.

The notion of dequantization is subtle, because classical algorithms are hard to compare with inherently quantum ones. To get a better sense of it, we begin with some examples of dequantization. These examples will not give formal analyses, but we encourage the interested reader to perform these themselves, as they are good warmups for subsequent arguments.[2]

Notation To begin with, we define notation to be used throughout. For natural numbers $n \in \mathbb{N}$, $[n] := \{1, \dots, n\}$. For complex numbers $z \in \mathbb{C}$, its absolute value is $|z| = \sqrt{z^* z}$, where z^* is the complex conjugate of z. $f \lesssim g$ denotes the ordering induced by big O notation, $f = \mathcal{O}(g)$

[2]Though the motivation of this work is to understand quantum speedups, quantum computing is not necessary to understand the results in this thesis: outside of the discussions in this prelude, we will only reference QML algorithms in their broad strokes. Readers wishing to understand this work from the perspective of classical algorithms can safely ignore this material.

(and respectively for $\gtrsim$ with $\Omega(\cdot)$ and $\eqsim$ with $\Theta(\cdot)$). $\widetilde{\mathcal{O}}(g)$ is shorthand for $\mathcal{O}(g \operatorname{poly}(\log g))$.

Let $A \in \mathbb{C}^{m\times n}$ be a complex matrix. For $i \in [m], j \in [n]$, $A(i,\cdot)$ denotes the i-th row of A, $A(\cdot, j)$ denotes the j-th column of A, and $A(i,j)$ denotes the (i,j)-th element of A. The conjugate transpose of A is $A^\dagger$, the *Frobenius norm* of A is $\|A\|_{\mathrm{F}} \coloneqq \|\mathrm{vec}(A)\| = (\sum_{i=1}^m \sum_{j=1}^n |A(i,j)|^2)^{1/2}$ and the *spectral norm* of A is $\|A\| \coloneqq \|A\|_{\mathrm{Op}} \coloneqq \sup_{x\in\mathbb{C}^n, \|x\|=1} \|Ax\|$. The *stable rank* of A is $\|A\|_{\mathrm{F}}^2/\|A\|^2$: when we refer to "low-rank" matrices, it means that stable rank is small.

For vectors $v \in \mathbb{C}^n$, $\|v\|$ denotes standard Euclidean norm (so $\|v\| \coloneqq (\sum_{i=1}^n |v(i)|^2)^{1/2}$). The quantum state encoding the entries of v in its amplitudes is denoted $|v\rangle = \frac{1}{\|v\|}\sum_{i=1}^n v_i |i\rangle$, where $|i\rangle$ are the computational basis vectors.

1.2 Example 1: The swap test, and access models

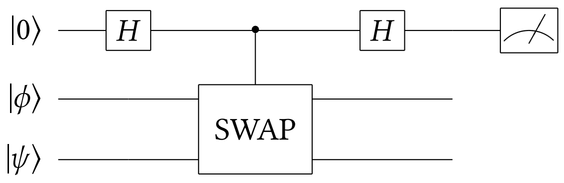


Figure 3: The quantum circuit for the swap test, taken from [BCWW01, Figure 1].

It seems counterintuitive that classical linear algebra algorithms can perform nearly as well as quantum ones, even on classical data. In some sense, what dequantization shows is that some quantum linear algebra algorithms do not fully exploit "quantumness," since they can be mimicked classically using sampling procedures. We'll investigate a simple example of a quantum linear algebra algorithm: the *swap test* [BCWW01].

Suppose we have two d-dimensional vectors $\phi, \psi \in \mathbb{C}^d$, both with unit norm. We wish to compute their overlap $|\langle\phi|\psi\rangle|^2$. There is a quantum algorithm, the swap test (shown in Fig. 3), to solve this: prepare the $\log(d)$-qubit quantum states $|\phi\rangle = \sum_{i=1}^d \phi_i |i\rangle$ and $|\psi\rangle = \sum_{i=1}^d \psi_i |i\rangle$, along with one additional qubit in the state $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Then, apply a controlled SWAP between $|\phi\rangle$ and $|\psi\rangle$, with the additional qubit as the control, and then measure this qubit in the Hadamard basis; the measurement produces 1 with probability $\frac{1}{2} - \frac{1}{2}|\langle\phi|\psi\rangle|^2$, so we can use it to estimate the overlap. Averaging over more runs of this circuit gives an estimate to 0.01 error with only $\mathcal{O}(\log(d))$ quantum gates and a constant number of copies of the input states. Even approximating overlaps using classical computers requires $\Omega(d)$ time, since we need to read this many entries of the input to distinguish the two cases $\phi = e_i$, $\psi = e_i$ and $\phi = e_i$, $\psi = e_j$. So, we might naively conclude that the swap test achieves an exponential quantum advantage in the task of "computing overlaps". This is not as farfetched a claim as it might appear: the general version of this task, where we wish to estimate $|\langle 0|^{\otimes n} U |0\rangle^{\otimes n}|$ for $U \in \mathbb{C}^{2^n \times 2^n}$ a unitary matrix encoded as a poly(n)-sized quantum circuit, indeed gives a quantum advantage (since this task is BQP-hard). Further, this idea has been proposed before in QML: a preprint of Lloyd, Mohseni, and Rebentrost claims to achieve an exponential quantum advantage for clustering with the swap test, computing the distance of a vector to a centroid by estimating the overlap of states like the above [LMR13].

However, the comparison between $\mathcal{O}(\log(d))$ and $\Omega(d)$ hides the difference in input models: the quantum algorithm requires copies of the states $|\phi\rangle$ and $|\psi\rangle$, and the classical lower bound assumes that we are only given the input vectors as lists of entries. For applications to machine learning, it's reasonable to receive the data in the latter form, since the data is classical (in that it comes from classical sources, as is the case for the vast majority of data). For example, machine learning datasets are stored in this way. This leads us to the question: given ϕ and ψ classically, how can we efficiently prepare their corresponding quantum states? Though state preparation assumptions like these are common in quantum linear algebra, they cannot be satisfied in general: the typical way of satisfying them is to assume preprocessing to load the input into a certain kind of data structure in *quantum random access memory* (QRAM) [GLM08; Pra14; JR23]. QRAM is a speculative piece of quantum hardware which supports storing n bits of data and subsequently querying that data in superposition in (functionally) polylog(n) time, similarly to how we consider classical RAM; for the sake of comparison, we assume the existence of QRAM.[3] If we assume that input is given in this data structure (see Fig. 4) for the sake of the quantum computer, then for a fair comparison, we should give our classical computer this same data structure.

[3]Of course, neither forms of RAM could be "truly" polylog(n) time, since storing n bits of data requires poly(n) space and therefore poly(n) time for the information to travel across that amount of space. The goal would be to optimize QRAM as well as classical RAM, so that accesses can be treated as $\mathcal{O}(\log(n))$ time, the cost of simply writing down the pointer into the data.

$$\begin{tikzcd}
 & & & \|v\|^2 \arrow[dll] \arrow[drr] & & & \\
 & |v(1)|^2 + |v(2)|^2 \arrow[dl] \arrow[dr] & & & & |v(3)|^2 + |v(4)|^2 \arrow[dl] \arrow[dr] & \\
|v(1)|^2 \arrow[d] & & |v(2)|^2 \arrow[d] & & |v(3)|^2 \arrow[d] & & |v(4)|^2 \arrow[d] \\
v(1) & & v(2) & & v(3) & & v(4)
\end{tikzcd}$$

Figure 4: Dynamic data structure used to perform efficient state preparation of a vector $v \in \mathbb{C}^4$. The values displayed are stored in QRAM, along with pointers to other values as designated by the entries. Observe that, by starting from the root of the tree and recursing appropriately, we can sample $i \in [4]$ with probability proportional to $|v(i)|^2$ using only *classical* access to the data structure. See Remark 4.12 part (b) for more information. A variety of data structures have similar properties, but this one has the advantage of supporting updating entries in $\mathcal{O}(\log n)$ accesses.

If ϕ in this data structure, a classical computer can draw independent samples $i \in [n]$ with probability proportional to $|\phi(i)|^2$ with $\mathcal{O}(\log(n))$ accesses. Equipped with this additional type of input access, we can estimate the overlap much faster via a Monte Carlo method: pull one sample, s, from $|\phi\rangle$, and then compute the estimator ψ_s/ϕ_s. This estimator has expected value $\langle\phi|\psi\rangle$ and variance 1, so by averaging over a constant number of runs, we can estimate of the overlap to 0.01 error using $O(\log d)$ classical gates, assuming that the entries of ϕ and ψ are specified with $O(\log d)$ bits. The swap test achieves the same dependence on dimension as the dequantized swap test, so it does not give an exponential speedup in this setting. (A more precise analysis would reveal that a quadratic quantum speedup in error is possible, from $\mathcal{O}(1/\varepsilon^2)$ to $\mathcal{O}(1/\varepsilon)$.) This argument against exponential quantum speedup remains valid provided we want to run the quantum algorithm in a setting where we could also perform the quantum-inspired algorithm.

The general principle of the dequantized swap test extends to other QML algorithms. In the typical RAM access model, we assume only that we can query entries efficiently. In other words, we receive our input $v \in \mathbb{C}^n$ as $\mathrm{Q}(v)$ with $\boldsymbol{q}(v) = 1$.

Definition 1.1 (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 query for $v(i)$. Let $\boldsymbol{q}(v)$ denote the (time) cost of such a query.

For comparison to quantum algorithms, we use a stronger input model, sampling and query access, which supports the queries we need to perform the overlap estimation algorithm.

Definition 1.2 (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 $i \in [n]$ where the probability of sampling i is $|v(i)|^2/\|v\|^2$;
3. query for $\|v\|$.

Let $\boldsymbol{sq}(v)$ denote the time cost of any query.

If we only have $\mathrm{Q}(v)$, then responding to queries from $\mathrm{SQ}(v)$ (or preparing the state $|v\rangle$) requires linear-time pre-processing. When quantum algorithms use $|v\rangle$, it's sensible to give classical algorithms access to $\mathrm{SQ}(v)$, since this is $\mathrm{Q}(v)$ with access to computational basis measurements of $|v\rangle$, also known as *importance samples from* v. In fact, as far as we know, if input data is given *classically*,[4] classical algorithms in the sampling and query model can be run whenever the corresponding algorithms in the quantum model can (Remark 4.12). For example, if input is loaded in the QRAM data structure, as commonly assumed in QML in order to satisfy state preparation assumptions [Pra14; Cil+18], then we have log-time sampling and query access to it. So, a fast algorithm for a problem in this classical model implies lack of quantum speedup for the problem, at least in the usual settings explored in the QML literature.

As the inner product estimation protocol suggests, $\mathrm{SQ}(v)$ is a much more powerful access model than $\mathrm{Q}(v)$. Classical algorithms can exploit the measurements of input data possible with sampling and query access to speed up linear algebra to become time-independent of the dimension. Specifically, sketching algorithms explore how to use randomness to perform a dimensionality reduction and "sketch" a large matrix A down to a constant-sized matrix normalized submatrix of A which behaves similarly to the full matrix [Woo14]. The computational basis measurements one can produce in the quantum-inspired input model allow for the efficient estimation of matrix products through Monte Carlo methods [DKM06], which can be applied iteratively to produce dequantized algorithms that achieve surprisingly similar

[4]This assumption is important. When input data is quantum (say, it is coming directly from a quantum system), a classical computer has little hope of performing linear algebra on it efficiently, see for example [ACQ22; HKP21].

bounds to their quantum counterparts. We explore this in our next example.

1.3 Example 2: QSVT, matrix-vector products, and approximate closure

We now introduce QSVT, our target of dequantization. QSVT is based on the primitive of computing matrix-vector products: given a matrix $A \in \mathbb{C}^{m\times n}$ and a vector $b \in \mathbb{C}^n$, compute Ab. To do this in quantum linear algebra, consider the quantum state $|b\rangle$: a unitary matrix $U \in \mathbb{C}^{n\times n}$ can be applied to b by running a quantum circuit implementing U on $|b\rangle$, giving $|Ub\rangle$ as output. QSVT extends this to general (bounded) matrices by including the non-unitary operation of measurement. Suppose we have a circuit implementing a unitary matrix $U \in \mathbb{C}^{2n\times 2n}$ where $A \in \mathbb{C}^{n\times n}$ is the top-left block of U. In other words, $(\langle 0|\langle u|)U(|0\rangle|v\rangle) = \langle u|A|v\rangle$. Then by taking the state $|0\rangle\,|b\rangle$, applying U, and measuring the first qubit, we will see $|Ab\rangle$ if the outcome is $|0\rangle$, which occurs with probability $1/\|Ab\|^2$ (see Fig. 5). This extends to the notion of an *α-block-encoding* of A, which is a circuit implementing U which has A/α as a submatrix in a known location. If we have an α-block encoding of A and copies of the state $|b\rangle$, the quantum algorithm can produce a sample from Ab in $\mathcal{O}(\frac{\alpha^2\|b\|^2}{\|Ab\|^2})$ time, ignoring $\log(mn)$ factors.[5] Roughly, we can think about an α-block encoding of A as a block-encoding of A where the quantum circuit is a factor of $\mathcal{O}(\alpha)$ larger.[6] QSVT combines this observation about *multiplying* block-encodings with one about *adding* block-encodings to prove that we can efficiently prepare the state $|p(A)b\rangle$ for any bounded low-degree polynomial p.

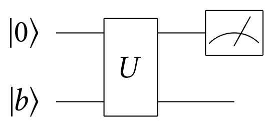


Figure 5: A basic QSVT circuit. If U is a block-encoding of the matrix $A \in \mathbb{C}^{n\times n}$, then provided the outcome of the measurement on the first wire is 0, then the output of the circuit is $|Ab\rangle$.

We dequantize the simple step of computing Ab. First, we should understand precisely when we can get efficient block-encodings of an input matrix. Efficient block-encodings do

[5]We will not concern ourselves with $\log(mn)$ and $\log\frac{1}{\varepsilon}$ factors: quantum algorithms typically count bit complexity where classical algorithms count word RAM complexity, which muddles any such comparisons.

[6]Precisely, by [GSLW19, Theorem 30], we can get a block-encoding to an ε-approximation to $A/2$ with $\mathcal{O}(\alpha\log\frac{\alpha}{\varepsilon})$ overhead.

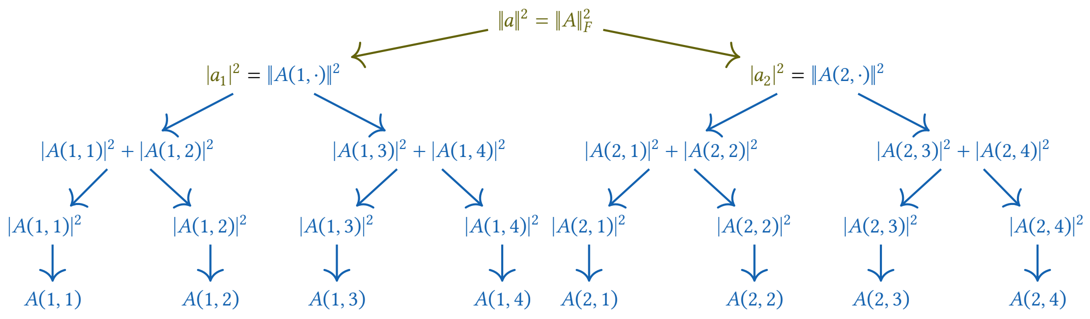


Figure 6: Dynamic data structure for a matrix $A \in \mathbb{C}^{2\times 4}$ discussed in Remark 4.12 part (b). We compose the data structure for a, the vector of row norms, with the data structure for A's rows.

not exist in general, but they do exist for two broad classes of matrices, assuming appropriately strong forms of coherent access: matrices with low sparsity (a typical setting for quantum simulation) and matrices with low stable rank (a typical setting for quantum machine learning). If A is s-sparse with elements that we can compute in superposition (i.e. from a quantum circuit or QRAM) and $\|A\| \leq 1$, then we can implement an s-block-encoding of A efficiently [GSLW19, Lemma 48]. If A is in a state preparation data structure in QRAM (like the vector case, see Fig. 6), we can implement a block-encoding of $A/\|A\|_F$ efficiently [GSLW19, Lemma 50]. This type of block-encoding is the one commonly used for quantum linear algebra algorithms on classical data [KPS21; CW23], since it works for arbitrary matrices and vectors, paying only a $\|A\|_F/\|A\|$ (square root of stable rank) factor in sub-normalization. We will use the term *QRAM-based QSVT* to refer to the family of quantum algorithms possible in the QSVT framework when all input matrices & vectors are given in the QRAM data structure.

So, assuming A and b are in appropriate data structures in QRAM, we can implement a $\|A\|_F$-block-encoding of A and prepare copies of $|b\rangle$ efficiently, so we can quantumly produce a sample from Ab in $\mathcal{O}(\frac{\|A\|_F^2\|b\|^2}{\|Ab\|^2})$ time. We can dequantize this algorithm! Classically, under identical assumptions, we can produce a sample from a v such that $\|v - Ab\| \leq \varepsilon\|Ab\|$ in $\mathcal{O}(\frac{\|A\|_F^4\|b\|^4}{\varepsilon^2\|Ab\|^4})$ time, only polynomially slower than quantum.

We note here that a dependence on error ε appears here where it does not in the quantum setting. However, this is not a realizable quantum speedup (except possibly for sampling tasks) since the output is a quantum state: estimating some statistic of the quantum state requires incurring a polynomial dependence on ε. For example, if the goal is to estimate $|\langle v|Ab\rangle|^2$, where v is a given vector, then this can be done with $1/\varepsilon^2$ invocations of a swap test (or $1/\varepsilon$ if one uses amplitude amplification). More generally, distinguishing a state from one ε-far in trace distance requires $\Omega(1/\varepsilon)$ additional overhead, even when given an oracle efficiently preparing that state, so estimating quantities to this sensitivity requires polynomial dependence on ε.

To see this, we first consider a simple case: where b is a constant-sized vector, so $n = \mathcal{O}(1)$. Then we simply wish to sample from a linear combination of columns of A, since $Ab = \sum_{t=1}^{n} b(t)A(\cdot,t)$. If A is in the QRAM data structure (i.e. storing $A^\dagger$ in Fig. 6), then this means its columns are in the vector QRAM data structures (Fig. 4), so classically we have sampling and query access to the columns of A, $\mathrm{SQ}(A(\cdot,t))$ for all $t \in [n]$. This implies we have sampling and query access to Ab, up to some overhead. We encode this overhead in the notion of *oversampling and query access*:

Definition 1.3. We have *ϕ-oversampling and query access* to a vector $v \in \mathbb{C}^n$, $\mathrm{SQ}_\phi(v)$, if:

1. we can query for entries of v, $\mathrm{Q}(v)$, and;
2. we have sampling and query access to an "entry-wise upper bound" vector $\tilde{v}$, $\mathrm{SQ}(\tilde{v})$, where $\|\tilde{v}\|^2 = \phi\|v\|^2$ and $|\tilde{v}(i)| \geq |v(i)|$ for all indices $i \in [n]$.

Let $\boldsymbol{sq}_\phi(v)$ denote the time cost of any query.

The parameter ϕ is the classical analogue of α for block-encodings. These appear in running times of algorithms because they correspond to overhead in rejection sampling and post-selection, respectively.

Oversampling and query access has *closure properties* that we can exploit here. Given access to a constant number of vectors $\mathrm{SQ}(A(\cdot,1)),\dots,\mathrm{SQ}(A(\cdot,n))$, we have access to linear combinations $\mathrm{SQ}_\phi(Ab)$ with $\phi = n\frac{\sum_{t=1}^{n}|b(t)|^2\|A(\cdot,t)\|^2}{\|Ab\|^2} \leq \frac{n\|A\|_{\mathrm{F}}^2\|b\|^2}{\|Ab\|^2}$ and $\boldsymbol{sq}_\phi(Ab) = \mathcal{O}(n)$ (Lemma 4.6; the inequality follows from Cauchy-Schwarz). Finally, from $\mathrm{SQ}_\phi(Ab)$ we can perform approximate versions of all the queries of $\mathrm{SQ}(Ab)$ with a factor ϕ of overhead (Lemma 4.5). This is possible with rejection sampling: given $\mathrm{SQ}_\phi(v)$, pull a sample i from $\tilde{v}$; accept it with probability $|v(i)|^2/|\tilde{v}(i)|^2$, and restart otherwise; the output will be a sample from v. In particular, we can sample from Ab in $\phi n = \mathcal{O}(n^2\frac{\|A\|_{\mathrm{F}}^2\|b\|^2}{\|Ab\|^2})$ time in expectation, which is good when $n = \mathcal{O}(1)$.

Now, consider when n is too large to iterate over. In this setting, we can use the *approximate matrix product* property of importance sampling to reduce the number of vectors in the linear combination. Consider pulling a sample $s \in [n]$ where we sample i with probability $p(i)$. Then $\frac{1}{p(s)} b(s)A(\cdot, s)$, a rescaled random column of A, has expectation $\sum_i b(i)A(\cdot, i) = Ab$. If the sampling distribution is chosen to be $p(i) = \frac{|b(i)|^2}{\|b\|^2}$, an importance sample from $\operatorname{SQ}(b)$, then a variance computation shows that the average of $\tau = \Theta(\frac{\|A\|_{\mathrm{F}}^2}{\varepsilon^2\|A\|^2})$ copies of this random vector is $\varepsilon\|A\|\|b\|$-close to Ab with probability ≥ 0.9 (Lemma 5.4). This average, which we denote v, is now a linear combination of only τ columns of A, each of which we have sampling and query access to. So, we can use the aforementioned closure properties to get $\operatorname{SQ}_\phi(v)$ for $\phi = \mathcal{O}(\frac{\|A\|_{\mathrm{F}}^2\|b\|^2}{\|v\|^2})$ and $\boldsymbol{sq}_\phi(v) = \mathcal{O}(\tau)$, and a sample from v in $\phi\tau = \mathcal{O}(\frac{\|A\|_{\mathrm{F}}^4\|b\|^2}{\varepsilon^2\|A\|^2\|v\|^2})$ time in expectation. Rescaling ε by a factor of $\frac{\|Ab\|}{\|A\|\|b\|}$ gives the result stated above.

In effect, what we have proven is an approximate closure property for taking products; a more careful analysis will give that, given $\operatorname{SQ}(A)$ and $\operatorname{SQ}(B)$, we have $\operatorname{SQ}_\phi(A^\dagger B)$ for a value of ϕ that is independent of input dimension. The thrust of our main results is to demonstrate that *oversampling and query access is approximately closed under arithmetic operations.* These closure properties together imply that, given input matrices and vectors in data structures in QRAM, we can get oversampling and query access to low-degree polynomials of the input via closure properties; in the same setting, QSVT gives block-encodings of low-degree polynomials of the input, through similar closure properties. The classical algorithm's runtime is only polynomially slower than the corresponding quantum algorithm (except in the ε parameter). This dequantizes QSVT.

1.4 Results

This thesis builds from and tightens up these basic ideas to give a variety of dequantization results. From these results, we argue that *QRAM-based QSVT does not give exponential speedups on classical data*, or at least that it does not under the current state of the art. Our first type of results are closure properties, which show that, like block-encodings in the QSVT framework, if we are given our form of access to input matrices and vectors, we can get access to linear algebraic expressions involving that input. We list these properties, along with the corresponding closure properties proven for block-encodings in [GSLW19]. For all of these, the query time for access to the output is just polynomial in the query times for access to the input, so in particular, these procedures run in time independent of input dimension. We will compare the subnormalization overhead between the two, which is the value ϕ in the classical setting and what α in the quantum setting. Specifically, for a matrix A in an α-block-encoding, α is an upper bound on $\|A\|$, and for the access $\mathrm{SQ}_\phi(A)$, $\sqrt{\phi}\|A\|_{\mathrm{F}}$ is an upper bound on $\|A\|_{\mathrm{F}}$; they characterize overhead because both $\alpha/\|A\|$ and $\sqrt{\phi}$ appear in the running time of their respective algorithms.

Small linear combinations Given access to a constant number of vectors $v_1, \dots, v_\tau$, we have access to linear combinations $\sum_{t=1}^{\tau} \lambda_t v_t$, and analogously with linear combinations of matrices (Lemmas 4.6 and 4.9). This is a classical analogue to the "linear combinations of unitaries" technique for block-encodings [GSLW19, Lemma 52]. With this technique, given block-encodings of $A^{(1)}/\alpha_1, \dots, A^{(\tau)}/\alpha_\tau$, we can form a block-encoding of $(\sum_{t=1}^{\tau} \lambda_t A^{(\tau)})/\alpha$ with $\alpha = \sum_{t=1}^{\tau} |\lambda_t \alpha_t|$. On the other hand, given $\mathrm{SQ}_{\varphi^{(t)}}(A^{(t)})$ for all $t \in [\tau]$, we can get $\mathrm{SQ}_\phi(\sum_{t=1}^{\tau} \lambda_t A^{(t)})$ with an upper bound of $\Phi = (\tau \sum_{t=1}^{\tau} |\lambda_t \Phi^{(t)}|^2)^{1/2}$, where $\Phi := \phi^{1/2} \|\sum_{t=1}^{\tau} \lambda_t A^{(t)}\|_{\mathrm{F}}$ and $\Phi^{(t)} := (\varphi^{(t)})^{1/2} \|A^{(t)}\|_{\mathrm{F}}$. By Cauchy-Schwarz, in the quantum setting there is less overhead, in the sense that if $\alpha_t = \Phi^{(t)}$ then $\alpha \leq \Phi$. However, $\Phi/\sqrt{\tau} \leq \alpha$ also holds, which make the quantum and classical overheads equivalent up to a factor of $\sqrt{\tau}$. Since both classical and quantum methods incur factors of τ in their running time (though LCU for special types of linear combinations need not incur this factor), these properties can be considered equivalent.

Approximate matrix products Given access to two matrices $\mathrm{SQ}_{\phi_1}(A), \mathrm{SQ}_{\phi_2}(B)$, we have access to a matrix $\mathrm{SQ}_\phi(Z)$ such that $\|Z - A^\dagger B\|_{\mathrm{F}} \leq \varepsilon \|A\|_{\mathrm{F}} \|B\|_{\mathrm{F}}$ (Lemma 5.7 and Remark 5.8). In the quantum setting, closure of block-encodings under products is almost immediate [GSLW19, Lemma 53] and is not approximate. In both cases the individual input overheads of A and B are multiplied: $\sqrt{\phi}\|Z\|_{\mathrm{F}} = (\sqrt{\phi_1}\|A\|_{\mathrm{F}})(\sqrt{\phi_2}\|B\|_{\mathrm{F}})$, and given α_1- and α_2-block-encodings of A and B, we can form a $(\alpha_1\alpha_2)$-block-encoding of $A^\dagger B$. With the same overheads one can also form Kronecker products $A \otimes B$ exactly—this is immediate both in the classical and quantum case [CB20]. In particular, given access to two vectors u and v, we have access to their outer product $uv^\dagger$ (Lemma 4.8).

Singular value transformation Given access to a matrix $\mathrm{SQ}_\varphi(A)$ and a vector $\mathrm{SQ}(b)$, we have access to a vector $\mathrm{SQ}_\phi(y)$ such that $\|y - p(A)b\| \leq \varepsilon\|b\|$, where $p(x)$ is an even or odd polynomial that is constant-degree and satisfies $|p(x)| \leq 1$ for $x \in [-\|A\|, \|A\|]$ (Theorem 6.1). QSVT provides an analogous guarantee without approximation: given A in an α-block encoding, we can produce an exact block-encoding of $p(A/\alpha)$, which we can use to convert copies of $|b\rangle$ to copies of $|p(A/\alpha)b\rangle$. There is classical overhead with the upper bound $\sqrt{\phi}\|y\| = \mathcal{O}(d\log(d)\sqrt{\varphi}\|A\|_{\mathrm{F}}\|b\|)$ (Theorems 6.22 and 6.26), but it's not obvious how to compare this to the quantum overhead, since the norm $p(A/\alpha)$ is not straightforwardly related to α. However, as seen in the applications, they lead to comparable running times.

More singular value transformation Given access to a matrix A with Frobenius norm at most one and a Lipschitz function f, we have access to a matrix Z ε-close to $f(A^\dagger A)$ in Frobenius norm (Theorem 7.1)[7]. An even polynomial of A is precisely $f(A^\dagger A)$ for f a low-degree polynomial (which means f is Lipschitz), and odd polynomials can be decomposed into a product of an even polynomial with A, so our closure property is as strong as the quantum one. This is a slightly more general statement than the previous, which makes it useful for applications that give a matrix A without a corresponding vector b to apply against. The full details are derived in Section 7.

To summarize, every arithmetic operation of matrices with block-encodings in [GSLW19] can be mimicked by matrices with oversampling and query access, up to Frobenius norm error, provided that an input matrix in a block-encoding corresponds to having[8] $\mathrm{SQ}(A)$ and $\mathrm{SQ}(A^\dagger)$.

Our second type of dequantization results are dequantized versions of notable QML algorithms. First, as mentioned above, we give an optimized algorithm for the fundamental QSVT task, computing $p(A)b$ for $p : [-1,1] \to [-1,1]$ a bounded degree-d polynomial, $A \in \mathbb{C}^{m\times n}$

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

[8]We take some care here to distinguish whether we have oversampling and query access to A or $A^\dagger$. We don't need to: we show that having either one of them implies having the other, up to approximation (Remark 6.3). However, the accesses assumed in our closure properties are in some sense the most natural choices and require the least overhead.

	quantum algorithm	classical algorithm
simple QSVT [GSLW19], §6	$d\|A\|_F$	$\frac{d^{11}\|A\|_F^4}{\varepsilon^2}$
recommendation systems [KP17; CGJ19], [Tan19], §8.1	$\frac{\|A\|_F}{\sigma}$	$\frac{\|A\|_F^4\|A\|^7}{\sigma^{11}\varepsilon^2}$
supervised clustering [LMR13], [Tan21], §8.2	$\frac{\|M\|_F^2\|w\|^2(\clubsuit)}{\varepsilon}$	$\frac{\|M\|_F^4\|w\|^4}{\varepsilon^2}$
principal component analysis [LMR14; CGJ19], [Tan21], §8.3	$\frac{\|X\|_F\|X\|}{\lambda_k\varepsilon}$	$\frac{\|X\|_F^6}{\|X\|^2\lambda_k^2\eta^6\varepsilon^6}$
matrix inversion [GSLW19], [CGLLTW20], §8.4	$\frac{\|A\|_F}{\sigma}$	$\frac{\|A\|_F^4\|A\|^7}{\sigma^{11}\varepsilon^2}$
support vector machines [RML14], [DBH22], §8.5	$\frac{1}{\lambda^3\varepsilon^3}^{(\diamond)}$	$\frac{1}{\lambda^{28}\varepsilon^6}$
Hamiltonian simulation [GSLW19], §8.6	$\|H\|_F$	$\frac{\|H\|_F^4\|H\|^{11}}{\varepsilon^2}$
semidefinite program solving [BKLLSW19; AG19], [CLLW20], §8.7	$\frac{\|A^{(\cdot)}\|_F^7}{\varepsilon^{7.5}}+\frac{\sqrt{m}\|A^{(\cdot)}\|_F^2}{\varepsilon^4}$	$\frac{\|A^{(\cdot)}\|_F^{22}}{\varepsilon^{46}}+\frac{m\|A^{(\cdot)}\|_F^{14}}{\varepsilon^{28}}$
discriminant analysis [CD16], §8.8	$\frac{\|B\|_F^7}{\varepsilon^3\sigma^7}+\frac{\|W\|_F^7(\diamond)}{\varepsilon^3\sigma^7}$	$\frac{\|B\|_F^6\|B\|^4}{\varepsilon^6\sigma^{10}}+\frac{\|W\|_F^6\|W\|^{10}}{\varepsilon^6\sigma^{16}}$

Figure 7: The time complexity for our algorithms and the quantum algorithms they are based on. We assume that we get linear-time pre-processing of the input, so that we can construct a data structure supporting $\mathcal{O}(1)$ time sampling and query accesses to the input (Remark 4.12). If we get the input in a QRAM data structure instead of with pre-processing, the runtime increases by at most small polynomial factors; see Section 8 for details.
We list the runtime of the algorithm, not including the time it takes to access the output (denoted with $\overline{sq}$). The runtimes as listed ignore polylog terms, particularly those in error parameters (ε and δ) and dimension parameters (m and n). The matrices and vectors referenced in these runtimes are always the input, σ refers to a singular value threshold of the input matrices, λ refers to an eigenvalue threshold (which can be thought of here as σ^2), and $\eta > \varepsilon$ is a (dimensionless) gap parameter.
($\clubsuit$) indicates that the error analyses of the corresponding results are incomplete; we list the runtime they achieve for completeness.
($\diamond$) indicates that the corresponding results only hold in the restricted setting where the input matrices are strictly rank k. For the quantum algorithms with this tag, they allow for general matrices, but only have an informal error analysis arguing that singular values outside the range considered don't affect the final result.

a matrix with $\|A\| \leq 1$, and $b \in \mathbb{C}^n$ a vector. In Section 6, we show that, after linear-time pre-processing, we can sample from y such that $\|y - p(A)b\| \leq \varepsilon\|b\|$ in $\widetilde{\mathcal{O}}(\frac{d^{11}\|A\|_F^4\|b\|^2}{\varepsilon^2\|y\|^2}\log(mn))$ time, which we can compare to the $\mathcal{O}(d\|A\|_F\frac{\|b\|^2}{\|p(A)b\|^2})$ time of QSVT. Either as a corollary of this result or via more flexible machinery, we dequantize quantum algorithms in the domains of recommendation systems [KP17; CGJ19], supervised clustering [LMR13], principal component analysis [LMR14; Pra14; CGJ19], low-rank matrix inversion [WZP18; GSLW19], support vector machines [RML14], low-rank Hamiltonian simulation [GSLW19], low-rank semidefinite program solving [BKLLSW19; AG19], and discriminant analysis [CD16]. Fig. 7 has a summary of our results, along with a comparison of runtimes to the corresponding quantum algorithms. The reach of dequantizing techniques is surprisingly wide: in many of these applications, either the authors had asserted that their algorithm attains an exponential quantum speedup or the community had expressed hope for exponential quantum speedup from these algorithms [Pre18, Sections 6.7 and 6.8].

The proofs for these dequantization results follow the same general structure: consider the quantum algorithm and formulate the problem that this algorithm solves, and in particular, the linear algebra expression that the quantum algorithm computes. From there, repeatedly use importance sampling to approximate this expression by either a small linear combination of vectors, as in Section 1.3, or a small linear combination of rank-one matrices, sometimes called an *RUR decomposition*. Finally, use closure properties to gain oversampling and query access to that output decomposition. This procedure is straightforward and, with some practice, much of it can be done by rote.

2 Discussion

The ideal mathematician's work is intelligible only to a small group of specialists, numbering a few dozen or at most a few hundred. This group has existed only for a few decades, and there is every possibility that it may become extinct in another few decades. However, the mathematician regards his work as part of the very structure of the world, containing truths which are valid forever, from the beginning of time, *even in the most remote corner of the universe.*

—Philip J. Davis and Reuben Hersh [DH80]

2.1 Quantum machine learning

Our work has major implications for the landscape of quantum machine learning. Though we have presented many dequantized versions of QML algorithms, the broader goal remains of finding quantum advantage for a machine learning task. In view of this goal, we discuss here when our assumptions do not hold, and therefore, dequantization results do not apply.

First, quantum-inspired linear algebra crucially relies on the input data being *classical*, meaning that, for example, we are given input data as a list of entries, rather than as a quantum state, which does not have its amplitudes easily accessible. This has been pointed out by, for example, Cotler, Huang, and McClean [CHM21], who give simple problems that are exponentially hard when given vectors only via their corresponding states, but become trivial when given access to the vector's entries. A dequantized algorithm simply proves that, if its quantum counterpart does achieve a, say, exponential speedup, then the task of estimating entries or sampling from the input, which we need to run a dequantized algorithm, must be exponentially hard on a classical computer. This explains why quantum principal component analysis has a dequantization [Tan21] when given classical access to input as well as a proof of exponential quantum advantage [Hua+22] in a setting without such access. Dequantized algorithms cannot work without being given an explicit list of amplitudes, suggesting that QML has the best chance of achieving large speedups whenever classical computation cannot get access to this data (which occurs when input states come from quantum circuits and other physical quantum systems). More generally, an algorithm with a dequantization is still useful when run on "quantum data".

Even when run on classical data, algorithms can resist dequantization by using high-degree sparsity-based QSVT rather than QRAM-based QSVT, as techniques do not extend to this regime. Further, sparsity-based QSVT is BQP-complete (indeed, quantum computation can be described as applying block-encodings of low-sparsity unitary matrices), so we would not expect this to be dequantized in full. Though this avoids the dequantization barrier to large quantum speedups, its potential for practical quantum speedups remains to be seen, since the decision of whether the speedups are "practical" or "useful" will ultimately come down to the particular choice of dataset and hardware. For example, for the HHL algorithm [HHL09] and its derivatives, the matrix needs to be represented by a concise quantum circuit and have a small (poly-logarithmic in input dimension) condition number in order to gain an exponential speedup over classical algorithms. This doesn't happen in typical datasets. The collection of these demanding requirements hamstrings most attempts to find applications of HHL with the potential for practical super-polynomial speedups. We note that the current proposals that resist dequantization and potentially obtain a super-polynomial quantum speedup include Zhao, Fitzsimons, and Fitzsimons on Gaussian process regression [ZFF19] and Lloyd, Garnerone, and Zanardi on topological data analysis [LGZ16]. In fact, this latter proposal has recently drawn attention as some argue it will give practical advantage when run on near-term quantum computers [GCD22; Akh+22; Ber+22].

Finally, even if a quantum algorithm can be simulated by a classical algorithm, there are certain problems for which having $|v\rangle$ is better than having the succinct representation of v. For example, estimating the Forrelation [AA18; AC17] of $|v\rangle$ with another vector $|u\rangle$ requires exponentially many queries to v when it is given as a classical list of entries, even with the ability to produce importance samples. If the desired task is to output the Forrelation of an output of low-rank QSVT, this could potentially give a large speedup despite our results on low-rank QSVT. However, this is a case of artificially adding hardness: we are unaware of problems in machine learning where Forrelation-type quantities are desired. Such a problem would be a good candidate for QML speedup.

Our results give evidence for the lack of exponential speedup for QRAM-based QSVT algorithms. However, dequantization does not yet rule out the possibility of large polynomial speedups on classical data, which could still lead to significant performance improvements in practice with sufficiently good quantum computers. This is still an area of active research.

2.2 Randomized numerical linear algebra

All of the results presented here are more or less randomized numerical linear algebra algorithms [Mah11; Woo14]. The kind of sampling we get from sampling and query access is called *importance sampling* or *length-square sampling* in that body of work: see the survey by Kannan and Vempala [KV17] for more on importance sampling. Importance sampling, and specifically, its approximate matrix product property, is the core primitive of this work. In addition to the low-rank approximation algorithms [FKV04] used in the quantum-inspired literature, others have used importance sampling for, e.g., orthogonal tensor decomposition [DM07; MMD08; SWZ16] (generalizing low-rank approximation [FKV04]) and support vector machines [HKS11]. However, the extensibility of importance sampling via "closure properties" and the significantly widened scope of computation shown here is new to this work, to our knowledge; the most similar result we know of is Van den Nest's work on "computationally tractable" states [Van11] in the setting of quantum simulation.

From a sketching perspective, our model encompasses "the set of algorithms that can be performed in time independent of input dimension, using only ℓ_2^2 importance sampling", since this is a decent classical analogue for the input given to a quantum machine learning algorithms operating on classical data. This quantum-inspired model is weaker than the standard sketching algorithm model (Remark 4.12): 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 $\mathcal{O}(\text{nnz}(A) + T)$, where $\text{nnz}(A)$ is the number of nonzero entries of A. So, we can also think about an $\mathcal{O}(T)$-time quantum-inspired algorithm as an $\mathcal{O}(\text{nnz}(A) + T)$-time sketching algorithm, where the $\text{nnz}(A)$ portion of the runtime can *only* be used to facilitate importance sampling.[9] This restriction makes for algorithms that may perform worse in generic sketching settings, but work in more settings, and so demonstrate lack of exponential quantum speedup for a wider range of problems.

A natural question is whether more modern sketching techniques 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*

[9]The same holds for quantum algorithms using the QRAM data structure input model: the data structure itself can be built during an $\mathcal{O}(\text{nnz}(A))$-time (classical) preprocessing phase.

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 CountSketch and Johnson-Lindenstrauss still take time linear in input-sparsity. Importance sampling can be thought of as oversampling leverage scores [CCHLW22], but since we always use it to approximate matrix products, where the optimal sampling distribution is indeed importance sampling [DKM06], this perspective is not the most natural for this setting.

One may expect that these exponential speedups of dequantized algorithms may improve over existing classical algorithms. However, we do *not* claim any meaningful breakthroughs for the problems QML algorithms solve in the classical literature: the problems that these QML algorithms solve differ substantially from their usual classical counterparts. For example, the quantum recommendation systems algorithm of Kerenidis and Prakash [KP17] performs sampling from a low-rank approximation of the input instead of low-rank matrix completion, which is the typical formalization of the recommendation systems problem [Tan19]. Evaluating these quantum algorithms' justifications for their versions of problems is outside the scope of this work: instead, we argue that these algorithms would likely not give exponential speedups when implemented, regardless of whether such implementations would be useful. The goal of our framework is to demonstrate what can be done classically and establish a classical frontier for quantum algorithms to push past.

2.3 Reality

A work by Arrazola, Delgado, Bardhan, and Lloyd [ADBL20] implements and benchmarks quantum-inspired algorithms for regression and recommendation systems. The aforementioned paper of Chepurko, Clarkson, Horesh, Lin, and Woodruff [CCHLW22] does the same for the quantum-inspired algorithms they introduce. The former work makes various conclusions, including that the ε^2 scaling in the number of rows/columns taken in our recommendation systems algorithm is inherent and that the quantum-inspired algorithms performed slower and worse than direct computation for practical datasets. The latter work finds that their algorithms perform faster than direct algorithms, with an accompanying increase in error comparable to that of other sketching algorithms [DKW18]. This improvement appears to come from both a better-performing implementation as well as an algorithm with better asymptotic runtime. Nevertheless, it is difficult to draw definitive conclusions about the practicality of quantum-inspired algorithms as a whole from these experimental results. Since quantum-inspired algorithms are a restricted, weaker form of computation than classical randomized numerical linear algebra algorithms (see the comparison made above), it seems possible that they perform worse than standard sketching algorithms, despite seemingly having exponentially improved runtime in theory.

Modern sketching algorithms use similar techniques to quantum-inspired algorithms, but are more natural to run on a classical computer and are likely to be faster. For example, Dahiya, Konomis, and Woodruff [DKW18] 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." Sketching algorithms like those in [DKW18] may become a relevant point of reference for benchmarking quantum linear algebra, when the implementation of these quantum algorithms on actual quantum hardware becomes possible. In a sense, our work shows using asymptotic runtime bounds that in many scenarios sketching and sampling techniques give similar computational power to quantum linear algebra, which is a counterintuitive point since the former typically leads to linear runtimes and the latter leads to poly-logarithmic ones.

2.4 Open problems

We present some natural open problems arising from our work. Since this work presents evidence that certain classes of QML algorithms will not give practical speedups, most of these open problems aim to make this evidence more convincing.

(a) Are there other ways to construct block-encodings or prepare states that prevent dequantization? For example, the QRAM data structure used here has alternatives which in some sense vary the "norm" in which one stores the input matrix [KP20, Theorem IV.4], [CGJ19, Lemma 25]. We do not expect that algorithms using these alternatives could be fully dequantized, since these data structures generalize and strengthen the sparse-access input model, which is known to be BQP-complete [HHL09]. Are there interesting QML problems that are efficient in these "intermediate" data structure regimes?

(b) Our algorithms still have significant slowdown as compared to their quantum counterparts. What is the right running time for simple QSVT, as shown in Section 6? Could we get a d^5 degree dependence?

(c) The techniques used to improve the time complexity of simple QSVT rely on being able to maintain a vector, and so do not apply for applications like low-rank semidefinite programming Section 8.7, where we wish to estimate a trace inner product. How can we make this dequantization more efficient?

(d) Do the matrix arithmetic closure properties we showed for ℓ_2-norm importance sampling hold for other kinds of sampling and sketching distributions, like leverage score or ℓ_p-norm sampling?

Outline The rest of this thesis proceeds as follows. We introduce some preliminaries (Section 3), which can be safely skipped and referred to as needed. We then introduce oversampling and query access, describe when we can get such access efficiently, and prove some of its closure properties (Section 4). Section 5 contains all of the main propositions regarding sketching and approximating matrix products. Section 6 shows how to dequantize the basic problem of QSVT, using ideas from stably computing polynomials to get an algorithm with good dependence on $\|A\|_F$ and $1/\varepsilon$. This will not be enough for all of our applications, so in Section 7 we show some versions of SVT that are slower but more general. We use these to dequantize QML algorithms appearing in the literature in Section 8.

3 Preliminaries

3.1 Linear algebra

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 say that U is an isometry if $\|Ux\| = \|x\|$ for all x, or equivalently, if U is a subset of columns of a unitary. We can also write this decomposition as $A = \sum_{i=1}^{N} \sigma_i u_i v_i^\dagger$, where $u_i := U(\cdot, i)$ and $v_i := V(\cdot, i)$. We denote the set of singular values of A by $\operatorname{Spec}(A) := \{\sigma_k\}_{k\in[N]}$. For Hermitian A, an *(unitary) eigendecomposition* of A is a singular value decomposition where $U = V$, except the entries of D are allowed to be negative.

Using SVD, we can define the rank-k approximation of A to be $A_k := \sum_{i=1}^{k} \sigma_i u_i v_i^\dagger$ and the pseudoinverse of A to be $A^+ := \sum_{i=1}^{\operatorname{rank}(A)} \frac{1}{\sigma_i} v_i u_i^\dagger$. We now formally define singular value transformation:

Definition 3.1. For a function $f : [0, \infty) \to \mathbb{C}$ such that $f(0) = 0$ and a matrix $A \in \mathbb{C}^{m\times n}$, we define the *singular value transform* of A via a singular value decomposition $A = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^\dagger$:

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

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

Definition 3.2. For a function $f : \mathbb{R} \to \mathbb{C}$ and a Hermitian matrix $A \in \mathbb{C}^{n\times n}$, we define the eigenvalue transform of A via a *unitary eigendecomposition* $A = \sum_{i=1}^{n} \lambda_i v_i v_i^\dagger$:

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

Since we only consider eigenvalue transformations of Hermitian matrices, where singular vectors/values and eigenvectors/values (roughly) coincide, the key difference between singular value transformation and eigenvalue transformation is that the latter can distinguish eigenvalue sign. As eigenvalue transformation 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 3.3. We say $f : \mathbb{R} \to \mathbb{C}$ is *L-Lipschitz* on $\mathfrak{F} \subseteq \mathbb{R}$ if for all $x, y \in \mathfrak{F}$, $|f(x) - f(y)| \leq L|x - y|$.

We define approximate isometry as follows:[10]

Definition 3.4. Let $m, n \in \mathbb{N}$ and $m \geq n$. A matrix $V \in \mathbb{C}^{m\times n}$ is an *α-approximate isometry* if $\|V^\dagger V - I\| \leq \alpha$. It is an *$\alpha$-approximate projective isometry* if $\|V^\dagger V - \Pi\| \leq \alpha$ for Π an orthogonal projector.

If V is an α-approximate isometry, among other things, it implies that $|\|V\|^2 - 1| \leq \alpha$ and that there exists an isometry $U \in \mathbb{C}^{m\times n}$ with $\mathrm{im}(U) = \mathrm{im}(V)$ such that $\|U - V\| \leq \alpha$. We show this and other basic facts in the following lemma.

Lemma 3.5. If $\hat{X} \in \mathbb{C}^{m\times n}$ is an α-approximate isometry, then there is an exact isometry $X \in \mathbb{C}^{m\times n}$ with the same columnspace as $\hat{X}$ such that $\|\hat{X} - X\| \leq \alpha$. Furthermore, for any matrix $Y \in \mathbb{C}^{n\times n}$,

$$\|\hat{X}Y\hat{X}^\dagger - XYX^\dagger\| \leq (2\alpha + \alpha^2)\|Y\|.$$

If $\alpha < 1$, then $\|\hat{X}^+\| \leq (1-\alpha)^{-1}$ and

$$\|\hat{X}Y\hat{X}^\dagger - XYX^\dagger\| \leq \alpha\frac{2-\alpha}{(1-\alpha)^2}\|\hat{X}Y\hat{X}^\dagger\|.$$

Proof. Let $\hat{X} = UDV^\dagger$ be a singular value decomposition of $\hat{X}$, with singular values $\sigma_1, \dots, \sigma_n$ and $U \in \mathbb{C}^{m\times n}$, $D \in \mathbb{R}^{n\times n}$, $V \in \mathbb{C}^{n\times n}$. We set $X := UV^\dagger$. X is an isometry since $(UV^\dagger)^\dagger UV^\dagger = I$, it has the same columnspace as $\hat{X}$, and

$$\begin{aligned}\|UV^\dagger - \hat{X}\| = \|UV^\dagger - UDV^\dagger\| = \|I - D\| = \max_{i\in[n]}|1 - \sigma_i| \leq \max_{i\in[n]}|1 - \sigma_i||1 + \sigma_i| \\ = \max_{i\in[n]}|1 - \sigma_i^2| = \|I - D^2\| = \|I - VDU^\dagger UDV^\dagger\| = \|I - \hat{X}^\dagger\hat{X}\| \leq \alpha.\end{aligned}$$

[10]This is the notion of approximate orthonormality as given by the first arXiv version of [Tan19].

Consequently,

$$\begin{aligned}
\|\hat{X}Y\hat{X}^\dagger - XYX^\dagger\| &\leq \|\hat{X}Y(\hat{X}-X)^\dagger\| + \|(\hat{X}-X)YX\| \\
&\leq \alpha(\|\hat{X}Y\| + \|YX\|) \\
&\leq \alpha(\|XY\| + \alpha\|Y\| + \|YX\|) \\
&= (2\alpha + \alpha^2)\|Y\|
\end{aligned}$$

Suppose $\alpha < 1$, ruling out the possibility that $\hat{X}$ is the zero matrix. Then by Lemma 5.16 we have

$$\begin{aligned}
\|\hat{X}^+\| &= \max_{i\in[n]} \frac{1}{\sigma_i} \leq \frac{1}{1-\alpha}, \text{ and consequently} \\
\|\hat{X}Y\hat{X}^\dagger - XYX^\dagger\| &\leq \alpha(\|\hat{X}Y\| + \|Y\|) \\
&\leq \alpha(\|\hat{X}Y\hat{X}^\dagger\|\|\hat{X}^+\| + \|\hat{X}Y\hat{X}^\dagger\|\|\hat{X}^+\|^2) \\
&\leq \alpha\frac{1-\alpha+1}{(1-\alpha)^2}\|\hat{X}Y\hat{X}^\dagger\|.
\end{aligned}$$

□

We also define some miscellaneous notation. $(A \mid B)$ 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. The number of non-zero entries of v is denoted $\|z\|_0$. We use the Iverson bracket, where $[\![P]\!]$ is one if the predicate P is true and zero otherwise. For example, $\sum_{i=1}^d \sum_{j=i}^d a_{ij} = \sum_{i=1}^d \sum_{j=1}^d a_{ij}[\![j \geq i]\!]$. 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 4) are at least unit time cost.

3.2 Polynomials and the Chebyshev basis

We consider polynomials with complex coefficients, $p \in \mathbb{C}[x]$. For a Hermitian matrix A, $p(A)$ refers to evaluating the polynomial with x replacing A; this is equivalent to applying p to the eigenvalues of A. The right definition for applying p to a general non-square matrix is subtle; as done in QSVT, we restrict to settings where the matrix formed by evaluating p on the singular values of A coincides with the evaluation of a corresponding polynomial in A.

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

$$p(A) := q(A^\dagger A) = p(\sqrt{A^\dagger A}).$$

2. If p is *odd*, meaning that we can express $p(x) = x \cdot q(x^2)$ for some polynomial $q(x)$, then

$$p(A) := A \cdot q(\sqrt{A^\dagger A}).$$

For example, if $p(x) = x^2 + 1$, then $p(A) = A^\dagger A + I$, and if $p(x) = x^3 + x$, then $p(A) = AA^\dagger A + A$. Looking at a singular value decomposition $A = \sum \sigma_i U(\cdot, i)V_{(}\cdot, i)^\dagger$, $p(A) = \sum p(\sigma_i)U(\cdot, i)V(\cdot, i)^\dagger$ when p is odd and $p(A) = \sum p(\sigma_i)V(\cdot, i)V(\cdot, i)^\dagger$ when p is even, thus making this definition coincide with the singular value transformation as given in [GSLW19, Definition 16].

We work in the Chebyshev basis of polynomials throughout. Let $T_\ell(x)$ and $U_\ell(x)$ denote the degree-ℓ Chebyshev polynomials of the first and second kind, respectively. They can be defined on $[-1, 1]$ via

$$T_\ell(\cos(\theta)) = \cos(\ell\theta) \tag{3}$$

$$U_\ell(\cos(\theta)) = \sin((\ell + 1)x)/\sin(x), \tag{4}$$

but we will give attention to their recursive definitions, since we use them for computation.

$$\begin{aligned}
T_0(x) &= 1 & U_0(x) &= 1 \\
T_1(x) &= x & U_1(x) &= 2x \\
T_k(x) &= 2x \cdot T_{k-1}(x) - T_{k-2}(x) & U_k(x) &= 2x \cdot U_{k-1}(x) - U_{k-2}(x)
\end{aligned} \tag{5}$$

For a function $f : [-1, 1] \to \mathbb{R}$, we denote $\|f\|_{\sup} := \sup_{x\in[-1,1]}|f(x)|$. In this norm, the Chebyshev polynomials have $\|T_k(x)\|_{\sup} = 1$ and $\|U_k(x)\|_{\sup} = n + 1$. More generally, for a function $f : S \to \mathbb{R}$ for $S \subset \mathbb{R}$, we denote $\|f\|_S := \sup_{x \in S} |f(x)|$, so that $\|f\|_{\sup} = \|f\|_{[-1,1]}$.

We use the following well-known properties of Chebyshev polynomials from Mason and Handscomb [MH02].

$$T_i(x) = \tfrac{1}{2}(U_i(x) - U_{i-2}(x)) \text{ for } i \geq 1 \tag{6}$$

$$U_i(x) = \sum_{j \geq 0} T_{i-2j}(x)(1 + [\![i - 2j \neq 0]\!]) \tag{7}$$

$$T_{jk}(x) = T_j(T_k(x)) \tag{8}$$

$$U_{2k+1}(x) = U_k(T_2(x))U_1(x) = U_k(T_2(x))2x \tag{9}$$

$$\tfrac{d}{dx} T_k(x) = kU_{k-1}(x) \tag{10}$$

Any Lipschitz continuous function[11] $f : [-1,1] \to \mathbb{R}$ can be written as a (unique) linear combination of Chebyshev polynomials, $f(x) = \sum_\ell a_\ell T_\ell(x)$ (where we interpret $T_\ell(x) \equiv 0$ for negative ℓ). When f is a degree-d polynomial, then $a_\ell = 0$ for all $\ell > d$. A common way to approximate a function is by truncating its Chebyshev series expansion; we denote this operation by $f_k(x) := \sum_{\ell=0}^{k} a_\ell T_\ell(x)$, and we denote the remainder to be $\bar{f}_k(x) := f(x) - f_k(x) = \sum_{\ell=k+1}^{\infty} a_\ell T_\ell(x)$. Standard results in approximation give bounds on $\|f - f_k\|_{\sup}$ for various smoothness assumptions on f. We recommend the book by Trefethen on this topic [Tre19], and use results from it throughout. We list here some basic lemmas for future use.

Lemma 3.7 (Coefficient bound, consequence of [Tre19, Eq. (3.12)]). Let $f : [-1,1] \to \mathbb{R}$ be a Lipschitz continuous function. Then all its Chebyshev coefficients a_k are bounded: $|a_k| \leq 2\|f\|_{\sup}$.

Lemma 3.8. For a degree-d polynomial p, and $\delta = \frac{1}{4d^2}$,

$$\|p\|_{[-1-\delta, 1+\delta]} \leq e\|p\|_{[-1,1]}.$$

Proof. Without loss of generality, take $\|p\|_{\sup} = 1$. By [SV14, Proposition 2.4] and basic prop-

[11] We call a function $f : [-1,1] \to \mathbb{R}$ Lipschitz continuous if there exists a constant C such that $|f(x) - f(y)| \leq C|x - y|$ for $x, y \in [-1,1]$.

erties of Chebyshev polynomials,

$$\|p(x)\|_{[-1-\delta,1+\delta]} \leq \|T_d(x)\|_{[-1-\delta,1+\delta]} = T_d(1+\delta).$$

Further, by Proposition 2.5 in [SV14], we can evaluate $T_d(1+\delta)$ via the formula

$$\begin{aligned}
T_d(x) &= \frac{1}{2}\Big(x+\sqrt{x^2-1}\Big)^d + \frac{1}{2}\Big(x-\sqrt{x^2-1}\Big)^d \\
T_d(1+\delta) &= \frac{1}{2}\Big(1+\delta+\sqrt{2\delta+\delta^2}\Big)^d + \frac{1}{2}\Big(1+\delta-\sqrt{2\delta+\delta^2}\Big)^d \\
&\leq \exp\Big(d(\delta+\sqrt{2\delta+\delta^2})\Big) \\
&\leq \exp\Big(\frac{1}{4d}+\sqrt{\frac{1}{2}+\frac{1}{16d^2}}\Big) \leq e
\end{aligned}$$

□

4 Data access models

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

Definition 4.1 (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 query for $v(i)$. 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 query for $A(i,j)$. Let $\boldsymbol{q}(v)$ (respectively $\boldsymbol{q}(A)$) 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 $\boldsymbol{q}(v) = 1$. For brevity, we will sometimes abuse this notation (and other access notations) and, for example, abbreviate "$\mathrm{Q}(A)$ for $A \in \mathbb{C}^{m\times n}$" as "$\mathrm{Q}(A) \in \mathbb{C}^{m\times n}$". We will also sometimes abuse complexity notation like $\boldsymbol{q}$ to refer to known bounds on the complexity, instead of the complexity itself.

Definition 4.2 (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 $\boldsymbol{q}(v)$, $\boldsymbol{s}(v)$, and $\boldsymbol{n}(v)$ denote the cost of querying entries, sampling indices, and querying the norm respectively. Further define $\boldsymbol{sq}(v) := \max(\boldsymbol{q}(v), \boldsymbol{s}(v), \boldsymbol{n}(v))$.

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.[12]

Definition 4.3. For $p, q \in \mathbb{R}^n_{\geq 0}$ that are distributions, meaning $\sum_i p(i) = \sum_i q(i) = 1$, we say that p *ϕ-oversamples* q if, for all $i \in [n]$, $p(i) \geq q(i)/\phi$.

The motivation for this definition is the following: if p ϕ-oversamples q, then we can convert a sample from p to a sample from q with probability $1/\phi$ using rejection sampling: sample an i distributed as p, then accept the sample with probability $q(i)/(\phi p(i))$ (which is ≤ 1 by definition).

Definition 4.4 (Oversampling and query access). For $v \in \mathbb{C}^n$ and $\phi \geq 1$, we have $\mathrm{SQ}_\phi(v)$, *ϕ-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 $\|\tilde{v}\|^2 = \phi\|v\|^2$ and $|\tilde{v}(i)|^2 \geq |v(i)|^2$ for all $i \in [n]$. Denote $\boldsymbol{s}_\phi(v) := \boldsymbol{s}(\tilde{v})$, $\boldsymbol{q}_\phi(v) := \boldsymbol{q}(\tilde{v})$, $\boldsymbol{n}_\phi(v) := \boldsymbol{n}(\tilde{v})$, and $\boldsymbol{sq}_\phi(v) := \max(\boldsymbol{s}_\phi(v), \boldsymbol{q}_\phi(v), \boldsymbol{q}(v), \boldsymbol{n}_\phi(v))$.

The distribution $\mathcal{D}_{\tilde{v}}$ ϕ-oversamples $\mathcal{D}_v$, since for all $i \in [n]$,

$$\mathcal{D}_{\tilde{v}}(i) = \frac{|\tilde{v}_i|^2}{\|\tilde{v}\|^2} = \frac{|\tilde{v}_i|^2}{\phi\|v\|^2} \geq \frac{|v_i|^2}{\phi\|v\|^2} = \frac{1}{\phi}\mathcal{D}_v(i).$$

For this reason, we call $\mathcal{D}_{\tilde{v}}$ a *ϕ-oversampled importance sampling distribution* of v. $\mathrm{SQ}(v)$ is the same as $\mathrm{SQ}_1(v)$, by taking $\tilde{v} = v$. Note that we do not assume knowledge of ϕ (though it can be

[12]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.

estimated, (though it can be estimated as shown in Lemma 4.5). However, we do need to know $\|\tilde{v}\|$ (even if $\|v\|$ is known), as it cannot be deduced from a small number of queries, samples, or probability computations. So, we will be choosing $\tilde{v}$ (and, correspondingly, ϕ) 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 ϕ).

Intuitively speaking, estimators that use $\mathcal{D}_v$ can also use $\mathcal{D}_{\tilde{v}}$ via rejection sampling at the expense of a factor ϕ increase in the number of utilized samples. From this observation we can prove that oversampling access implies an approximate version of the usual sampling access:

Lemma 4.5. Suppose we are given $\mathrm{SQ}_\phi(v)$ and some $\delta \in (0,1]$. Denote $\overline{\boldsymbol{sq}}(v) := \phi\, \boldsymbol{sq}_\phi(v) \log \frac{1}{\delta}$. We can sample from $\mathcal{D}_v$ with probability $\geq 1-\delta$ in $\mathcal{O}(\overline{\boldsymbol{sq}}(v))$ time. We can also estimate $\|v\|$ to ν multiplicative error for $\nu \in (0,1]$ with probability $\geq 1-\delta$ in $\mathcal{O}(\frac{1}{\nu^2}\,\overline{\boldsymbol{sq}}(v))$ time.

Proof. Consider the following rejection sampling algorithm to generate samples: sample an index i from $\tilde{v}$, and output it as the desired sample with probability $r(i) := \frac{|v(i)|^2}{|\tilde{v}(i)|^2}$. Otherwise, restart. We can perform this: we can compute $r(i)$ in $\mathcal{O}(\boldsymbol{sq}_\phi(v))$ time and $r(i) \leq 1$ since $\tilde{v}$ bounds v.

The probability of accepting a sample in a round is $\sum_i \mathcal{D}_{\tilde{v}}(i) r(i) = \|v\|^2/\|\tilde{v}\|^2 = \phi^{-1}$ and, conditioned on a sample being accepted, the probability of it being i is $|v(i)|^2/\|v\|^2$, so the output distribution is $\mathcal{D}_v$ as desired. So, to get a sample with $\geq 1-\delta$ probability, run rejection sampling for at most $2\phi \log \frac{1}{\delta}$ rounds.

To estimate $\|v\|^2$, notice that we know $\|\tilde{v}\|^2$, so it suffices to estimate $\|v\|^2/\|\tilde{v}\|^2$ which is ϕ^{-1}. The probability of accepting the rejection sampling routine is ϕ^{-1}, so we run $3\nu^{-2}\phi \log \frac{2}{\delta}$ rounds of it for estimating ϕ^{-1}. Let Z denote the fraction of them which end in acceptance. Then, by a Chernoff bound we have

$$\Pr[|Z - \phi^{-1}| \geq \nu\phi^{-1}] \leq 2\exp\Big(-\frac{\nu^2 z \phi^{-1}}{2+\nu}\Big) \leq \delta,$$

so $Z\|\tilde{v}\|^2$ is a good multiplicative approximation to $\|v\|^2$ with probability $\geq 1-\delta$. □

Generally, compared to a quantum algorithm that can output (and measure) a desired vector $|v\rangle$, our algorithms will output $\mathrm{SQ}_\phi(u)$ such that $\|u - v\|$ is small. So, $\overline{\boldsymbol{sq}}(u)$ 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 for "obtaining" $\mathrm{SQ}_\phi(u)$, and then pay $\overline{\boldsymbol{sq}}(u)$ cost per additional measurement. As for error, bounds on $\|u - v\|$ imply that measurements from u and v follow distributions that are close in total variation distance [Tan19, Lemma 4.1]. Now, we show that oversampling and query access of vectors is closed under taking small linear combinations.

Lemma 4.6 (Linear combinations, Proposition 4.3 of [Tan19]). Given $\mathrm{SQ}_{\varphi_t}(v_t) \in \mathbb{C}^n$ and $\lambda_t \in \mathbb{C}$ for all $t \in [\tau]$, we have $\mathrm{SQ}_\phi(\sum_{t=1}^{\tau} \lambda_t v_t)$ for $\phi = \tau \frac{\sum \varphi_t \|\lambda_t v_t\|^2}{\|\sum \lambda_t v_t\|^2}$ and $\boldsymbol{sq}_\phi(\sum \lambda_t v_t) = \max_{t\in[\tau]} \boldsymbol{s}_{\varphi_t}(v_t) + \sum_{t=1}^{\tau} \boldsymbol{q}(v_t)$ (after paying $\mathcal{O}(\sum_{t=1}^{\tau} \boldsymbol{n}_{\varphi_t}(v_t))$ one-time pre-processing cost to query for norms).

Proof. Denote $u := \sum \lambda_t v_t$. To compute $u(s)$ for some $s \in [n]$, we just need to query $v_t(s)$ for all $t \in [\tau]$, paying $\mathcal{O}(\sum \boldsymbol{q}(v_t))$ cost. So, it suffices to get $\mathrm{SQ}(\tilde{u})$ for an appropriate bound $\tilde{u}$. We choose

$$\tilde{u}(s) = \sqrt{\tau \textstyle\sum_{t=1}^{\tau} |\lambda_t \tilde{v}_t(s)|^2},$$

so that $|\tilde{u}(s)| \geq |u(s)|$ by Cauchy–Schwarz, and $\|\tilde{u}\|^2 = \tau \sum_{t=1}^{\tau} \|\lambda_t \tilde{v}_t\|^2 = \tau \sum_{t=1}^{\tau} \varphi_t \|\lambda_t v_t\|^2$, giving the desired value of ϕ.

We have $\mathrm{SQ}(\tilde{u})$: we can compute $\|\tilde{u}\|^2$ by querying for all norms $\|\tilde{v}_t\|$, compute $\tilde{u}(s)$ by querying $\tilde{v}_t(s)$ for all $t \in [\tau]$. We can sample from $\tilde{u}$ by first sampling $t \in [\tau]$ with probability $\frac{\|\lambda_t \tilde{v}_t\|^2}{\sum_\ell \|\lambda_\ell \tilde{v}_\ell\|^2}$, and then taking our sample to be $j \in [n]$ from $\tilde{v}_t$. The probability of sampling $j \in [n]$ is correct:

$$\sum_{t=1}^{\tau} \frac{\|\lambda_t \tilde{v}_t\|^2}{\sum_\ell \|\lambda_\ell \tilde{v}_\ell\|^2} \frac{|\tilde{v}_t(j)|^2}{\|\tilde{v}_t\|^2} = \frac{\sum_{t=1}^{\tau} |\lambda_t \tilde{v}_t(j)|^2}{\sum_{\ell=1}^{\tau} \|\lambda_\ell \tilde{v}_\ell\|^2} = \frac{|\tilde{u}(j)|^2}{\|\tilde{u}\|^2}.$$

If we pre-process by querying all the norms $\|\tilde{v}_\ell\|$ in advance, we can sample from the distribution over i's in $\mathcal{O}(1)$ time, using an alias sampling data structure for the distribution (Remark 4.12), and we can sample from $\tilde{v}_t$ using our assumed access to it, $\mathrm{SQ}_{\varphi_t}(v_t)$. □

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 4.5, where we pay an additional "cancellation constant" factor of $\phi = \tau \frac{\sum \varphi_t \|\lambda_t v_t\|^2}{\|\sum \lambda_t v_t\|^2}$. This factor is only large when the linear combination has significantly smaller norm than the components v_t 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 the same kind of overhead. Namely, the QSVT framework encodes this in the subnormalization constant α of block-encodings, and the overhead from the subnormalization appears during post-selection [GSLW19]. When this cancellation is not too large, the resulting overhead typically does not affect too badly the runtime of our applications.

We also define oversampling and query access for a matrix. The same model (under an alternative definition) is also discussed in prior work [FKV04; DKR02] and is the right notion for the sampling procedures we will use.

Definition 4.7 (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 $\|\tilde{A}\|_\mathrm{F}^2 = \phi\|A\|_\mathrm{F}^2$ and $|\tilde{A}(i,j)|^2 \geq |A(i,j)|^2$ for all $(i,j) \in [m]\times[n]$.

The complexity of (over)sampling and querying from the matrix A is denoted by $\boldsymbol{s}_\phi(A) := \max(\boldsymbol{s}(\tilde{A}(i,\cdot)), \boldsymbol{s}(\tilde{a}))$, $\boldsymbol{q}_\phi(A) := \max(\boldsymbol{q}(\tilde{A}(i,\cdot)), \boldsymbol{q}(\tilde{a}))$, $\boldsymbol{q}(A) := \max(\boldsymbol{q}(A(i,\cdot)))$, and $\boldsymbol{n}_\phi(A) := \boldsymbol{n}(\tilde{a})$ respectively. We also denote $\boldsymbol{sq}_\phi(A) := \max(\boldsymbol{s}_\phi(A), \boldsymbol{q}_\phi(A), \boldsymbol{q}(A), \boldsymbol{n}_\phi(A))$. We omit subscripts if $\phi = 1$.

Access to a matrix, $\mathrm{SQ}_\phi(A)$, implies access to its vectorized version, $\mathrm{SQ}_\phi(\mathrm{vec}(A))$: 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, one can think of $\mathrm{SQ}_\phi(A)$ as $\mathrm{SQ}_\phi(\mathrm{vec}(A))$, with the addition of having access to samples (i,j) from $\mathrm{vec}(A)$, conditioned on fixing a particular row i and also knowing the probabilities of these conditional samples.

Now we prove that oversampling and query access is closed under taking outer products. The same idea also extends to taking Kronecker products of matrices.

Lemma 4.8. Given vectors $\mathrm{SQ}_{\varphi_u}(u) \in \mathbb{C}^m$ and $\mathrm{SQ}_{\varphi_v}(v) \in \mathbb{C}^n$, we have $\mathrm{SQ}_{\phi}(A)$ for their outer product $A := uv^\dagger$ with $\phi = \varphi_u\varphi_v$ and $\boldsymbol{s}_\phi(A) = \boldsymbol{s}_{\varphi_u}(u) + \boldsymbol{s}_{\varphi_v}(v)$, $\boldsymbol{q}_\phi(A) = \boldsymbol{q}_{\varphi_u}(u) + \boldsymbol{q}_{\varphi_v}(v)$, $\boldsymbol{q}(A) = \boldsymbol{q}(u) + \boldsymbol{q}(v)$, and $\boldsymbol{n}_\phi(A) = \boldsymbol{n}_{\varphi_u}(u) + \boldsymbol{n}_{\varphi_v}(v)$,

Proof. We can query an entry $A(i,j) = u(i)v(j)^\dagger$ by querying once from u and v. Our choice of upper bound is $\tilde{A} = \tilde{u}\tilde{v}^\dagger$. Clearly, this is an upper bound on $uv^\dagger$ and $\|\tilde{A}\|_{\mathrm{F}}^2 = \|\tilde{u}\|^2\|\tilde{v}\|^2 = \varphi_u\varphi_v\|A\|_{\mathrm{F}}^2$. We have $\mathrm{SQ}(\tilde{A})$ in the following manner: $\tilde{A}(i,\cdot) = \tilde{u}(i)\tilde{v}^\dagger$, so we have $\mathrm{SQ}(\tilde{A}(i,\cdot))$ from $\mathrm{SQ}(\tilde{v})$ after querying for $\tilde{u}(i)$, and $\tilde{a} = \|\tilde{v}\|^2\tilde{u}$, so we have $\mathrm{SQ}(\tilde{a})$ from $\mathrm{SQ}(\tilde{u})$ after querying for $\|\tilde{v}\|$. □

Using the same ideas as in Lemma 4.6, we can extend sampling and query access of input matrices to linear combinations of those matrices.

Lemma 4.9. Given $\mathrm{SQ}_{\varphi^{(t)}}(A^{(t)}) \in \mathbb{C}^{m\times n}$ and $\lambda_t \in \mathbb{C}$ for all $t \in [\tau]$, we have $\mathrm{SQ}_\phi(A) \in \mathbb{C}^{m\times n}$ for $A := \sum_{t=1}^\tau \lambda_t A^{(t)}$ with $\phi = \tau\frac{\sum_{t=1}^\tau \varphi^{(t)}\|\lambda_t A^{(t)}\|_{\mathrm{F}}^2}{\|A\|_{\mathrm{F}}^2}$ and $\boldsymbol{s}_\phi(A) = \max_{t\in[\tau]} \boldsymbol{s}_{\varphi^{(t)}}(A^{(t)}) + \sum_{t=1}^\tau \boldsymbol{q}_{\varphi^{(t)}}(A^{(t)})$, $\boldsymbol{q}_\phi(A) = \sum_{t=1}^\tau \boldsymbol{q}_{\varphi^{(t)}}(A^{(t)})$, $\boldsymbol{q}(A) = \sum_{t=1}^\tau \boldsymbol{q}(A^{(t)})$, and $\boldsymbol{n}_\phi(A) = 1$ (after paying $\mathcal{O}(\sum_{t=1}^\tau \boldsymbol{n}_{\varphi^{(t)}}(A^{(t)}))$ one-time pre-processing cost).

Proof. To compute $A(i,j) = \sum_{t=1}^\tau \lambda_t A^{(t)}(i,j)$ for $(i,j) \in [m]\times[n]$, we just need to query $A^{(t)}(i,j)$ for all $t \in [\tau]$, paying $\mathcal{O}(\sum_t \boldsymbol{q}(A^{(t)}))$ cost. So, it suffices to get $\mathrm{SQ}(\tilde{A})$ for an appropriate bound $\tilde{A}$. We choose

$$\tilde{A}(i,j) = \sqrt{\tau \textstyle\sum_{t=1}^\tau |\lambda_t \tilde{A}^{(t)}(i,j)|^2}.$$

That $|\tilde{A}(i,j)| \geq |A(i,j)|$ follows from Cauchy–Schwarz, and we get the desired value of ϕ:

$$\|\tilde{A}\|_{\mathrm{F}}^2 = \tau\sum_{t=1}^\tau \|\lambda_i \tilde{A}^{(t)}\|_{\mathrm{F}}^2 = \tau\sum_{t=1}^\tau \varphi^{(t)}\|\lambda_i A^{(t)}\|_{\mathrm{F}}^2.$$

We have $\mathrm{SQ}(\tilde{A})$: we can compute $\|\tilde{A}\|_{\mathrm{F}}$ by querying for all norms $\|\tilde{A}^{(t)}\|_{\mathrm{F}}$, compute $\tilde{a}(i) = \|\tilde{A}(i,\cdot)\| = \sqrt{\tau\sum_{t=1}^\tau \|\lambda_t \tilde{A}^{(t)}(i,\cdot)\|^2}$ by querying $\tilde{a}^{(t)}(i)$ for all $t \in [\tau]$, and compute $\tilde{A}(i,j)$ by querying $\tilde{A}^{(t)}(i,j)$ for all $t \in [\tau]$. Analogously to Lemma 4.6, we can sample from $\tilde{a}$ by first sampling $s \in [\tau]$ with probability $\frac{\|\lambda_s \tilde{A}^{(s)}\|_{\mathrm{F}}^2}{\sum_t \|\lambda_t \tilde{A}^{(t)}\|_{\mathrm{F}}^2}$, then taking our sample to be $i \in [m]$ from $\mathcal{D}_{\tilde{a}^{(s)}}$. If we pre-process by querying all the Frobenius norms $\|\tilde{A}^{(t)}\|_{\mathrm{F}}$ in advance, we can sample from $\tilde{a}$ in $\mathcal{O}(\max_{t\in[\tau]} \boldsymbol{s}_{\varphi^{(t)}}(A^{(t)}))$ time. We can sample from $\tilde{A}(i,\cdot)$ by first sampling $s \in [\tau]$

with probability $\frac{\|\lambda_s \tilde{A}^{(s)}(i,\cdot)\|^2}{\sum_t \|\lambda_t \tilde{A}^{(t)}(i,\cdot)\|^2}$, then taking our sample to be $j \in [n]$ from $\mathcal{D}_{\tilde{A}^{(s)}(i,\cdot)}$. This takes $\mathcal{O}(\sum_{t=1}^{\tau} \boldsymbol{q}_{\varphi^{(t)}}(A^{(t)}) + \max_{t\in[\tau]} \boldsymbol{s}_{\varphi^{(t)}}(A^{(t)}))$ time. □

We will use this result to attain sampling and query access to an output vector when it is implicitly represented by input (here, a matrix A and a vector b).

Corollary 4.10. *Suppose we are given access to a matrix* $\mathrm{SQ}(A) \in \mathbb{C}^{m\times n}$ *and a vector* $\mathrm{SQ}(b) \in \mathbb{C}^n$, *where we can respond to queries in* $\mathcal{O}(1)$ *time. Further suppose we have a vector* $\mathrm{Q}(u) \in C^n$ *implicitly represented by* $v \in \mathbb{C}^m$ *and* η, *with* $u = A^\dagger v + \eta b$. *Then we can:*

(i) Compute entries of u *in* $\mathcal{O}(\|v\|_0)$ *time;*

(ii) Sample $i \in [n]$ *with probability* $\frac{|u(i)|^2}{\|u\|^2}$ *in* $\mathcal{O}\Big(\|v\|_0 \frac{\|v\|_0 \sum_k \|v(k)A(k,\cdot)\|^2 + \eta^2\|b\|^2}{\|A^\dagger v + \eta b\|^2} \log\frac{1}{\delta}\Big)$ *time with probability* $\geq 1-\delta$*;*

(iii) Estimate $\|u\|^2$ *to* ν *relative error in* $\mathcal{O}\Big(\|v\|_0 \frac{\|v\|_0 \sum_k \|v(k)A(k,\cdot)\|^2 + \eta^2\|b\|^2}{\nu^2\|A^\dagger v + \eta b\|^2} \log\frac{1}{\delta}\Big)$ *time with probability* $\geq 1-\delta$.

Proof. By Lemma 4.6, we have $\mathrm{SQ}_\phi(A^\dagger v)$ for

$$\phi = \|v\|_0 \frac{\sum_k \|v(k)A(k,\cdot)\|^2}{\|A^\dagger v\|^2}$$

and a query cost of $\mathcal{O}(\|v\|_0)$. Applying Lemma 4.6 again, we have $\mathrm{SQ}_\varphi(A^\dagger v + \eta b)$ for

$$\varphi = 2\frac{\|v\|_0 \sum_k \|v(k)A(k,\cdot)\|^2 + \eta^2\|b\|^2}{\|A^\dagger v + \eta b\|^2}$$

By Lemma 4.5, we can draw one sample from u with probability $\geq 1-\delta$ with $\mathcal{O}(\varphi \log\frac{1}{\delta})$ queries to $\mathrm{SQ}_\phi(A^\dagger v)$, each of which takes $\mathcal{O}(\|v\|_0)$ time. Similarly, we can estimate $\|u\|^2$ to ν multiplicative error with $\mathcal{O}(\frac{\varphi}{\nu^2}\log\frac{1}{\delta})$ queries to $\mathrm{SQ}_\phi(A^\dagger v)$. □

Remark 4.11. With the lemmas we've introduced, we can already get oversampling and query access to some modest expressions. For example, consider RUR decompositions, which show up frequently in our results: suppose we have $\mathrm{SQ}(A)$ for $A \in \mathbb{C}^{m\times n}$, $R \in \mathbb{C}^{r\times n}$ a (possibly normalized) subset of rows of A, and a matrix $U \in \mathbb{C}^{r\times r}$. Then

$$R^\dagger U R = \sum_{i=1}^{r}\sum_{j=1}^{r} U(i,j) R(i,\cdot)^\dagger R(j,\cdot),$$

which is a linear combination of r^2 outer products involving rows of A. So, by Lemma 4.8 and Lemma 4.9, we have $\mathrm{SQ}_\phi(R^\dagger U R)$.

For us, the most interesting scenario is when our sampling and query oracles take poly-logarithmic time, since this corresponds to the scenarios where quantum state preparation procedures can run in time polylog(n). In these scenarios, quantum machine learning have the potential to achieve exponential speedups. We can provide such classical access in various ways.

Remark 4.12. Below, we list 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, i.e., we do not count each operation bitwise, their runtimes may be missing log factors that should be counted for a fair comparison between classical and quantum.

(a) (Data structure) Given $v \in \mathbb{C}^n$ in the standard RAM model, the alias method [Vos91] 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 $\boldsymbol{sq}(v) = \Theta(1)$ in $\mathcal{O}(n)$ time, and by extension, for a matrix $A \in \mathbb{C}^{m\times n}$, $\mathrm{SQ}(A)$ with $\boldsymbol{sq}(A) = \Theta(1)$ in $\mathcal{O}(mn)$ time. If the input vector (resp. matrix) is given as a list of $\mathrm{nnz}(v)$ (resp. $\mathrm{nnz}(A)$) of its non-zero entries, then the pre-processing time is linear in that number of entries. Therefore, the quantum-inspired setting can be directly translated to a basic randomized numerical linear algebra algorithm. More precisely, with this data structure, a fast quantum-inspired algorithm (say, one running in time $\mathcal{O}(T\, \boldsymbol{sq}(A))$ for T independent of input size) implies an algorithm in the standard computational model (running in $\mathcal{O}(\mathrm{nnz}(A) + T)$ time).

(b) (Dynamic data structure) QML algorithms often assume their input is in a data structure with a certain kind of quantum access [Pra14; KP20; WZP18; RSWPL19; CGJ19]. They argue that, since this data structure allows for circuits preparing input states with linear gate count but polylog *depth*, hardware called QRAM might be able to parallelize these circuits enough so that they run in effectively 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. 6) 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 $\boldsymbol{q}(A) = \mathcal{O}(1)$, $\boldsymbol{s}(A) = \mathcal{O}(\log mn)$, and $\boldsymbol{n}(A) = \mathcal{O}(1)$. 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 $\mathcal{O}(\log mn)$ time, which is used in applications of QML where data accumulates over time [KP17].

(c) (Integrability assumption) For $v \in \mathbb{C}^n$, suppose we can compute entries $v(i)$ 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 $\boldsymbol{q}(v) = \mathcal{O}(T)$, $\boldsymbol{s}(v) = \mathcal{O}(T\log n)$, and $\boldsymbol{n}(v) = \mathcal{O}(T)$. Analogously, the quantum state that encodes v in its amplitudes, $|v\rangle = \sum_i \frac{v_i}{\|v\|}|i\rangle$, can be prepared in time $\mathcal{O}(T\log n)$ via Grover-Rudolph state preparation [GR02]. (One can think about the QRAM data structure as pre-computing all the necessary sums for this protocol.)

(d) (Uniformity assumption) Given $\mathcal{O}(1)$-time $\mathrm{Q}(v) \in \mathbb{C}^n$ and a β such that $\max|v(i)|^2 \leq \beta/n$, we have $\mathrm{SQ}_\phi(v)$ with $\phi = \beta/\|v\|^2$ and $\boldsymbol{sq}_\phi(v) = \mathcal{O}(1)$, by using the vector whose entries are all $\sqrt{\beta/n}$ as the upper bound $\tilde{v}$. Assuming the ability to query entries of v in superposition, a quantum state corresponding to v can be prepared in time $\mathcal{O}\big(\sqrt{\phi}\log n\big)$.

(e) (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 $|A(i,j)| \leq c$ (and efficiently computable), then we have $\mathrm{SQ}_\phi(A)$ for $\phi = c^2\frac{sm}{\|A\|_\mathrm{F}^2}$, 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).

Note that similar sparse-access assumptions are often seen in the QML and Hamiltonian simulation literature [HHL09]. Also, if A is not much smaller than we expect, then ϕ can be 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 $\phi \leq (c/c')^2$.

(f) (CT states) In 2009, Van den Nest defined the notion of a "computationally tractable" (CT) state [Van11]. Using our notation, $|\psi\rangle \in \mathbb{C}^n$ is a CT state if we have $\mathrm{SQ}(\psi)$ with $\boldsymbol{sq}(\psi) = \mathrm{polylog}(n)$. 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. For more details on how can one get efficient sampling and query access to such vectors we direct the reader to [Van11].

5 Sketching matrices to reduce dimension

We now introduce the workhorse of our algorithms: the matrix sketch. Using sampling and query access, we can generate these sketches efficiently, and these allow one to reduce the dimensionality of a problem, up to some approximation. Most of the results presented in this section are known in the classical sketching literature: we present them here for completeness, and to restate them in the context of sampling and query access.

Definition 5.1. For a distribution $p \in \mathbb{R}^m$, we say that a matrix $S \in \mathbb{R}^{s\times m}$ is *sampled according to p* if each row of S is independently chosen to be $e_i/\sqrt{s \cdot p(i)}$ with probability $p(i)$, where e_i is the vector that is one in the ith position and zero elsewhere. If p is an ℓ_2-norm sampling distribution $\mathcal{D}_v$ as defined in Definition 4.2, then we also say S is *sampled according to v*.

We call S an *importance sampling sketch* for $A \in \mathbb{C}^{m\times n}$ if it is sampled according to A's row norms a, and we call S a *ϕ-oversampled importance sampling sketch* if it is sampled according to the bounding row norms from $\mathrm{SQ}_\phi(A)$, $\tilde{a}$ (or, more generally, from a ϕ-oversampled importance sampling distribution of a).

One should think of S as a description of how to sketch A down to SA. The following lemma shows that $\|SA\|_\mathrm{F}$ approximates $\|A\|_\mathrm{F}$, giving a simple example of the phenomenon that SA

approximates A in certain senses: it shows that $\|SA\|_F = \Theta(\|A\|_F)$ with probability ≥ 0.9 when S has $\Omega(\frac{1}{\phi^2})$ rows. We show later (Lemma 5.9) that a similar statement holds for spectral norm: $\|SA\| = \Theta(\|A\|)$ with probability ≥ 0.9 when S has $\tilde{\Omega}(\phi^2\|A\|_F^2/\|A\|^2)$ rows.

Lemma 5.2 (Frobenius norm bounds for matrix sketches). Let $S \in \mathbb{C}^{r\times m}$ be a ϕ-oversampled importance sampling sketch of $A \in \mathbb{C}^{m\times n}$. Then $\|[SA](i,\cdot)\| \leq \sqrt{\phi/r}\|A\|_F$ for all $i \in [r]$, so $\|SA\|_F^2 \leq \phi\|A\|_F^2$ (unconditionally). Equality holds when $\phi = 1$. Further,

$$\Pr\left[|\|SA\|_F^2 - \|A\|_F^2| \geq \sqrt{\frac{\phi^2\ln(2/\delta)}{2r}}\|A\|_F^2\right] \leq \delta.$$

Proof. Let p be the distribution used to create S, and let s_i be the sample from p used for row i of S. Then $\|SA\|_F^2$ is the sum of the row norms $\|[SA](i,\cdot)\|^2$ over all $i \in [r]$, and

$$\|[SA](i,\cdot)\|^2 = \frac{\|A(s_i,\cdot)\|^2}{r\cdot p(s_i)} \leq \frac{\phi}{r}\|A\|_F^2$$
$$\boldsymbol{E}[\|[SA](i,\cdot)\|^2] = \sum_{s=1}^{m} p(s)\frac{\|A(s,\cdot)\|^2}{r\cdot p(s)} = \frac{1}{r}\|A\|_F^2$$

The first equation shows the unconditional bounds on $\|SA\|_F$. When $\phi = 1$, $p(i) = \|A(i,\cdot)\|^2/\|A\|_F^2$ so the inequality becomes an equality. By the second equation, $\|SA\|_F^2 - \|A\|_F^2$ has expected value zero and is the sum of independent random variables bounded in $[-\|A\|_F^2, (\phi-1)\|A\|_F^2]$, so the probabilistic bound follows immediately from Hoeffding's inequality. □

In the standard algorithm setting, computing an importance sampling sketch requires reading all of A, since we need to sample from $\mathcal{D}_a$. If we have $\mathrm{SQ}_\phi(A)$, though, we can efficiently create a ϕ-oversampling sketch S in $\mathcal{O}(s(\boldsymbol{s}_\phi(A) + \boldsymbol{q}_\phi(A)) + \boldsymbol{n}_\phi(A))$ time: for each row of S, we pull a sample from p, and then compute $\sqrt{p(i)}$. After finding this sketch S, we have an implicit description of SA: it is a normalized multiset of rows of A, so we can describe it with the row indices and corresponding normalization, $(i_1, c_1), \dots, (i_s, c_s)$.

SA can be used to approximate matrix expressions involving A. Further, we can chain sketches using the lemma below, which shows that from $\mathrm{SQ}_\phi(A)$, we have $\mathrm{SQ}_{\leq 2\phi}((SA)^\dagger)$, under a mild assumption on the size of the sketch S. This can be used to find a sketch $T^\dagger$ of $(SA)^\dagger$. The resulting expression SAT is small enough that we can compute functions of it in time independent of dimension, and so will be used extensively. When we discuss sketching A down to SAT, we are referring to the below lemma for the method of sampling T.

Lemma 5.3. Consider $\mathrm{SQ}_\varphi(A) \in \mathbb{C}^{m\times n}$ and $S \in \mathbb{R}^{r\times m}$ sampled according to $\tilde{a}$, described as pairs $(i_1, c_1), \dots, (i_r, c_r)$. If $r \geq 2\varphi^2 \ln\frac{2}{\delta}$, then with probability $\geq 1-\delta$, we have $\mathrm{SQ}_\phi(SA)$ and $\mathrm{SQ}_\phi((SA)^\dagger)$ for some ϕ satisfying $\phi \leq 2\varphi$. If $\varphi = 1$, then for all r, we have $\mathrm{SQ}(SA)$ and $\mathrm{SQ}((SA)^\dagger)$.

After $\mathcal{O}(\boldsymbol{n}_\varphi(A))$ pre-processing cost, the runtimes for $\mathrm{SQ}_\phi(SA)$ are $\boldsymbol{q}(SA) = \boldsymbol{q}(A)$, $\boldsymbol{s}_\phi(SA) = \boldsymbol{s}_\varphi(A)$, $\boldsymbol{q}_\phi(SA) = \boldsymbol{q}_\varphi(A)$, and $\boldsymbol{n}_\phi(SA) = \mathcal{O}(1)$. The runtimes for $\mathrm{SQ}_\phi((SA)^\dagger)$ are $\boldsymbol{q}((SA)^\dagger) = \boldsymbol{q}(A)$, $\boldsymbol{s}_\phi((SA)^\dagger) = \boldsymbol{s}_\varphi(A) + r\,\boldsymbol{q}_\varphi(A)$, $\boldsymbol{q}_\phi((SA)^\dagger) = r\,\boldsymbol{q}_\varphi(A)$, and $\boldsymbol{n}_\phi((SA)^\dagger) = \boldsymbol{n}_\varphi(A)$.

Proof. By Lemma 5.2, $\|SA\|_F^2 \geq \|A\|_F^2/2$ with probability $\geq 1-\delta$. Suppose this bound holds. To get $\mathrm{SQ}_\phi(SA)$, we take $\widetilde{SA} = S\tilde{A}$, which bounds SA by inspection. Further, $\|S\tilde{A}\|_F^2 = \|\tilde{A}\|_F^2$ by Lemma 5.2, so $\phi = \|S\tilde{A}\|_F^2/\|SA\|_F^2 = \varphi\|A\|_F^2/\|SA\|_F^2 \leq 2\varphi$. Analogously, $(S\tilde{A})^\dagger$ works as a bound for $\mathrm{SQ}_\phi((SA)^\dagger)$. We can query an entry of SA by querying the corresponding entry of A, so all that suffices is to show that we have $\mathrm{SQ}(S\tilde{A})$ and $\mathrm{SQ}((S\tilde{A})^\dagger)$ from $\mathrm{SQ}(\tilde{A})$. (When $\varphi = 1$, we can ignore the above argument: the rest of the proof will show that we have $\mathrm{SQ}(SA)$ and $\mathrm{SQ}((SA)^\dagger)$ from $\mathrm{SQ}(A)$.)

We have $\mathrm{SQ}(S\tilde{A})$. Because the rows of $S\tilde{A}$ are rescaled rows of $\tilde{A}$, we have SQ access to them from SQ access to $\tilde{A}$. Because $\|S\tilde{A}\|_F^2 = \|\tilde{A}\|_F^2$ and $\|[S\tilde{A}](i,\cdot)\|^2 = \|\tilde{A}\|_F^2/r$, after precomputing $\|\tilde{A}\|_F^2$, we have SQ access to the vector of row norms of $S\tilde{A}$ (pulling samples simply by pulling samples from the uniform distribution).

We have $\mathrm{SQ}((S\tilde{A})^\dagger)$. (This proof is similar to one from [FKV04].) Since the rows of $(S\tilde{A})^\dagger$ are length r, we can respond to SQ queries to them by reading all entries of the row and performing some linear-time computation. $\|(S\tilde{A})^\dagger\|_F^2 = \|\tilde{A}\|_F^2$, so we can respond to a norm query by querying the norm of $\tilde{A}$. Finally, we can sample according to the row norms of $(S\tilde{A})^\dagger$ by first querying an index $i \in [r]$ uniformly at random, then outputting the index $j \in [n]$ sampled from $[S\tilde{A}](i,\cdot)$ (which we can sample from because it is a row of $\tilde{A}$). The distribution of the samples output by this procedure is correct: the probability of outputting j is

$$\frac{1}{r}\sum_{i=1}^{r}\frac{|[S\tilde{A}](i,j)|^2}{\|[S\tilde{A}](i,\cdot)\|^2} = \sum_{i=1}^{r}\frac{|[S\tilde{A}](i,j)|^2}{\|S\tilde{A}\|_F^2} = \frac{\|[S\tilde{A}](\cdot,j)\|^2}{\|S\tilde{A}\|_F^2}. \qquad \square$$

5.1 Approximation results

Here, we present approximation results on sketched matrices that we will use heavily throughout our results. 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. We formalize this with two variance bounds, which we can use together with Chebyshev's inequality.

Lemma 5.4 (Asymmetric matrix multiplication to Frobenius norm error, [DKM06, Lemma 4]). Consider $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{m\times p}$, and take $S \in \mathbb{R}^{r\times m}$ to be sampled according to $p \in \mathbb{R}^m$ a ϕ-oversampled importance sampling distribution from X or Y. Then,

$$\boldsymbol{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F^2] \le \frac{\phi}{r}\|X\|_F^2\|Y\|_F^2 \quad \text{and} \quad \boldsymbol{E}\Big[\sum_{i=1}^r \|[SX](i,\cdot)\|^2\|[SY](i,\cdot)\|^2\Big] \le \frac{\phi}{r}\|X\|_F^2\|Y\|_F^2.$$

Proof. To show the first equation, we use that $\boldsymbol{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F^2]$ is a sum of variances, one for each entry (i,j), since $\boldsymbol{E}[X^\dagger S^\dagger SY - XY]$ is zero in every entry. Furthermore, for every entry (i,j), the matrix expression is the sum of r independent, mean-zero terms, one for each row of S:

$$[X^\dagger S^\dagger SY - XY](i,j) = \sum_{s=1}^r \Big([SX](s,i)^\dagger [SY](s,j) - \frac{1}{r}[X^\dagger Y](i,j)\Big).$$

So, we can use standard properties of variances[13] to conclude that

$$\begin{aligned}\boldsymbol{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F^2] &= r\cdot \boldsymbol{E}[\|[SX](1,\cdot)^\dagger[SY](1,\cdot) - \tfrac{1}{r}X^\dagger Y\|_F^2] \le r\cdot\boldsymbol{E}[\|[SX](1,\cdot)^\dagger[SY](1,\cdot)\|_F^2] \\ &= r\sum_{i=1}^m p(i)\frac{\|X(i,\cdot)^\dagger Y(i,\cdot)\|_F^2}{r^2p(i)^2} = \frac{1}{r}\sum_{i=1}^m \frac{\|X(i,\cdot)\|^2\|Y(i,\cdot)\|^2}{p(i)} \le \frac{\phi}{r}\|X\|_F^2\|Y\|_F^2.\end{aligned}$$

The second other inequality follows by the same computation:

$$\boldsymbol{E}\Big[\sum_{i=1}^r \|[SX](i,\cdot)\|^2\|[SY](i,\cdot)\|^2\Big] = r\cdot\boldsymbol{E}[\|[SX](1,\cdot)\|^2\|[SY](1,\cdot)\|^2] \le \frac{\phi}{s}\|X\|_F^2\|Y\|_F^2. \qquad \square$$

The above result shows that, given SQ(X), $X^\dagger Y$ can be approximated by a sketch with constant failure probability. If we have SQ(X) *and* SQ(Y), we can make the failure probability

[13] See the proof of Lemma 5.5 for this kind of computation done with more detail.

exponential small. To show this tighter error bound, we use an argument of Drineas, Kannan, and Mahoney for approximating matrix multiplication. We state their result in a slightly stronger form, which is actually proved in their paper.

Lemma 5.5 (Matrix multiplication by subsampling [DKM06, Theorem 1]). Suppose we are given $X \in \mathbb{C}^{n\times m}, Y \in \mathbb{C}^{n\times p}, r \in \mathbb{N}$ and a distribution $p \in \mathbb{R}^n$ satisfying the oversampling condition that, for some $\phi \geq 1$,

$$p(k) \geq \frac{\|X(k,\cdot)\|\|Y(k,\cdot)\|}{\phi\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|}.$$

Let $S \in \mathbb{R}^{r\times n}$ be sampled according to p. Then $X^\dagger S^\dagger SY$ is an unbiased estimator for $X^\dagger Y$ and

$$\Pr\left[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F < \sqrt{\frac{8\phi^2\ln(2/\delta)}{r}}\underbrace{\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|}_{\leq \|X\|_F\|Y\|_F}\right] > 1-\delta.$$

Proof. Using that the rows of S are selected independently, we can conclude the following:

$$\boldsymbol{E}[(SX)^\dagger(SY)] = r\cdot\boldsymbol{E}[[SX](1,\cdot)^\dagger[SY](1,\cdot)] = r\sum_{i=1}^n p(i)\frac{X(i,\cdot)^\dagger Y(i,\cdot)}{rp(i)} = X^\dagger Y$$

$$\begin{aligned}\boldsymbol{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F^2] &= \sum_{i=1}^m\sum_{j=1}^p \boldsymbol{E}\left[\left|[X^\dagger S^\dagger SY - X^\dagger Y](i,j)\right|^2\right]\\
&= r\sum_{i=1}^m\sum_{j=1}^p \boldsymbol{E}\left[\left|[SX](1,i)^\dagger[SY](1,j) - [X^\dagger Y](i,j)\right|^2\right]\\
&\leq r\sum_{i=1}^m\sum_{j=1}^p \boldsymbol{E}\left[\left|[SX](1,i)^\dagger[SY](1,j)\right|^2\right]\\
&= r\,\boldsymbol{E}\left[\|[SX](1,\cdot)\|^2\|[SY](1,\cdot)\|^2\right]\\
&= r\sum_{k=1}^n p(k)\frac{\|X(k,\cdot)\|^2}{r\cdot p(k)}\frac{\|Y(k,\cdot)\|^2}{r\cdot p(k)}\\
&\leq \frac{1}{r}\sum_{k=1}^n \frac{\phi\sum_\ell\|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|}{\|X(k,\cdot)\|\|Y(k,\cdot)\|}\|X(k,\cdot)\|^2\|Y(k,\cdot)\|^2\\
&= \frac{\phi}{r}\Big(\sum_k \|X(k,\cdot)\|\|Y(k,\cdot)\|\Big)^2.\end{aligned}$$

To prove concentration, we use McDiarmid's inequality [McD89].

Lemma 5.6 ([McD89, Lemma (1.2)]). Let $X_1, \dots, X_c$ be independent random variables with X_s taking values in a set A_s for all $s \in [c]$. Suppose that f is a real valued measurable function on the product set $\Pi_s A_s$ such that $|f(x) - f(x')| \leq b_s$ whenever the vectors x and x' differ only in the s-th coordinate. Let Y be the random variable $f[X_1, \dots, X_c]$. Then for any $\gamma > 0$:

$$\Pr[|Y - \boldsymbol{E}[Y]| \geq \gamma] \leq 2\exp\Big(-\frac{2\gamma^2}{\sum_s b_s^2}\Big).$$

To use Lemma 5.6, we think about this expression as a function of the indices that are randomly chosen from p. That is, let f be the function $[n]^r \to \mathbb{R}$ defined to be

$$f(i_1, i_2, \dots, i_r) := \Big\| X^\dagger Y - \sum_{s=1}^{r} \frac{1}{r \cdot p(i_s)} X(i_s, \cdot)^\dagger Y(i_s, \cdot) \Big\|_F,$$

Then, by Jensen's inequality, we have

$$\boldsymbol{E}[f] = \boldsymbol{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F] \leq \sqrt{\boldsymbol{E}[\|X^\dagger S^\dagger SY - XY\|_F^2]} \leq \sqrt{\frac{\phi}{r}} \sum_k \|X(k, \cdot)\| \|Y(k, \cdot)\|.$$

Now suppose that the index sequences $\vec{i}$ and $\vec{i}'$ only differ at the s-th position. Then by the triangle inequality,

$$\begin{aligned} |f(\vec{i}) - f(\vec{i}')| &\leq \frac{1}{r} \Big\| \frac{1}{p(i_s)} X(i_s, \cdot)^\dagger Y(i_s, \cdot) - \frac{1}{p(i_s')} X(i_s', \cdot)^\dagger Y(i_s', \cdot) \Big\|_F \\ &\leq \frac{2}{r} \max_{k \in [n]} \Big\| \frac{1}{p(k)} X(k, \cdot)^\dagger Y(k, \cdot) \Big\|_F \leq \frac{2\phi}{r} \sum_{k=1}^{n} \|X(k, \cdot)\| \|Y(k, \cdot)\|. \end{aligned}$$

Now, by Lemma 5.6, we conclude that

$$\Pr\Bigg[|f - E[f]| \geq \sqrt{\frac{2\phi^2 \ln(2/\delta)}{r}} \sum_k \|X(\cdot, k)\| \|Y(k, \cdot)\| \Bigg] \leq \delta.$$

So, with probability $\geq 1 - \delta$,

$$\begin{aligned}
\|X^\dagger S^\dagger SY - X^\dagger Y\|_F &\leq \boldsymbol{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F] + \sqrt{\frac{2\phi^2 \ln(2/\delta)}{r}} \sum_k \|X(\cdot,k)\|\|Y(k,\cdot)\| \\
&\leq \Big(\sqrt{\frac{\phi}{r}} + \sqrt{\frac{2\phi^2 \ln(2/\delta)}{r}}\Big) \sum_k \|X(\cdot,k)\|\|Y(k,\cdot)\| \\
&\leq \sqrt{\frac{8\phi^2 \ln(2/\delta)}{r}} \sum_k \|X(\cdot,k)\|\|Y(k,\cdot)\|. \qquad \square
\end{aligned}$$

From a simple application of Lemma 5.5, we get a key lemma used frequently in Section 8.

Lemma 5.7 (Approximating matrix multiplication to Frobenius norm error; corollary of [DKM06, Theorem 1]). Consider $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{m\times p}$, and take $S \in \mathbb{R}^{r\times m}$ to be sampled according to $q := \frac{q_1+q_2}{2}$, where $q_1, q_2 \in \mathbb{R}^m$ are ϕ_1, ϕ_2-oversampled importance sampling distributions from x, y, the vector of row norms for X, Y, respectively. Then S is a $2\phi_1, 2\phi_2$-oversampled importance sampling sketch of X, Y, respectively. Further,

$$\Pr\Big[\|X^\dagger S^\dagger SY - X^\dagger Y\|_F < \sqrt{\frac{8\phi_1\phi_2 \log 2/\delta}{r}}\|X\|_F\|Y\|_F\Big] > 1 - \delta.$$

Proof. First, notice that $2q(i) \geq q_1(i)$ and $2q(i) \geq q_2(i)$, so q oversamples the importance sampling distributions for X and Y with constants $2\phi_1$ and $2\phi_2$, respectively. We get the bound by using Lemma 5.5; q satisfies the oversampling condition with $\phi = \frac{\sqrt{\phi_1\phi_2}\|X\|_F\|Y\|_F}{\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|}$, using the inequality of arithmetic and geometric means:

$$\begin{aligned}
\frac{1}{q(i)} \frac{\|X(i,\cdot)\|\|Y(i,\cdot)\|}{\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|} &= \frac{2}{q_1(i)+q_2(i)} \frac{\|X(i,\cdot)\|\|Y(i,\cdot)\|}{\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|} \\
&\leq \frac{1}{\sqrt{q_1(i)q_2(i)}} \frac{\|X(i,\cdot)\|\|Y(i,\cdot)\|}{\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|} \\
&\leq \frac{\sqrt{\phi_1\phi_2}\|X\|_F\|Y\|_F}{\|X(i,\cdot)\|\|Y(i,\cdot)\|} \frac{\|X(i,\cdot)\|\|Y(i,\cdot)\|}{\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|} \\
&= \frac{\sqrt{\phi_1\phi_2}\|X\|_F\|Y\|_F}{\sum_\ell \|X(\ell,\cdot)\|\|Y(\ell,\cdot)\|}. \qquad \square
\end{aligned}$$

Remark 5.8. Lemma 5.7 implies that, given $\mathrm{SQ}_{\phi_1}(X)$ and $\mathrm{SQ}_{\phi_2}(Y)$, we can get $\mathrm{SQ}_\phi(M)$ for M a sufficiently good approximation to $X^\dagger Y$, with $\phi \leq \phi_1\phi_2 \frac{\|X\|_F^2\|Y\|_F^2}{\|M\|_F^2}$. This is an approximate closure property for oversampling and query access under matrix products.

Given the above types of accesses, we can compute the sketch S necessary for Lemma 5.7 by taking $p = \mathcal{D}_{\tilde{x}}$ and $q = \mathcal{D}_{\tilde{y}}$, thereby finding a desired $M := X^\dagger S^\dagger SY$. We can compute entries of M with only r queries each to X and Y, so all we need is to get $\mathrm{SQ}(\tilde{M})$ for $\tilde{M}$ the appropriate bound. We choose $|\tilde{M}(i,j)|^2 := r\sum_{\ell=1}^{r} |[S\tilde{X}](\ell,i)^\dagger[S\tilde{Y}](\ell,j)|^2$; showing that we have $\mathrm{SQ}(M)$ follows from the proofs of Lemmas 4.8 and 4.9, since M is simply a linear combination of outer products of rows of $\tilde{X}$ with rows of $\tilde{Y}$. Finally, this bound has the appropriate norm. Notating the rows sampled by the sketch as $s_1, \dots, s_r$, we have

$$\begin{aligned}\|\tilde{M}\|_{\mathrm{F}}^2 = r\sum_{\ell=1}^{r} \|[S\tilde{X}](\ell,\cdot)\|^2\|[S\tilde{Y}](\ell,\cdot)\|^2 &= r\sum_{\ell=1}^{r} \frac{\|\tilde{X}(s_\ell,\cdot)\|^2\|\tilde{Y}(s_\ell,\cdot)\|^2}{r^2(\frac{\|\tilde{X}(s_\ell,\cdot)\|^2}{2\|\tilde{X}\|_{\mathrm{F}}^2} + \frac{\|\tilde{Y}(s_\ell,\cdot)\|^2}{2\|\tilde{Y}\|_{\mathrm{F}}^2})^2} \\ &\le \sum_{\ell=1}^{r} \frac{\|\tilde{X}(s_\ell,\cdot)\|^2\|\tilde{Y}(s_\ell,\cdot)\|^2}{r(\frac{\|\tilde{X}(s_\ell,\cdot)\|\|\tilde{Y}(s_\ell,\cdot)\|}{\|\tilde{X}\|_{\mathrm{F}}\|\tilde{Y}\|_{\mathrm{F}}})^2} = \|\tilde{X}\|_{\mathrm{F}}^2\|\tilde{Y}\|_{\mathrm{F}}^2 = \phi_1\phi_2\|X\|_{\mathrm{F}}^2\|Y\|_{\mathrm{F}}^2.\end{aligned}$$

This argument more generally shows that we have $\mathrm{SQ}_\phi(X^\dagger S^\dagger SY)$ for the same sketch but with arbitrary size r (with query times dependent on r). Thus, later in this section, when we prove tighter approximation bounds using this sketch with smaller r, we can conclude that we also get SQ_ϕ to those matrix products.

If $X = Y$, we can get an improved spectral norm bound: instead of depending on $\|X\|_{\mathrm{F}}^2$, error depends on $\|X\|\|X\|_{\mathrm{F}}$.

Lemma 5.9 (Approximating matrix multiplication to spectral norm error [RV07, Theorem 3.1]). Suppose we are given $A \in \mathbb{R}^{m\times n}, \varepsilon > 0, \delta \in [0,1]$, and $S \in \mathbb{R}^{r\times n}$ a ϕ-oversampled importance sampling sketch of A. Then

$$\Pr\left[\|A^\dagger S^\dagger SA - A^\dagger A\| \lesssim \sqrt{\frac{\phi^2 \log r \log 1/\delta}{r}}\|A\|\|A\|_{\mathrm{F}}\right] > 1-\delta.$$

We also prove an asymmetric version of this result.

Theorem 5.10 (Asymmetric Approximate Matrix Multiplication). *Given matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times d}$, let $S \in \mathbb{C}^{s\times n}$ be sampled according to $p \in \mathbb{R}^n_{\ge 0}$ with $p_i \ge \frac{1}{2\phi}(\frac{\|A(\cdot,i)\|^2}{\|A\|_{\mathrm{F}}^2} + \frac{\|B(\cdot,i)\|^2}{\|B\|_{\mathrm{F}}^2})$ for* *some $\phi \geq 1$. Let* $\mathrm{sr} = \frac{\|A\|_F^2}{\|A\|^2} + \frac{\|B\|_F^2}{\|B\|^2}$*. Then, with probability at least* $1 - \delta > 0.75$*,*

$$\|ASS^\dagger B^\dagger - AB^\dagger\| \leq \sqrt{\frac{2}{s}\log\Big(\frac{\mathrm{sr}}{\delta}\Big)\phi(\|A\|_F^2\|B\|^2 + \|A\|^2\|B\|_F^2)} + \frac{1}{s}\log\Big(\frac{\mathrm{sr}}{\delta}\Big)\phi\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}.$$

We will use the following consequence of this theorem.

Corollary 5.11. *Given matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times d}$, let $S \in \mathbb{C}^{s\times n}$ be sampled according to $p \in \mathbb{R}^n_{\geq 0}$ with $p_i \geq \frac{1}{2\phi}(\frac{\|A(\cdot,i)\|^2}{\|A\|_{\mathrm{F}}^2} + \frac{\|B(\cdot,i)\|^2}{\|B\|_{\mathrm{F}}^2})$ for some $\phi \geq 1$. For $\varepsilon \in (0,1]$ and $\delta \in (0, 0.25]$, when $s = \frac{4\phi}{\varepsilon^2}(\frac{\|A\|_{\mathrm{F}}^2}{\|A\|^2} + \frac{\|B\|_{\mathrm{F}}^2}{\|B\|^2})\log(\frac{1}{\delta}(\frac{\|A\|_{\mathrm{F}}^2}{\|A\|^2} + \frac{\|B\|_{\mathrm{F}}^2}{\|B\|^2}))$, then $\|ASS^\dagger B^\dagger - AB^\dagger\| \leq \varepsilon\|A\|\|B\|$ with probability $\geq 1-\delta$.*

The symmetric version of this result was previously stated in [KV17; RV07], and this asymmetric version was stated in [MZ11]. However, the final theorem statement was weaker, so we reprove it here. To obtain it, we prove the following key lemma:

Lemma 5.12 (Concentration of Asymmetric Random Outer Products). Let $\{(X_i, Y_i)\}_{i\in[s]}$ be s independent copies of the tuple of random vectors (X, Y), with $X \in \mathbb{C}^m$ and $Y \in \mathbb{C}^d$. In particular, $(X, Y) = (a^{(i)}, b^{(i)})$ with probability p_i for $i \in [n]$. Let $M, L \geq 0$ be such that

$$\begin{aligned}
L &\geq \max_{i\in[n]}\|a^{(i)}(b^{(i)})^\dagger\| \\
M^2 &\geq \max_{i\in[n]}\|b^{(i)}\|^2\|\boldsymbol{E}[XX^\dagger]\| + \max_{i\in[n]}\|a^{(i)}\|^2\|\boldsymbol{E}[YY^\dagger]\|, \\
\mathrm{sr} &\geq \frac{\max_{i\in[n]}\|b^{(i)}\|^2\,\boldsymbol{E}[\|X\|_{\mathrm{F}}^2] + \max_{i\in[n]}\|a^{(i)}\|^2\,\boldsymbol{E}[\|Y\|_{\mathrm{F}}^2]}{\max(\max_{i\in[n]}\|b^{(i)}\|^2\|\boldsymbol{E}[XX^\dagger]\|, \max_{i\in[n]}\|a^{(i)}\|^2\|\boldsymbol{E}[YY^\dagger]\|)}
\end{aligned}$$

Then, for any $t \geq M/\sqrt{s} + 2L/(3s)$,

$$\Pr\Big[\Big\|\frac{1}{s}\sum_{i\in[s]} X_iY_i^\dagger - \boldsymbol{E}[XY^\dagger]\Big\| \geq t\Big] \leq 4\,\mathrm{sr}\exp\Big(-\frac{st^2}{2(M^2 + Lt)}\Big).$$

Proof. For $i \in [s]$, let $Z_i = \frac{1}{s}(X_iY_i^\dagger - \boldsymbol{E}[XY^\dagger])$. Then $\|Z_i\| \leq \frac{2}{s}\|X_iY_i^\dagger\| \leq \frac{2L}{s}$. Next, we bound the variance:

$$\sigma^2 := \max\Bigg(\underbrace{\Big\|\sum_{i\in[n]}\boldsymbol{E}[Z_iZ_i^\dagger]\Big\|}_{(i)}, \underbrace{\Big\|\sum_{i\in[n]}\boldsymbol{E}[Z_i^\dagger Z_i]\Big\|}_{(ii)}\Bigg)$$

We can observe that

$$\begin{aligned}
\sum_{i\in[s]} \mathbf{E}[Z_iZ_i^\dagger] &= \frac{1}{s}\mathbf{E}\Big[\Big(X_iY_i^\dagger - \mathbf{E}[XY^\dagger]\Big)\Big(X_iY_i^\dagger - \mathbf{E}[XY^\dagger]\Big)^\dagger\Big] \\
&= \frac{1}{s}\mathbf{E}\Big[\|Y_i\|^2 X_iX_i^\dagger - \mathbf{E}[XY^\dagger]\,\mathbf{E}[YX^\dagger]\Big] \\
&\preceq \frac{1}{s}(\max_{i\in[n]}\|b^{(i)}\|^2)\,\mathbf{E}[XX^\dagger] =: V_1 \\
\sum_{i\in[s]} \mathbf{E}[Z_i^\dagger Z_i] &= \frac{1}{s}\mathbf{E}\Big[\Big(X_iY_i^\dagger - \mathbf{E}[XY^\dagger]\Big)^\dagger\Big(X_iY_i^\dagger - \mathbf{E}[XY^\dagger]\Big)\Big] \\
&= \frac{1}{s}\mathbf{E}\Big[\|X_i\|^2 Y_iY_i^\dagger - \mathbf{E}[YX^\dagger]\,\mathbf{E}[XY^\dagger]\Big] \\
&\preceq \frac{1}{s}(\max_{i\in[n]}\|a^{(i)}\|^2)\,\mathbf{E}[YY^\dagger] =: V_2
\end{aligned}$$

We can use this to bound term (i):

$$\Big\|\sum_{i\in[s]} \mathbf{E}[Z_iZ_i^\dagger]\Big\| \leq \frac{1}{s}\|\mathbf{E}[\|Y_i\|^2X_iX_i^\dagger]\| \leq \frac{1}{s}(\max_{i\in[n]}\|b^{(i)}\|^2)\|\mathbf{E}[XX^\dagger]\|$$

We bound term (ii) as follows:

$$\Big\|\sum_{i\in[s]} \mathbf{E}[Z_i^\dagger Z_i]\Big\| \leq \frac{1}{s}\|\mathbf{E}[\|X_i\|^2Y_iY_i^\dagger]\| \leq \frac{1}{s}(\max_{i\in[n]}\|a^{(i)}\|^2)\|\mathbf{E}[YY^\dagger]\|$$

Altogether, we have shown that $\sigma^2 \leq M^2/s$. Applying Matrix Bernstein (see Fact 5.13) with upper bounds of V_1 and V_2 and parameters $L \leftarrow \frac{2L}{s}$ and $v \leftarrow M^2/s$, we get

$$\begin{aligned}
\Pr\Big[\Big\|\frac{1}{s}\sum_{i\in[s]} X_iY_i^\dagger - \mathbf{E}[XY^\dagger]\Big\| \geq t\Big] &= \Pr\Big[\Big\|\sum_{i\in[s]} Z_i\Big\| \geq t\Big] \\
&\leq 4\,\mathrm{sr}\exp\Big(-\frac{t^2/2}{M^2/s + 2Lt/(3s)}\Big) \leq 4\,\mathrm{sr}\exp\Big(-\frac{st^2}{2(M^2+Lt)}\Big),
\end{aligned}$$

where

$$\mathrm{sr} = \frac{\mathrm{tr}(V_1)+\mathrm{tr}(V_2)}{\max(\|V_1\|,\|V_2\|)} = \frac{\max_{i\in[n]}\|b^{(i)}\|^2\,\mathbf{E}[\|X\|_{\mathrm{F}}^2] + \max_{i\in[n]}\|a^{(i)}\|^2\,\mathbf{E}[\|Y\|_{\mathrm{F}}^2]}{\max(\max_{i\in[n]}\|b^{(i)}\|^2\|\mathbf{E}[XX^\dagger]\|, \max_{i\in[n]}\|a^{(i)}\|^2\|\mathbf{E}[YY^\dagger]\|)}$$ □

Fact 5.13 (Intrinsic Matrix Bernstein, [Tro15, Theorem 7.3.1]). *Consider a finite sequence $\{Z_k\}_{k\in[s]}$ of random complex matrices with the same size, and assume that $\mathbf{E}[Z_k] = 0$ and $\|Z_k\| \leq L$. Let V_1 and V_2 be semidefinite upper bounds for the corresponding matrix-valued variances:*

$$V_1 \succeq \mathbf{E}[\sum_{k=1}^{s} Z_k Z_k^\dagger] \qquad\qquad V_2 \succeq \mathbf{E}[\sum_{k=1}^{s} Z_k^\dagger Z_k]$$

Define an intrinsic dimension bound and a variance bound,

$$\mathrm{sr} = \frac{\mathrm{tr}(V_1 + V_2)}{\max(\|V_1\|, \|V_2\|)} \qquad\qquad v = \max\{\|V_1\|, \|V_2\|\}$$

Then, for $t \geq \sqrt{v} + L/3$,

$$\Pr\left[\Big\|\sum_{i\in[k]} Z_i\Big\| \geq t\right] \leq 4\,\mathrm{sr}\exp\Big(-\frac{t^2/2}{v + Lt/3}\Big).$$

It is now straight-forward to prove Theorem 5.10 using the aforementioned lemma:

Proof of Theorem 5.10. We apply Lemma 5.12 with $a^{(i)} = \sqrt{1/p_i}\cdot A(\cdot,i)$ and $b^{(i)} = \sqrt{1/p_i}\cdot B(\cdot,i)$. As assumed the sampling distribution p_i satisfies $p_i \geq \frac{1}{2\phi}(\frac{\|A(\cdot,i)\|^2}{\|A\|_\mathrm{F}^2} + \frac{\|B(\cdot,i)\|^2}{\|B\|_\mathrm{F}^2}) \geq \frac{\|A(\cdot,i)\|\|B(\cdot,i)\|}{\phi\|A\|_\mathrm{F}\|B\|_\mathrm{F}}$, so

$$\begin{aligned}
\|a^{(i)}\| &= \|A(\cdot,i)\|/\sqrt{p_i} \leq \|A(\cdot,i)\|\sqrt{\frac{2\phi\|A\|_\mathrm{F}^2}{\|A(\cdot,i)\|^2}} = \sqrt{2\phi}\|A\|_\mathrm{F}\\
\|b^{(i)}\| &= \|B(\cdot,i)\|/\sqrt{p_i} \leq \|B(\cdot,i)\|\sqrt{\frac{2\phi\|B\|_\mathrm{F}^2}{\|B(\cdot,i)\|^2}} = \sqrt{2\phi}\|B\|_\mathrm{F}\\
\|a^{(i)}(b^{(i)})^\dagger\| &= \frac{\|A(\cdot,i)\|\|B(\cdot,i)\|}{p_i} \leq \phi\|A\|_\mathrm{F}\|B\|_\mathrm{F}
\end{aligned}$$

Further,

$$\begin{aligned}
\|\mathbf{E}[XX^\dagger]\| &= \Big\|\sum_{i\in[n]} A(\cdot,i)A(\cdot,i)^\dagger\Big\| = \|AA^\dagger\| = \|A\|^2\\
\|\mathbf{E}[YY^\dagger]\| &= \Big\|\sum_{i\in[n]} B(\cdot,i)B(\cdot,i)\Big\| = \|BB^\dagger\| = \|B\|^2
\end{aligned}$$

Finally,

$$\begin{aligned}\frac{\max_{i\in[n]}\|b^{(i)}\|^2\,\boldsymbol{E}\,[\|X\|_{\mathrm{F}}^2]+\max_{i\in[n]}\|a^{(i)}\|^2\,\boldsymbol{E}\,[\|Y\|_{\mathrm{F}}^2]}{\max(\max_{i\in[n]}\|b^{(i)}\|^2\|\boldsymbol{E}\,[XX^\dagger]\|,\max_{i\in[n]}\|a^{(i)}\|^2\|\boldsymbol{E}\,[YY^\dagger]\|)} &\leq \frac{\boldsymbol{E}\,[\|X\|_{\mathrm{F}}^2]}{\|\boldsymbol{E}\,[XX^\dagger]\|}+\frac{\boldsymbol{E}\,[\|Y\|_{\mathrm{F}}^2]}{\|\boldsymbol{E}\,[YY^\dagger]\|}\\ &= \frac{\|A\|_{\mathrm{F}}^2}{\|A\|^2}+\frac{\|B\|_{\mathrm{F}}^2}{\|B\|^2}\end{aligned}$$

So, in Lemma 5.12, we can set $L=\phi\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}$, $M^2=2\phi\|A\|_{\mathrm{F}}^2\|B\|^2+2\phi\|B\|_{\mathrm{F}}^2\|A\|^2$, and $\mathrm{sr}=\frac{\|A\|_{\mathrm{F}}^2}{\|A\|^2}+\frac{\|B\|_{\mathrm{F}}^2}{\|B\|^2}$ to get that, for all $t\geq M/\sqrt{s}+2L/(3s)$,

$$\begin{aligned}&\Pr\left[\Big\|\frac{1}{s}\sum_{i\in[s]}X_iY_i^\dagger-\boldsymbol{E}\,[XY^\dagger]\Big\|\geq t\right]\\ &\leq 4\Big(\frac{\|A\|_{\mathrm{F}}^2}{\|A\|}+\frac{\|B\|_{\mathrm{F}}^2}{\|B\|}\Big)\exp\Big(\frac{-st^2}{2\phi(\|A\|_{\mathrm{F}}^2\|B\|^2+\|A\|^2\|B\|_{\mathrm{F}}^2)+\phi\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}t}\Big).\end{aligned}$$

To get the right-hand side of the above equation to be $\leq\delta$, it suffices to choose

$$t=\sqrt{\frac{2}{s}\log\Big(\frac{\mathrm{sr}}{\delta}\Big)\phi(\|A\|_{\mathrm{F}}^2\|B\|^2+\|A\|^2\|B\|_{\mathrm{F}}^2)}+\frac{1}{s}\log\Big(\frac{\mathrm{sr}}{\delta}\Big)\phi\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}.$$

This choice of t is greater than $M/\sqrt{s}+2L/(3s)$ when $\delta<1/e$, so with this assumption, we can conclude that with probability $\geq 1-\delta$,

$$\|ASS^\dagger B^\dagger-AB^\dagger\|\leq\sqrt{\frac{2}{s}\log\Big(\frac{\mathrm{sr}}{\delta}\Big)\phi(\|A\|_F^2\|B\|^2+\|A\|^2\|B\|_F^2)}+\frac{1}{s}\log\Big(\frac{\mathrm{sr}}{\delta}\Big)\phi\|A\|_{\mathrm{F}}\|B\|_{\mathrm{F}}. \qquad \square$$

The above results can be used to approximate singular values, simply by directly translating the bounds on matrix product error to bounds on singular value error.

Lemma 5.14 (Approximating singular values). Given $\mathrm{SQ}_\phi(A)\in\mathbb{C}^{m\times n}$ and $\varepsilon\in(0,1]$, we can form importance sampling sketches $S\in\mathbb{R}^{r\times m}$ and $T^\dagger\in\mathbb{R}^{c\times n}$ in $\mathcal{O}((r+c)\,\boldsymbol{sq}_\phi(A))$ time satisfying the following property. Take $r,c\geq s$ for some sufficiently large $s=\widetilde{\mathcal{O}}(\frac{\phi^2}{\varepsilon^2}\log\frac{1}{\delta})$. Then, if σ_i and $\hat{\sigma}_i$ are the singular values of A and SAT, respectively (where $\hat{\sigma}_i=0$ for $i>\min(r,c)$), we have with probability $\geq 1 - \delta$ that

$$\sqrt{\sum_{i=1}^{\min(m,n)} (\hat{\sigma}_i^2 - \sigma_i^2)^2} \leq \varepsilon \|A\|_F^2.$$

If we additionally assume that $\varepsilon \lesssim \|A\|/\|A\|_F$, we can conclude $|\sigma_i^2 - \hat{\sigma}_i^2| \leq \varepsilon \|A\| \|A\|_F$ for all i.

This result follows from results bounding the error between singular values by errors of matrix products. For notation, let $\sigma_i(M)$ be the ith largest singular value of M. We will use the following inequalities relating norm error of matrices to error in their singular values:

Lemma 5.15 (Hoffman-Wielandt inequality [KV17, Lemma 2.7]). For symmetric $X, Y \in \mathbb{R}^{n \times n}$,

$$\sum |\sigma_i(X) - \sigma_i(Y)|^2 \leq \|X - Y\|_F^2.$$

Lemma 5.16 (Weyl's inequality [Bha97, Corollary III.2.2]). For $A, B \in \mathbb{C}^{m \times n}$, $|\sigma_k(A) - \sigma_k(B)| \leq \|A - B\|$. When A, B are Hermitian, the same bound holds for their eigenvalues.[14]

Proof of Lemma 5.14. We use known theorems, plugging in the values of r and c. By Lemma 5.7 for the sketch S, we know that

$$\Pr\left[\|A^\dagger S^\dagger S A - A^\dagger A\|_F \leq \frac{\varepsilon}{2}\|A\|_F^2\right] \geq 1 - \delta;$$

by Lemma 5.3, $T^\dagger$ is an $\leq 2\phi$-oversampled importance sampling sketch of $(SA)^\dagger$, so using Lemma 5.7 for $T^\dagger$,

$$\Pr\left[\|SATT^\dagger A^\dagger S^\dagger - SAA^\dagger S^\dagger\|_F \leq \frac{\varepsilon}{4}\|SA\|_F^2\right] \geq 1 - \delta,$$

and from Lemma 5.2,

$$\Pr\left[\|SA\|_F^2 \leq 2\|A\|_F^2\right] \geq 1 - \delta.$$

By rescaling δ and union bounding, we can have all events happen with probability $\geq 1 - \delta$.

[14] [Bha97, Corollary III.2.2] actually proves the Hermitian version. The result about singular values is an easy consequence, see for example the blog of Terence Tao [Tao10, Exercise 22(iv)].

Then, from triangle inequality followed by Lemma 5.15,

$$\begin{aligned}\sqrt{\sum|\sigma_i(SAT)^2-\sigma_i(A)^2|^2} &\leq \sqrt{\sum|\sigma_i(SAT)^2-\sigma_i(SA)^2|^2}+\sqrt{\sum|\sigma_i(SA)^2-\sigma_i(A)^2|^2}\\ &\leq \|(SAT)(SAT)^\dagger-(SA)(SA)^\dagger\|_{\mathrm{F}}+\|(SA)^\dagger(SA)-A^\dagger A\|_{\mathrm{F}}\\ &\leq \varepsilon\|A\|_{\mathrm{F}}^2.\end{aligned}$$

The analogous result holds for spectral norm via Lemma 5.9 and Lemma 5.16; the only additional complication is that we need to assert that $\|SA\| \lesssim \|A\|$. We use the following argument, using the upper bound on ε:

$$\|SA\|^2=\|A^\dagger S^\dagger SA\|\leq\|A^\dagger S^\dagger SA-A^\dagger A\|+\|A^\dagger A\|\leq\|A\|^2+\varepsilon\|A\|\|A\|_{\mathrm{F}}\lesssim\|A\|^2. \qquad \square$$

Finally, if we wish to approximate a vector inner product $u^\dagger v$, a special case of matrix product, we can do so with only sampling and query access to one of the vectors while still getting $\log\frac{1}{\delta}$ dependence on failure probability.

Lemma 5.17 (Inner product estimation, [Tan19, Proposition 4.2]). Given $\mathrm{SQ}_\phi(u), \mathrm{Q}(v) \in \mathbb{C}^n$, we can output an estimate $c \in \mathbb{C}$ such that $|c - \langle u, v\rangle| \leq \varepsilon$ with probability $\geq 1-\delta$ in time $\mathcal{O}(\phi\|u\|^2\|v\|^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}(\boldsymbol{sq}_\phi(u)+\boldsymbol{q}(v)))$.

Proof. Define a random variable Z by sampling an index from the distribution p given by $\mathrm{SQ}_\phi(u)$, and setting $Z := u(i)v(i)/p(i)$. Then

$$\boldsymbol{E}[Z]=\langle u,v\rangle \quad\text{and}\quad \boldsymbol{E}[|Z|^2]=\sum_{i=1}^n p(i)\frac{|u(i)v(i)|^2}{p(i)^2}\leq\sum_{i=1}^n|u(i)v(i)|^2\frac{\phi\|u\|^2}{|u(i)|^2}=\phi\|u\|^2\|v\|^2.$$

So, we just need to boost the quality of this random variable. Consider taking $\bar{Z}$ to be the mean of $x := 8\phi\|u\|^2\|v\|^2\frac{1}{\varepsilon^2}$ independent copies of Z. Then, by Chebyshev's inequality (stated here for complex-valued random variables),

$$\Pr[|\bar{Z}-\boldsymbol{E}[\bar{Z}]|\geq\varepsilon/\sqrt{2}]\leq\frac{2\,\boldsymbol{Var}[Z]}{x\varepsilon^2}\leq\frac{1}{4}.$$

Next, we take the (component-wise) median of $y := 8\log\frac{1}{\delta}$ independent copies of $\bar{Z}$, which we call $\tilde{Z}$, to decrease failure probability. Consider the median of the real parts of $\bar{Z}$. The key observation is that if $\Re(\tilde{Z} - \boldsymbol{E}[Z]) \geq \varepsilon/\sqrt{2}$, then at least half of the $\bar{Z}$'s satisfy $\Re(\bar{Z} - \boldsymbol{E}[Z]) \geq \varepsilon/\sqrt{2}$. Let $E_i = \chi(\Re(\bar{Z}_i - \boldsymbol{E}[Z]) \geq \varepsilon/\sqrt{2})$ be the characteristic function for this event for a particular mean. The above argument implies that $\Pr[E_i] \leq \frac{1}{4}$. So, by Hoeffding's inequality,

$$\Pr\left[\frac{1}{q}\sum_{i=1}^{q} E_i \geq \frac{1}{2}\right] \leq \Pr\left[\frac{1}{q}\sum_{i=1}^{q} E_i \geq \frac{1}{4} + \Pr[E_i]\right] \leq \exp(-q/8) \leq \frac{\delta}{2}.$$

With this combined with our key observation, we can conclude that $\Pr[\Re(\tilde{Z} - \langle u, v\rangle) \geq \varepsilon/\sqrt{2}] \leq \delta/2$. From a union bound together with the analogous argument for the imaginary component, we have $\Pr[|\tilde{Z} - \langle u, v\rangle| \geq \varepsilon] \leq \delta$ as desired. The time complexity is the number of samples multiplied by the time to create one instance of the random variable Z, which is $\mathcal{O}(\boldsymbol{sq}(u) + \boldsymbol{q}(v))$. $\square$

Remark 5.18. Lemma 5.17 also applies to higher-order tensor inner products:

(a) (Trace inner products, [GLT18, Lemma 11]) Given $\mathrm{SQ}_\phi(A) \in \mathbb{C}^{n\times n}$ and $\mathrm{Q}(B) \in \mathbb{C}^{n\times n}$, we can estimate $\mathrm{tr}[AB^\dagger]$ to additive error ε with probability at least $1-\delta$ by using

$$\mathcal{O}(\phi\frac{\|A\|_{\mathrm{F}}^2\|B\|_{\mathrm{F}}^2}{\varepsilon^2}\big(\boldsymbol{sq}_\phi(A) + \boldsymbol{q}(B)\big)\log\frac{1}{\delta})$$

time. To do this, note that $\mathrm{SQ}_\phi(A)$ and $\mathrm{Q}(B)$ imply $\mathrm{SQ}_\phi(\mathrm{vec}(A))$ and $\mathrm{Q}(\mathrm{vec}(B))$. $\mathrm{tr}[AB] = \langle \mathrm{vec}(B), \mathrm{vec}(A)\rangle$, so we can just apply Lemma 5.17 to conclude.

(b) (Expectation values) Given $\mathrm{SQ}_\phi(A) \in \mathbb{C}^{n\times n}$ and $\mathrm{Q}(x), \mathrm{Q}(y) \in \mathbb{C}^n$, we can estimate $x^\dagger Ay$ to additive error ε with probability at least $1-\delta$ in

$$\mathcal{O}(\phi\frac{\|A\|_{\mathrm{F}}^2\|x\|^2\|y\|^2}{\varepsilon^2}\big(\boldsymbol{sq}_\phi(A) + \boldsymbol{q}(x) + \boldsymbol{q}(y)\big)\log\frac{1}{\delta})$$

time. To do this, observe that $x^\dagger Ay = \mathrm{tr}(x^\dagger Ay) = \mathrm{tr}(Ayx^\dagger)$ and that $\mathrm{Q}(yx^\dagger)$ can be simulated with $\mathrm{Q}(x), \mathrm{Q}(y)$. So, we just apply the trace inner product procedure.

We will also apply the inner product sketch to matrices in order to sparsify it.

Definition 5.19 (Bi-linear Entry-wise Sparsifying Transform). For a matrix $A \in \mathbb{C}^{m\times n}$, the

BEST of A with parameter T is a matrix sampled as follows: for all $k \in [T]$,

$$M^{(k)} = \frac{1}{p_{i,j}} A(i,j) e_i e_j^\dagger \qquad \text{with probability } p_{i,j} = \frac{|A(i,j)|^2}{\|A\|_F^2}$$

Then,

$$\textsc{best}_T(A) = \frac{1}{T} \sum_{k \in [T]} M^{(k)}.$$

Lemma 5.20 (Basic properties of the Bi-Linear Entry-wise Sparsifying Transform). For a matrix $A \in \mathbb{C}^{m \times n}$, let $M = \textsc{best}(A)$ with parameter T. Then, for $X \in \mathbb{C}^{m \times m}$, $u \in \mathbb{C}^m$ and $v \in \mathbb{C}^n$,

$$\operatorname{nnz}(M) \leq T \tag{11}$$

$$\boldsymbol{E}[M] = A \tag{12}$$

$$\boldsymbol{E}\left[M^\dagger X M - A^\dagger X A\right] = \frac{1}{T}\left(\operatorname{tr}(X)\|A\|_F^2 I - A^\dagger X A\right) \tag{13}$$

Proof. Observe, since M is an average of T sub-samples, each of which are 1-sparse, M has at most T non-zero entries. Next,

$$\boldsymbol{E}[M] = \frac{1}{T} \sum_{k \in T} \boldsymbol{E}[M^{(k)}] = \sum_{i \in [m]} \sum_{j \in [n]} p_{i,j} \frac{A(i,j)}{p_{i,j}} e_i e_j^\dagger = A$$

Similarly,

$$\begin{aligned}
\boldsymbol{E}\left[M^\dagger X M\right] &= \frac{1}{T^2} \boldsymbol{E}\left[\Big(\sum_{k \in [T]} M^{(k)}\Big)^\dagger X \Big(\sum_{k \in [T]} M^{(k)}\Big)\right] \\
&= \frac{1}{T^2} \boldsymbol{E}\left[\Big(\sum_{k,k' \in [T]} (M^{(k)})^\dagger X M^{(k')}\Big)\right] \\
&= \frac{1}{T^2}\Big(\Big(\sum_{k \neq k' \in [T]} \boldsymbol{E}[M^{(k)}]^\dagger \cdot X \cdot \boldsymbol{E}[M^{(k')}]\Big) + \Big(\sum_{k \in [T]} \boldsymbol{E}[(M^{(k)})^\dagger X M^{(k)}]\Big)\Big) \\
&= \Big(1 - \frac{1}{T}\Big) A^\dagger X A + \frac{1}{T} \sum_{i \in [m], j \in [n]} p_{i,j} \frac{A(i,j)^2}{p_{i,j}^2} e_j e_i^\dagger X e_i e_j^\dagger \\
&= \Big(1 - \frac{1}{T}\Big) A^\dagger X A + \frac{\|A\|_F^2}{T} \sum_{i \in [m], j \in [n]} X(i,i) e_j e_j^\dagger \\
&= \Big(1 - \frac{1}{T}\Big) A^\dagger X A + \frac{\|A\|_F^2 \operatorname{tr}(X)}{T} I.
\end{aligned}$$

□

We list a simple consequence of these bounds that we use later.

Corollary 5.21. *For a matrix $A \in \mathbb{C}^{m\times n}$, let $M = \textsc{best}(A)$ with parameter T. Then, for matrices $X \in \mathbb{C}^{\ell\times m}$ and $Y \in \mathbb{C}^{n\times d}$,*

$$\Pr\Big[\|XMY - XAY\|_F \geq \frac{\|X\|_F\|A\|_F\|Y\|_F}{\sqrt{\delta T}}\Big] \leq \delta$$

Finally, we observe a simple technique to convert importance sampling sketches into approximate isometries, by inserting the appropriate pseudoinverse. This will be used in some of the more involved applications.

Lemma 5.22. Given $A \in \mathbb{C}^{m\times n}$, $S \in \mathbb{C}^{r\times m}$ sampled from a ϕ-oversampled importance sampling distribution of A, and $T^\dagger \in \mathbb{C}^{n\times c}$ sampled from an $\leq \phi$-oversampled importance sampling distribution of $(SA)^\dagger$, let $R := SA$ and $C := SAT$. Let σ_k be the kth singular value of A. If, for $\alpha \in (0,1]$, $r = \tilde{\Omega}(\frac{\phi^2\|A\|^2\|A\|_F^2}{\sigma_k^4}\log\frac{1}{\delta})$ and $c = \tilde{\Omega}(\frac{\phi^2\|A\|^2\|A\|_F^2}{\sigma_k^4\alpha^2}\log\frac{1}{\delta})$, then with probability $\geq 1-\delta$, $((C_k)^+R)^\dagger$ is an α-approximate projective isometry onto the image of $(C_k)^+$. Further, $(DV^\dagger R)^\dagger$ is an α-approximate isometry, where $C_k^+ = UDV^\dagger$ is a singular value decomposition truncated so that $D \in \mathbb{R}^{k'\times k'}$ is full rank (so $k' \leq \min(k, \operatorname{rank}(A))$).

Proof. The following occurs with probability $\geq 1-\delta$. By Lemma 5.14, $\|C_k^+\| \lesssim \frac{1}{\sigma_k^2}$. By Lemma 5.9, $\|R^\dagger R - A^\dagger A\| \lesssim \|A\|^2$, which implies that $\|R\| \lesssim \|A\|$, and by Lemma 5.2, $\|R\|_F \lesssim \|A\|_F$. Further, $\|RR^\dagger - CC^\dagger\| \leq \alpha\sigma_k^2\frac{\|R\|\|R\|_F}{\|A\|\|A\|_F} \lesssim \alpha\sigma_k^2$. Finally, $C_k^+C = C_k^+C_k$ is an orthogonal projector. So, with probability $\geq 1-\delta$,

$$\|(C_k^+R)(C_k^+R)^\dagger - (C_k^+C)(C_k^+C)^\dagger\| = \|C_k^+(RR^\dagger - CC^\dagger)(C_k^+)^\dagger\| \leq \|C_k^+\|^2\|RR^\dagger - CC^\dagger\| = O(\alpha).$$

We get the computation for the α-approximate isometry by restricting attention to the span of U:

$$\|(DV^\dagger R)(DV^\dagger R)^\dagger - I\| = \|DV^\dagger(RR^\dagger - CC^\dagger)VD^\dagger\| \leq \|UDV^\dagger\|^2\|RR^\dagger - CC^\dagger\| = O(\alpha). \qquad \square$$

One can also observe that, for a sufficiently good sketch C, $R \approx C_k(C_k)^+R$ in spectral norm, giving a generic way to approximate a sketch R by a product of a small matrix with an approximate projective isometry. We do not need it in our proofs, so this computation is not included.

6 Dequantizing the quantum singular value transformation

We begin our exploration of dequantizing algorithms by dequantizing the quantum singular value transformation described by Gilyén, Su, Low, and Wiebe [GSLW19] for close-to-low-rank matrices. Our goal is to prove the following theorem:

Theorem 6.1. *Suppose we are given sampling and query access to $A \in \mathbb{C}^{m\times n}$ and $b \in \mathbb{C}^n$ with $\|A\| \leq 1$; a an even or odd degree-d polynomial p with $p(0) = 0$, given as its Chebyshev coefficients; and a sufficiently small accuracy parameter $\varepsilon > 0$. Then we can output a description of a vector y (in $\mathbb{C}^m$ if p is odd, in $\mathbb{C}^n$ if p is even) such that $\|y - p(A)b\| \leq \varepsilon\|p\|_{\sup}\|b\|$ with probability ≥ 0.9 in time*

$$\mathcal{O}\left(\min\left\{\operatorname{nnz}(A), \frac{\|A\|_F^4}{\varepsilon^4}d^{12}\log^8(d)\log^2\frac{\|A\|_\mathrm{F}}{\|A\|}\right\} + \frac{\|A\|_F^4}{\varepsilon^2}d^{11}\log^4(d)\log\frac{\|A\|_\mathrm{F}}{\|A\|}\right).$$

We can get $\mathrm{SQ}_\phi(y)$ such that we can access the output description in the following way:

(i) Compute entries of y in $\mathcal{O}\big(\frac{\|A\|_\mathrm{F}^2}{\varepsilon^2}d^6\log^4(d)\log\frac{\|A\|_\mathrm{F}}{\|A\|}\big)$ time;

(ii) Sample $i \in [n]$ with probability $\frac{|y_i|^2}{\|y\|^2}$ in $\mathcal{O}\big(\frac{\|p\|_{\sup}^2\|A\|_\mathrm{F}^4\|b\|^2}{\varepsilon^2\|y\|^2}d^8\log^8(d)\log\frac{\|A\|_\mathrm{F}}{\|A\|}\big)$ time with probability ≥ 0.9;

(iii) Estimate $\|y\|^2$ to ν relative error in $\mathcal{O}\big(\frac{\|p\|_{\sup}^2\|A\|_\mathrm{F}^4\|b\|^2}{\nu^2\varepsilon^2\|y\|^2}d^8\log^8(d)\log\frac{\|A\|_\mathrm{F}}{\|A\|}\big)$ time with probability ≥ 0.9.

From this result it follows that QSVT, as described in [GSLW19, Theorem 17], has no exponential speedup when the block-encoding of A comes from a quantum-accessible “QRAM” data structure as in [GSLW19, 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 $\operatorname{polylog}(mn)$ time. Then one can apply (quantum) SVT by a degree-d polynomial on A and apply the resulting map to $|b\rangle$ with $d \cdot \operatorname{polylog}(mn)$ gates and finally project down to get the state $|p(A)b\rangle$ with probability $\geq 1 - \delta$ after $\Theta(\frac{1}{\|p(A)b\|}\log\frac{1}{\delta})$ iterations of the circuit. So, getting a sample from

$|p(A)b\rangle$ takes $\Theta\big(d\frac{1}{\|p(A)b\|}\operatorname{polylog}(mn/\delta)\big)$ 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 4.12, having A and b in (Q)RAM implies having $\mathrm{SQ}(A)$ and $\mathrm{SQ}(b)$ with $\boldsymbol{sq}(A) = \mathcal{O}(\log mn)$ and $\boldsymbol{sq}(b) = \mathcal{O}(\log n)$. Since QSVT also needs to assume $\max_{x\in[-1,1]}|p(x)| \leq 1$, 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, $\frac{1}{\|p(A)b\|}$, and $\log(mn)$. We are exponentially slower in ε and δ (these errors are conflated for the quantum algorithm). However, this exponential advantage vanishes if the desired output is not a quantum state but some fixed value (or an estimate of one). In that case, the quantum algorithm must also pay $\frac{1}{\varepsilon}$ during the sampling or tomography procedures and the classical algorithm can boost a constant success probability to $\geq 1-\delta$, only paying a $\log\frac{1}{\delta}$ factor. 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 6.1 also dequantizes QSVT for block-encodings of density operators when the density operator comes from some well-structured classical data. Indeed, [GSLW19, Lemma 45] assumes we can efficiently prepare a purification of the density operator ρ. 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 $\operatorname{tr}(\rho) = 1$, we have $\|A\|_{\mathrm{F}} = 1$. Then, $p(\rho) = r(A)$ 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.

Remark 6.2. We can also compute estimate $\langle u|y\rangle$ to $\varepsilon\|u\|\|b\|$ error without worsening the $1/\varepsilon^2$ dependence. This does not follow from the above; rather, we can observe that, from the description of our output, $y = (AS)v + \eta b$, it suffices to estimate $u^\dagger(AS)v$ and $u^\dagger b$. Because of properties of the sketches and the later analysis, $\|AS\|_{\mathrm{F}} \lesssim \|A\|_{\mathrm{F}}$ and $\|v\| \lesssim d\log^2(d)\|b\|$, so by Lemma 5.17 and Remark 5.18 we can estimate these values to the desired error with $\mathcal{O}(d^2\log^4(d)\|A\|_{\mathrm{F}}^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ queries to u, v, A and samples to AS. All of these queries can be done in $\mathcal{O}(1)$ time.

Remark 6.3. Here, we make a brief remark about a technical detail we previously elided. Technically, QSVT can use $A^\dagger$ in QRAM instead of A (cf. [GSLW19, Lemma 50]), leaving open the possibility that there is a quantum algorithm that does not give an exponential speedup when A is in QRAM, but does when $A^\dagger$ is in QRAM. We sketch an argument why this is impossible by showing that, given $\mathrm{SQ}(A)$, we can simulate $\mathrm{SQ}_\phi(B)$ (and $\mathrm{SQ}_\phi(B^\dagger)$) for B such that $\|B - A^\dagger\| \leq \varepsilon\|A\|$ with probability $\geq 1 - \delta$.

Proof sketch of Remark 6.3. Recall that we wish to show that, given $\mathrm{SQ}(A)$, we can simulate $\mathrm{SQ}_\phi(B)$ for B such that $\|B - A^\dagger\| \leq \varepsilon\|A\|$ with probability $\geq 1 - \delta$.

Following the argument from Remark 8.6, we can find a $B \coloneqq AR^\dagger\bar{t}(CC^\dagger)R$ satisfying the above property in $\widetilde{\mathcal{O}}(\frac{\|A\|_F^{28}}{\|A\|^{28}\varepsilon^{22}}\log^3\frac{1}{\delta})$ (rescaling ε appropriately). Here, R and $\bar{t}(CC^\dagger)$ come from an application of Theorem 7.1 with $t : \mathbb{R} \to \mathbb{C}$ a smooth step function that goes from zero to one around $(\varepsilon\|A\|)^2$. If we had sampling and query access to the columns of $AR^\dagger$, we would be done, since then $B = \sum_{i=1}^r \sum_{j=1}^r [\bar{t}(CC^\dagger)](i,j)[A'R'^\dagger](\cdot,i)R(j,\cdot)$, and we can express B as a sum of r^2 outer products of vectors that we have sampling and query access to. This gives us both $\mathrm{SQ}_\phi(B)$ and $\mathrm{SQ}_\phi(B^\dagger)$.

We won't get exactly this, but using that $\bar{t}(CC^\dagger) = (C^+_{\varepsilon\|A\|/2})^\dagger t(C^\dagger C)C^+_{\varepsilon\|A\|/2}$, for $UDV^\dagger$ the SVD of C and $U_{\varepsilon\|A\|/2}D_{\varepsilon\|A\|/2}V^\dagger_{\varepsilon\|A\|/2}$ the SVD truncated to singular values at least $\varepsilon\|A\|/2$, we can rewrite

$$B = A(R^\dagger U_{\varepsilon\|A\|/2}D^+_{\varepsilon\|A\|/2})(t(D^2)D^+_{\varepsilon\|A\|/2}U^\dagger_{\varepsilon\|A\|/2})R.$$

Now it suffices to get sampling and query access to the columns of $A(R^\dagger U_{\varepsilon\|A\|/2}D^+_{\varepsilon\|A\|/2})$, and by Lemma 5.22, $R^\dagger U_{\varepsilon\|A\|/2}D^+_{\varepsilon\|A\|/2}$ is an ε^3-approximate isometry. Further, we can lower bound the norms of these columns, using that $R^\dagger R \approx A^\dagger A$ and $CC^\dagger \approx RR^\dagger$.

$$\begin{aligned}
\|A(R^\dagger U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2})\|^2 &= \|(U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2})^\dagger R A^\dagger A R^\dagger (U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2})\| \\
&\approx \|(U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2})^\dagger R R^\dagger R R^\dagger (U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2})\| \\
&= \|RR^\dagger (U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2})\|^2 \\
&\approx \|CC^\dagger U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2}\|^2 \\
&= \|UD^2U^\dagger U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2}\|^2 \\
&\geq \varepsilon^2\|A\|^2
\end{aligned}$$

Consider one particular column $v := [R^\dagger U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2}](\cdot, \ell)$; summarizing our prior arguments, we know $\|v\| \geq \frac{1}{2}$ from approximate orthonormality and $\|Av\| \gtrsim \varepsilon\|A\|$, which we just showed. We can also query for entries of v since it is a linear combination of rows of R. We make one more approximation $Av \approx u$, using Lemma 5.17 as we do in Corollary 8.5. That is, if we want to know $[Av](i) = A(i,\cdot)v$, we use our inner product protocol to approximate it to $\gamma\|A(i,\cdot)\|\|v\|$ error, and declare it to be $u(i)$. This implicitly defines u via an algorithm to compute its entries from $\mathrm{SQ}(A)$ and $\mathrm{Q}(v)$. Let B' be the version of B, with the columns of $AR^\dagger U_{\varepsilon\|A\|/2} D^+_{\varepsilon\|A\|/2}$ replaced with their u versions. One can set γ such that the correctness bound $\|B' - A^\dagger\| \lesssim \varepsilon$ and our lower bound $u \gtrsim \varepsilon\|A\|$ both still hold. All we need now to get $\mathrm{SQ}_\phi(u)$ (thereby completing our proof sketch) is a bound $\tilde{u}$ such that we have $\mathrm{SQ}(\tilde{u})$. We will take $\tilde{u}(i) := 2\|A(i,\cdot)\|$. We have $\mathrm{SQ}(\tilde{u})$ immediately from $\mathrm{SQ}(A)$, $\phi = \|\tilde{u}\|^2/\|u\|^2 \lesssim \varepsilon^2\|A\|_\mathrm{F}^2/\|A\|^2$ (from our lower bound on $\|u\|$), and $|\tilde{u}(i)| \geq \|A(i,\cdot)\| + \gamma\|A(i,\cdot)\|\|v\| \geq |u(i)|$ (from our correctness bound from Lemma 5.17). □

Our proof of Theorem 6.1 requires establishing some statements about *stability of computing scalar polynomials*: we begin by bounding quantities coming from bounded polynomials, and then use them to bound the error propagation of the Clenshaw recurrence for computing polynomials. Finally, we consider polynomials of matrices, and then give a fast algorithm for computing $p(A)b$ by applying sketches to A, and then bounding the error using ideas from scalar stability analyses. We prove the even and odd cases separately. We can conclude the above theorem as a consequence of Theorems 6.22 and 6.26 and Corollaries 6.23 and 6.27.

6.1 Sums of Chebyshev coefficients

To give improved stability bounds for the Clenshaw recurrence, we need to bound various sums of Chebyshev coefficients. Since we aim to give bounds that hold for all degree-d polynomials, we use no property of the function beyond that it has a unique Chebyshev expansion; of course, for any particular choice of function f, the bounds in this section can be improved by explicitly computing its Chebyshev coefficients, or in some cases, by using smoothness properties of the function [Tre19, Theorems 7.2 and 8.2].

Let $f : [-1,1] \to \mathbb{R}$ be a Lipschitz continuous function. Then it can be expressed uniquely as a linear combination of Chebyshev polynomials $f(x) = \sum_{i=0}^{\infty} a_i T_i(x)$. A broad topic of interest in approximation theory is bounds for linear combinations of these coefficients, $\sum a_i c_i$, in terms of $\|f\|_{\text{sup}}$; this was one motivation of Vladimir Markov in proving the Markov brothers' inequality [Sch41, p575]. Our goal for this section will be to investigate this question in the case where these sums are arithmetic progressions of step four. This will be necessary for later stability analyses, and is one of the first non-trivial progressions to bound. We begin with some straightforward assertions (see [Tre19] for background).

Fact 6.4. *Let $f : [-1,1] \to \mathbb{R}$ be a Lipschitz continuous function. Then its Chebyshev coefficients $\{a_\ell\}_\ell$ satisfy*

$$\Big|\sum_\ell a_\ell\Big| = |f(1)| \le \|f\|_{\text{sup}}$$

$$\Big|\sum_\ell (-1)^\ell a_\ell\Big| = |f(-1)| \le \|f\|_{\text{sup}}$$

$$\Big|\sum_\ell a_\ell [\![\ell \textit{ is even}]\!]\Big| = \Big|\sum_\ell a_\ell \frac{1}{2}(1 + (-1)^\ell)\Big| \le \|f\|_{\text{sup}}$$

$$\Big|\sum_\ell a_\ell [\![\ell \textit{ is odd}]\!]\Big| = \Big|\sum_\ell a_\ell \frac{1}{2}(1 - (-1)^\ell)\Big| \le \|f\|_{\text{sup}}$$

We use the following result on Lebesgue constants to bound truncations of the Chebyshev coefficient sums.

Lemma 6.5 ([Tre19, Theorem 15.3]). Let $f : [-1,1] \to \mathbb{R}$ be a Lipschitz continuous function, let $f_k(x) = \sum_{\ell=0}^{k} a_\ell T_\ell(x)$, and let the optimal degree-k approximating polynomial to f be denoted f_k^*. Then

$$\begin{aligned}\|f - f_k\|_{\sup} &\leq \Big(4 + \frac{4}{\pi^2}\log(k+1)\Big)\|f - f_k^*\|_{\sup} \\ &\leq \Big(4 + \frac{4}{\pi^2}\log(k+1)\Big)\|f\|_{\sup}.\end{aligned}$$

Similarly,

$$\|f_k\|_{\sup} \leq \|f - f_k\|_{\sup} + \|f\|_{\sup} \leq \Big(5 + \frac{4}{\pi^2}\log(k+1)\Big)\|f\|_{\sup}.$$

This implies bounds on sums of coefficients.

Fact 6.6. *Consider a function $f(x) = \sum_\ell a_\ell T_\ell(x)$. Then*

$$\begin{aligned}\Big|\sum_{\ell=k}^{\infty} a_\ell [\![\ell - k \textit{ is even}]\!]\Big| &\leq \|f - f_{k-1}\|_{\sup} \leq \Big(4 + \frac{4}{\pi^2}\log(k)\Big)\|f\|_{\sup}, \\ \Big|\sum_{\ell=k}^{\infty} a_\ell (-1)^\ell\Big| &\leq \|f - f_{k-1}\|_{\sup} \leq \Big(4 + \frac{4}{\pi^2}\log(k)\Big)\|f\|_{\sup},\end{aligned}$$

where the inequalities follow from Fact 6.4 and Lemma 6.5. When $k = 0$, then the sum is bounded by $\|f\|_{\sup}$, as shown in Fact 6.4.

Now, we prove similar bounds in the case that $f(x)$ is an odd function. In particular, we want to obtain a bound on alternating signed sums of the Chebshyev coefficients and we incur a blowup that scales logarithmically in the degree.

Lemma 6.7. Let $f : [-1,1] \to \mathbb{R}$ be an *odd* Lipschitz continuous function with Chebyshev coefficients $\{a_\ell\}_\ell$, so that $a_k = 0$ for all even k. Then the Chebyshev coefficient sum is bounded as

$$\begin{aligned}\Big|\sum_{\ell=0}^{d}(-1)^\ell a_{2\ell+1}\Big| &\leq (\log(d) + 2)\max_{0\leq k\leq 2d+1}\|f_k\|_{\sup} \\ &\leq (\log(d)+2)\Big(5 + \frac{4}{\pi^2}\log(2d+2)\Big)\|f\|_{\sup} \\ &\leq \Big(16 + 4\log^2(d+1)\Big)\|f\|_{\sup}.\end{aligned}$$

We first state the following relatively straight-forward corollary:

Corollary 6.8. *Lemma 6.7 gives bounds on arithmetic progressions with step size four. Let $f : [-1,1] \to \mathbb{R}$ be a Lipschitz continuous function, and consider nonnegative integers $c \leq d$. Then*

$$\Big|\sum_{\ell=c}^{d} a_\ell [\![\ell - c \equiv 0 \pmod 4]\!]\Big| \leq (32 + 8\log^2(d+1))\|f\|_{\sup}$$

Proof. Define $f^{\text{odd}} := \frac{1}{2}(f(x) - f(-x))$ and $f^{\text{even}} := \frac{1}{2}(f(x) + f(-x))$ to be the odd and even parts of f respectively. Triangle inequality implies that $\|f^{\text{odd}}\|_{\sup}, \|f^{\text{even}}\|_{\sup} \leq \|f\|_{\sup}$. Suppose c, d are odd. Then

$$\begin{aligned}
&\Big|\sum_{\ell=c}^{d} a_\ell [\![\ell - c \equiv 0 \pmod 4]\!]\Big| \\
&= \frac{1}{2}\Big|\sum_{\ell=0}^{\lfloor (d-c)/2 \rfloor} a_{c+2\ell}(1 \pm (-1)^\ell)\Big| \\
&\leq \frac{1}{2}\Big(\Big|\sum_{\ell=0}^{\lfloor (d-c)/2 \rfloor} a_{c+2\ell}\Big| + \Big|\sum_{\ell=0}^{\lfloor (d-c)/2 \rfloor} (-1)^\ell a_{c+2\ell}\Big|\Big) \\
&= \frac{1}{2}\Big(\Big|f_d^{\text{odd}}(1) - f_{c-2}^{\text{odd}}(1)\Big| + \Big|\sum_{\ell=0}^{(d-1)/2} (-1)^\ell a_{2\ell+1} - \sum_{\ell=0}^{(c-3)/2} (-1)^\ell a_{2\ell+1}\Big|\Big) \\
&\leq \frac{1}{2}\Big(\|f_{c-2}^{\text{odd}}\|_{\sup} + \|f_d^{\text{odd}}\|_{\sup} + 2(\log(d) + 2)\max_{0 \leq k \leq d}\|f_k^{\text{odd}}\|_{\sup}\Big) \\
&\leq (32 + 8\log^2(d+1))\|f^{\text{odd}}\|_{\sup} \\
&\leq (32 + 8\log^2(d+1))\|f\|_{\sup}
\end{aligned}$$

The case when c is even is easier: by Eq. (8), we know that

$$\Big\|\sum_\ell a_{2\ell}T_\ell(x)\Big\|_{\sup} = \Big\|\sum_\ell a_{2\ell}T_\ell(T_2(x))\Big\|_{\sup} = \Big\|\sum_\ell a_{2\ell}T_{2\ell}(x)\Big\|_{\sup} = \Big\|f^{\text{even}}(x)\Big\|_{\sup} \leq \|f\|_{\sup},$$

so by Fact 6.6,

$$\begin{aligned}\Big|\sum_{\ell\geq c} a_\ell [\![\ell - c \equiv 0 \pmod 4]\!]\Big| &= \Big|\sum_{\ell\geq c/2} a_{2\ell} [\![\ell - c/2 \text{ is even}]\!]\Big| \\ &\leq \Big(4 + \frac{4}{\pi^2}\log(c/2-1)\Big)\Big\|\sum_\ell a_{2\ell}T_\ell(x)\Big\|_{\sup} \\ &\leq \Big(4 + \frac{4}{\pi^2}\log(c/2-1)\Big)\|f\|_{\sup}. \end{aligned} \tag{14}$$

From the above, we can bound the type of sums in the problem statement, paying an additional factor of two:

$$\begin{aligned}\Big|\sum_{\ell=c}^{d} a_\ell [\![\ell - c \equiv 0 \pmod 4]\!]\Big| &\leq \Big|\sum_{\ell\geq c} a_\ell [\![\ell - c \equiv 0 \pmod 4]\!]\Big| + \Big|\sum_{\ell\geq d+1} a_\ell [\![\ell - c \equiv 0 \pmod 4]\!]\Big| \\ &\leq \Big(8 + \frac{4}{\pi^2}(\log(c/2-1) + \log(d/2+1))\Big)\|f\|_{\sup}, \end{aligned} \tag{15}$$

giving the desired bound. □

We note that Lemma 6.7 will be significantly harder to prove. See Remark 6.11 for an intuitive explanation why. We begin with two structural lemmas on how the solution to a unitriangular linear system behaves, which might be of independent interest.

Lemma 6.9 (An entry-wise positive solution). Suppose that $A \in \mathbb{R}^{d\times d}$ is an upper unitriangular matrix such that, for all $i \leq j$, $A(i,j) > 0$, $A(i,j) > A(i-1,j)$. Then $A^{-1}\vec{1}$ is a vector with positive entries. The same result holds when A is a lower unitriangular matrix such that, for all $i \geq j$, $A(i,j) > 0$, $A(i,j) > A(i+1,j)$.

Proof. Let $x = A^{-1}\vec{1}$. Then $x_d = 1 \geq 0$. The result follows by induction:

$$\begin{aligned}
x(i) &= 1 - \sum_{j=i+1}^{d} A(i,j)x(j) \\
&= \sum_{j=i+1}^{d} (A(i+1,j) - A(i,j))x(j) + 1 - \sum_{j=i+1}^{d} A(i+1,j)x(j) \\
&= \sum_{j=i+1}^{d} (A(i+1,j) - A(i,j))x(j) + 1 - [Ax](i+1) \\
&= \sum_{j=i+1}^{d} (A(i+1,j) - A(i,j))x(j) \\
&> 0
\end{aligned}$$

For lower unitriangular matrices, the same argument follows. The inverse satisfies $x(1) = 1$ and

$$\begin{aligned}
x(i) &= 1 - \sum_{j=1}^{i-1} A(i,j)x(j) \\
&= \sum_{j=i+1}^{d} (A(i-1,j) - A(i,j))x(j) + 1 - \sum_{j=i+1}^{d} A(i-1,j)x(j) > 0
\end{aligned}$$ □

Next, we characterize how the solution to a unitriangular linear system behaves when we consider a partial ordering on the matrices.

Lemma 6.10. Let A be a nonnegative upper unitriangular matrix such that $A(i,j) > A(i-1,j)$ and $A(i,j) > A(i,j+1)$ for all $i \leq j$. Let B be a matrix with the same properties, such that $A \geq B$ entrywise. By Lemma 6.9, $x^{(A)} = A^{-1}\vec{1}$ and $x^{(B)} = B^{-1}\vec{1}$ are nonnegative. It further holds that $\sum_{i=1}^{d}[A^{-1}\vec{1}](i) \leq \sum_{i=1}^{d}[B^{-1}\vec{1}](i)$.

Proof. We consider the line between A and B, $A(t) = A(1-t)+Bt$ for $t \in [0,1]$. Let $x(t) = A^{-1}\vec{1}$; we will prove that $\vec{1}^{\dagger}x(t)$ is monotonically increasing in t. The gradient of $x(t)$ has a simple form [Tao13]:

$$\begin{aligned}
A(t)x(t) &= \vec{1} \\
\partial[A(t)x(t)] &= \partial_t[\vec{1}] \\
(B - A)x(t) + A(t)\partial_t x(t) &= 0 \\
\partial_t x(t) &= A^{-1}(t)(A - B)x(t).
\end{aligned}$$

So,

$$\begin{aligned}
\vec{1}^\dagger \partial_t x(t) &= \vec{1}^\dagger A^{-1}(t)(A - B)A^{-1}(t)\vec{1} \\
&= [([A(t)]^{-1})^\dagger \vec{1}]^\dagger (A - B)[A^{-1}(t)\vec{1}].
\end{aligned}$$

Since A and B satisfy the entry constraints, so do every matrix along the line. Consequently, the column constraints in Lemma 6.9 are satisfied for both A and $A^\dagger$, so both $([A(t)]^{-1})^\dagger \vec{1}$ and $A^{-1}(t)\vec{1}$ are positive vectors. Since $A \geq B$ entrywise, this means that $\vec{1}^\dagger \partial_t x(t)$ is positive, as desired. □

Proof of Lemma 6.7. We first observe that the following sorts of sums are bounded. Let $x_k := \cos(\frac{\pi}{2}(1 - \frac{1}{2k+1}))$. Then, using that $T_\ell(\cos(x)) = \cos(\ell x)$,

$$\begin{aligned}
f_{2k+1}(x_k) = \sum_{\ell=0}^{2k+1} a_\ell T_\ell(x_k) &= \sum_{\ell=0}^{k} a_{2\ell+1} T_{2\ell+1}(x_k) \\
&= \sum_{\ell=0}^{k} a_{2\ell+1} \cos\Big(\frac{\pi}{2}\Big(2\ell + 1 - \frac{2\ell+1}{2k+1}\Big)\Big) = \sum_{\ell=0}^{k} (-1)^\ell a_{2\ell+1} \sin\Big(\frac{\pi}{2}\frac{2\ell+1}{2k+1}\Big).
\end{aligned}$$

We have just shown that

$$\Big|\sum_{\ell=0}^{k} a_{2\ell+1}(-1)^\ell \sin\Big(\frac{\pi}{2}\frac{2\ell+1}{2k+1}\Big)\Big| \leq \|f_{2k+1}\|_{\sup}. \tag{16}$$

We now claim that there exist non-negative c_k for $k \in \{0, 1, \dots, d\}$ such that

$$\sum_{\ell=0}^{d} (-1)^\ell a_{2\ell+1} = \sum_{k=0}^{d} c_k f_{2k+1}(x_k). \tag{17}$$

The $f_{2k+1}(x_k)$'s can be bounded using Lemma 6.5. The rest of the proof will consist of showing that the c_k's exist, and then bounding them.

To do this, we consider the coefficient of each $a_{2\ell+1}$ separately; let $A^{(k)} \in [0,1]^{d+1}$ (index starting at zero) be the vector of coefficients associated with $p_{2k+1}(x_k)$:

$$A^{(k)}_\ell = \sin\Big(\frac{\pi}{2}\frac{2\ell+1}{2k+1}\Big) \text{ for } 0 \le \ell \le k,\ 0 \text{ otherwise} \tag{18}$$

Note that the $A^{(k)}_\ell$ is always non-negative and increasing with ℓ up to $A^{(k)}_k = 1$. Then Eq. (17) holds if and only if

$$c_0 A^{(0)} + \dots + c_d A^{(d)} = \vec{1},$$

or in other words, the equation $Ac = \vec{1}$ is satisfied, where A is the matrix with columns $A^{(k)}$ and c is the vector of c_ℓ's. Since A is upper triangular (in fact, with unit diagonal), this can be solved via backwards substitution: $c_d = 1$, then c_{d-1} can be deduced from c_d, and so on. More formally, the sth row gives the following constraint that can be rewritten as a recurrence.

$$\sum_{t=s}^{d} \sin\Big(\frac{\pi}{2}\frac{2s+1}{2t+1}\Big) c_t = 1 \tag{19}$$

$$c_s = 1 - \sum_{t=s+1}^{d} \sin\Big(\frac{\pi}{2}\frac{2s+1}{2t+1}\Big) c_t \tag{20}$$

Because the entries of A increase in ℓ, the c_ℓ's are all positive.

Invoking Lemma 6.9 with the matrix A establishes that such c_s exist; our goal now is to bound them. Doing so is not as straightforward as it might appear: since the recurrence Eq. (20) *subtracts* by c_t's, an upper bound on c_t for $t \in [s+1, d]$ does not give an upper bound on c_s; it gives a lower bound. So, an induction argument to show bounds for c_s's fails. Further, we were unable to find any closed form for this recurrence. However, since all we need to know is the sum of the c_s's, we show that we *can* bound this via a generic upper bound on the recurrence.

Here, we apply Lemma 6.10 to A as previously defined, and the bounding matrix is (for $i \leq j$)

$$B(i,j) = \frac{i}{j} \leq \frac{2i+1}{2j+1} \leq \sin\Big(\frac{\pi}{2}\frac{2i+1}{2j+1}\Big) = A(i,j),$$

using that $\sin(\frac{\pi}{2}x) \geq x$ for $x \in [0,1]$. Let $\hat{c} = B^{-1}\vec{1}$. Then $\hat{c}(i) = \frac{1}{i+1}$ for $i \neq d$ and $\hat{c}(d) = 1$.

$$[B\hat{c}](i) = \sum_{j=i}^{d} B(i,j)\hat{c}(j) = \sum_{j=i}^{d-1} \frac{i}{j}\frac{1}{j+1} + \frac{i}{d} = i\sum_{j=i}^{d-1}\Big(\frac{1}{j} - \frac{1}{j+1}\Big) + \frac{i}{d} = i\Big(\frac{1}{i} - \frac{1}{d}\Big) + \frac{i}{d} = 1$$

By Lemma 6.10, $\sum_i c(i) \leq \sum_i \hat{c}(i) \leq \log(d) + 2$. So, altogether, we have

$$\begin{aligned}
\Big|\sum_{\ell=0}^{d}(-1)^{\ell}a_{2\ell+1}\Big| &= \Big|\sum_{k=0}^{d} c(k) f_{2k+1}(x_k)\Big| \\
&\leq \sum_{k=0}^{d} c(k)\|f_{2k+1}\|_{\text{sup}} \\
&\leq \Big(\sum_{k=0}^{d} c(k)\Big)\max_{0\leq k\leq d}\|f_{2k+1}\|_{\text{sup}} \\
&= \Big(\sum_{k=0}^{d} c(k)\Big)\max_{0\leq k\leq 2d+1}\|f_k\|_{\text{sup}} \\
&\leq (\log(d)+2)\max_{0\leq k\leq 2d+1}\|f_k\|_{\text{sup}}
\end{aligned}$$ □

Remark 6.11. A curious reader will (rightly) wonder whether this proof requires this level of difficulty. Intuition from the similar Fourier analysis setting suggests that arithmetic progressions of any step size at any offset are easily bounded. We can lift to the Fourier setting by considering, for an $f : [-1,1] \to \mathbb{R}$, a corresponding 2π-periodic $g : [0,2\pi] \to \mathbb{R}$ such that

$$g(\theta) := f(\cos(\theta)) = \sum_{k=0}^{\infty} a_k T_k(\cos(\theta)) = \sum_{k=0}^{\infty} a_k \cos(k\theta) = \sum_{k=0}^{\infty} a_k \frac{e^{ik\theta} + e^{-ik\theta}}{2}$$

This function has the property that $|g(\theta)| \leq \|f\|_{\sup}$ and $\hat{g}(k) = a_{|k|}/2$ (except $\hat{g}(0) = a_0$). Consequently,

$$\begin{aligned}\frac{1}{t}\sum_{j=0}^{t-1} f\Big(\cos(\frac{2\pi j}{t})\Big) = \frac{1}{t}\sum_{j=0}^{t-1} g\Big(\frac{2\pi j}{t}\Big) = \frac{1}{t}\sum_{j=0}^{t-1}\sum_{k=-\infty}^{\infty} \hat{g}(k)e^{2\pi ijk/t} &= \sum_{k=-\infty}^{\infty}\hat{g}(k)\sum_{j=0}^{t-1}\frac{1}{t}e^{2\pi ijk/t} \\ &= \sum_{k=-\infty}^{\infty}\hat{g}(k)[\![k \text{ is divisible by } t]\!] = \sum_{k=-\infty}^{\infty}\hat{g}(kt),\end{aligned}$$

so we can bound arithmetic progressions $|\sum_k \hat{g}(kt)| \leq \|f\|_{\sup}$, and this generalizes to other offsets, to bound $|\sum_k \hat{g}(kt+o)|$ for some $o \in [t-1]$. Notably, though, this approach does not say anything about sums like $\sum_k a_{4k+1}$. The corresponding progression of Fourier coefficients doesn't give it, for example, since we pick up unwanted terms from the negative Fourier coefficients.[15]

$$\begin{aligned}\sum_k \hat{g}(4k+1) &= (\hat{g}(1)+\hat{g}(5)+\hat{g}(9)+\cdots) + (\hat{g}(-3)+\hat{g}(-7)+\hat{g}(-11)+\cdots) \\ &= \frac{1}{2}(a_1+a_5+a_9+\cdots) + \frac{1}{2}(a_3+a_7+a_{11}+\cdots) = \sum_{k\geq 0} a_{2k+1}.\end{aligned}$$

In fact, by inspection of the distribution[16] $D(x) = \sum_{k=0}^{\infty} T_{4k+1}(x)$, it appears that this arithmetic progression cannot be written as a linear combination of evaluations of $f(x)$. Since the shape of the distribution appears to have $1/x$ behavior near $x = 0$, we conjecture that our analysis losing a log factor is, in some respect, necessary.

Conjecture 6.12. *For any step size $t > 1$ and offset $o \in [t-1]$ such that $o \neq t/2$, there exists a function $f : [-1,1] \to \mathbb{R}$ such that $\|f\|_{\sup} = 1$ but $|\sum_{k=0}^{n} a_{tk+o}| = \Omega(\log(n))$.*

[15]These sums are related to the Chebyshev coefficients one gets from interpolating a function at Chebyshev points [Tre19, Theorem 4.2].

[16]This is the functional to integrate against to compute the sum, $\frac{2}{\pi}\int_{-1}^{1} f(x)D(x)/\sqrt{1-x^2} = \sum a_{4k+1}$. The distribution is not a function, but can be thought of as the limit object of $D_n(x) = \sum_{k=0}^{n} T_{4k+1}(x)$ as $n \to \infty$, analogous to Dirichlet kernels and the Dirac delta distribution.

6.2 The Clenshaw recursion

Suppose we are given as input a degree-d polynomial as a linear combination of Chebyshev polynomials:

$$p(x) = \sum_{k=0}^{d} a_k T_k(x). \tag{21}$$

Then this can be computed with the *Clenshaw algorithm*, which is the following recurrence.

$$\begin{aligned} q_{d+1} &= q_{d+2} = 0 \\ q_k &= 2xq_{k+1} - q_{k+2} + a_k \\ \tilde{p} &= \tfrac{1}{2}(a_0 + q_0 - q_2) \end{aligned} \tag{Clenshaw}$$

Lemma 6.13. The recursion in Eq. (Clenshaw) computes $p(x)$. That is, in exact arithmetic, $\tilde{p} = p(x)$. In particular,

$$q_k = \sum_{i=k}^{d} a_i U_{i-k}(x). \tag{22}$$

Proof. We show Eq. (22) by induction.

$$\begin{aligned} q_k &= 2xq_{k+1} - q_{k+2} + a_k \\ &= 2x\Big(\sum_{i=k+1}^{d} a_i U_{i-k-1}(x)\Big) - \Big(\sum_{i=k+2}^{d} a_i U_{i-k-2}(x)\Big) + a_k \\ &= a_k + 2xa_{k+1}U_0(x) + \sum_{i=k+2}^{d} a_i(2xU_{i-k-1}(x) - U_{i-k-2}(x)) \\ &= \sum_{i=k}^{d} a_i U_{i-k}(x). \end{aligned}$$

Consequently, we have

$$\begin{aligned}\frac{1}{2}(a_0+u_0-u_2) &= \frac{1}{2}\Big(a_0+\sum_{i=0}^{d} a_iU_i(x)-\sum_{i=2}^{d} a_iU_{i-2}(x)\Big)\\ &= a_0+a_1x+\sum_{i=2}^{d}\frac{a_i}{2}(U_i(x)-U_{i-2}(x))=\sum_{i=0}^{d} a_iT_i(x).\end{aligned}$$ □

Remark 6.14. Though the aforementioned discussion is specialized to the scalar setting, it extends to the the matrix setting almost entirely syntactically: consider a Hermitian $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$ with $\|A\|, \|b\| \leq 1$. Then $p(A)b$ can be computed in the following way:

$$\begin{aligned} u_{d+1} &= \vec{0}\\ u_d &= a_d b\\ u_k &= 2Au_{k+1}-u_{k+2}+a_kb\\ u := p(A)b &= \frac{1}{2}(a_0b+u_0-u_2)\end{aligned} \tag{23}$$

The proof that this truly computes $p(A)b$ is the same as the proof of correctness for Clenshaw's algorithm shown above. We will also be generalizing to non-Hermitian $A \in \mathbb{C}^{m\times n}$, in which case the only additional wrinkle is that in the recurrence we will need to choose either A or $A^\dagger$ such that dimensions are consistent. Provided that the polynomial being computed is even or odd, no issues will arise. Consequently, Clenshaw-like recurrences will give matrix polynomials where x^k are replaced with $A^\dagger AA^\dagger \cdots Ab$, which corresponds to the definition of singular value transformation from Definition 3.6.

We will be considering evaluating odd and even polynomials. We again focus on the scalar setting and note that this extends to the matrix setting in the obvious way. The previous recurrence Eq. (Clenshaw) can work in this setting, but it'll be helpful for our analysis if the recursion multiplies by x^2 each time, instead of x [MH02, Chapter 2, Problem 7]. So, in the case where the degree-$(2d+1)$ polynomial $p(x)$ is *odd* (so $a_{2k} = 0$ for every k), it can be computed with the iteration

$$\begin{aligned}
q_{d+1} &= q_{d+2} = 0 \\
q_k &= 2T_2(x)q_{k+1} - q_{k+2} + a_{2k+1}U_1(x) && \text{(Odd Clenshaw)} \\
\tilde{p} &= \tfrac{1}{2}(q_0 - q_1)
\end{aligned}$$

When $p(x)$ is a degree-$(2d)$ *even* polynomial (so $a_{2k+1} = 0$ for every k), it can be computed via the same recurrence, replacing $a_{2k+1}U_1(x)$ with a_{2k}. However, we will use an alternative form that's more convenient for us (since we can reuse the analysis of the odd case).

$$\begin{aligned}
\tilde{a}_{2k} &:= a_{2k} - a_{2k+2} + a_{2k+4} - \cdots \pm a_{2d} && (24) \\
q_{d+1} &= q_{d+2} = 0 \\
q_k &= 2T_2(x)q_{k+1} - q_{k+2} + \tilde{a}_{2k+2}U_1(x)^2 && \text{(Even Clenshaw)} \\
\tilde{p} &= \tilde{a}_0 + \tfrac{1}{2}(q_0 - q_1)
\end{aligned}$$

These recurrences correctly compute p follows from a similar analysis to the standard Clenshaw algorithm, formalized below.

Lemma 6.15. The recursions in Eq. (Odd Clenshaw) and Eq. (Even Clenshaw) correctly compute $p(x)$ for even and odd polynomials, respectively. That is, in exact arithmetic, $\tilde{p} = p(x)$. In particular,

$$q_k = \sum_{i=k}^{d} a_i U_{i-k}(x). \tag{25}$$

Proof. We can prove these statements by applying Eq. (22). In the odd case, Eq. (Odd Clenshaw) is identical to Eq. (Clenshaw) except that x is replaced by $T_2(x)$ and a_k is replaced by

$a_{2k+1}U_1(x)$, so by making the corresponding changes in the iterate, we get that

$$q_k = \sum_{i=k}^{d} a_{2i+1}U_1(x)U_{i-k}(T_2(x)) = \sum_{i=k}^{d} a_{2i+1}U_{2(i-k)+1}(x) \qquad \text{by Eq. (9)}$$

$$\tilde{p} = \tfrac{1}{2}(q_0 - q_1) = \sum_{i=0}^{d} \frac{a_{2i+1}}{2}\Big(U_{2i+1}(x) - U_{2i-1}(x)\Big) = p(x). \qquad \text{by Eq. (6)}$$

Similarly, in the even case, Eq. (Even Clenshaw) is identical to Eq. (Clenshaw) except that x is replaced by $T_2(x)$ and a_k is replaced by $4\tilde{a}_{2k}x^2$ (see Definition 24), so that

$$\begin{aligned} q_k &= \sum_{i=k}^{d} \tilde{a}_{2i+2}U_1(x)^2 U_{i-k}(T_2(x)) \\ &= \sum_{i=k}^{d} \tilde{a}_{2i+2}U_1(x)U_{2(i-k)+1}(x) && \text{by Eq. (9)} \\ &= \sum_{i=k}^{d} \tilde{a}_{2i+2}(U_{2(i-k)}(x) + U_{2(i-k+1)}(x)) && \text{by Eq. (5)} \\ &= \sum_{i=k}^{d+1} \tilde{a}_{2i+2}U_{2(i-k)}(x) + \sum_{i=k+1}^{d+1} \tilde{a}_{2i}U_{2(i-k)}(x) && \text{noticing that } \tilde{a}_{2d+2} = 0 \\ &= \tilde{a}_{2k+2} + \sum_{i=k+1}^{d+1} (\tilde{a}_{2i} + \tilde{a}_{2i+2})U_{2(i-k)}(x) \\ &= \tilde{a}_{2k+2} + \sum_{i=k+1}^{d} a_{2i}U_{2(i-k)}(x) \\ &= -\tilde{a}_{2k} + \sum_{i=k}^{d} a_{2i}U_{2(i-k)}(x) \end{aligned}$$

Finally, observe

$$\tilde{a}_0 + \frac{1}{2}(q_0 - q_1) = \tilde{a}_0 + \frac{1}{2}(a_0 - \tilde{a}_0 + \tilde{a}_2) + \sum_{i=1}^{d} \frac{a_{2i}}{2}(U_{2i}(x) - U_{2i-2}(x)) = p(x). \qquad \square$$

Remark 6.16. We can further compute what happens to all these recursions with some additive $\varepsilon^{(k)}$ error in iteration k. This follows just by adding $\varepsilon^{(k)}$ to the constant term in the recursion and chasing the resulting changes through the analysis of Clenshaw, as is done for

Lemma 6.15.

$$\begin{aligned}
&\text{for standard Clenshaw:} && \tilde{q}_k = \sum_{i=k}^{d}(a_i + \varepsilon^{(i)})U_{i-k}(x)\\
&\text{for odd Clenshaw:} && \tilde{q}_k = \sum_{i=k}^{d}(a_{2i+1}U_{2(i-k)+1}(x) + \varepsilon^{(i)}U_{i-k}(T_2(x)))\\
&\text{for even Clenshaw:} && \tilde{q}_k = -\tilde{a}_{2k} + \sum_{i=k}^{d}(a_{2i}U_{2(i-k)}(x) + \varepsilon^{(i)}U_{i-k}(T_2(x)))
\end{aligned}$$

Propagating this error to the full result gives the following results:

$$\begin{aligned}
&\text{for standard Clenshaw:} && \tilde{p} - p(x) = \frac{1}{2} + \sum_{i=1}^{d}\varepsilon^{(i)}T_i(x)\\
&\text{for odd Clenshaw:} && \tilde{p} - p(x) = \frac{1}{2}\varepsilon^{(0)} + \frac{1}{2}\sum_{i=1}^{d}\varepsilon^{(i)}(U_i(T_2(x)) - U_{i-1}(T_2(x)))\\
&\text{for even Clenshaw:} && \tilde{p} - p(x) = \frac{1}{2}\varepsilon^{(0)} + \frac{1}{2}\sum_{i=1}^{d}\varepsilon^{(i)}(U_i(T_2(x)) - U_{i-1}(T_2(x)))
\end{aligned}$$

Because $\|T_i(x)\|_{\sup} = 1$ but $\|U_i(T_2(x)) - U_{i-1}(T_2(x))\|_{\sup} = 2i+1$, this suggests that these parity-specific recurrences are less stable than the standard recursion. However, they will be more amenable to our sketching techniques.

6.3 Stability of the scalar Clenshaw recursion

Before we move to the matrix setting, we warmup with a stability analysis of the scalar Clenshaw recurrence. Suppose we perform Eq. (Clenshaw) to compute a degree-d polynomial p, except every addition, subtraction, and multiplication incurs ε relative error. Typically, this has been analyzed in the finite precision setting, where the errors are caused by truncation. These standard analyses show that this finite precision recursion gives $p(x)$ to $d^2(\sum|a_i|)\varepsilon = \mathcal{O}(d^3\|p\|_{\sup}\varepsilon)$ error. This bound $\sum|a_i|$ is not easily improved in settings where p is a polynomial approximation of a smooth function, since standard methods only give bounds of the form $|a_k| = \Theta((1-\log(1/\varepsilon)/d)^{-k})$, [Tre19, Theorem 8.1], giving only constant bounds for the coefficients.

Such a bound on $\sum|a_k|$ is not tight, however. A use of Parseval's formula [MH02, Theorem 5.3] improves on this by a factor of $\sqrt{d}$:

$$\sum|a_i| \leq \sqrt{d}\sqrt{\sum a_i^2} = \mathcal{O}(\sqrt{d}\|p\|_{\mathrm{sup}}). \tag{26}$$

Testing $\|\sum_{\ell=1}^d s_\ell \frac{1}{\sqrt{\ell}} T_\ell(x)\|_{\mathrm{sup}}$ for random signs $s_\ell \in \{\pm 1\}$ suggests that this bound is tight, meaning coefficient-wise bounds can only prove an error overhead of $\widetilde{\Theta}(d^{2.5}\|p\|_{\mathrm{sup}})$ for the Clenshaw recurrence.

We improve on prior stability analyses to show that the Clenshaw recurrence for Chebyshev polynomials only incurs an error overhead of $d^2\log(d)\|p\|_{\mathrm{sup}}$. This is tight up to a logarithmic factor. This, for example, could be used to improve the bound in [MMS18, Lemma 9] from k^3 to $k^2\log(k)$ (where in that paper, k denotes degree). As we do for the upcoming matrix setting, we proceed by performing an error analysis on the recursion with a stability parameter μ, and then showing that for any bounded polynomial, $1/\mu$ can be chosen to be $\mathcal{O}(d^2\log d)$.

The following is a simple analysis of Clenshaw, with some rough resemblance to an analysis of Oliver [Oli79].

Theorem 6.17 (Stability Analysis for Scalar Clenshaw). *Consider a degree-d polynomial $p : [-1,1] \to \mathbb{R}$ with Chebyshev coefficients $p(x) = \sum_{k=0}^d a_k T_k(x)$. Let $\oplus, \ominus, \odot : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$ be binary operations representing addition, subtraction, and multiplication to $\mu\varepsilon$ relative error, for $0 < \varepsilon < 1$:*

$$|(x \oplus y) - (x + y)| \leq \mu\varepsilon(|x| + |y|)$$
$$|(x \ominus y) - (x - y)| \leq \mu\varepsilon(|x| + |y|)$$
$$|x \odot y - x \cdot y| \leq \mu\varepsilon|x||y| = \mu\varepsilon|xy|.$$

Given an $x \in [-1,1]$, consider performing the Clenshaw recursion with these noisy operations:

$$\begin{aligned} \tilde{q}_{d+1} &= \tilde{q}_{d+2} = 0 \\ \tilde{q}_k &= (2 \odot x) \odot \tilde{q}_{k+1} \ominus (\tilde{q}_{k+2} \ominus a_k) \\ \tilde{\tilde{p}} &= \tfrac{1}{2} \odot ((a_0 \oplus q_0) \ominus q_2) \end{aligned} \tag{Finite-Precision Clenshaw}$$

Then Eq. (Finite-Precision Clenshaw) *outputs $p(x)$ up to $50\varepsilon\|p\|_{\sup}$ error*[17], *provided that $\mu > 0$ satisfies the following three criterion.*

(a) $\mu\varepsilon \leq \frac{1}{50(d+2)^2}$;

(b) $\mu \sum_{i=0}^{d} |a_i| \leq \|p\|_{\sup}$;

(c) $\mu|q_k| = \mu|\sum_{i=k}^{d} a_i U_{i-k}(x)| \leq \frac{1}{d}\|p\|_{\sup}$ *for all $k \in \{0, \dots, d\}$.*

This analysis shows that arithmetic operations incurring $\mu\varepsilon$ error result in computing $p(x)$ to ε error. In particular, the stability of the scalar Clenshaw recurrence comes down to understanding how small we can take μ. Note that if we ignored coefficient sign, $|\sum_{i=k}^{d} a_i U_{i-k}(x)| \leq |\sum_{i=k}^{d} |a_i| U_{i-k}(x)| = \sum_{i=k}^{d}(i-k+1)|a_i|$, this would require setting $\mu = \Theta(1/d^3)$. We show later that we can set $\mu = \Theta((d^2 \log(d))^{-1})$ for all $x \in [-1,1]$ and polynomials p.

Lemma 6.18. In Theorem 6.17, it suffices to take $\mu = \Theta((d^2 \log(d))^{-1})$.

Proof of Theorem 6.17. We will expand out these finite precision arithmetic to get error intervals for each iteration.

$$\tilde{q}_{d+1} = \tilde{q}_{d+2} = 0, \tag{27}$$

[17] We did not attempt to optimize the constants for this analysis.

and

$$\begin{aligned}
\tilde{q}_k &= (2 \odot x) \odot \tilde{q}_{k+1} \ominus (\tilde{q}_{k+2} \ominus a_k) \\
&= (2x \pm 2\mu\varepsilon|x|) \odot \tilde{q}_{k+1} \ominus (\tilde{q}_{k+2} - a_k \pm \mu\varepsilon(|\tilde{q}_{k+2}| + |a_k|)) \\
&= ((2x\tilde{q}_{k+1} \pm 2\mu\varepsilon|x|\tilde{q}_{k+1}) \pm \mu\varepsilon|(2x \pm 2\mu\varepsilon|x|)\tilde{q}_{k+1}|) \ominus (\tilde{q}_{k+2} - a_k \pm \mu\varepsilon(|\tilde{q}_{k+2}| + |a_k|)) \\
&\in (2x\tilde{q}_{k+1} \pm (2\mu\varepsilon + \mu^2\varepsilon^2)2|x\tilde{q}_{k+1}|) \ominus (\tilde{q}_{k+2} - a_k \pm \mu\varepsilon(|\tilde{q}_{k+2}| + |a_k|)) \\
&\in (2x\tilde{q}_{k+1} \pm 6\mu\varepsilon|x\tilde{q}_{k+1}|) \ominus (\tilde{q}_{k+2} - a_k \pm \mu\varepsilon(|\tilde{q}_{k+2}| + |a_k|)) \\
&= 2x\tilde{q}_{k+1} - \tilde{q}_{k+2} + a_k \pm \mu\varepsilon(6|x\tilde{q}_{k+1}| + |\tilde{q}_{k+2}| + |a_k|) \\
&\qquad + \mu\varepsilon|2x\tilde{q}_{k+1} \pm 6\mu\varepsilon|x\tilde{q}_{k+1}|| + \mu\varepsilon|\tilde{q}_{k+2} - a_k \pm \mu\varepsilon(|\tilde{q}_{k+2}| + |a_k|)| \\
&\in 2x\tilde{q}_{k+1} - \tilde{q}_{k+2} + a_k \pm \mu\varepsilon(14|x\tilde{q}_{k+1}| + 3|\tilde{q}_{k+2}| + 3|a_k|),
\end{aligned}$$

and,

$$\begin{aligned}
\tilde{\tilde{p}} &= \tfrac{1}{2} \odot ((a_0 \oplus q_0) \ominus q_2) \\
&= \tfrac{1}{2} \odot ((a_0 + q_0 \pm \mu\varepsilon(|a_0| + |q_0|)) \ominus q_2) \\
&= \tfrac{1}{2} \odot ((a_0 + q_0 - q_2 \pm \mu\varepsilon(|a_0| + |q_0|)) \pm \mu\varepsilon(|a_0 + q_0 \pm \mu\varepsilon(|a_0| + |q_0|)| + |q_2|)) \\
&\in \tfrac{1}{2} \odot (a_0 + q_0 - q_2 \pm \mu\varepsilon(3|a_0| + 3|q_0| + |q_2|)) \\
&= \tfrac{1}{2}(a_0 + q_0 - q_2 \pm \mu\varepsilon(3|a_0| + 3|q_0| + |q_2|)) \pm \mu\varepsilon\tfrac{1}{2}|a_0 + q_0 - q_2 \pm \mu\varepsilon(3|a_0| + 3|q_0| + |q_2|)| \\
&\in \tfrac{1}{2}(a_0 + q_0 - q_2) \pm \tfrac{1}{2}\mu\varepsilon(7|a_0| + 7|q_0| + 3|q_2|).
\end{aligned}$$

To summarize, we have

$$\begin{aligned}
\tilde{q}_{d+1} &= \tilde{q}_{d+2} = 0 \\
\tilde{q}_k &= 2x\tilde{q}_{k+1} - \tilde{q}_{k+2} + a_k + \delta_k, &&\text{where } |\delta_k| \leq \mu\varepsilon(14|x\tilde{q}_{k+1}| + 3|\tilde{q}_{k+2}| + 3|a_k|) && (28) \\
\tilde{\tilde{p}} &= \tfrac{1}{2}(a_0 + q_0 - q_2) + \delta, &&\text{where } |\delta| \leq \tfrac{1}{2}\mu\varepsilon(7|a_0| + 7|q_0| + 3|q_2|) && (29)
\end{aligned}$$

By Lemma 6.13, this recurrence satisfies

$$\begin{aligned}
\tilde{q}_k &= \sum_{i=k}^{d} U_{i-k}(x)(a_i + \delta_i) \\
q_k - \tilde{q}_k &= \sum_{i=k}^{d} U_{i-k}(x)\delta_i \\
q - \tilde{q} &= \delta + \frac{1}{2}\Big(\sum_{i=0}^{d} U_i(x)\delta_i - \sum_{i=2}^{d} U_{i-2}(x)\delta_i\Big) \\
&= \delta + \frac{1}{2}\delta_0 + \sum_{i=1}^{d} T_i(x)\delta_i \\
|q - \tilde{q}| &\leq |\delta| + \frac{1}{2}|\delta_0| + \sum_{i=1}^{d} |T_i(x)\delta_i| \leq |\delta| + \sum_{i=0}^{d} |\delta_i|. \qquad (30)
\end{aligned}$$

This analysis so far has been fully standard. Let's continue bounding.

$$\begin{aligned}
&\leq \mu\varepsilon\Big(\tfrac{7}{2}|a_0| + \tfrac{7}{2}|q_0| + \tfrac{3}{2}|q_2| + \sum_{i=0}^{d}(14|x\tilde{q}_{i+1}| + 3|\tilde{q}_{i+2}| + 3|a_i|)\Big) \\
&\leq \mu\varepsilon \sum_{i=0}^{d}(20|\tilde{q}_i| + 10|a_i|). \qquad (31)
\end{aligned}$$

Now, we will bound all of the δ_k's. Combining previous facts, we have

$$\begin{aligned}
|\tilde{q}_k| &= |\sum_{i=k}^{d} U_{i-k}(x)(a_i + \delta_i)| \\
&\leq \Big|\sum_{i=k}^{d} U_{i-k}(x)a_i\Big| + \sum_{i=k}^{d}\Big|U_{i-k}(x)\delta_i\Big| \\
&\leq \Big|\sum_{i=k}^{d} U_{i-k}(x)a_i\Big| + \sum_{i=k}^{d}(i-k+1)|\delta_i| \\
&\leq \Big|\sum_{i=k}^{d} U_{i-k}(x)a_i\Big| + \mu\varepsilon\sum_{i=k}^{d}(i-k+1)(14|\tilde{q}_{i+1}| + 3|\tilde{q}_{i+2}| + 3|a_i|) \\
&\leq \Big(\frac{1}{\mu d} + 3\mu\varepsilon\frac{d-k+1}{\mu}\Big)\|p\|_{\sup} + \mu\varepsilon\sum_{i=k}^{d}(i-k+1)(14|\tilde{q}_{i+1}| + 3|\tilde{q}_{i+2}|) \\
&\leq \frac{1.5}{\mu d}\|p\|_{\sup} + \mu\varepsilon\sum_{i=k}^{d}(i-k+1)(14|\tilde{q}_{i+1}| + 3|\tilde{q}_{i+2}|)
\end{aligned}$$

Note that $|\tilde{q}_k| \leq c_k$, where

$$c_d = 0;$$
$$c_k = \frac{1.5}{\mu d}\|p\|_{\sup} + \mu\varepsilon \sum_{i=k}^{d}(i-k+1)(14c_{i+1} + 3c_{i+2})$$

Solving this recurrence, we have that $c_k \leq \frac{2}{\mu d}\|p\|_{\sup}$, since by strong induction,

$$\begin{aligned} c_k &\leq \Big(\frac{1.5}{\mu d} + \mu\varepsilon \sum_{i=k}^{d}(i-k+1)17\frac{2}{\mu d}\Big)\|p\|_{\sup} \\ &= \Big(\frac{1.5}{\mu d} + 17\mu\varepsilon\frac{1}{\mu d}(d-k+1)(d-k+2)\Big)\|p\|_{\sup} \leq \frac{2}{\mu d}\|p\|_{\sup} \end{aligned}$$

Returning to Equation (31):

$$\begin{aligned} |q - \tilde{q}| &\leq \mu\varepsilon \sum_{i=0}^{d}(20|\tilde{q}_i| + 10|a_i|) \leq \mu\varepsilon \sum_{i=0}^{d}(20c_i + 10|a_i|) \\ &\leq 40\varepsilon\|p\|_{\sup} + 10\mu\varepsilon \sum_{i=0}^{d}|a_i| \leq 50\varepsilon\|p\|_{\sup} \end{aligned}$$ □

We now prove Lemma 6.18. In particular, we wish to show that for $\mu = \Theta((d^2\log(d))^{-1})$, the following criteria hold:

(a) $\mu\varepsilon \leq \frac{1}{50(d+2)^2}$;

(b) $\mu\sum_{i=0}^{d}|a_i| \leq \|p\|_{\sup}$;

(c) $\mu|q_k| = \mu|\sum_{i=k}^{d} a_i U_{i-k}(x)| \leq \frac{1}{d}\|p\|_{\sup}$ for all $k \in \{0, \dots, d\}$.

For this choice of μ, (a) is clearly satisfied, and since $|a_i| \leq 2\|p\|_{\sup}$ (Lemma 3.7), $\mu\sum_{i=0}^{d}|a_i| \leq 2(d+1)\|p\|_{\sup} \leq \|p\|_{\sup}$, so (b) is satisfied. In fact, both of these criterion are satisfied for $\mu = \Omega(1/d)$, provided ε is sufficiently small.

Showing (c) requires bounding $\|\sum_{\ell=k}^{d} a_\ell U_{\ell-k}(x)\|_{\sup}$ for all $k \in [d]$. These expressions are also the iterates of the Clenshaw algorithm (Lemma 6.13), so we are in fact trying to show that in the process of our algorithm we never produce a value that's much larger than the final value. From testing computationally, we believe that the following holds true.

Conjecture 6.19. *Let $p(x)$ be a degree-d polynomial with $p(x) = \sum_{\ell=0}^{d} a_\ell T_\ell(x)$. Then, for all k*

from 0 *to* d,

$$\Big\|\sum_{\ell=k}^{d} a_\ell U_{\ell-k}(x)\Big\|_{\sup} \leq (d-k+1)\|p\|_{\sup},$$

maximized for the Chebyshev polynomial $p(x) = T_d(x)$.

Conjecture 6.19 would imply that it suffices to take $\mu = \Theta(1/d^2)$. We prove it up to a log factor.

Theorem 6.20. *For a degree-d polynomial $p(x) = \sum_{\ell=0}^{d} a_\ell T_\ell(x)$, consider the degree-$(d-k)$ polynomial $q_k(x) = \sum_{\ell=k}^{d} a_\ell U_{\ell-k}(x)$. Then*

$$\|q_k\|_{\sup} \leq (d-k+1)\Big(16 + \frac{16}{\pi^2}\log(d)\Big)\|p\|_{\sup}.$$

Proof. We proceed by carefully bounding the Chebyshev coefficients of q_k, which turn out to be arithmetic progressions of the a_k's which we bounded in Section 6.1.

$$\begin{aligned}
q_k(x) &= \sum_i a_i U_{i-k}(x) \\
&= \sum_i \sum_{j\geq 0} a_i T_{i-k-2j}(x)(1 + [\![i-k-2j \neq 0]\!]) \\
&= \sum_i \sum_{j\geq 0} a_{i+k+2j} T_i(x)(1 + [\![i \neq 0]\!]) \\
&= \sum_i T_i(x)(1 + [\![i \neq 0]\!]) \sum_{j\geq 0} a_{i+k+2j} \\
|q_k(x)| &\leq \sum_i [\![i \geq 0]\!](1 + [\![i \neq 0]\!])\Big|\sum_{j\geq 0} a_{i+k+2j}\Big| \\
&\leq 2\sum_{i=0}^{d-k}\Big|\sum_{j\geq 0} a_{i+k+2j}\Big| \\
&\leq 4\sum_{i=0}^{d-k}\Big(4 + \frac{4}{\pi^2}\log(i+k-1)\Big)\|p\|_{\sup} && \text{by Fact 6.6} \\
&\leq (d-k+1)\Big(16 + \frac{16}{\pi^2}\log(d)\Big)\|p\|_{\sup}.
\end{aligned}$$

□

Remark 6.21. We spent some time trying to prove Conjecture 6.19, since its form is tantalizingly close to that of the Markov brothers' inequality [Sch41],

$$\|\tfrac{d}{dx}p(x)\|_{\sup} = \|\textstyle\sum_{\ell=0}^{d} a_\ell \ell U_{\ell-1}(x)\|_{\sup} \leq d^2\|p(x)\|_{\sup},$$

except with the linear differential operator $\frac{d}{dx} : T_\ell \mapsto \ell U_{\ell-1}$ replaced with the linear operator $T_\ell \mapsto U_{\ell-k}$. However, calculations suggest that the variational characterization of $\max_{\|p\|_{\sup}=1}|\frac{d}{dx}p(x)|$ underlying proofs of the Markov brothers' inequality [Sha04] does not hold here, and from our shallow understanding of these proofs, it seems that they strongly use properties of the derivative.

6.4 Computing matrix polynomials

We prove now begin our proof of Theorem 6.1, beginning with the odd case. The statement involves a parameter μ which depends on the polynomial being evaluated; this parameter is between 1 and $(d\log d)^{-2}$, depending on how well-conditioned the polynomial is.

Theorem 6.22. *Suppose we are given sampling and query access to $A \in \mathbb{C}^{m\times n}$ and $b \in \mathbb{C}^n$ with $\|A\| \leq 1$; a $(2d+1)$-degree odd polynomial p, written in its Chebyshev coefficients as*

$$p(x) = \sum_{i=0}^{d} a_{2i+1}T_{2i+1}(x);$$

an accuracy parameter $\varepsilon > 0$; a failure probability parameter $\delta > 0$; and a stability parameter $\mu > 0$. Then we can output a vector $x \in \mathbb{C}^n$ such that $\|Ax - p(A)b\| \leq \varepsilon\|p\|_{\sup}\|b\|$ with probability $\geq 1-\delta$ in time

$$\mathcal{O}\left(\min\left\{\mathrm{nnz}(A), \frac{d^4\|A\|_F^4}{(\mu\varepsilon)^4}\log^2\left(\frac{\|A\|_F}{\delta\|A\|}\right)\right\} + \frac{d^7\|A\|_F^4}{(\mu\varepsilon)^2\delta}\log\left(\frac{\|A\|_F}{\delta\|A\|}\right)\right),$$

assuming $\mu\varepsilon < \min(\frac{1}{4}d\|A\|, \frac{1}{100d})$ and μ satisfies the following bounds:

(a) $\mu\sum_{i=0}^{d}|a_{2i+1}| \leq \|p\|_{\sup}$;

(b) $\mu\|\sum_{i=k}^{d} a_{2i+1}U_{i-k}(T_2(x))\|_{\sup} \leq \frac{1}{d}\|p\|_{\sup}$ *for all* $0 \leq k \leq d$;

The output description has the additional properties

$$\sum_j \|A(\cdot,j)\|^2|x(j)|^2 \lesssim \frac{\varepsilon^2\|p\|_{\sup}^2\|b\|^2}{d^4\log\frac{\|A\|_F}{\delta\|A\|}} \qquad \|x\|_0 \lesssim \frac{d^2\|A\|_F^2}{(\mu\varepsilon)^2}\log\frac{\|A\|_F}{\delta\|A\|},$$

so that by Corollary 4.10, for the output vector $y := Ax$, we can:

(i) Compute entries of y in $\mathcal{O}(\|x\|_0) = \mathcal{O}\Big(\frac{d^2\|A\|_F^2}{(\mu\varepsilon)^2}\log\frac{\|A\|_F}{\delta\|A\|}\Big)$ time;

(ii) Sample $i \in [n]$ with probability $\frac{|y_i|^2}{\|y\|^2}$ in $\mathcal{O}\Big(\frac{\|p\|_{\sup}^2\|A\|_F^4\|b\|^2}{\mu^4\varepsilon^2\|y\|^2}\log\frac{\|A\|_F}{\delta\|A\|}\log\frac{1}{\delta}\Big)$ time with probability $\geq 1-\delta$;

(iii) Estimate $\|y\|^2$ to ν relative error in $\mathcal{O}\Big(\frac{\|p\|_{\sup}^2\|A\|_F^4\|b\|^2}{\nu^2\mu^4\varepsilon^2\|y\|^2}\log\frac{\|A\|_F}{\delta\|A\|}\log\frac{1}{\delta}\Big)$ time with probability $\geq 1-\delta$.

Algorithm 1 (Odd singular value transformation).

Input (pre-processing): A matrix $A \in \mathbb{C}^{m\times n}$, vector $b \in \mathbb{C}^n$, and parameters ε, δ, $\mu > 0$.

Pre-processing sketches: Let $s, t = \Theta(\frac{d^2\|A\|_F^2}{(\mu\varepsilon)^2}\log(\frac{\|A\|_F}{\delta\|A\|}))$. This phase will succeed with probability $\geq 1-\delta$.

```
P1. If SQ($A^\dagger$) and SQ($b$) are not given, compute data structures to simulate them
    in $\mathcal{O}(1)$ time;
P2. Sample $S \in \mathbb{C}^{n\times s}$ from $\{\frac{1}{2}(\frac{\|A(\cdot,j)\|^2}{\|A\|_F^2} + \frac{|b(j)|^2}{\|b\|^2})\}_{j\in[n]}$;
P3. Sample $T^\dagger \in \mathbb{C}^{m\times t}$ from $\{\frac{\|[AS](i,\cdot)\|}{\|AS\|_F^2}\}_{i\in[m]}$;
P4. Compute a data structure that can respond to SQ($TAS$) queries in $\mathcal{O}(1)$ time;
```

Input: A degree $2d+1$ polynomial $p(x) = \sum_{i=0}^d a_{2i+1}T_{2i+1}(x)$ given as its coefficients a_{2i+1}.

Clenshaw iteration: Let $r = \Theta(d^4\|A\|_F^2(s+t)\frac{1}{\delta}) = \Theta(d^6\frac{\|A\|_F^4}{(\mu\varepsilon)^2\delta}\log\frac{\|A\|_F}{\delta\|A\|})$. This phase will succeed with probability $\geq 1-\delta$. Starting with $v_{d+1} = v_{d+2} = \vec{0}^s$ and going until v_0,

```
I1. Let $B^{(k)} = \textsc{best}_r(TAS)$ and $B_\dagger^{(k)} = \textsc{best}_r((TAS)^\dagger)$ (Definition 5.19);
I2. Compute $v_k = 2(2B_\dagger^{(k)}B^{(k)} - I)v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b$.
```

Output: Output $x = \frac{1}{2}S(v_0 - v_1)$ satisfying $\|Ax - p(A)b\| \leq \varepsilon\|p\|_{\sup}\|b\|$.

The criterion for what μ need to be are somewhat non-trivial; the important requirement is Item 6.22(b), which states that $1/\mu$ is a bound on the norm of various polynomials. These polynomials turn out to be iterates of Eq. (Odd Clenshaw) when computing $p(x)/x$, so roughly speaking, the algorithm we present depends on the numerical stability of evaluating $p(x)/x$. This is necessary because we primarily work in the "dual" space, maintaining our Clenshaw iterate u_k as Av_k where v_k is a sparse vector. For any bounded polynomial, we can always take μ to be $\Omega((d\log(d))^{-2})$.

Corollary 6.23 (Corollary of Proposition 6.28). *In Theorem 6.22, we can always take* $1/\mu \lesssim d^2\log^2(d)$ *for* $d > 1$.

This bound is achieved up to log factors by $p(x) = T_{2k+1}(x)$. We are now ready to dive into the proof of Theorem 6.22. Without loss of generality, we assume $\|p\|_{\sup} = 1$.

Pre-processing sketches, running time. Given a matrix $A \in C^{m\times n}$ and a vector $b \in \mathbb{C}^n$, the pre-processing phase of Algorithm 1 can be performed in $\mathcal{O}(\operatorname{nnz}(A) + \operatorname{nnz}(b))$ time. First, we build a data structure to respond to $\mathrm{SQ}(A^\dagger)$ and $\mathrm{SQ}(b)$ queries in $\mathcal{O}(1)$ time using the alias data structure described in Remark 4.12 (A1.P1). Then, we use these accesses to construct an AMP sketch S for Ab, where we produce the samples for the sketch by sampling from b and sampling from the row norms of $A^\dagger$, each with probability $\frac{1}{2}$ (A1.P2). Since we need s samples, this takes $\mathcal{O}(s)$ queries to the data structure. The sketch S is defined such that AS is a subset of the columns of A, with each column rescaled according to the probability it was sampled. So, with another pass through A, we can construct a data structure for $\mathrm{SQ}(AS)$ in $\mathcal{O}(1)$ time using Remark 4.12 and use this to construct an AMP sketch $T^\dagger$ for $(AS)^\dagger AS$ (A1.P3). The samples for the sketch are drawn from the row norms of AS. The final matrix TAS is a rescaled submatrix of A; another $\mathcal{O}(\operatorname{nnz}(A))$ pass through A suffices to construct a data structure to respond to $\mathrm{SQ}(TAS)$ queries in $\mathcal{O}(1)$ time (A1.P4). The running time is $\mathcal{O}(\operatorname{nnz}(A) + \operatorname{nnz}(b) + s + t)$ or, alternatively, three passes through A and one pass with b, using $\mathcal{O}(st)$ space. We can assume that $s, t \leq \operatorname{nnz}(A)$, though: if $s \geq n$, then we can take $S = I$, and if $t \geq m$, then we can take $T = I$, and this will satisfy the same guarantees; further, without loss, $\operatorname{nnz}(A) \geq \max(m, n)$, since otherwise there is an empty row or column that we can ignore.

If we are given A, b such that $\mathrm{SQ}(A^\dagger)$ and $\mathrm{SQ}(b)$ queries can be performed in $\mathcal{O}(Q)$ time, then the pre-processing phase of Algorithm 1 can be performed in $\mathcal{O}(Qst)$ time. The main difference from the description above is that we use that, given $\mathrm{SQ}(A^\dagger)$, we can simulate queries to $\mathrm{SQ}(AS)$ with only $\mathcal{O}(s)$ overhead. To produce a sample $i \in [m]$ with probability $\|[AS](i,\cdot)\|^2/\|AS\|_{\mathrm{F}}^2$, sample a column index $j \in [s]$ with probability $\|[AS](\cdot,j)\|^2/\|AS\|_{\mathrm{F}}^2$ and then query $\mathrm{SQ}(A)$ to get a row index $i \in [m]$ with probability $|[AS](i,j)|^2/\|[AS](\cdot,j)\|^2$.

Remark 6.24. If we are not given b until after the pre-processing phase, or if we are only given b as a list of entries without $\mathrm{SQ}(b)$, then we can take the S sketch to just be sampling from the row norms of $A^\dagger$. This will decrease the success probability of the following phase (specifically, because of the guarantee in Eq. (Ab AMP)) to 0.99.

Pre-processing sketches, correctness. We list the guarantees of the sketch that we will use in the error analysis, and point to where they come from in Section 5. Recall that, with $S \in \mathbb{C}^{n\times s}$ taken to be an AMP sketch of $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{n\times d}$, by Corollary 5.11, with probability $\geq 1-\delta$, $\|XSS^\dagger Y^\dagger - XY^\dagger\| \leq \varepsilon\|X\|\|Y\|$ provided $s = \Omega(\frac{1}{\varepsilon^2}(\frac{\|X\|_{\mathrm{F}}^2}{\|X\|^2}+\frac{\|Y\|_{\mathrm{F}}^2}{\|Y\|^2})\log(\frac{m+d}{\delta}))$ and $\varepsilon \leq 1$. We will use this with $s,t = \Theta\Big(\frac{d^2\|A\|_{\mathrm{F}}^2}{(\mu\varepsilon)^2}\log(\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|})\Big)$, where $\frac{\mu\varepsilon}{d} \leq \frac{1}{4}\|A\|$. The guarantees of Corollary 5.11 individually fail with probability $\mathcal{O}(\delta)$, so we will rescale to say that they all hold with probability $\geq 1-\delta$. The following bounds hold for the sketch S.

$\|[AS](\cdot,j)\|^2 \leq 2\|A\|_{\mathrm{F}}^2/s$ for all $j \in [s]$	by Lemma 5.2	($\|[AS](\cdot,j)\|$ bd)
$\|AS\|_{\mathrm{F}}^2 \leq 2\|A\|_{\mathrm{F}}^2$	by Eq. ($\|[AS](\cdot,j)\|$ bd)	($\|AS\|_F$ bd)
$\|S^\dagger b\|^2 \leq 2\|b\|^2$	by Lemma 5.2	($\|S^\dagger b\|$ bd)
$\|Ab - ASS^\dagger b\| \leq \frac{\mu\varepsilon}{d}\|b\|$	by Corollary 5.11	(Ab AMP)
$\|AA^\dagger - AS(AS)^\dagger\| \leq \frac{\mu\varepsilon}{d}\|A\|$	by Corollary 5.11	($AA^\dagger$ AMP)
$\|AS\|^2 = \|AS(AS)^\dagger\| \leq (1+\frac{\mu\varepsilon}{d\|A\|})\|A\|^2$	by Eq. ($AA^\dagger$ AMP)	($\|AS\|$ bd)

The following bounds hold for the sketch T.

$$\begin{aligned}
\|TAS\|_{\mathrm{F}}^2 &= \|AS\|_{\mathrm{F}}^2 && \text{by Lemma 5.2} \\
&\leq 2\|A\|_{\mathrm{F}}^2 && \text{by Eq. } (\|AS\|_F \text{ bd}) && (\|TAS\|_F \text{ bd}) \\
\|(AS)^\dagger AS - (TAS)^\dagger TAS\| &\leq \tfrac{\mu\varepsilon}{d}\|AS\| && \text{by Corollary 5.11} \\
&\leq 2\tfrac{\mu\varepsilon}{d}\|A\| && \text{by Eq. } (\|AS\| \text{ bd}) && ((AS)^\dagger AS \text{ AMP}) \\
\|TAS\|^2 = \|(TAS)^\dagger TAS\| &\leq (1 + \tfrac{\mu\varepsilon}{d\|AS\|})\|AS\|^2 && \text{by Eq. } ((AS)^\dagger AS \text{ AMP}) \\
&\leq (1 + 2\tfrac{\mu\varepsilon}{d\|A\|})\|A\|^2 && \text{by Eq. } (\|AS\| \text{ bd}) && (\|TAS\| \text{ bd})
\end{aligned}$$

One Clenshaw iteration. We are trying to perform the odd Clenshaw recurrence defined in Eq. (Odd Clenshaw). The matrix analogue of this is

$$\begin{aligned}
u_k &= 2(2AA^\dagger - I)u_{k+1} - u_{k+2} + 2a_{2k+1}Ab, && \text{(Odd Matrix Clenshaw)} \\
u &= \tfrac{1}{2}(u_0 - u_1).
\end{aligned}$$

We now show how to compute the next iterate u_k given b and the previous two iterates as $v_{k+1}, v_{k+2} \in \mathbb{C}^s$ where $u_{k+1} = ASv_{k+1}$ and $u_{k+2} = ASv_{k+2}$, for all $k \geq 0$. The analysis begins by showing that u_k is $\varepsilon\|v_{k+1}\|$-close to the output of an exact, zero-error iteration. Next, we show that $\|v_k\|$ is $\mathcal{O}(d)$-close to its zero-error iteration value. Finally, we show that these errors don't accumulate too much towards the final outcome.

Starting from Eq. (Odd Matrix Clenshaw), we take the following series of approximations by applying intermediate sketches:

$$
\begin{aligned}
4AA^\dagger u_{k+1} &- 2u_{k+1} - u_{k+2} + a_{2k+1}Ab \\
&\approx_1 4AS(AS)^\dagger u_{k+1} - 2u_{k+1} - u_{k+2} + a_{2k+1}Ab \\
&= AS(4(AS)^\dagger(AS)v_{k+1} - 2v_{k+1} - v_{k+2}) + a_{2k+1}Ab \\
&\approx_2 AS(4(TAS)^\dagger(TAS)v_{k+1} - 2v_{k+1} - v_{k+2}) + a_{2k+1}Ab \\
&\approx_3 AS(4(TAS)^\dagger(TAS)v_{k+1} - 2v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b) \qquad (32) \\
&\approx_4 AS(4(TAS)^\dagger B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b) \\
&\approx_5 AS(4B_\dagger^{(k)}B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b). \qquad (33)
\end{aligned}
$$

The final expression, Eq. (33), is what the algorithm computes, taking

$$v_k := (4B_\dagger^{(k)}B^{(k)} - 2I)v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b. \qquad (34)$$

Here, $B^{(k)}$ and $B_\dagger^{(k)}$ are taken to be $\textsc{best}_r(TAS)$ and $\textsc{best}_r((TAS)^\dagger)$ for $r = \Theta(d^4\|A\|_F^2(s+t)\frac{1}{\delta}) = \Theta(d^6\frac{\|A\|_F^4}{(\mu\varepsilon)^2\delta}\log\frac{\|A\|_F}{\delta\|A\|})$. The runtime of each iteration is $\mathcal{O}(r)$, since the cost of producing the sketches $B^{(k)}$ and $B_\dagger^{(k)}$ using SQ(TAS) is $\mathcal{O}(r)$ (A1.I1), and actually computing the iteration costs $\mathcal{O}(r+s+t) = \mathcal{O}(r)$ (A1.I2).

Remark 6.25. We could have stopped sketching at Eq. (32), without using $\textsc{best}$, and instead took $v_k := 4(TAS)^\dagger(TAS)v_{k+1} - 2v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b$. The time to compute each iteration increases to $\mathcal{O}(st)$, and the final running time is

$$\mathcal{O}\Big(\frac{d^5\|A\|_F^4}{\mu^4\varepsilon^4}\log^2\frac{\|A\|_F}{\delta\|A\|}\Big)$$

to achieve the guarantees of Theorem 6.22 with probability $\geq 1-\delta$. This running time is worse by a factor of d^2/ε^2, but scales logarithmically in failure probability.

As for approximation error, let $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4$ and ε_5 be the errors introduced in the approximation steps for Eq. (33). Using the previously established bounds on S and T,

$$\varepsilon_1 \leq 4\|AA^\dagger - AS(AS)^\dagger\|\|u_{k+1}\| \leq 4\tfrac{\mu\varepsilon}{d}\|A\|\|u_{k+1}\| \qquad \text{by Eq. } (AA^\dagger \text{ AMP})$$

$$\varepsilon_2 \leq 4\|AS\|\|((AS)(AS)^\dagger - (TAS)(TAS)^\dagger)v_{k+1}\| \leq 16\tfrac{\mu\varepsilon}{d}\|A\|^2\|v_{k+1}\| \qquad \text{by Eq. } ((AS)^\dagger AS \text{ AMP})$$

$$\varepsilon_3 \leq |a_{2k+1}|\|Ab - ASS^\dagger b\| \leq |a_{2k+1}|\tfrac{\mu\varepsilon}{d}\|b\| \qquad \text{by Eq. } (Ab \text{ AMP})$$

The bounds on ε_4 and ε_5 follow from the bounds in Lemma 5.20 applied to TAS. With probability $\geq 1 - \delta/d$, the following hold:

$$\begin{aligned}
\varepsilon_4 &\leq 4\|AS(TAS)^\dagger(TAS - B^{(k)})v_{k+1}\| \\
&\leq 4\|AS(TAS)^\dagger\|_F\|TAS\|_F\|v_{k+1}\|\sqrt{d/(r\delta)} && \text{by Corollary 5.21} \\
&\leq \|AS(TAS)^\dagger\|_F\|TAS\|_F\|v_{k+1}\|\frac{\mu\varepsilon}{12\|A\|_F^2 d^{5/2}} \\
&\leq d^{-5/2}\mu\varepsilon\|A\|\|v_{k+1}\| && \text{by Eqs. } (\|AS\| \text{ bd}) \text{ and } (\|TAS\|_F \text{ bd}) \\
\varepsilon_5 &\leq 4\|AS((TAS)^\dagger - B_\dagger^{(k)})B^{(k)}v_{k+1}\| \\
&\leq 4\|AS\|_F\|TAS\|_F\|B^{(k)}v_{k+1}\|\sqrt{d/(r\delta)} && \text{by Corollary 5.21} \\
&\leq 4\sqrt{d/(r\delta)}\|AS\|_F\|TAS\|_F\Big(\|TASv_{k+1}\| + \sqrt{d/(r\delta)}\|I_t\|_F\|TAS\|_F\|v_{k+1}\|\Big) && \text{by Corollary 5.21} \\
&\leq \tfrac{1}{3}d^{-5/2}\mu\varepsilon(\|TASv_{k+1}\| + d^{-3/2}\|v_{k+1}\|) && \text{by Eqs. } (\|TAS\|_F \text{ bd}) \text{ and } (\|TAS\| \text{ bd}) \\
&\leq d^{-5/2}\mu\varepsilon\|v_{k+1}\|. && \text{by } \|A\| \leq 1
\end{aligned}$$

In summary, we can view the iterate of A1.I2 as computing

$$\tilde{u}_k = 2(2AA^\dagger - I)\tilde{u}_{k+1} - \tilde{u}_{k+2} + 2a_{2k+1}Ab + \varepsilon^{(k)} \tag{35}$$

Where $\varepsilon^{(k)} \in \mathbb{C}^m$ is the error of the approximation in the iterate Eq. (33). We have showed that

$$\|\varepsilon^{(k)}\| \leq \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 + \varepsilon_5 \lesssim \tfrac{\mu\varepsilon}{d}\Big(\|v_{k+1}\| + |a_{2k+1}|\|b\|\Big).$$

Upon applying a union bound, we see that this bound on $\varepsilon^{(k)}$ holds for every k from 0 to $d-1$ with probability $\geq 1 - 3\delta$.

Error accumulation across iterations. Now, we analyze how the error from one iteration affects to the final output. Using the formulation of the iterate from Eq. (35), we notice that this is the standard Clenshaw iteration Eq. (Clenshaw) with x replaced with $T_2(A^\dagger) = 2AA^\dagger - I$ and a_k replaced with $2a_{2k+1}Ab + \varepsilon^{(k)}$. Following Remark 6.16 and Lemma 6.15, we conclude that the output of Algorithm 1 satisfies

$$\begin{aligned}
\tilde{u}_k &= \sum_{i=k}^{d} U_{i-k}(T_2(A^\dagger))(2a_{2i+1}Ab + \varepsilon^{(k)}) \\
\tilde{u} &:= \frac{1}{2}(\tilde{u}_0 - \tilde{u}_1) \\
&= \sum_{i=0}^{d} \tfrac{1}{2}(U_i(T_2(A^\dagger)) - U_{i-1}(T_2(A^\dagger)))(2a_{2i+1}Ab + \varepsilon^{(k)}) \\
&= \sum_{i=0}^{d} a_{2i+1}T_{2i+1}(A)b + \sum_{i=0}^{d} \tfrac{1}{2}(U_i(T_2(A^\dagger)) - U_{i-1}(T_2(A^\dagger)))\varepsilon^{(k)}
\end{aligned}$$

In other words, after completing the iteration, we have a vector $\tilde{u}$ such that

$$\begin{aligned}
\|\tilde{u} - p(A)b\| &\leq \Big\| \sum_{i=0}^{d} \tfrac{1}{2}(U_i(T_2(A^\dagger)) - U_{i-1}(T_2(A^\dagger)))\varepsilon^{(k)} \Big\| \\
&\leq \sum_{i=0}^{d} (2i+1)\|\varepsilon^{(k)}\| \\
&\lesssim \mu\varepsilon \sum_{k=0}^{d} (\|v_{k+1}\| + |a_{2k+1}|\|b\|) \\
&\leq \varepsilon\|b\| + \mu\varepsilon \sum_{k=1}^{d} \|v_k\| \qquad (36)
\end{aligned}$$

The last step follows from Item 6.22(a). So, it suffices to bound the v_k's. Recalling from Eq. (34), the recursions defining them is

$$\begin{aligned}
v_k &= 4B_\dagger^{(k)}B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b \\
&= 2(2(TAS)^\dagger(TAS) - I)v_{k+1} - v_{k+2} + a_{2k+1}S^\dagger b + 4(B_\dagger^{(k)}B^{(k)} - (TAS)^\dagger(TAS))v_{k+1}
\end{aligned}$$

This is Eq. (Odd Matrix Clenshaw) on the matrix $(TAS)^\dagger$ with an additional error term. Following Remark 6.16, this solves to

$$v_k = \sum_{i=k}^{d} U_{i-k}(T_2(TAS))\Big(a_{2i+1}S^\dagger b + 4(B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\Big).$$

Since $B^{(k)}$ and $B_\dagger^{(k)}$ are all drawn independently, $\boldsymbol{E}[B_\dagger^{(k)}B^{(k)} - (TAS)^\dagger(TAS)]$ is the zero matrix, where the expectation is over the randomness of $B^{(k)}$ and $B_\dagger^{(k)}$. We use the following bound on the variance of the error. In this computation, we use Eq. (13), which states that, for $B = \textsc{best}(A)$ with parameter r and X positive semi-definite, $\boldsymbol{E}[B^\dagger XB] \leq A^\dagger XA + \frac{1}{r}\operatorname{tr}(X)\|A\|_\mathrm{F}^2 I$.

$$\begin{aligned}
&\mathop{\boldsymbol{E}}_k\Big[\|(B_\dagger^{(k)}B^{(k)} - (TAS)^\dagger(TAS))v_{k+1}\|^2\Big] \\
&= \mathop{\boldsymbol{E}}_k\Big[\|B_\dagger^{(k)}B^{(k)}v_{k+1}\|^2\Big] - \|(TAS)^\dagger(TAS)v_{k+1}\|^2 \\
&= \mathop{\boldsymbol{E}}_k\Big[(B^{(k)}v_{k+1})^\dagger(B_\dagger^{(k)})^\dagger B_\dagger^{(k)}(B^{(k)}v_{k+1})\Big] - \|(TAS)^\dagger(TAS)v_{k+1}\|^2 \\
&\leq \mathop{\boldsymbol{E}}_k\Big[(B^{(k)}v_{k+1})^\dagger((TAS)(TAS)^\dagger + \frac{1}{r}\operatorname{tr}(I_s)\|TAS\|_\mathrm{F}^2 I_t)(B^{(k)}v_{k+1})\Big] - \|(TAS)^\dagger(TAS)v_{k+1}\|^2 \\
&\leq \mathop{\boldsymbol{E}}_k\Big[v_{k+1}^\dagger(TAS)^\dagger((TAS)(TAS)^\dagger + \frac{s}{r}\|TAS\|_\mathrm{F}^2 I_t)(TAS)v_{k+1} \\
&\qquad + v_{k+1}^\dagger\frac{1}{r}\operatorname{tr}((TAS)(TAS)^\dagger + \frac{s}{r}\|TAS\|_\mathrm{F}^2 I_t)\|TAS\|_\mathrm{F}^2 v_{k+1}\Big] - \|(TAS)^\dagger(TAS)v_{k+1}\|^2 \\
&= \frac{s}{r}\|TAS\|_\mathrm{F}^2\|TASv_{k+1}\|^2 + \Big(\frac{1}{r} + \frac{st}{r^2}\Big)\|TAS\|_\mathrm{F}^4\|v_{k+1}\|^2 \\
&\leq 4\Big(\frac{s\|A\|_\mathrm{F}^2\|A\|^2}{r} + \frac{\|A\|_\mathrm{F}^4}{r} + \frac{st\|A\|_\mathrm{F}^4}{r^2}\Big)\|v_{k+1}\|^2 \qquad \text{by Eqs. } (\|TAS\|_F \text{ bd}) \text{ and } (\|TAS\| \text{ bd}) \\
&\leq \frac{\delta}{1000d^4}\|A\|^2\|v_{k+1}\|^2, \qquad (37)
\end{aligned}$$

where the last line uses $r = \Theta(d^4\|A\|_\mathrm{F}^2(s+t)\frac{1}{\delta})$ (and is the bottleneck for the choice of r). Let $\boldsymbol{E}_{[k,d]}$ denote taking the expectation over $B^{(i)}$ and $B_\dagger^{(i)}$ for i between k and d (treating T, S as fixed).

$$\bar{v}_k := \mathop{\boldsymbol{E}}_{[k,d]}[v_k] = \sum_{i=k}^{d} U_{i-k}(T_2(TAS))a_{2i+1}S^\dagger b.$$

We first bound the recurrence in expectation, then we bound the second moment.

$$\begin{aligned}
\|\bar{v}_k\| &\leq \Big\|\sum_{i=k}^{d} U_{i-k}(T_2(TAS))a_{2i+1}\Big\|\|S^\dagger b\| && \text{by sub-multiplicativity of } \|\cdot\| \\
&= \Big\|\sum_{i=k}^{d} U_{i-k}(T_2(x))a_{2i+1}\Big\|_{\mathrm{Spec}(TAS)}\|S^\dagger b\| \\
&\leq \Big\|\sum_{i=k}^{d} U_{i-k}(T_2(x))a_{2i+1}\Big\|_{[-1-2\frac{\mu\varepsilon}{d},1+2\frac{\mu\varepsilon}{d}]}\|S^\dagger b\| && \text{by Eq. } (\|TAS\| \text{ bd}) \\
&\leq e\Big\|\sum_{i=k}^{d} U_{i-k}(T_2(x))a_{2i+1}\Big\|_{\sup}\|S^\dagger b\| && \text{by Lemma 3.8, } \mu\varepsilon \leq \tfrac{1}{100d} \\
&\leq 4\Big\|\sum_{i=k}^{d} U_{i-k}(T_2(x))a_{2i+1}\Big\|_{\sup}\|b\| && \text{by Eq. } (\|S^\dagger b\| \text{ bd}) \\
&\leq 4\frac{1}{\mu d}\|b\| && \text{by Item 6.22(b)}
\end{aligned}$$

We now compute the second moment of v_k.

$$\begin{aligned}
\mathop{\boldsymbol{E}}_{[k,d]}\left[\|v_k - \bar{v}_k\|^2\right] &= \mathop{\boldsymbol{E}}_{[k,d]}\left[\Big\|\sum_{i=k}^{d} U_{i-k}(T_2(TAS))4(B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\Big\|^2\right] \\
&= \sum_{i=k}^{d} \mathop{\boldsymbol{E}}_{[k,d]}\left[\|U_{i-k}(T_2(TAS))4(B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\|^2\right] \\
&\leq 16\sum_{i=k}^{d}\Big\|U_{i-k}(T_2(TAS))\Big\|^2 \mathop{\boldsymbol{E}}_{[k,d]}\left[\|(B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\|^2\right] \\
&\leq 16\sum_{i=k}^{d} e^2d^2 \mathop{\boldsymbol{E}}_{[k,d]}\left[\|(B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\|^2\right] \\
&\leq 16e^2\sum_{i=k}^{d}\frac{d^2\delta}{1000d^4}\|A\|^2 \mathop{\boldsymbol{E}}_{[i+1,d]}\left[\|v_{i+1}\|^2\right] \\
&\leq \frac{\delta}{2d^2}\|A\|^2\sum_{i=k}^{d}\mathop{\boldsymbol{E}}_{[i+1,d]}\left[\|v_{i+1}\|^2\right] \\
&= \frac{\delta}{2d^2}\|A\|^2\sum_{i=k}^{d}\left(\mathop{\boldsymbol{E}}_{[i+1,d]}\left[\|v_{i+1}-\bar{v}_{i+1}\|^2\right] + \|\bar{v}_{i+1}\|^2\right) \\
&\leq \frac{\delta}{2d^2}\|A\|^2\sum_{i=k}^{d}\left(\mathop{\boldsymbol{E}}_{[i+1,d]}\left[\|v_{i+1}-\bar{v}_{i+1}\|^2\right] + \frac{16}{\mu^2d^2}\|b\|^2\right) && \text{by Eq. (37)}
\end{aligned}$$

In the second line, we used that the $B^{(i)}$'s are independent, so the variance of the sum is the sum of the variances. To bound this recurrence, we define the following recurrence c_k to satisfy $\boldsymbol{E}_{[k,d]}[\|v_k - \bar{v}_k\|^2] \leq c_k$:

$$c_k = \gamma \sum_{i=k}^{d} (c_{i+1} + \Gamma) \qquad \gamma = \frac{\delta}{2d^2}\|A\|^2,\ \Gamma = \frac{16}{\mu^2 d^2}\|b\|^2.$$

For this recurrence, $c_k \leq d\gamma\Gamma$ for all k between 0 and d provided that $d\gamma \leq \frac{1}{2}$.

$$\leq \frac{8\delta}{\mu^2 d^3}\|A\|^2\|b\|^2$$

We have shown that $\boldsymbol{E}[\|v_k - \bar{v}_k\|^2] \leq \frac{8\delta}{d}\Big(\frac{\|A\|\|b\|}{\mu d}\Big)^2$. By Markov's inequality, with probability $\geq 1 - \delta/100$, we have that for all k, $\|v_k\| \lesssim \frac{\|b\|}{\mu d}$. Returning to the final error bound Eq. (36),

$$\|\tilde{u} - p(A)b\| \lesssim \varepsilon\|b\| + \mu\varepsilon \sum_{k=1}^{d} \|v_k\| \lesssim \varepsilon\|b\|. \tag{38}$$

Output description properties. After the iteration concludes, we can compute u by computing $x = \frac{1}{2}S(v_0 - v_1)$ in linear $\mathcal{O}(s)$ time. Then, $u = \frac{1}{2}(u_0 - u_1) = \frac{1}{2}AS(v_0 - v_1) = Ax$. Note that though $x \in \mathbb{C}^n$, its sparsity is at most the sparsity of x, which is bounded by s.

Further, using the prior bounds on v_0 and v_1, we have that

$$\begin{aligned}
\sum_{j=1}^{n} \|A(\cdot,j)\|^2 |x(i)|^2 &= \sum_{j=1}^{s} \|[SA](\cdot,j)\|^2 |\tfrac{1}{2}[v_0 - v_1](i)|^2 \\
&\leq \sum_{j=1}^{s} \tfrac{2}{s}\|A\|_{\mathrm{F}}^2 |\tfrac{1}{2}[v_0 - v_1](i)|^2 \\
&\leq \tfrac{2}{s}\|A\|_{\mathrm{F}}^2 \|\tfrac{1}{2}(v_0 - v_1)\|^2 \\
&\leq \tfrac{2}{s}\|A\|_{\mathrm{F}}^2 (\|v_0\|^2 + \|v_1\|^2) \\
&\lesssim \|A\|_{\mathrm{F}}^2 \|b\|^2 / (\sqrt{s}\mu d)^2 \\
&\lesssim \varepsilon^2 \|b\|^2 / (d^4 \log \tfrac{\|A\|_{\mathrm{F}}}{\delta\|A\|}).
\end{aligned}$$

This completes the proof for Theorem 6.22.

We also obtain an analogous result for even polynomials. For the most part, changes are superficial; the purple text indicates differences from Theorem 6.22. The major difference is that the representation of the output is $Ax + \eta b$ instead of Ax, which results from constant terms being allowed when p is even. We state the theorem for estimating $p(A^\dagger)b$ because it makes the similarities with the odd setting more apparent.

Theorem 6.26. *Suppose we are given sampling and query access to $A \in \mathbb{C}^{m\times n}$ and $b \in \mathbb{C}^m$ with $\|A\| \le 1$; a $(2d)$-degree even polynomial, written in its Chebyshev coefficients as*

$$p(x) = \sum_{i=0}^{d} a_{2i} T_{2i}(x);$$

an accuracy parameter $\varepsilon > 0$; a failure probability parameter $\delta > 0$; and a stability parameter $\mu > 0$. Then we can output a vector $x \in \mathbb{C}^n$ and $\eta \in \mathbb{C}$ such that $\|Ax + \eta b - p(A^\dagger)b\| \le \varepsilon\|p\|_{\sup}\|b\|$ with probability $\ge 1-\delta$ in time

$$\mathcal{O}\left(\min\left\{\operatorname{nnz}(A), \frac{d^4\|A\|_F^4}{(\mu\varepsilon)^4}\log^2\left(\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|}\right)\right\} + \frac{d^7\|A\|_F^4}{(\mu\varepsilon)^2\delta}\log\left(\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|}\right)\right),$$

assuming $\mu\varepsilon < \min(\frac{1}{4}d\|A\|, \frac{1}{100d})$ and μ satisfies the following bounds. Below, $\tilde{a}_{2k} := a_{2k} - a_{2k+2} + \cdots \pm a_{2d}$.

(a) *$\mu\sum_{i=1}^{d}|\tilde{a}_{2i}| \le \|p\|_{\sup}$ and $d\mu^2\sum_{i=1}^{d}|\tilde{a}_{2i}|^2 \le \|p\|_{\sup}^2$;*

(b) *$\mu\|\sum_{i=k}^{d} 4\tilde{a}_{2i+2}x \cdot U_{i-k}(T_2(x))\|_{\sup} \le \frac{1}{d}\|p\|_{\sup}$ for all $0 \le k \le d$.*

The output description has the additional properties

$$\sum_j \|A(\cdot,j)\|^2|x(j)|^2 \lesssim \frac{\varepsilon^2\|p\|_{\sup}^2\|b\|^2}{d^4\log\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|}} \qquad \|x\|_0 \lesssim \frac{d^2\|A\|_{\mathrm{F}}^2}{(\mu\varepsilon)^2}\log\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|},$$

so that by Corollary 4.10, for the output vector $y := Ax + \eta b$, we can:

(i) *Compute entries of y in $\mathcal{O}(\|x\|_0) = \mathcal{O}\Big(\frac{d^2\|A\|_{\mathrm{F}}^2}{(\mu\varepsilon)^2}\log\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|}\Big)$ time;*

(ii) *Sample $i \in [n]$ with probability $\frac{|y_i|^2}{\|y\|^2}$ in $\mathcal{O}\Big(\big(\frac{\|p\|_{\sup}^2\|A\|_{\mathrm{F}}^2}{(\mu d)^2} + p(0)^2\big)\frac{d^2\|A\|_{\mathrm{F}}^2\|b\|^2}{\mu^2\varepsilon^2\|y\|^2}\log(\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|})\log\frac{1}{\delta}\Big)$ time with probability $\ge 1-\delta$;*

(iii) *Estimate $\|y\|^2$ to ν relative error in $\mathcal{O}\Big(\big(\frac{\|p\|_{\sup}^2\|A\|_{\mathrm{F}}^2}{(\mu d)^2} + p(0)^2\big)\frac{d^2\|A\|_{\mathrm{F}}^2\|b\|^2}{\nu^2\mu^2\varepsilon^2\|y\|^2}\log(\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|})\log\frac{1}{\delta}\Big)$ time with probability $\ge 1-\delta$.*

Corollary 6.27 (Corollary of Proposition 6.29). *In Theorem 6.26, we can always take* $1/\mu \lesssim d^2 \log(d)$ *for* $d > 1$.

Algorithm 2 (Even singular value transformation).

Input (pre-processing): A matrix $A \in \mathbb{C}^{m\times n}$, vector $b \in \mathbb{C}^m$, and parameters ε, δ, $\mu > 0$.

Pre-processing sketches: Let $s,t = \Theta(\frac{d^2\|A\|_F^2}{(\mu\varepsilon)^2}\log(\frac{\|A\|_F}{\delta\|A\|}))$. This phase will succeed with probability $\geq 1-\delta$.

P1. If $\mathrm{SQ}(A^\dagger)$ and $\mathrm{SQ}(b)$ are not given, compute data structures to simulate them in $\mathcal{O}(1)$ time;

P2. Sample $S \in \mathbb{C}^{n\times s}$ from $\{\frac{\|A(\cdot,j)\|^2}{\|A\|_F^2}\}_{j\in[n]}$;

P3. Sample $T^\dagger \in \mathbb{C}^{m\times t}$ from $\{\frac{1}{2}(\frac{\|[AS](i,\cdot)\|^2}{\|AS\|_F^2} + \frac{|b(i)|^2}{\|b\|^2})\}_{i\in[m]}$;

P4. Compute a data structure that can respond to $\mathrm{SQ}(TAS)$ queries in $\mathcal{O}(1)$ time;

Input: A degree $2d$ polynomial $p(x) = \sum_{i=0}^{d} a_{2i}T_{2i}(x)$ given as its coefficients a_{2i}.

Compute all $\tilde{a}_{2k} = a_{2k} - a_{2k+2} + \cdots \pm a_{2d}$.

Clenshaw iteration: Let $r = \Theta(d^4\|A\|_F^2(s+t)\frac{1}{\delta}) = \Theta(d^6\frac{\|A\|_F^4}{(\mu\varepsilon)^2\delta}\log\frac{\|A\|_F}{\delta\|A\|})$. This phase will succeed with probability $\geq 1-\delta$. Starting with $v_{d+1} = v_{d+2} = \vec{0}^s$ and going until v_0,

I1. Let $B^{(k)} = \textsc{best}_r(TAS)$ and $B_\dagger^{(k)} = \textsc{best}_r((TAS)^\dagger)$ (Definition 5.19);

I2. Compute $v_k = 2(2B_\dagger^{(k)}B^{(k)} - I)v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}B_\dagger^{(k)}Tb$.

Output: Output $x = \frac{1}{2}S(v_0 - v_1)$ and $\eta = \tilde{a}_0$ satisfying $\|Ax + \eta b - p(A^\dagger)b\| \leq \varepsilon\|p\|_{\sup}\|b\|$.

Recall the odd and even recurrences defined in Eqs. (Odd Clenshaw) and (Even Clenshaw).

$$u_k = 2(2AA^\dagger - I)u_{k+1} - u_{k+2} + 2a_{2k+1}Ab, \qquad \text{(Odd Matrix Clenshaw)}$$
$$p(A)b = u = \tfrac{1}{2}(u_0 - u_1).$$

The matrix analogue of the even recurrence is identical except that the final term is $4\tilde{a}_{2k+2}AA^\dagger b$

instead of $2a_{2k+1}Ab$.

$$\begin{aligned} \tilde{a}_{2k} &:= a_{2k} - a_{2k+2} + a_{2k+4} - \cdots \pm a_{2d} \\ u_k &= 2(2AA^\dagger - 1)u_{k+1} - u_{k+2} + 4\tilde{a}_{2k+2}AA^\dagger b, \qquad \text{(Even Matrix Clenshaw)} \\ p(A^\dagger)b = u &= \tilde{a}_0 b + \tfrac{1}{2}(u_0 - u_1). \end{aligned}$$

So, a roughly identical analysis works upon making the appropriate changes. As before, we assume $\|p\|_{\sup} = 1$ without loss of generality.

Pre-processing sketches, correctness. Though the sketches are chosen to be slightly different because of the different parity, all of the sketching bounds used for the odd SVT analysis hold here, up to rescaling s, t by constant factors. This includes Eqs. ($\|[AS](\cdot, j)\|$ bd), ($\|AS\|_F$ bd), ($AA^\dagger$ AMP), ($\|AS\|$ bd), ($\|TAS\|_F$ bd), ($(AS)^\dagger AS$ AMP) and ($\|TAS\|$ bd). What remains (Eqs. ($\|S^\dagger b\|$ bd) and (Ab AMP)) have analogues that follow from the same argument:

$$\|Tb\|^2 \leq 2\|b\|^2 \qquad (\|Tb\| \text{ bd})$$

$$\|(AS)^\dagger b - (TAS)^\dagger Tb\| \leq \frac{\mu\varepsilon}{d}\|b\| \qquad ((AS)^\dagger b \text{ AMP})$$

All this holds with probability $\geq 1 - \delta$.

One (even) Clenshaw iteration. As with the odd case, we perform the recurrence on u_k by updating v_k such that $u_k = (AS)v_k$. The error analysis proceeds by bounding

$$\begin{aligned}
&4AA^\dagger u_{k+1} - 2u_{k+1} - u_{k+2} + 4\tilde{a}_{2k+2}AA^\dagger b \\
&\qquad \approx_1 4AS(AS)^\dagger u_{k+1} - 2u_{k+1} - u_{k+2} + 4\tilde{a}_{2k+2}AA^\dagger b \\
&\qquad = AS(4(AS)^\dagger(AS)v_{k+1} - 2v_{k+1} - v_{k+2}) + 4\tilde{a}_{2k+2}AA^\dagger b \\
&\qquad \approx_2 AS(4(TAS)^\dagger(TAS)v_{k+1} - 2v_{k+1} - v_{k+2}) + 4\tilde{a}_{2k+2}AA^\dagger b \\
&\qquad \approx_3 AS(4(TAS)^\dagger(TAS)v_{k+1} - 2v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}(AS)^\dagger b) \\
&\qquad \approx_4 AS(4(TAS)^\dagger B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}(AS)^\dagger b) \\
&\qquad \approx_5 AS(4B_\dagger^{(k)}B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}(AS)^\dagger b) \\
&\qquad \approx_6 AS(4(TAS)^\dagger(TAS)v_{k+1} - 2v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}(TAS)^\dagger Tb) \\
&\qquad \approx_7 AS(4B_\dagger^{(k)}B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}B_\dagger^{(k)}Tb)
\end{aligned} \tag{39}$$

So, our update is

$$v_k = 4B_\dagger^{(k)}B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}B_\dagger^{(k)}Tb. \tag{40}$$

As before, we can label the error incurred by each approximation in Eq. (39) with $\varepsilon_1, \dots, \varepsilon_7$. The approximations in $\varepsilon_1, \varepsilon_2, \varepsilon_4, \varepsilon_5$ do not involve the constant term and so can be bounded identically to the odd case.

$$\begin{aligned}
\varepsilon_1 &\leq 4\|AA^\dagger - AS(AS)^\dagger\|\|u_{k+1}\| \leq 4\tfrac{\mu\varepsilon}{d}\|A\|\|u_{k+1}\| \\
\varepsilon_2 &\leq 4\|AS\|\|((AS)(AS)^\dagger - (TAS)(TAS)^\dagger)v_{k+1}\| \leq 16\tfrac{\mu\varepsilon}{d}\|A\|^2\|v_{k+1}\| \\
\varepsilon_4 &\leq 4\|AS(TAS)^\dagger(TAS - B^{(k)})v_{k+1}\| \leq d^{-5/2}\mu\varepsilon\|A\|\|v_{k+1}\| \\
\varepsilon_5 &\leq 4\|AS((TAS)^\dagger - B_\dagger^{(k)})B^{(k)}v_{k+1}\| \leq d^{-5/2}\mu\varepsilon\|v_{k+1}\|.
\end{aligned}$$

The approximation in ε_3 goes through with a slight modification.

$$\varepsilon_3 \leq |4\tilde{a}_{2k+2}|\|AA^\dagger b - AS(AS)^\dagger b\| \leq 4|\tilde{a}_{2k+2}|\tfrac{\mu\varepsilon}{d}\|A\|\|b\| \qquad \text{by Eq. } (AA^\dagger \text{ AMP})$$

The approximations ε_6 and ε_7 follow from similar arguments.

$$\begin{aligned}
\varepsilon_6 &\leq |4\tilde{a}_{2k+2}|\|AS\|\|(AS)^\dagger b - (TAS)^\dagger Tb\| \\
&\leq |8\tilde{a}_{2k+2}|\tfrac{\mu\varepsilon}{d}\|b\| && \text{by Eqs. } (\|[AS](\cdot,j)\| \text{ bd}) \text{ and } ((AS)^\dagger b \text{ AMP}) \\
\varepsilon_7 &\leq |4\tilde{a}_{2k+2}|\|AS((TAS)^\dagger - B_\dagger^{(k)})Tb\| \\
&\leq |4\tilde{a}_{2k+2}|\|AS\|_{\mathrm{F}}\|TAS\|_{\mathrm{F}}\|Tb\|\sqrt{d/(r\delta)} && \text{by Corollary 5.21} \\
&\leq |\tilde{a}_{2k+2}|d^{-5/2}\mu\varepsilon\|b\| && \text{by Eqs. } (\|TAS\| \text{ bd}) \text{ and } (\|Tb\| \text{ bd})
\end{aligned}$$

In summary, we can view the iterate of A1.I2 as computing

$$\tilde{u}_k = 2(2AA^\dagger - I)\tilde{u}_{k+1} - \tilde{u}_{k+2} + 4\tilde{a}_{2k+2}AA^\dagger b + \varepsilon^{(k)} \tag{41}$$

Where $\varepsilon^{(k)} \in \mathbb{C}^m$ is the error of the approximation in the iterate Eq. (33), and

$$\begin{aligned}
\|\varepsilon^{(k)}\| &\leq \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 + \varepsilon_5 + \varepsilon_6 + \varepsilon_7 \\
&\lesssim \tfrac{\mu\varepsilon}{d}\Big(\|v_{k+1}\| + |\tilde{a}_{2k+2}|\|b\|\Big).
\end{aligned}$$

Upon applying a union bound, we see that this bound on $\varepsilon^{(k)}$ holds for every k from 0 to $d-1$ with probability $\geq 1 - 4\delta$.

Error accumulation across iterations. The error accumulates in the same way as in the odd setting. Using the formulation of the iterate from Eq. (41), we notice that this is the standard Clenshaw iteration Eq. (Clenshaw) with x replaced with $T_2(A^\dagger) = 2AA^\dagger - I$ and a_k replaced with $4\tilde{a}_{2k+2}AA^\dagger b + \varepsilon^{(k)}$. Following Remark 6.16 and Lemma 6.15, we conclude that the output of Algorithm 2 satisfies

$$\begin{aligned}
\tilde{u}_k &= \sum_{i=k}^{d} U_{i-k}(T_2(A^\dagger))(4\tilde{a}_{2i+2}AA^\dagger b + \varepsilon^{(i)}) \\
\tilde{u} &:= \tilde{a}_0 + \frac{1}{2}(\tilde{u}_0 - \tilde{u}_1) \\
&= \tilde{a}_0 + \sum_{i=0}^{d} \tfrac{1}{2}(U_i(T_2(A^\dagger)) - U_{i-1}(T_2(A^\dagger)))(4\tilde{a}_{2i+2}AA^\dagger b + \varepsilon^{(k)}) \\
&= \sum_{i=0}^{d} a_{2i}T_{2(i-k)}(A^\dagger)b + \sum_{i=0}^{d} \tfrac{1}{2}(U_i(T_2(A^\dagger)) - U_{i-1}(T_2(A^\dagger)))\varepsilon^{(k)}
\end{aligned}$$

In other words, after completing the iteration, we have a vector $\tilde{u}$ such that

$$\begin{aligned}
\|\tilde{u} - p(A^\dagger)b\| &\leq \Big\|\sum_{i=0}^{d} \tfrac{1}{2}(U_i(T_2(A^\dagger)) - U_{i-1}(T_2(A^\dagger)))\varepsilon^{(k)}\Big\| \\
&\leq \sum_{i=0}^{d}(2i+1)\|\varepsilon^{(k)}\| \\
&\lesssim \mu\varepsilon \sum_{k=0}^{d}(\|v_{k+1}\| + |\tilde{a}_{2k+2}|\|b\|) \\
&\leq \varepsilon\|b\| + \mu\varepsilon \sum_{k=1}^{d} \|v_k\| \qquad (42)
\end{aligned}$$

In the last line, we use the assumption Item 6.26(a). It suffices to bound the v_k's. Recalling from Eq. (40), the recursions defining them is

$$\begin{aligned}
v_k &= 4B_\dagger^{(k)}B^{(k)}v_{k+1} - 2v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}B_\dagger^{(k)}Tb \\
&= 2(2(TAS)^\dagger(TAS) - I)v_{k+1} - v_{k+2} + 4\tilde{a}_{2k+2}B_\dagger^{(k)}Tb + 4(B_\dagger^{(k)}B^{(k)} - (TAS)^\dagger(TAS))v_{k+1}
\end{aligned}$$

Following Remark 6.16, this solves to

$$v_k = \sum_{i=k}^{d} U_{i-k}(T_2(TAS))\Big(4\tilde{a}_{2i+2}B_\dagger^{(i)}Tb + 4(B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\Big).$$

From here, the identical analysis applies.

$$\begin{aligned}\bar{v}_k := \mathop{\boldsymbol{E}}_{[k,d]}[v_k] &= \sum_{i=k}^{d} U_{i-k}(T_2(TAS))4\tilde{a}_{2i+2}(TAS)^\dagger Tb\\ \|\bar{v}_k\| &\leq \Big\|\sum_{i=k}^{d} U_{i-k}(T_2(TAS))4\tilde{a}_{2i+2}(TAS)^\dagger\Big\|\|Tb\| \leq 4\frac{1}{\mu d}\|b\|.\end{aligned}$$

We now compute the second moment of v_k. The main difference from the odd setting is that there is an additional term, $4\tilde{a}_{2k+2}(B_\dagger^{(k)} - (TAS)^\dagger)Tb$, where

$$\begin{aligned}\mathop{\boldsymbol{E}}_{k}\|(B_\dagger^{(k)} - (TAS)^\dagger)Tb\|^2 &\leq \frac{1}{r}\operatorname{tr}(I_s)\|TAS\|_F^2\|Tb\|^2 && \text{by Eq. (13)}\\ &\leq \frac{\delta}{1000d^4}\|b\|^2. && \text{by Eq. } (\|Tb\| \text{ bd}),\ r = \Theta(d^4\|A\|_F^2(s+t)\tfrac{1}{\delta})\end{aligned}$$

We use this and the derivation in Eq. (37) to conclude

$$\begin{aligned}&\mathop{\boldsymbol{E}}_{[k,d]}\left[\|v_k - \bar{v}_k\|^2\right]\\ &= \mathop{\boldsymbol{E}}_{[k,d]}\left[\Big\|\sum_{i=k}^{d} U_{i-k}(T_2(TAS))4(\tilde{a}_{2i+2}(B_\dagger^{(i)} - (TAS)^\dagger)Tb + (B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1})\Big\|^2\right]\\ &= \sum_{i=k}^{d}\mathop{\boldsymbol{E}}_{[k,d]}\left[\|U_{i-k}(T_2(TAS))4(\tilde{a}_{2i+2}(B_\dagger^{(i)} - (TAS)^\dagger)Tb + (B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1})\|^2\right]\\ &\leq 16\sum_{i=k}^{d}\Big\|U_{i-k}(T_2(TAS))\Big\|^2\mathop{\boldsymbol{E}}_{[k,d]}\left[\|\tilde{a}_{2i+2}(B_\dagger^{(i)} - (TAS)^\dagger)Tb + (B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\|^2\right]\\ &\leq 32\sum_{i=k}^{d} e^2d^2\mathop{\boldsymbol{E}}_{[k,d]}\left[\|\tilde{a}_{2i+2}(B_\dagger^{(i)} - (TAS)^\dagger)Tb\|^2 + \|(B_\dagger^{(i)}B^{(i)} - (TAS)^\dagger(TAS))v_{i+1}\|^2\right]\\ &\leq 32\sum_{i=k}^{d} e^2d^2\mathop{\boldsymbol{E}}_{[k,d]}\left[\frac{\delta|\tilde{a}_{2i+2}|^2}{1000d^4}\|b\|^2 + \frac{\delta}{1000d^4}\|A\|^2\|v_{k+1}\|^2\right]\\ &\leq \frac{\delta}{4d^2}\|b\|^2\sum_{i=k}^{d}|\tilde{a}_{2i+2}|^2 + \frac{\delta\|A\|^2}{4d^2}\sum_{i=k}^{d}\mathop{\boldsymbol{E}}_{[k,d]}\left[\|v_{k+1}\|^2\right]\\ &\leq \frac{\delta}{4\mu^2d^3}\|b\|^2 + \frac{\delta\|A\|^2}{4d^2}\sum_{i=k}^{d}\mathop{\boldsymbol{E}}_{[k,d]}\left[\|v_{k+1}\|^2\right] \qquad \text{by Item 6.26(a)}\\ &= \frac{\delta}{4\mu^2d^3}\|b\|^2 + \frac{\delta\|A\|^2}{4d^2}\sum_{i=k}^{d}\left(\mathop{\boldsymbol{E}}_{[i+1,d]}\left[\|v_{i+1} - \bar{v}_{i+1}\|^2\right] + \|\bar{v}_{i+1}\|^2\right)\end{aligned}$$

$$\leq \frac{\delta}{4\mu^2 d^3}\|b\|^2 + \frac{\delta\|A\|^2}{4d^2}\sum_{i=k}^{d}\left(\underset{[i+1,d]}{\boldsymbol{E}}\left[\|v_{i+1}-\bar{v}_{i+1}\|^2\right] + \frac{16}{\mu^2 d^2}\|b\|^2\right) \qquad \text{by Eq. (37)}$$

$$\leq \frac{\delta\|A\|^2}{4d^2}\sum_{i=k}^{d}\left(\underset{[i+1,d]}{\boldsymbol{E}}\left[\|v_{i+1}-\bar{v}_{i+1}\|^2\right] + \frac{17}{\mu^2 d^2}\|b\|^2\right)$$

By the same recurrence argument, this is

$$\leq \frac{9\delta}{\mu^2 d^3}\|A\|^2\|b\|^2 \tag{43}$$

We have shown that $\boldsymbol{E}[\|v_k - \bar{v}_k\|^2] \leq \frac{9\delta}{d}\left(\frac{\|A\|\|b\|}{\mu d}\right)^2$. By Markov's inequality, with probability $\geq 1-\delta/100$, we have that for all k, $\|v_k\| \lesssim \frac{\|b\|}{\mu d}$. Returning to the final error bound Eq. (42),

$$\|\tilde{u} - p(A^\dagger)b\| \lesssim \varepsilon\|b\| + \mu\varepsilon\sum_{k=1}^{d}\|v_k\| \lesssim \varepsilon\|b\|. \tag{44}$$

Output description properties. The argument from the odd case shows that

$$\sum_j \|A(\cdot,j)\|^2 |x(j)|^2 \lesssim \frac{\varepsilon^2\|b\|^2}{d^4\frac{\|A\|_{\mathrm{F}}}{\delta\|A\|}}$$

and $\|x\|_0 \leq s$. By Corollary 4.10 we get the desired bounds. Notice that $\tilde{a}_0 = a_0 - a_2 + a_4 - \cdots \pm a_{2d} = p(0)$. This completes the proof of Theorem 6.26.

Finally, we will give bounds on the value of μ that suffices for a generic polynomial. Though bounds may be improvable for specific functions, they are tight up to log factors for Chebyshev polynomials $T_k(x)$, and improve by a factor of d over naive coefficient-wise bounds.

Proposition 6.28. *Let $p(x)$ be an odd polynomial with degree $2d+1$, with a Chebyshev series expansion of $p(x) = \sum_{i=0}^{d} a_{2i+1}T_{2i+1}(x)$. Then $\sum_{i=0}^{d}|a_{2i+1}| \leq 2d\|p\|_{\sup}$ and, for all integers $k \leq d$,*

$$\Big\|\sum_{i=k}^{d} a_{2i+1}U_{i-k}(T_2(x))\Big\|_{\sup} \lesssim (d-k+1)(1+\log^2(d+1))\|p\|_{\sup}$$

Proof. Without loss of generality, we take $\|p\|_{\sup} = 1$. By Lemma 3.7, $|a_{2i+1}| \leq 2$, giving the first conclusion. Towards the second conclusion, we first note that

$$\Big\|\sum_{i=k}^{d} a_{2i+1}U_{i-k}(T_2(x))\Big\|_{\sup} = \Big\|\sum_{i=k}^{d} a_{2i+1}U_{i-k}(x)\Big\|_{\sup},$$

since T_2 maps $[-1,1]$ to $[-1,1]$. Then, we use the strategy from Theorem 6.20, writing out U_{i-k} with Eq. (7) and then bounding the resulting coefficients. Note that, by convention, a_i and U_i are zero for any integer $i \in \mathbb{Z}$ for which they are not defined.

$$\begin{aligned}
\sum_{i=k}^{d} a_{2i+1}U_{i-k}(x) &= \sum_{i} a_{2i+1}U_{i-k}(x) \\
&= \sum_{i} a_{2i+1} \sum_{j\geq 0} T_{i-k-2j}(x)(1 + [\![i-k-2j \neq 0]\!]) \\
&= \sum_{j\geq 0}\sum_{i} a_{2i+1} T_{i-k-2j}(x)(1 + [\![i-k-2j \neq 0]\!]) \\
&= \sum_{j\geq 0}\sum_{i} a_{2(i+k+2j)+1} T_{i}(x)(1 + [\![i \neq 0]\!]) \\
&= \sum_{i} T_i(x)(1 + [\![i \neq 0]\!]) \sum_{j\geq 0} a_{2(i+k+2j)+1} \\
\Big\|\sum_{i=k}^{d} a_{2i+1}U_{i-k}(x)\Big\|_{\sup} &\leq \sum_{i}(1 + [\![i \neq 0]\!])\|T_i(x)\|_{\sup}\Big|\sum_{j\geq 0} a_{2(i+k+2j)+1}\Big| \\
&= \sum_{i=0}^{d-k}(1 + [\![i \neq 0]\!])\Big|\sum_{j\geq 0} a_{2(i+k)+1+4j}\Big| \\
&\leq \sum_{i=0}^{d-k}(1 + [\![i \neq 0]\!])(32 + 8\log^2(2d+2)) && \text{by Corollary 6.8} \\
&= (2(d-k)+1)(32 + 8\log^2(2d+2)) \\
&\lesssim (d-k+1)(1 + \log^2(d+1)) && \square
\end{aligned}$$

Proposition 6.29. *Let $p(x)$ be an even polynomial with degree $2d$, written as $p(x) = \sum_{i=0}^{d} a_{2i}T_{2i}(x)$.*

Let $\tilde{a}_{2k} := a_{2k} - a_{2k+2} + \cdots \pm a_{2d}$. Then

$$\sum_{i=0}^{d} |\tilde{a}_{2i}| \leq 4(d+1)(1+\log(d+1))\|p\|_{\sup}$$
$$\sum_{i=0}^{d} |\tilde{a}_{2i}|^2 \leq 32(d+1)(1+\log^2(d+1))\|p\|_{\sup}$$

and, for all integers $k \leq d$,

$$\Big\|\sum_{i=k}^{d} 4\tilde{a}_{2i+2}x \cdot U_{i-k}(T_2(x))\Big\|_{\sup} \lesssim (d-k+1)(1+\log(d+1))\|p\|_{\sup}.$$

Proof. Without loss of generality, we take $\|p\|_{\sup} = 1$. Consider the polynomial $q(x) = \sum_{i=0}^{d} b_i T_i(x)$ where $b_i := a_{2i}$. By Eq. (8), $p(x) = q(T_2(x))$, and because T_2 maps $[-1,1]$ to $[-1,1]$, $\|q\|_{\sup} = \|q\|_{\sup} = 1$. Then by Fact 6.6,

$$|\tilde{a}_{2k}| = |b_k - b_{k+1} + \cdots \pm b_d| \leq \Big(4 + \frac{4}{\pi^2}\log(\max(k,1))\Big)\|q\|_{\sup} = 4 + \frac{4}{\pi^2}\log(\max(k,1))$$

From this follows the first two conclusions. For the final conclusion, by Eq. (9),

$$\sum_{i=k}^{d} 4\tilde{a}_{2i+2}xU_{i-k}(T_2(x)) = \sum_{i=k}^{d} 2\tilde{a}_{2i+2}U_{2(i-k)+1}(x) = \sum_{i} 2\tilde{a}_{2i+2}U_{2(i-k)+1}(x)$$

From here, we proceed as in the proof of Theorem 6.20.

$$\begin{aligned}
\sum_i \tilde{a}_{2i+2}U_{2(i-k)+1}(x) &= \sum_i \sum_{r\geq 0} (-1)^r b_{i+r+1} 2 \sum_s T_{2s+1}(x)[\![s \leq i-k]\!] \\
&= 2\sum_s T_{2s+1}(x) \sum_i \sum_{r\geq 0} [\![s \leq i-k]\!](-1)^r b_{i+r+1} \\
&= 2\sum_s T_{2s+1}(x) \sum_t b_t \sum_i \sum_r [\![r \geq 0]\!][\![t = i+r+1]\!][\![s \leq i-k]\!](-1)^r \\
&= 2\sum_s T_{2s+1}(x) \sum_t b_t \sum_r [\![r \geq 0]\!][\![s \leq t-r-1-k]\!](-1)^r \\
&= 2\sum_s T_{2s+1}(x) \sum_t b_t \sum_{r=0}^{t-s-k-1} (-1)^r
\end{aligned}$$

$$
\begin{aligned}
&= 2\sum_s T_{2s+1}(x)\sum_t b_t [\![t-s-k-1 \in 2\mathbb{Z}_{\geq 0}]\!] \\
&= 2\sum_s T_{2s+1}(x)\sum_{t\geq 0} b_{2t+s+k+1} \\
\Big\|\sum_i \tilde{a}_{2i+2}U_{2(i-k)+1}(x)\Big\|_{\sup} &\leq 2\sum_{s=0}^{d-k}\Big|\sum_{t\geq 0} b_{2t+s+k+1}\Big| \\
&\leq 2\sum_{s=0}^{d-k}\Big(4+\frac{4}{\pi^2}\log(\max(s+k,1))\Big) && \text{by Fact 6.6} \\
&\lesssim (d-k+1)(1+\log(d+1)) && \square
\end{aligned}
$$

7 Singular value transformation

The algorithm presented in Section 6 suffices to dequantize basic applications singular value transformation framework, but is less flexible because it requires a vector b to apply matrices to. We will now present more flexible tools for singular value transformation. Our main result is that, given $\mathrm{SQ}_\phi(A)$ and a smooth function f, we can approximate $f(A^\dagger A)$ by a decomposition $R^\dagger UR + f(0)I$. This primitive is based on the even singular value transformation used by Gilyén, Su, Low, and Wiebe [GSLW19].

Theorem 7.1 (Even singular value transformation). *Let $A \in \mathbb{C}^{m\times n}$ and $f : \mathbb{R}^+ \to \mathbb{C}$ be such that $f(x)$ and $\bar{f}(x) := (f(x)-f(0))/x$ are L-Lipschitz and $\bar{L}$-Lipschitz, respectively, on $\cup_{i=1}^{\min(m,n)}[\sigma_i^2 - d, \sigma_i^2 + d]$ for some $d > 0$. Take parameters ε and δ such that $0 < \varepsilon \lesssim \min(L\|A\|_*^2, \bar{L}\|A\|_*^2\|A\|^2)$ and $\delta \in (0,1]$. Choose a norm $* \in \{\mathrm{F}, \mathrm{Op}\}$.*

Suppose we have $\mathrm{SQ}_\phi(A)$. Consider the importance sampling sketch $S \in \mathbb{R}^{r\times m}$ corresponding to $\mathrm{SQ}_\phi(A)$ and the importance sampling sketch $T^\dagger \in \mathbb{R}^{c\times n}$ corresponding to $\mathrm{SQ}_{\leq 2\phi}((SA)^\dagger)$ (which we have by Lemma 5.3). Then, for $R := SA$ and $C := SAT$, we can achieve the bound

$$
\Pr\Big[\|R^\dagger \bar{f}(CC^\dagger)R + f(0)I - f(A^\dagger A)\|_* > \varepsilon\Big] < \delta, \tag{45}
$$

if $r, c > \|A\|^2\|A\|_\mathrm{F}^2\phi^2\frac{1}{d^2}\log\frac{1}{\delta}$ (or, equivalently, $d > \bar{\varepsilon} := \|A\|_*\|A\|_\mathrm{F}(\frac{\phi^2\log(1/\delta)}{\min(r,c)})^{1/2}$) and

$$
r = \tilde{\Omega}\Big(\phi^2L^2\|A\|_*^2\|A\|_\mathrm{F}^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big) \qquad c = \tilde{\Omega}\Big(\phi^2\bar{L}^2\|A\|^4\|A\|_*^2\|A\|_\mathrm{F}^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big). \tag{46}
$$

First, we make some technical remarks. The assumption that $\varepsilon \lesssim L\|A\|_*^2$ is for non-degeneracy: if $\varepsilon \geq L\|A\|^2$, then the naive approximation $f(0)I$ of $f(A^\dagger A)$ would suffice, since $\|f(0)I - f(A^\dagger A)\| \leq L\|A\|^2 \leq \varepsilon$ as desired.[18] The parameter d (or, rather, the parameter $\bar{\varepsilon}$) specifies the domain where $f(x)$ and $\bar{f}(x)$ should be smooth: the condition in the theorem is that they should be Lipschitz on the spectrum of $A^\dagger A$, with $\bar{\varepsilon}$ room for approximation. This will not come into play often, though, since we can often design our singular value transforms such that we can take $d = \infty$. For example, if our desired transform f becomes non-smooth outside the relevant interval $[0, \|A\|^2]$, we can apply Theorem 7.1 with $d = \infty$ and the function g such that $g(x) = g(\|A\|^2)$ for $x \geq \|A\|^2$ and $g(x) = f(x)$ otherwise. Then $g(A^\dagger A) = f(A^\dagger A)$ and g is smooth everywhere, so we do not need to worry about the d parameter. Finally, we note that no additional log terms are necessary (i.e., $\tilde{\Omega}$ becomes Ω) when the Frobenius norm is used.

By our discussion in Section 5, finding the sketches S and T for Theorem 7.1 takes time $\mathcal{O}((r+c)\,\boldsymbol{s}_\phi(A) + rc\,\boldsymbol{q}_\phi(A) + \boldsymbol{n}_\phi(A))$, querying for all of the entries of C takes additional time $\mathcal{O}(rc\,\boldsymbol{q}(A))$, and computing $\bar{f}(CC^\dagger)$ takes additional time $\mathcal{O}(\min(r^2c, rc^2))$ (if done naively). For our applications, this final matrix function computation will dominate the runtime, and the rest of the cost we will treat as $\mathcal{O}(rc\,\boldsymbol{sq}_\phi(A))$.

For some intuition on error bounds and time complexity, we consider how the parameters in our main theorem behave in a restricted setting: suppose we have $\mathrm{SQ}(A)$ with minimum singular value σ and such that $\|A\|_\mathrm{F}/\sigma$ is dimension-independent.[19] This condition simultaneously bounds the rank and condition number of A. Further suppose[20] that f is L-Lipschitz on the interval $[0, \|A\|^2]$ and satisfies

$$L\|A\|^2 < \Gamma D \text{ where } D := \max_{x\in[0,\|A\|^2]} f(x) - \min_{y\in[0,\|A\|^2]} f(y),$$

for some dimension-independent Γ. Γ must be at least one, so we can think about such an f as being at most Γ times "steeper" compared to the least possible "steepness". Under these

[18]The choice $f(0)I$ assumes that f is Lipschitz on $\{0, \|A\|^2\}$. More generally, we can choose $f(x)I$ for any $x \in \cup_{i=1}^{\min(m,n)}[\sigma_i^2 - d, \sigma_i^2 + d]$ in order to get a sufficiently good naive approximation.

[19]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 $A \leftarrow \alpha A$.

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

assumptions, we can get a decomposition satisfying

$$\|R^\dagger \bar{f}(CC^\dagger)R + f(0)I - f(A^\dagger A)\| > \varepsilon D$$

with probability $\geq 1-\delta$ by taking

$$r = \tilde{\Theta}\Big(\Gamma^2 \frac{\|A\|_{\mathrm{F}}^2}{\|A\|^2}\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big) \text{ and } c = \tilde{\Theta}\Big(\Gamma^2 \frac{\|A\|^2\|A\|_{\mathrm{F}}^2}{\sigma^4}\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big).$$

The time to compute the decomposition is

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

These quantities are all dimensionless. Dependence on σ arises because we bound $\bar{L} \leq L/\sigma^2$: our algorithm's dependence on $\bar{L}$ implicitly enforces a low-rank constraint in this case. This bears a resemblance to the dependence on the smoothness of the "dual" polynomial recurrence in Theorem 6.22. All of our analyses give qualitatively similar results to this, albeit in more general settings allowing approximately low-rank input.

To perform error analyses, we will need bounds on the norms of the matrices in our decomposition. The following lemma gives the bounds we need for Section 8.

Lemma 7.2 (Norm bounds for even singular value transformation). Suppose the assumptions from Theorem 7.1 hold. Then with probability at least $1-\delta$, the event in Eq. (45) occurs (that is, $R^\dagger \bar{f}(CC^\dagger)R \approx f(A^\dagger A) - f(0)I$) and moreover, *the following bounds also hold*:

$$\|R\| = \mathcal{O}(\|A\|) \quad \text{and} \quad \|R\|_{\mathrm{F}} = \mathcal{O}(\|A\|_{\mathrm{F}}), \tag{47}$$

$$\|\bar{f}(CC^\dagger)\| \leq \max\Big\{|\bar{f}(x)| \,\Big|\, x \in \bigcup_{i=1}^{\min(r,c)} [\sigma_i^2 - \bar{\varepsilon}, \sigma_i^2 + \bar{\varepsilon}]\Big\}, \tag{48}$$

$$\text{when } * = \mathrm{Op}, \Big\|R^\dagger \sqrt{\bar{f}(CC^\dagger)}\Big\| \leq \sqrt{\|f(A^\dagger A) - f(0)I\| + \varepsilon}. \tag{49}$$

Eq. (49) is typically a better bound than combining Eqs. (47) and (48). For intuition, notice this is true if $\varepsilon, \bar{\varepsilon} = 0$: the left-hand and right-hand sides of the following inequality are the two ways to bound $\|R^\dagger\sqrt{\bar{f}(CC^\dagger)}\|^2$, up to constant factors (σ below runs over the singular values of A):

$$\|f(A^\dagger A) - f(0)I\| \leq \max_\sigma |f(\sigma^2) - f(0)| \leq \max_\sigma \sigma^2 \max_\sigma \frac{|f(\sigma^2) - f(0)|}{\sigma^2} = \|A\|^2 \max_\sigma |\bar{f}(\sigma^2)|.$$

The rest of this section will be devoted to proving Theorem 7.1 and Lemma 7.2. A mathematical tool we will need is a matrix version of the defining inequality of L-Lipschitz functions, $|f(x) - f(y)| \leq L|x - y|$ when f is L-Lipschitz. The Frobenius norm version of this bound (Lemma 7.3) follows by computing matrix derivatives; the spectral norm version (Lemma 7.4) has a far less obvious proof.

Lemma 7.3 ([Gil10, Corollary 2.3]). Let A and B be Hermitian matrices and let $f : \mathbb{R} \to \mathbb{C}$ be L-Lipschitz continuous on the eigenvalues of A and B. Then $\|f^{(\mathrm{EV})}(A) - f^{(\mathrm{EV})}(B)\|_\mathrm{F} \leq L\|A - B\|_\mathrm{F}$.

Lemma 7.4 ([AP11, Theorem 11.2]). Let A and B be Hermitian matrices and let $f : \mathbb{R} \to \mathbb{C}$ be L-Lipschitz continuous on the eigenvalues of A and B. Then

$$\|f^{(\mathrm{EV})}(A) - f^{(\mathrm{EV})}(B)\| \lesssim L\|A - B\| \log \min(\operatorname{rank} A, \operatorname{rank} B).$$

Proof of Theorem 7.1 and Lemma 7.2. Since $g(A^\dagger A) = f(A^\dagger A) + g(0)I$ for $f(x) := g(x) - g(0)$, we can assume without loss of generality that $f(0) = 0$. As a reminder, in the statement of Theorem 7.1 we take

$$r = \tilde{\Omega}\Big(\phi^2 L^2 \|A\|_*^2 \|A\|_\mathrm{F}^2 \frac{1}{\varepsilon^2} \log\frac{1}{\delta}\Big) \qquad c = \tilde{\Omega}\Big(\phi^2 \bar{L}^2 \|A\|^4 \|A\|_*^2 \|A\|_\mathrm{F}^2 \frac{1}{\varepsilon^2} \log\frac{1}{\delta}\Big).$$

These values are chosen such that the following holds with probability $\geq 1 - \delta$ simultaneously.

1. The ith singular value of $CC^\dagger$ does not differ from the ith singular value of $A^\dagger A$ by more than $\bar{\varepsilon}$. This follows from Lemma 5.14 with error parameter $\varepsilon/(\|A\|_\mathrm{F}\|A\|_* \min(L, \bar{L}\|A\|^2))$. This immediately implies Eq. (48).
2. $\|R\|^2 = \mathcal{O}(\|A\|^2)$. This is the spectral norm bound in Eq. (47) (the Frobenius norm bound

follows from Lemma 5.2). We use Lemma 5.9:

$$\|R\|^2 \leq \|A\|^2 + \|R^\dagger R - A^\dagger A\| \leq \|A\|^2 + \frac{\varepsilon\|A\|^2}{L\|A\|_*^2} = O(\|A\|^2).$$

3. $\|f(R^\dagger R) - f(A^\dagger A)\|_* = \mathcal{O}(\varepsilon)$. We need the polylog factors in our number of samples to deal with the $\log r$ that arises from Lemma 7.4 in the spectral norm case.

$$\begin{aligned}
&\|f(R^\dagger R) - f(A^\dagger A)\|_* \\
&\lesssim L\|R^\dagger R - A^\dagger A\|_* \log \operatorname{rank}(R^\dagger R) && \text{(Lemma 7.4 or Lemma 7.3)} \\
&\lesssim L\sqrt{\frac{\phi^2 \log r \log(1/\delta)}{r}}\|A\|_*\|A\|_\mathrm{F} \log r && \text{(Lemma 5.9 or Lemma 5.7)} \\
&\lesssim \varepsilon. && \text{(plugging in value for } r\text{)}
\end{aligned}$$

4. $\|\bar{f}(CC^\dagger) - \bar{f}(RR^\dagger)\|_* = \mathcal{O}(\varepsilon/\|A\|^2)$. This follows similarly to the above point.

When all of the above bounds hold, we can conclude:

$$\begin{aligned}
&\|R^\dagger \bar{f}(CC^\dagger)R - f(A^\dagger A)\|_* \\
&\leq \|R^\dagger \bar{f}(RR^\dagger)R - f(A^\dagger A)\|_* + \|R^\dagger(\bar{f}(RR^\dagger) - \bar{f}(CC^\dagger))R\|_* \\
&= \|f(R^\dagger R) - f(A^\dagger A)\|_* + \|R^\dagger(\bar{f}(RR^\dagger) - \bar{f}(CC^\dagger))R\|_* && \text{(Definition of } \bar{f}\text{)} \\
&\leq \|f(R^\dagger R) - f(A^\dagger A)\|_* + \|R\|^2\|\bar{f}(RR^\dagger) - \bar{f}(CC^\dagger)\|_* \\
&\lesssim \varepsilon + \|R\|^2\varepsilon/\|A\|^2 \\
&\lesssim \varepsilon.
\end{aligned}$$

This gives Eq. (45) after rescaling ε by an appropriate constant factor. When $* = \mathrm{Op}$, we also have Eq. (49), since

$$\left\|R^\dagger\sqrt{\bar{f}(CC^\dagger)}\right\| = \sqrt{\|R^\dagger \bar{f}(CC^\dagger)R\|} \leq \sqrt{\|f(A^\dagger A) - f(0)I\| + \varepsilon}. \qquad \square$$

We remark here that the log term in Lemma 7.4 unfortunately cannot be removed (because some Lipschitz functions are not operator Lipschitz). However, several bounds hold under various mild assumptions, and for particular functions, the log term can be improved to log log or completely removed. For example, the QSVT literature [GSLW19] cites the following result:

Lemma 7.5 ([AP10, Corollary 7.4])**.** Let A and B be Hermitian matrices such that $aI \preceq A, B \preceq bI$, and let $f : \mathbb{R} \to \mathbb{C}$ be L-Lipschitz continuous on the interval $[a, b]$. Then

$$\|f^{(\mathrm{EV})}(A) - f^{(\mathrm{EV})}(B)\| \lesssim L\|A - B\| \log\Big(e\frac{b-a}{\|A-B\|}\Big).$$

Though we will not use it, we can extend these results on eigenvalue transformation of Hermitian matrices to singular value transformation of general matrices via the reduction from [GSLW19, Corollary 21]. For example, Lemma 7.4 implies the following:

Lemma 7.6. Let $A, B \in \mathbb{C}^{m\times n}$ be matrices and let $f : [0, \infty) \to \mathbb{C}$ be L-Lipschitz continuous on the singular values of A and B such that $f(0) = 0$. Then

$$\|f^{(\mathrm{SV})}(A) - f^{(\mathrm{SV})}(B)\| \lesssim L\|A - B\| \log \min(\operatorname{rank} A, \operatorname{rank} B).$$

In Section 7.1, we prove results on generic singular value transformation and eigenvalue transformation by bootstrapping Theorem 7.1. Since these are slower, though, we will use primarily the even singular value transformation results that we just proved to recover "dequantized QML" results. This will be the focus of next section.

7.1 More singular value transformation

In this section, we present more general versions of our algorithm for even SVT to get results for generic SVT (Theorem 7.7) and eigenvalue transformation (Theorem 7.8). In applications we mainly use even SVT to allow for more fine-tuned control over runtime, but we do use eigenvalue transformation in Section 8.7.

For generic SVT: consider a matrix $A \in \mathbb{C}^{m\times n}$ and a function $f : [0, \infty) \to \mathbb{C}$ satisfying $f(0) = 0$ (so the singular value transformation $f^{(\mathrm{SV})}(A)$ is well-defined as in Definition 3.1). Given $\mathrm{SQ}(A)$ and $\mathrm{SQ}(A^\dagger)$, we give an algorithm to output a CUR decomposition approximating $f^{(\mathrm{SV})}(A)$.

Theorem 7.7 (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 $f : [0,\infty) \to \mathbb{C}$ be a function such that $f(0) = 0$, $g(x) := f(\sqrt{x})/\sqrt{x}$ is L-Lipschitz, and $\bar{g}(x) := (g(x) - g(0))/x$ is $\bar{L}$-Lipschitz. Then, for $0 < \varepsilon \le \min(L\|A\|^3, \bar{L}\|A\|^5)$, we can output sketches $R := SA \in \mathbb{C}^{r\times n}$ and $C := AT \in \mathbb{C}^{m\times c}$, along with $M \in \mathbb{C}^{r\times c}$ such that*

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

with $r = \widetilde{\mathcal{O}}(\phi^2 L^2\|A\|^2\|A\|_\mathrm{F}^4\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ and $c = \widetilde{\mathcal{O}}(\phi^2 L^2\|A\|^4\|A\|_\mathrm{F}^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$. Finding S, M, and T takes time

$$\begin{aligned}\tilde{\mathcal{O}}\Big(&(\bar{L}^2\|A\|^8\|A\|_\mathrm{F}^2 + L^2\|A\|^2\|A\|_\mathrm{F}^4)\frac{\phi^2}{\varepsilon^2}\log\frac{1}{\delta}(\boldsymbol{s}_\phi(A) + \boldsymbol{s}_\phi(A^\dagger) + \boldsymbol{q}_\phi(A) + \boldsymbol{q}_\phi(A^\dagger))\\ &\qquad + (L^2\bar{L}^2\|A\|^{12}\|A\|_\mathrm{F}^4 + L^4\|A\|^6\|A\|_\mathrm{F}^6)\frac{\phi^4}{\varepsilon^4}\log^2\frac{1}{\delta}\,\boldsymbol{q}(A)\\ &\qquad\qquad + (L^4\bar{L}^2\|A\|^{16}\|A\|_\mathrm{F}^6 + L^6\|A\|^{10}\|A\|_\mathrm{F}^8)\frac{\phi^6}{\varepsilon^6}\log^3\frac{1}{\delta} + \boldsymbol{n}_\phi(A)\Big).\end{aligned}$$

If we only wish to assume $\mathrm{SQ}_\phi(A)$, we can do so by using Lemma 5.4 instead of Lemma 5.7 in our proof, paying an additional factor of $\frac{1}{\delta}$.

Note that if $\boldsymbol{sq}_\phi(A), \boldsymbol{sq}_\phi(A^\dagger) = \mathcal{O}(1)$, then this runtime is dominated by the last term. Moreover, if A is strictly low-rank, with minimum singular value σ, or essentially equivalently, if $f(x) = 0$ for $x \le \sigma$ and so $g(x) = 0$ for $x \le \sigma^2$, then $L \le \ell/\sigma^2$ and $\bar{L} = 2\ell/\sigma^4$ for ℓ the Lipschitz constant of f. In this case the complexity is

$$\widetilde{\mathcal{O}}\Big(\Big(\frac{\|A\|^{10}\|A\|_\mathrm{F}^6}{\sigma^{16}} + \frac{\|A\|^4\|A\|_\mathrm{F}^8}{\sigma^{12}}\Big)\Big(\frac{\ell\|A\|}{\varepsilon}\Big)^6\phi^6\log^3\frac{1}{\delta}\Big). \tag{50}$$

Importantly, when $\varepsilon = \mathcal{O}(\ell\|A\|)$ (that is, if we want relative error), this runtime is independent of dimension. If one desires greater generality, where we only need to depend on the Lipschitz constant of f, we can use a simple trick: as we aim for a spectral norm bound, we can essentially treat A as if it had strictly low rank. Consider the variant of f, $f_{\ge\sigma}$, which is zero below $\sigma/2$, f above σ, and is a linear interpolation in between.

$$f_{\geq\sigma}(x) := \begin{cases} 0 & 0 \leq x < \sigma/2 \\ (2x/\sigma - 1)f(\sigma) & \sigma/2 \leq x < \sigma \\ f(x) & \sigma \leq x \end{cases}$$

Then $\|f^{(\mathrm{SV})}(A) - f^{(\mathrm{SV})}_{\geq\varepsilon/\ell}(A)\| \leq \varepsilon$, because $f(\varepsilon/\ell) \leq \varepsilon$. Further, the Lipschitz constant of $f_{\geq\varepsilon/\ell}$ is at most 2ℓ: the slope of the linear interpolation is $2f(\sigma)/\sigma \leq 2\ell\sigma/\sigma$. So, we can run our algorithm for arbitrary ℓ-Lipschitz f in the time given by Eq. (50), with $\sigma = \varepsilon/\ell$.

Our proof strategy is to apply our main result Theorem 7.1 to $g(A^\dagger A)$, for $g(x) := f(\sqrt{x})/\sqrt{x}$, and subsequently approximate matrix products with Lemma 5.7 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).

We do not use this theorem in our applications. Sometimes we implicitly use a similar strategy (e.g. in Section 8.4), but because we apply our matrix to a vector ($f(A^\dagger)b$) we can use Lemma 5.17 instead of Lemma 5.7 when approximating. This allows for the algorithm to work with only $\mathrm{SQ}_\phi(A)$ and still achieve a poly-logarithmic dependence on $\frac{1}{\delta}$.

Proof. If we want to compute $\hat{f}^{(\mathrm{SV})}(A)$, we can work with $f(x) := \hat{f}(x) - g(0)x$, so that $g(0) = 0$ without loss of generality. Notice that $\hat{f}^{(\mathrm{SV})}(A) = f^{(\mathrm{SV})}(A) + g(0)A$, so if we get a CUR decomposition for $f^{(\mathrm{SV})}(A)$ we can add $g(0)A$ after to get the decomposition in the theorem statement.

Consider the SVT $g(x) := f(\sqrt{x})/\sqrt{x}$, so that $f^{(\mathrm{SV})}(A) = Ag(A^\dagger A)$. First, use Theorem 7.1

to get $SA \in \mathbb{C}^{r\times n}$, $SAT \in \mathbb{C}^{r\times c}$ such that, with probability $\geq 1-\delta/4$,

$$\|(SA)^\dagger \bar{g}((SAT)(SAT)^\dagger)SA - g(A^\dagger A)\| \leq \frac{\varepsilon}{2\|A\|}. \tag{51}$$

Second, use Lemma 5.7 to get a sketch $T'^\dagger \in \mathbb{C}^{c'\times n}$ such that, with probability $\geq 1-\delta/4$,

$$\|A(SA)^\dagger - AT'(SAT')^\dagger\| \leq \varepsilon(3L\|A\|)^{-1}. \tag{52}$$

The choices of parameters necessary are as follows (using that $\|SA\|_\mathrm{F} = O(\|A\|_\mathrm{F})$ by Eq. (47) and we have a 2ϕ-oversampled distribution for $(SA)^\dagger$ by Lemma 5.3):

$$\begin{aligned} r &= \tilde{\Theta}\Big(\phi^2 L^2 \|A\|^4 \|A\|_\mathrm{F}^2 \frac{1}{\varepsilon^2} \log\frac{1}{\delta}\Big) \\ c &= \tilde{\Theta}\Big(\phi^2 \bar{L}^2 \|A\|^8 \|A\|_\mathrm{F}^2 \frac{1}{\varepsilon^2} \log\frac{1}{\delta}\Big) \\ c' &= \tilde{\Theta}\Big(\phi^2 L^2 \|A\|^2 \|A\|_\mathrm{F}^4 \frac{1}{\varepsilon^2} \log\frac{1}{\delta}\Big) \end{aligned}$$

This implies the desired bound through the following sequence of approximations:

$$\begin{aligned} f^{(\mathrm{SV})}(A) &= Ag(A^\dagger A) \\ &\approx A(SA)^\dagger \bar{g}((SAT)(SAT)^\dagger)SA \\ &\approx \underbrace{AT'}_{C}\,\underbrace{(SAT')^\dagger \bar{g}((SAT)(SAT)^\dagger)}_{M}\,\underbrace{SA}_{R}. \end{aligned}$$

This gives us a CUR decomposition of $f^{(\mathrm{SV})}(A)$. These two approximations only incur $\mathcal{O}(\varepsilon)$ error in spectral norm; for the first, notice that

$$\begin{aligned} &\|Ag(A^\dagger A) - A(SA)^\dagger \bar{g}((SAT)(SAT)^\dagger)SA\| \\ &\leq \|A\|\|(SA)^\dagger \bar{g}((SAT)(SAT)^\dagger)SA - g(A^\dagger A)\| \leq \frac{\varepsilon}{2}. \qquad \text{(by (51))} \end{aligned}$$

For the second approximation observe that $|g(x)| \leq L|x|$ (and, by corollary, $\bar{g}(x) \leq L$) due to g

being L-Lipschitz and $g(0) = 0$, therefore

$$\begin{aligned}
&\|(A(SA)^\dagger - AT'(SAT')^\dagger)\bar{g}((SAT)(SAT)^\dagger)SA\| \\
&\leq \|AT'(SAT')^\dagger - A(SA)^\dagger\|\|\sqrt{\bar{g}((SAT)(SAT)^\dagger)}\|\|\sqrt{\bar{g}((SAT)(SAT)^\dagger)}SA\| \\
&\leq \varepsilon(3L\|A\|)^{-1}\|\sqrt{\bar{g}((SAT)(SAT)^\dagger)}\|\|\sqrt{\bar{g}((SAT)(SAT)^\dagger)}SA\| && \text{(by (52))} \\
&\leq \varepsilon(3L\|A\|)^{-1}\sqrt{L}\sqrt{\|g(A^\dagger A)\| + \frac{\varepsilon}{2\|A\|}} && \text{(since } \bar{g}(x) \leq L \text{ and by (51))} \\
&\leq \varepsilon(3L\|A\|)^{-1}\sqrt{\frac{3}{2}L\|A\|} < \frac{\varepsilon}{2}. && \text{(since } |g(x)| \leq L|x| \text{ and } \varepsilon \leq L\|A\|^3)
\end{aligned}$$

The time complexity of this procedure is

$$\mathcal{O}(\boldsymbol{n}_\phi(A) + (r + c + c')(\boldsymbol{s}_\phi(A) + \boldsymbol{q}_\phi(A)) + c'(\boldsymbol{s}_\phi(A^\dagger) + \boldsymbol{q}_\phi(A^\dagger)) + (rc + rc')\,\boldsymbol{q}(A) + r^2c + r^2c'),$$

which comes from producing sketches, querying all the relevant entries of SAT and SAT', the singular value transformation of SAT, and the matrix multiplication in M. We get r factors in the latter two terms because we can separate $\bar{g}((SAT)(SAT)^\dagger) = \sqrt{\bar{g}}(SAT)(\sqrt{\bar{g}}(SAT))^\dagger$ where $\sqrt{\bar{g}}(x) := \sqrt{\bar{g}(x)}$. □

As for eigenvalue transformation, consider a function $f : \mathbb{R} \to \mathbb{C}$ and a Hermitian matrix $A \in \mathbb{C}^{n\times n}$, given $\mathrm{SQ}(A)$. If f is even (so $f(x) = f(-x)$), then $f(A) = f(\sqrt{A^\dagger A})$, so we can use Theorem 7.1 to compute the eigenvalue transform $f(A)$. For non-even f, we cannot use this result, and present the following algorithm to compute it.

Theorem 7.8 (Eigenvalue transformation). *Suppose we are given a Hermitian $\mathrm{SQ}_\phi(A) \in \mathbb{C}^{n\times n}$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$, 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 $\varepsilon \in (0, L\|A\|/2]$. Then we can output matrices $S \in \mathbb{C}^{r\times n}$, $N \in \mathbb{C}^{s\times r}$, and $D \in \mathbb{C}^{s\times s}$, with $r = \widetilde{\mathcal{O}}(\phi^2\|A\|^4\|A\|_{\mathrm{F}}^2\frac{L^6}{\varepsilon^6}\log\frac{1}{\delta})$ and $s = \widetilde{\mathcal{O}}(\|A\|_{\mathrm{F}}^2\frac{L^2}{\varepsilon^2})$, such that*

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

in time

$$\tilde{\mathcal{O}}\Big((L^{10}\varepsilon^{-10}\|A\|^8\|A\|_F^2\phi^2\log\frac{1}{\delta} + L^6\varepsilon^{-6}\|A\|_F^6\phi\log\frac{1}{\delta})(\boldsymbol{s}_\phi(A)+\boldsymbol{q}_\phi(A)) \\ + (L^{16}\varepsilon^{-16}\|A\|^{12}\|A\|_F^4\phi^4\log^2\frac{1}{\delta} + L^{18}\varepsilon^{-18}\|A\|^8\|A\|_F^{10}\phi^5\log^3\frac{1}{\delta})\,\boldsymbol{q}(A) \\ + L^{22}\varepsilon^{-22}\|A\|^{16}\|A\|_F^6\phi^6\log^3\frac{1}{\delta} + \boldsymbol{n}_\phi(A)\Big).$$

Moreover, this decomposition satisfies the following two 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 $i \in [n]$ (where $D(i,i) := 0$ for $i > s$).

Under the reasonable assumptions[21] that $\boldsymbol{sq}(A)$ and ϕ are small ($\mathcal{O}(1)$, say) and $\varepsilon \le L\|A\|\frac{\|A\|}{\|A\|_F}$, the complexity of this theorem is $\widetilde{\mathcal{O}}(L^{22}\varepsilon^{-22}\|A\|^{16}\|A\|_F^6\log^3\frac{1}{\delta})$.

We now outline our proof. Our strategy is similar to the one used for quantum-inspired semidefinite programming [CLLW20]: first we find the eigenvectors and eigenvalues of A and then apply f to the eigenvalues. Let $\pi(x)$ be a (smoothened) step function designed so it can zeroes out small eigenvalues of A (in particular, eigenvalues smaller than $\varepsilon/\sqrt{2}L$). Then

$$\begin{aligned} A &\approx \pi(A^\dagger A)A\pi(A^\dagger A) && \text{by definition of } \pi \\ &\approx R^\dagger\bar{\pi}(CC^\dagger)RAR^\dagger\bar{\pi}(CC^\dagger)R && \text{by Theorem 7.1} \\ &\approx R^\dagger\bar{\pi}(CC^\dagger)M\bar{\pi}(CC^\dagger)R && \text{by sketching } M \approx RAR^\dagger \\ &= R^\dagger(C_\sigma C_\sigma^+)^\dagger\bar{\pi}(CC^\dagger)M\bar{\pi}(CC^\dagger)C_\sigma C_\sigma^+R. && \text{where } \sigma = \varepsilon/\sqrt{2}L \end{aligned}$$

Here, C_σ is the low-rank approximation of C formed by transforming C according to the "filter" function on x that is 0 for $x < \sigma$ and x otherwise. $\hat{U} := C_\sigma^+R \in \mathbb{C}^{c\times n}$ is an approximate isometry by Lemma 5.22. 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 A of size independent of n, we can directly compute its unitary eigendecomposition $UDU^\dagger$. This gives the approximate decomposition $A \approx (\hat{U}U)D(\hat{U}U)^\dagger$, with $\hat{U}U$ and D acting as approximate eigenvectors and eigenvalues

[21]The correct way to think about ε 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 $\|A\|_F/\|A\|$ improvement over trivial, which is modest in the close-to-low-rank regime that we care about. Similar assumptions appear in Section 8.

of A, respectively. An application of Lemma 7.4 shows that $f(A) \approx (\hat{U}U)f(D)(\hat{U}U)^\dagger$ in the desired sense. Therefore, our output approximation of $f(A)$ comes in the form of an RUR decomposition that can be rewritten in the form of an approximate eigendecomposition. The only major difference between this proof sketch and the proof below is that we perform our manipulations on the SVD of C_σ, to save on computation time: note that the SVD can be made small in dimension, using that the rank of C_σ is bounded by $\|C\|_{\mathrm{F}}^2/\sigma^2$.

Proof. Throughout this proof ε is not dimensionless; if choices of parameters are confusing, try replacing ε with $\varepsilon\|A\|$. We will take $f(0) = 0$ without loss of generality. First, consider the "smooth projection" singular value transformation

$$\pi(x) = \begin{cases} 0 & x < \frac{\varepsilon^2}{2L^2} \\ \frac{2L^2}{\varepsilon^2}x - 1 & \frac{\varepsilon^2}{2L^2} \leq x < \frac{\varepsilon^2}{L^2} \\ 1 & \frac{\varepsilon^2}{L^2} \leq x \end{cases}$$

Since π is a projector onto the large eigenvectors of A, we can add these projectors to our expression without incurring too much spectral norm error.

$$\|\pi(A^\dagger A)A\pi(A^\dagger A) - A\| = \max_{i\in[n]}|\pi(\lambda_i^2)\lambda_i\pi(\lambda_i^2) - \lambda_i| \leq \varepsilon/L$$

Second, use Theorem 7.1 to get $SA \in \mathbb{C}^{r\times n}$, $SAT \in \mathbb{C}^{r\times c}$ such that, with probability $\geq 1-\delta$,

$$\|(SA)^\dagger\bar{\pi}((SAT)(SAT)^\dagger)SA - \pi(A^\dagger A)\| \leq \frac{\varepsilon}{L\|A\|}.$$

The necessary sizes for these bounds to hold are as follows (Lipschitz constants for π are $2L^2/\varepsilon^2$ and $4L^4/\varepsilon^4$, $\|SA\|_{\mathrm{F}} = O(\|A\|_{\mathrm{F}})$ by Eq. (47), and we have a 2ϕ-oversampled distribution for $(SA)^\dagger$ by Lemma 5.3):[22]

$$r = \tilde{\Theta}\Big(\phi^2 \frac{L^4}{\varepsilon^4}\|A\|^2\|A\|_{\mathrm{F}}^2 \frac{L^2\|A\|^2}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \tilde{\Theta}\Big(\phi^2\|A\|^4\|A\|_{\mathrm{F}}^2\frac{L^6}{\varepsilon^6}\log\frac{1}{\delta}\Big),$$
$$c = \tilde{\Theta}\Big(\phi^2 \frac{L^8}{\varepsilon^8}\|A\|^6\|A\|_{\mathrm{F}}^2 \frac{L^2\|A\|^2}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \tilde{\Theta}\Big(\phi^2\|A\|^8\|A\|_{\mathrm{F}}^2\frac{L^{10}}{\varepsilon^{10}}\log\frac{1}{\delta}\Big).$$

This approximation does not incur too much error:

$$\begin{aligned}
&\|R^\dagger\bar{\pi}(CC^\dagger)RAR^\dagger\bar{\pi}(CC^\dagger)R - \pi(A^\dagger A)A\pi(A^\dagger A)\| \\
&\leq \|\pi(A^\dagger A)A(\pi(A^\dagger A) - R^\dagger\bar{\pi}(CC^\dagger)R)\| + \|(\pi(A^\dagger A) - R^\dagger\bar{\pi}(CC^\dagger)R)AR^\dagger\bar{\pi}(CC^\dagger)R\| \\
&\leq \frac{\varepsilon}{L\|A\|}\Big(\|\pi(A^\dagger A)\|\|A\| + \|A\|\|R^\dagger\bar{\pi}(CC^\dagger)R\|\Big) \leq \frac{\varepsilon}{L\|A\|}\Big(\|A\| + \|A\|\Big(1+\frac{\varepsilon}{L\|A\|}\Big)\Big) \leq 3\frac{\varepsilon}{L}.
\end{aligned}$$

Third, apply Remark 5.18(b) r^2 times to approximate each entry of $RAR^\dagger$: pull t samples from $\mathrm{SQ}_\phi(A)$ for $t := \mathcal{O}(\phi\|A\|_{\mathrm{F}}^6\frac{L^6}{\varepsilon^6}\log\frac{r^2}{\delta})$ such that, given some $\mathrm{Q}(x), \mathrm{Q}(y)$, with probability $\geq 1-\frac{\delta}{r^2}$, one can output an estimate of $x^\dagger Ay$ up to $\frac{\varepsilon^3\|x\|\|y\|}{L^3\|A\|_{\mathrm{F}}^2}$ additive error with *no additional queries to* $\mathrm{SQ}_\phi(A)$. Then, by union bound, with probability $\geq 1-\delta$, using the same t samples from A each time, one can output an estimate of $R(i,\cdot)AR(j,\cdot)^\dagger$ up to $\frac{\varepsilon^3\|R(i,\cdot)\|\|R(j,\cdot)\|}{L^3\|A\|_{\mathrm{F}}^2}$ error for all $i,j \in [r]$ such that $i \leq j$. Let M be the matrix of these estimates. Then, using that $\|R\|_{\mathrm{F}} = \mathcal{O}(\|A\|_{\mathrm{F}})$ from Eq. (47),

$$\|M - RAR^\dagger\|_{\mathrm{F}}^2 \leq \sum_{i=1}^r\sum_{j=1}^r\Big(\frac{\varepsilon^3\|R(i,\cdot)\|\|R(j,\cdot)\|}{L^3\|A\|_{\mathrm{F}}^2}\Big)^2 = \frac{\varepsilon^6\|R\|_{\mathrm{F}}^4}{L^6\|A\|_{\mathrm{F}}^4} \lesssim \frac{\varepsilon^6}{L^6}.$$

From Eqs. (48) and (49),

$$\|R^\dagger\bar{\pi}(CC^\dagger)(RAR^\dagger - M)\bar{\pi}(CC^\dagger)R\| \lesssim \frac{\varepsilon^3}{L^3}\|R^\dagger\bar{\pi}(CC^\dagger)\|^2 \leq \frac{\varepsilon^3}{L^3}\Big(1+\frac{\varepsilon}{L\|A\|}\Big)\frac{L^2}{\varepsilon^2} \lesssim \frac{\varepsilon}{L}.$$

So far, we have shown that we can find an RUR approximation to A, with

$$\|R^\dagger\bar{\pi}(CC^\dagger)M\bar{\pi}(CC^\dagger)R - A\| \lesssim \frac{\varepsilon}{L}$$

[22]The constraint on the size of ε from Theorem 7.1 here is $\varepsilon(L\|A\|)^{-1} \lesssim \min(L^2\|A\|^2/\varepsilon^2, L^4\|A\|^4/\varepsilon^4)$, which is true since $\varepsilon \leq L\|A\|/2$.

However, if we wish to apply an eigenvalue transformation to A, we need to access the eigenvalues of A as well. To do this, we will express this decomposition as an approximate unitary eigendecomposition. Using that $\bar{\pi}$ zeroes out singular values that are smaller than $\frac{\varepsilon^2}{2L^2}$, we can write our expression as $\hat{U}\hat{D}\hat{U}^\dagger$, for $\hat{U} \in \mathbb{C}^{n\times s}$ and $\hat{D} \in \mathbb{C}^{s\times s}$:

$$\begin{aligned}
&R^\dagger \bar{\pi}(CC^\dagger) M \bar{\pi}(CC^\dagger) R \\
&= \big(C^+_{\frac{\varepsilon}{\sqrt{2}L}} R\big)^\dagger \big(C^\dagger_{\frac{\varepsilon}{\sqrt{2}L}} \bar{\pi}(CC^\dagger) M \bar{\pi}(CC^\dagger) C_{\frac{\varepsilon}{\sqrt{2}L}}\big)\big(C^+_{\frac{\varepsilon}{\sqrt{2}L}} R\big) \\
&= \underbrace{\big(R^\dagger U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}} (D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^{-1}\big)}_{\hat{U}} \underbrace{\big(D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}} (U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^\dagger \bar{\pi}(CC^\dagger) M \bar{\pi}(CC^\dagger) U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}} D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}}\big)}_{\hat{D}} \underbrace{\big((D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^{-1} (U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^\dagger R\big)}_{\hat{U}^\dagger}.
\end{aligned} \tag{53}$$

Here, we are using an SVD of C truncated to ignore singular values smaller than $\frac{\varepsilon}{\sqrt{2}L}$, where $U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}} \in \mathbb{C}^{r\times s}$, $D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}} \in \mathbb{C}^{s\times s}$, $V^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}} \in \mathbb{C}^{c\times s}$, where s is the number of singular values of C that are at least $\frac{\varepsilon}{\sqrt{2}L}$. Note that, as a result, $s \leq \|C\|_F^2/(\varepsilon/\sqrt{2}L)^2 \lesssim \|A\|_F^2 L^2 \varepsilon^{-2}$ and $s \leq \min(r, c, n)$. By Lemma 5.22 with our values of r and c, we get that $\hat{U} := R^\dagger U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}} (D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^{-1}$ is an $\mathcal{O}(\frac{\varepsilon^3}{L^3\|A\|^3})$-approximate isometry; we rescale ε until this is at most $\frac{1}{2}$.

The rest of this error analysis will show that, since $\hat{U}$ is an approximate isometry, $f(A) \approx \hat{U} f(\hat{D}) \hat{U}^\dagger$ in the senses required for the theorem statement. Though $\hat{D}$ is not diagonal, since it is $s\times s$, we can compute its unitary eigendecomposition $U^{(\hat{D})} D^{(\hat{D})} (U^{(\hat{D})})^\dagger$; so, we can take $D := f(D^{(\hat{D})})$ and $N := (U^{(\hat{D})})^\dagger (D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^+ (U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^\dagger$ to get the decomposition in the theorem statement. (Including the isometry $(U^{(\hat{D})})^\dagger$ in our expression for $\hat{U}$ does not change the value of α).

First, consider the eigenvalues of $\hat{D}$. Note that $\|\hat{U}^\dagger \hat{U} - I\| \leq \alpha$ since $\hat{U}$ is an α-approximate isometry, and by Lemma 3.5, there exists an isometry U such that $\|U - \hat{U}\| \leq \alpha$. We first observe that, using Lemma 3.5 and our bound on α,

$$\begin{aligned}
\|A - U\hat{D}U^\dagger\| &\leq \|A - \hat{U}\hat{D}\hat{U}^\dagger\| + \|\hat{U}\hat{D}\hat{U}^\dagger - U\hat{D}U^\dagger\| \\
&\leq \frac{\varepsilon}{L} + \alpha\frac{2-\alpha}{(1-\alpha)^2}\|\hat{U}\hat{D}\hat{U}^\dagger\| \\
&\leq \frac{\varepsilon}{L} + \alpha\frac{2-\alpha}{(1-\alpha)^2}\Big(\|A\| + \frac{\varepsilon}{L}\Big) \lesssim \frac{\varepsilon}{L}.
\end{aligned}$$

Consequently, by Weyl's inequality (Lemma 5.16), the eigenvalues of $U\hat{D}U^\dagger$, $\hat{\lambda}_1 \geq \cdots \geq \hat{\lambda}_n$

satisfy $|\lambda_i - \hat{\lambda}_i| \lesssim \frac{\varepsilon}{L}$ for all $i \in [n]$, and by assumption, f is L-Lipschitz on the spectrums of A and $U\hat{D}U$. From this, we can conclude that we can compute estimates for the eigenvalues of $f(A)$, since the eigenvalues of $U\hat{D}U^\dagger$ are the eigenvalues of $\hat{D}$ (padded with zero eigenvalues) which we know from our eigendecomposition of $\hat{D}$ Further, our estimates $f(\hat{\lambda}_i)$ satisfy the desired bound $|f(\hat{\lambda}_i) - f(\lambda_i)| \leq \varepsilon$. Finally, since f is Lipschitz on our spectrums of concern, the desired error bound for our approximation holds by the following computation (which uses Lemma 3.5 extensively):

$$\begin{aligned}
\|f(A) - \hat{U}f(\hat{D})\hat{U}^\dagger\| &\leq \|f(A) - Uf(\hat{D})U^\dagger\| + \|Uf(\hat{D})U^\dagger - \hat{U}f(\hat{D})\hat{U}^\dagger\| \\
&\leq \|f(A) - Uf(\hat{D})U^\dagger\| + (2\alpha + \alpha^2)\|f(\hat{D})\| \\
&\lesssim L(\|A - U\hat{D}U^\dagger\| + (2\alpha + \alpha^2)\|\hat{D}\|)\log s && \text{by Lemma 7.4} \\
&\leq L(\|A - \hat{U}\hat{D}\hat{U}^\dagger\| + 2(2\alpha + \alpha^2)\|\hat{D}\|)\log s \\
&\leq L\Big(\frac{\varepsilon}{L} + \frac{2(2\alpha + \alpha^2)}{(1-\alpha)^2}\|\hat{U}\hat{D}\hat{U}^\dagger\|\Big)\log s \\
&\leq L\Big(\frac{\varepsilon}{L} + \frac{2(2\alpha + \alpha^2)}{(1-\alpha)^2}\Big(\|A\| + \frac{\varepsilon}{L}\Big)\Big)\log s \lesssim \varepsilon \log s. && \text{by } \alpha \leq \frac{\varepsilon}{L\|A\|}
\end{aligned}$$

Finally, we rescale ε down by $\log^2 s$ so that this final bound is $\mathcal{O}(\varepsilon)$. This term is folded into the polylog terms of r, c, and s. (We need to scale by more than $\log s$ because s has a dependence on $\frac{1}{\varepsilon^2}$.) This completes the error analysis.

The complexity analysis takes some care: we want to compute our matrix expressions in the correct order. First, we will sample to get S and T, and then compute the *truncated* singular value decomposition of $C := SAT$, which we have denoted $C_{\frac{\varepsilon}{\sqrt{2L}}} = U^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} (V^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}})^\dagger$ for $U^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} \in \mathbb{C}^{r\times s}$, $D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} \in \mathbb{C}^{s\times s}$, $V^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} \in \mathbb{C}^{c\times s}$. Then, we will perform the inner product estimation protocol r^2 times to get our estimate $M \in \mathbb{C}^{r\times r}$, and compute the eigendecomposition of

$$\begin{aligned}
\hat{D} &= D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} (U^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}})^\dagger \bar{\pi}(CC^\dagger) M \bar{\pi}(CC^\dagger) U^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} \\
&= D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} \bar{\pi}((D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}})^2)(U^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}})^\dagger M U^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}} \bar{\pi}((D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}})^2) D^{(C)}_{\frac{\varepsilon}{\sqrt{2L}}}
\end{aligned}$$

via the final expression above, with the truncations propagated through the matrices, to get

$\hat{D} = U^{(\hat{D})} D^{(\hat{D})} (U^{(\hat{D})})^\dagger$. Then, we compute and output $D = \hat{D}$ and $N = (U^{(\hat{D})})^\dagger (D^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^+ (U^{(C)}_{\frac{\varepsilon}{\sqrt{2}L}})^\dagger$. By evaluating the expression for $\hat{D}$ from left-to-right, we only need to perform matrix multiplications that (naively) take s^3 or sr^2 time. The only cost of c we incur is in computing the SVD of C. The runtime is

$$\mathcal{O}((r + c + t)(\boldsymbol{s}_\phi(A) + \boldsymbol{q}_\phi(A)) + (rc + r^2 t)\, \boldsymbol{q}(A) + s^3 + r^2 s + r^2 c + \boldsymbol{n}_\phi(A)). \qquad \square$$

8 Dequantizing quantum machine learning

Now, with our framework, we can recover previous dequantization results: recommendation systems (Section 8.1), supervised clustering (Section 8.2), principal component analysis (Section 8.3), low-rank matrix inversion (Section 8.4), support vector machines (Section 8.5), and low-rank semidefinite programs (Section 8.7). We also propose new quantum-inspired algorithm for other applications, including Hamiltonian simulation (Section 8.6), and discriminant analysis (Section 8.8). We give applications in roughly chronological order; this also happens to be a rough difficulty curve, with applications that follow more easily from our main results being first.

Everywhere it occurs, $K := \|A\|_F^2/\sigma^2$, where A is the input matrix. $\kappa := \|A\|_2^2/\sigma^2$. For simplicity, we will often describe our runtimes as if we know spectral norms of input matrices (so, for example, we know κ). If we do not know the spectral norm, we can run Lemma 5.14 repeatedly with multiplicatively decreasing ε until we find a constant factor upper bound on the spectral norm, which suffices for our purposes. Alternatively, we can bound the spectral norm by the Frobenius norm, which we know from sampling and query access to input.

8.1 Recommendation systems

Our framework gives a simpler and faster variant of Tang's dequantization [Tan19] of Kerenidis and Prakash's quantum recommendation systems [KP17]. Tang's 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 [Pre18].

We want to find a product $j \in [n]$ that is a good recommendation for a particular user $i \in [m]$, given incomplete data on user-product preferences. If we store this data in a matrix $A \in \mathbb{R}^{m \times n}$ with sampling and query access, in the strong model described by Kerenidis and Prakash [KP17], finding good recommendations reduces to the following:

Problem 8.1. For a matrix $A \in \mathbb{R}^{m\times n}$, given $\mathrm{SQ}(A)$ and a row index $i \in [m]$, sample from $\hat{A}(i, \cdot)$ up to δ error in total variation distance, where $\|\hat{A} - A_{\sigma,\eta}\|_{\mathrm{F}} \le \varepsilon \|A\|_{\mathrm{F}}$.

Here, $A_{\sigma,\eta}$ is a certain type of low-rank approximation to A. The standard notion of low-rank approximation is that of $A_r := \sum_{i=1}^{r} \sigma_i U(\cdot, i) V(\cdot, i)^\dagger$, which is the rank-$r$ matrix closest to A in spectral and Frobenius norms. Using singular value transformation, we define an analogous notion thresholding singular values instead of rank.

Definition 8.2 ($A_{\sigma,\eta}$). We define $A_{\sigma,\eta}$ as a singular value transform of A satisfying:

$$A_{\sigma,\eta} := P^{(\mathrm{SV})}_{\sigma,\eta}(A) \qquad P_{\sigma,\eta}(\lambda) \begin{cases} = \lambda & \lambda \ge \sigma(1+\eta) \\ = 0 & \lambda < \sigma(1-\eta) \cdot \\ \in [0, \lambda] & \text{otherwise} \end{cases}$$

Note that $P_{\sigma,\eta}$ is not fully specified in the range $[\sigma(1-\eta), \sigma(1+\eta))$, so $A_{\sigma,\eta}$ is any of a family of matrices with error η.

For intuition, $P^{(\mathrm{SV})}_{\sigma,\eta}(A)$ is A for (right) singular vectors with value $\ge \sigma(1+\eta)$, zero for those with value $< \sigma(1-\eta)$, and something in between for the rest. The quantum algorithm for this attains this by taking a polynomial approximation of a threshold function and using quantum linear algebra techniques to apply it to A. The threshold function is as follows:

Lemma 8.3 (Polynomial approximations of the rectangle function [GSLW19, Lemma 29]). Let $\delta', \varepsilon' \in (0, \frac{1}{2})$ and $t \in [-1, 1]$. There exists an even polynomial $p \in \mathbb{R}[x]$ of degree

$\mathcal{O}(\log(\frac{1}{\varepsilon'})/\delta')$, such that $|p(x)| \leq 1$ for all $x \in [-1,1]$, and

$$p(x) \in \begin{cases} [0,\varepsilon'] & \text{for all } x \in [-1,-t-\delta'] \cup [t+\delta',1], \text{ and} \\ [1-\varepsilon',1] & \text{for all } x \in [-t-\delta', t+\delta'] \end{cases}$$

In particular, one obtains the quantum recommendation systems result from QSVT is by preparing the state $|A(i,*)\rangle$ and applying a block-encoding of $p(A)$ to get a copy of $|p(A)A(i,*)\rangle$, where p is the polynomial from Lemma 8.3.[23] We can obtain the same guarantee classically by appealing to Theorem 6.1.

Corollary 8.4 (Dequantizing recommendation systems). *Suppose we are given a matrix $A \in \mathbb{C}^{m\times n}$ such that $0.01 \leq \|A\| \leq 1$ and $0 < \varepsilon, \sigma < 1$, with $\mathcal{O}(\mathrm{nnz}(A))$ time pre-processing. Then there exists an algorithm that, given an index $i \in [m]$, computes a vector y such that $\|y - p(A)A(i,*)\| \leq \varepsilon\|A(i,*)\|$ with probability at least 0.9, where $p(x)$ is the rectangle polynomial from Lemma 8.3, with parameters $t = \sigma$, $\delta' = \sigma/6$, $\varepsilon' = \varepsilon$. Further, the running time to compute such a description of y is*

$$\widetilde{\mathcal{O}}\Big(\frac{\|A\|_F^4}{\sigma^{11}\varepsilon^2}\Big),$$

and from this description we can sample from y in $\widetilde{\mathcal{O}}\Big(\frac{\|A\|_F^4\|A(i,\cdot)\|^2}{\sigma^8\varepsilon^2\|y\|^2}\Big)$ time.

Thinking of $p(A)A$ as the low-rank approximation of A this algorithm gives access to, we see that the error guarantee of Corollary 8.4 implies the solution to Problem 8.1.

We can also solve this problem without appealing to polynomial approximation. Our algorithm uses that we can rewrite our target low-rank approximation as $A \cdot t(A^\dagger A)$, where t is a smoothened projector. So, we can use our main theorem, Theorem 7.1, to approximate $t(A^\dagger A)$ by some $R^\dagger UR$. Then, the i^{th} row of our low-rank approximation is $A(i,\cdot)R^\dagger UR$, which is a product of a vector with an RUR decomposition. Thus, using the sampling techniques described in Section 5.1, we have $\mathrm{SQ}_\phi(A(i,\cdot)R^\dagger UR)$, so we can get the sample from this row as desired.

[23]The original paper goes through a singular value estimation procedure; see the discussion in Section 3.6 of [GSLW19] for more details.

Corollary 8.5. *Suppose* $0 < \varepsilon \lesssim \|A\|/\|A\|_F$ *and* $\eta \leq 0.99$. *A classical algorithm can solve Problem 8.1 in time*

$$\widetilde{\mathcal{O}}\Big(\frac{K^3\kappa^5}{\eta^6\varepsilon^6}\log^3\frac{1}{\delta} + \frac{K^2\kappa\|A(i,\cdot)\|^2}{\eta^2\varepsilon^2\|\hat{A}(i,\cdot)\|^2}\log^2\frac{1}{\delta}\Big).$$

The assumption on ε is a weak non-degeneracy condition in the low-rank regime. For reference, $\eta = 1/6$ in the application of this algorithm to recommendation systems. So, supposing the first term of the runtime dominates, the runtime is $\widetilde{\mathcal{O}}(\frac{\|A\|_F^6\|A\|^{10}}{\sigma^{16}\varepsilon^6}\log^3\frac{1}{\delta})$, which improves on the previous runtime $\widetilde{\mathcal{O}}(\frac{\|A\|_F^{24}}{\sigma^{24}\varepsilon^{12}}\log^3\frac{1}{\delta})$ of [Tan19]. The quantum runtime for this problem is $\widetilde{\mathcal{O}}(\frac{\|A\|_F}{\sigma})$, up to polylog$(m,n)$ terms [CGJ19, Theorem 27].

Proof. Note that $A_{\sigma,\eta} = A \cdot t(A^\dagger A)$, where t is the thresholding function shown below.

$$t(x) = \begin{cases} 0 & x < (1-\eta)^2\sigma^2 \\ \frac{1}{4\eta\sigma^2}(x-(1-\eta)^2\sigma^2) & (1-\eta)^2\sigma^2 \leq x < (1+\eta)^2\sigma^2 \\ 1 & x \geq (1+\eta)^2\sigma^2 \end{cases}.$$

We will apply Theorem 7.1 with error parameter ε to get matrices R, C such that $AR^\dagger\bar{t}(CC^\dagger)R$ satisfies

$$\|A_{\sigma,1/6} - AR^\dagger\bar{t}(CC^\dagger)R\|_F \leq \|A\|_F\|t(A^\dagger A) - R^\dagger\bar{t}(CC^\dagger)R\| \leq \varepsilon\|A\|_F. \tag{54}$$

Since $t(x)$ is $(4\eta\sigma^2)^{-1}$-Lipschitz and $t(x)/x$ is $(4\eta(1-\eta)^2\sigma^4)^{-1}$-Lipschitz, the sizes of r and c are

$$r = \widetilde{\mathcal{O}}\Big(L^2\|A\|^2\|A\|_F^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{\|A\|^2\|A\|_F^2}{\sigma^4\eta^2\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{K\kappa}{\eta^2\varepsilon^2}\log\frac{1}{\delta}\Big);$$
$$c = \widetilde{\mathcal{O}}\Big(\bar{L}^2\|A\|^6\|A\|_F^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{\|A\|^6\|A\|_F^2}{\sigma^8\eta^2\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{K\kappa^3}{\eta^2\varepsilon^2}\log\frac{1}{\delta}\Big).$$

So, it suffices to compute the SVD of an $r \times c$ matrix, which has a runtime of

$$\widetilde{\mathcal{O}}\Big(\frac{K^3\kappa^5}{\eta^6\varepsilon^6}\log^3\frac{1}{\delta}\Big).$$

Next, we want to approximate $AR^\dagger \approx A'R'^\dagger$. If we had $\mathrm{SQ}(A^\dagger)$ (in particular, if we could compute column norms $\|A(\cdot,j)\|$), we could do this via Lemma 5.7, and if we were okay with paying factors of $\frac{1}{\delta}$, we could do this via Lemma 5.4. Here, we will instead implicitly define an approximation by approximating each row $[AR^\dagger](i,\cdot) = A(i,\cdot)R^\dagger$ via Lemma 5.7, since we then have $\mathrm{SQ}(A(i,\cdot)^\dagger)$ and $\mathrm{SQ}(R^\dagger)$. With this proposition, we can estimate $[AR^\dagger](i,\cdot) \approx (A(i,\cdot)S^\dagger S)R^\dagger$ to $\frac{\varepsilon}{\sqrt{K}}\|A(i,\cdot)\|\|R^\dagger\|_{\mathrm{F}} = \varepsilon\|A(i,\cdot)\|\sigma$ error using $r' := O(\varepsilon^{-2}K\log\frac{1}{\delta})$ samples[24]. Here, $A'(i,\cdot) := A(i,\cdot)S^\dagger S$ is our r'-sparse approximation, giving that

$$\|AR^\dagger - A'R^\dagger\|_{\mathrm{F}} = \sqrt{\textstyle\sum_{i=1}^m \|[AR^\dagger](i,\cdot) - [A'R^\dagger](i,\cdot)\|^2} \le \sqrt{\textstyle\sum_{i=1}^m \varepsilon^2\|A(i,\cdot)\|^2\sigma^2} = \varepsilon\sigma\|A\|_{\mathrm{F}}. \tag{55}$$

Using this and the observation that $\max_x \bar{t}(x) = (1+\eta)^{-2}\sigma^{-2} \le \sigma^{-2}$, we can bound the quality of our final approximation as

$$\begin{aligned}
\|\hat{A} - A_{\sigma,\eta}\|_{\mathrm{F}} &\le \|(A'R^\dagger - AR^\dagger)\bar{t}(CC^\dagger)R\|_{\mathrm{F}} + \|AR^\dagger\bar{t}(CC^\dagger)R - A_{\sigma,\eta}\|_{\mathrm{F}} && \text{by triangle inequality} \\
&\le \|A'R^\dagger - AR^\dagger\|_{\mathrm{F}}\Big\|\sqrt{\bar{t}(CC^\dagger)}\Big\|\Big\|\sqrt{\bar{t}(CC^\dagger)}R\Big\| + \varepsilon\|A\|_{\mathrm{F}} && \text{by Eq. (54)} \\
&\le \varepsilon\sigma\|A\|_{\mathrm{F}}\sigma^{-1}\sqrt{1+\varepsilon} + \varepsilon\|A\|_{\mathrm{F}} \lesssim \varepsilon\|A\|_{\mathrm{F}}. && \text{by Lemma 7.2 and Eq. (55)}
\end{aligned}$$

We can sample from $\hat{A}(i,\cdot) = A'(i,\cdot)R^\dagger\bar{f}(CC^\dagger)R$ by naively computing $x := A'(i,\cdot)R^\dagger\bar{f}(CC^\dagger)$, taking $O(r'r + rc)$ time. Then, we use Lemmas 4.5 and 4.6 to get a sample from xR with probability $\ge 1-\delta$ in $\mathcal{O}(\overline{\boldsymbol{sq}}_\phi(xR))$ time, which is $\mathcal{O}(\phi\,\boldsymbol{sq}_\phi(xR)\log\frac{1}{\delta})$, where $\boldsymbol{sq}_\phi(xR) = \mathcal{O}(r)$ and

$$\phi = r\frac{\sum_{j=1}^r |x(j)|^2\|R(j,\cdot)\|^2}{\|xR\|^2} \lesssim r\frac{\sum_{j=1}^r |x(j)|^2\|A\|_{\mathrm{F}}^2}{\|\hat{A}(i,\cdot)\|^2 r} = \frac{\|x\|^2\|A\|_{\mathrm{F}}^2}{\|\hat{A}(i,\cdot)\|^2}.$$

[24]Formally, to get a true approximation $AR \approx A'R$, we need to union bound the failure probability for each row, paying a $\log m$ factor in runtime. However, we will ignore this consideration: our goal is to sample from one row, so we only need to succeed in our particular row.

Then, using previously established bounds and bounds from Lemma 7.2, we have

$$\begin{aligned}\frac{\|x\|^2\|A\|_F^2}{\|\hat{A}(i,\cdot)\|^2} &= \frac{\|A'(i,\cdot)R^\dagger\bar{t}(CC^\dagger)\|^2\|A\|_F^2}{\|\hat{A}(i,\cdot)\|^2} \\ &\leq \left(\|A(i,\cdot)\|\left\|R^\dagger\sqrt{\bar{t}(CC^\dagger)}\right\|\left\|\sqrt{\bar{t}(CC^\dagger)}\right\| + \|A(i,\cdot)'R^\dagger - A(i,\cdot)R^\dagger\|\|\bar{t}(CC^\dagger)\|\right)^2\frac{\|A\|_F^2}{\|\hat{A}(i,\cdot)\|^2} \\ &\lesssim (\|A(i,\cdot)\|\sigma^{-1} + \varepsilon\sigma\|A(i,\cdot)\|\sigma^{-2})^2\frac{\|A\|_F^2}{\|\hat{A}(i,\cdot)\|^2} \\ &\lesssim \frac{\|A(i,\cdot)\|^2\|A\|_F^2}{\|\hat{A}(i,\cdot)\|^2\sigma^2}.\end{aligned}$$

This sampling procedure and the SVD dominate the runtime. Since the sampling is exact, the only error in total variation distance is the probability of failure. □

Remark 8.6. This algorithm implicitly assumes that the important singular values are $\geq \sigma$. Without such an assumption, we can take $\sigma = \varepsilon\|A\|_F$ and $\eta = 1/2$, and have meaningful bounds on the output matrix $\hat{A}$. Observe that, for $p(x) = x(t(\sqrt{x}) - 1)$,

$$\|A \cdot t(A^\dagger A) - A\| = \|p^{(\mathrm{SV})}(A)\| \leq \frac{3}{2}\varepsilon\|A\|_F.$$

So, our low-rank approximation output $\hat{A}$ satisfies $\|\hat{A} - A\| \lesssim \varepsilon\|A\|_F$, with no assumptions on A, in $\widetilde{\mathcal{O}}(\frac{\|A\|_F^6}{\|A\|^6\varepsilon^{22}}\log^3\frac{1}{\delta})$ time. This can be subsequently used to get $\mathrm{SQ}_\phi(\hat{A}(i,\cdot)) = \mathrm{SQ}_\phi(e_i\hat{A})$ where $\|\hat{A}(i,\cdot) - A(i,\cdot)\| \lesssim \varepsilon\|A\|_F$ (in a myopic sense, solving the same problem as Problem 8.1), or more generally, any product of $\hat{A}$ with a vector, in time independent of dimension.

8.2 Supervised clustering

The 2013 paper of Lloyd, Mohseni, and Rebentrost [LMR13] gives two algorithms for the machine learning problem of clustering. The first algorithm is a simple swap test procedure that was dequantized by Tang [Tan21] (the second is an application of the quantum adiabatic algorithm with no proven runtime guarantees). We will reproduce the algorithm from [Tan21] here: since the dequantization just uses the inner product protocol, so it rather trivially fits into our framework.

We have a dataset of points in $\mathbb{R}^d$ grouped into clusters, and we wish to classify a new data point by assigning it to the cluster with the nearest average, aka *centroid.* We do this by estimating the distance between the new point $p \in \mathbb{R}^d$ to the centroid of a cluster of points $q_1, \dots, q_{n-1} \in \mathbb{R}^d$, namely, $\|p - \frac{1}{n-1}(q_1 + \cdots + q_{n-1})\|^2$. This is equal to $\|wM\|^2$, where

$$M := \begin{bmatrix} p/\|p\| \\ -q_1/(\|q_1\|\sqrt{n-1}) \\ \vdots \\ -q_{n-1}/(\|q_{n-1}\|\sqrt{n-1}) \end{bmatrix} \in \mathbb{R}^{n\times d}, \qquad w := \left[\|p\|, \frac{\|q_1\|}{\sqrt{n-1}}, \dots, \frac{\|q_{n-1}\|}{\sqrt{n-1}}\right] \in \mathbb{R}^n.$$

Because the quantum algorithm assumes input in quantum states, we can assume sampling and query access to the data points, giving the problem

Problem 8.7. Given $\mathrm{SQ}(M) \in \mathbb{R}^{n\times d}, \mathrm{Q}(w) \in \mathbb{R}^n$, approximate $(wM)(wM)^T$ to additive ε error with probability at least $1 - \delta$.

Corollary 8.8 ([Tan21, Theorem 4]). *There is a classical algorithm to solve Problem 8.7 in* $\mathcal{O}(\|M\|_F^4\|w\|^4\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ *time.*

Note that $\|M\|_F^2 = 2$ and $\|w\|^2 = \|p\|^2 + \frac{1}{n-1}\sum_{i=1}^{n-1}\|q_i\|^2$. The quantum algorithm has a quadratically faster runtime of $\mathcal{O}(\|M\|_F^2\|w\|^2\frac{1}{\varepsilon})$, ignoring polylog$(n, d)$ factors [LMR13; Tan21].

Proof. Recall our notation for the vector of row norms $m := [\|M(1,\cdot)\|, \dots, \|M(n,\cdot)\|]$ coming from Definition 4.7. We can rewrite $(wM)(wM)^T$ as an inner product $\langle u, v\rangle$ where

$$u := \sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{k=1}^{n} M(i,j)\|M(k,\cdot)\| e_i \otimes e_j \otimes e_k = M \otimes m$$
$$v := \sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{k=1}^{n} \frac{w_i w_k M(j,k)}{\|M(k,\cdot)\|} e_i \otimes e_j \otimes e_k,$$

where u and v are three-dimension tensors. By flattening u and v, we can represent them as two vectors in $\mathbb{R}^{(n\cdot d\cdot n)\times 1}$. We clearly have $\mathrm{Q}(v)$ from queries to M and w. As for getting $\mathrm{SQ}(u)$ from $\mathrm{SQ}(M)$: to sample, we first sample i according to m, sample j according to $M(i,\cdot)$, and sample k according to m; to query, compute $u_{i,j,k} = M(i,j)m(k)$. Finally, we can apply Lemma 5.17 to estimate $\langle u, v\rangle$. $\|u\| = \|M\|_F^2$ and $\|v\| = \|w\|^2$, so estimating $\langle u, v\rangle$ to ε additive error with probability at least $1 - \delta$ requires $O(\|M\|_F^4\|w\|^4\varepsilon^{-2}\log\frac{1}{\delta})$ samples. □

8.3 Principal component analysis

Principal component analysis (PCA) is an important data analysis tool, first proposed to be feasible via quantum computation by Lloyd, Mohseni, and Rebentrost [LMR14]. Given copies of states with density matrix $\rho = X^\dagger X$, the quantum PCA algorithm can prepare the state $\sum \lambda_i |v_i\rangle\langle v_i| \otimes |\hat{\lambda}_i\rangle\langle\hat{\lambda}_i|$, where λ_i and v_i are the eigenvalues and eigenvectors of $X^\dagger X$, and $\hat{\lambda}_i$ are eigenvalue estimates (up to additive error). See Prakash's PhD thesis [Pra14, Section 3.2] for a full analysis and Chakraborty, Gilyén, and Jeffery for a faster version of this algorithm in the block-encoding model [CGJ19]. Directly measuring the eigenvalue register is called *spectral sampling*, but such sampling is not directly useful for machine learning applications.

Though we do not know how to dequantize this protocol exactly, we can dequantize it in the low-rank setting, which is the only useful poly-logarithmic time application that Lloyd, Mohseni, and Rebentrost [LMR14] suggests for quantum PCA.

Problem 8.9 (PCA for low-rank matrices). 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 $\{v_i\}_{i=1}^k$, with probability $\geq 1-\delta$, compute eigenvalue estimates $\{\hat{\lambda}_i\}_{i=1}^k$ such that $\sum_{i=1}^k |\hat{\lambda}_i - \lambda_i| \leq \varepsilon \operatorname{tr}(X^\dagger X)$ and eigenvectors $\{\mathrm{SQ}_\phi(\hat{v}_i)\}_{i=1}^k$ such that $\|\hat{v}_i - v_i\| \leq \varepsilon$ for all i.

Note that we should think of λ_i as σ_i^2, where σ_i is the ith largest singular value of X. To robustly avoid degeneracy conditions, our runtime must depend on parameters for condition number and spectral gap:

$$K := \operatorname{tr}(X^\dagger X)/\lambda_k \geq k \quad \text{and} \quad \eta := \min_{i\in[k]} |\lambda_i - \lambda_{i+1}|/\|X\|^2. \tag{56}$$

We also denote $\kappa := \|X\|^2/\lambda_k$. Dependence on K and η are necessary to reduce Problem 8.9 to spectral sampling. If $K = \mathrm{poly}(n)$, then $\lambda_k = \operatorname{tr}(X^\dagger X)/\mathrm{poly}(n)$, so distinguishing λ_k from λ_{k+1} necessarily takes $\mathrm{poly}(n)$ samples, and even sampling λ_k once takes $\mathrm{poly}(n)$ samples. As a result, learning v_k is also impossible. A straightforward coupon collector argument (given e.g. by Tang [Tan21]) shows that Problem 8.9 can be solved by a quantum algorithm performing spectral sampling[25], with runtime depending polynomially on K and $\frac{1}{\eta}$. We omit this

[25]The quantum analogue to $\mathrm{SQ}(X)$ is efficient state preparation of X, a purification of ρ.

argument for brevity. Classically, we can solve this PCA problem with quantum-inspired techniques, as first noted in [Tan21].

Corollary 8.10. *For $0 < \varepsilon \lesssim \eta\|X\|^2/\|X\|_F^2$, we can solve Problem 8.9 in $\widetilde{\mathcal{O}}(\frac{\|X\|_F^6}{\lambda_k^2\|X\|^2}\eta^{-6}\varepsilon^{-6}\log^3\frac{k}{\delta})$ time to get $\mathrm{SQ}_\phi(\hat{v}_i)$ where $\overline{\boldsymbol{sq}}(\hat{v}_i) = \widetilde{\mathcal{O}}(\frac{\|X\|_F^4}{\lambda_i\|X\|^2}\eta^{-2}\varepsilon^{-2}\log^2\frac{1}{\delta})$.*

This improves significantly over prior work [Tan21, Theorem 8], which achieves the runtime of $\widetilde{\mathcal{O}}(\frac{\|X\|_F^{36}}{\|X\|^{12}\lambda_k^{12}}\eta^{-6}\varepsilon^{-12}\log^3\frac{k}{\delta})$.[26] The best quantum algorithm for this problem runs in $\widetilde{\mathcal{O}}(\frac{\|X\|_F\|X\|}{\lambda_k\varepsilon})$ time, up to factors of polylog(m, n) [CGJ19, Theorem 27].[27]

We approach the problem as follows. First, we use that an importance-sampled submatrix of X has approximately the same singular values as X itself (Lemma 5.14) 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. Our runtime is quite good because these piecewise linear step functions have relatively tame derivatives, as opposed to the thresholded inverse function, whose Lipschitz constants must incur quadratic and cubic overheads in terms of condition number.

Proof. We will assume that we know λ_k and η. If both are unknown, then we can estimate them with the singular value estimation procedure described below (Lemma 5.14).

Notice that $\eta\|X\|^2 \leq \lambda_k$ follows from our definition of η. The algorithm will proceed as follows: first, consider $C := SXT \in \mathbb{C}^{c\times r}$ as described in Theorem 7.1, with parameters

$$r := \widetilde{\mathcal{O}}\Big(\frac{\|X\|_F^2}{\eta^2\|X\|^2\varepsilon^2}\log\frac{k}{\delta}\Big) \qquad c := \widetilde{\mathcal{O}}\Big(\frac{\|X\|_F^2\|X\|^2}{\eta^2\lambda_k^2\varepsilon^2}\log\frac{k}{\delta}\Big).$$

[26] This runtime comes from taking $\varepsilon_\sigma = \varepsilon_v = \varepsilon$ and changing the normalization of the gap parameter $\eta = \eta\|X\|^2/\|X\|_F^2$ to correspond to the problem as formulated here.

[27] Given X in QRAM, this follows from applying Theorem 27 to a quantum state with density matrix of $X^\dagger X$ with $\alpha = \|X\|_F$ and $\Delta = \frac{\varepsilon\|X\|_F^2}{\|X\|} \lesssim \frac{\eta\|X\|}{\|X\|_F}$. The output is some estimate of $\sqrt{\lambda_i}$ to Δ error, which when squared is an estimate of λ_i to $\Delta\|X\| = \varepsilon\|X\|_F^2$ error as desired. Then, the density matrix is a probability distribution over eigenvectors with their corresponding eigenvalue estimate (which is enough to identify the eigenvector). The coupon collector argument mentioned above gives us access to all the top k eigenvalues and eigenvectors by running this algorithm $\|X\|_F^2/\lambda_k$ times [Tan21].

Consider computing the eigenvalues of $CC^\dagger$; denote the i^{th} eigenvalue $\hat{\lambda}_i$. Since $r, c \gtrsim \frac{\|X\|_F^2}{\lambda_k \varepsilon^2} \log \frac{1}{\delta}$, by Lemma 5.14 with error parameter $\frac{\varepsilon\sqrt{\lambda_k}}{8\|X\|_F}$, with probability $\geq 1 - \delta$,

$$\sqrt{\sum_{i=1}^{\min(m,n)} (\hat{\lambda}_i - \lambda_i)^2} \leq \frac{\varepsilon\sqrt{\lambda_k}}{8\|X\|_F}\|X\|_F^2.$$

These $\hat{\lambda}_i$'s for $i \in [k]$ have the desired property for eigenvalue estimates:

$$\sum_{i=1}^{k} |\hat{\lambda}_i - \lambda_i| \leq \sqrt{k}\sqrt{\sum_{i=1}^{k} (\hat{\lambda}_i - \lambda_i)^2} \leq \varepsilon\sqrt{k\lambda_k}\|X\|_F \leq \varepsilon\|X\|_F^2.$$

This bound also implies that, for all i, $|\hat{\lambda}_i - \lambda_i| \leq \frac{\varepsilon}{8}\|X\|_F^2$. Next, consider the eigenvalue transformations f_i for $i \in [k]$, defined

$$f_i(x) := \begin{cases} 0 & x - \hat{\lambda}_i < -\frac{1}{4}\eta\|X\|^2 \\ 2 + \frac{8}{\eta\|X\|^2}(x - \hat{\lambda}_i) & -\frac{1}{4}\eta\|X\|^2 \leq x - \hat{\lambda}_i < -\frac{1}{8}\eta\|X\|^2 \\ 1 & -\frac{1}{8}\eta\|X\|^2 \leq x - \hat{\lambda}_i < \frac{1}{8}\eta\|X\|^2 \\ 2 - \frac{8}{\eta\|X\|^2}(x - \hat{\lambda}_i) & \frac{1}{8}\eta\|X\|^2 \leq x - \hat{\lambda}_i < \frac{1}{4}\eta\|X\|^2 \\ 0 & \frac{1}{4}\eta\|X\|^2 \leq x - \hat{\lambda}_i \end{cases}.$$

This is a function that is one when $|x - \hat{\lambda}_i| \leq \frac{1}{8}\eta\|X\|^2$, zero when $|x - \hat{\lambda}_i| \geq \frac{1}{4}\eta\|X\|^2$, and interpolates between them otherwise. From the eigenvalue gap and the aforementioned bound $|\hat{\lambda}_i - \lambda_i| \leq \frac{1}{8}\eta\|X\|^2$, we can conclude that $f_i(X^\dagger X) = v_i v_i^\dagger$ exactly. Further, by Theorem 7.1, we can conclude that $R^\dagger \bar{f}_i(CC^\dagger)R$ approximates $v_i v_i^\dagger$, with C, R the exact approximations used to estimate singular values. The conditions of Theorem 7.1 are satisfied because $\varepsilon \lesssim 8 \leq \frac{8}{\eta} = L\|X\|^2$ for L the Lipschitz constant of f_i. The values of r, c are chosen so that $\|R^\dagger \bar{f}_i(CC^\dagger)R - f_i(X^\dagger X)\| \leq \varepsilon/2$ (note $f_i(0) = 0$):

$$r = \widetilde{\mathcal{O}}\Big(L^2\|X\|^2\|X\|_F^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{\|X\|_F^2}{\|X\|^2\eta^2\varepsilon^2}\log\frac{1}{\delta}\Big)$$
$$c = \widetilde{\mathcal{O}}\Big(\bar{L}^2\|X\|^6\|X\|_F^2\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{\|X\|^6\|X\|_F^2}{\eta^2\|X\|^4(\hat{\lambda}_i - \frac{1}{4}\eta\|X\|^2)^2\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{\|X\|^2\|X\|_F^2}{\eta^2\lambda_k^2\varepsilon^2}\log\frac{1}{\delta}\Big).$$

Further, f_i is chosen with respect to $\hat{\lambda}_i$ such that $R^\dagger \bar{f}_i(CC^\dagger)R$ is rank one, since $CC^\dagger$ has one eigenvalue between $\hat{\lambda}_i - \frac{1}{4}\eta\|X\|^2$ and $\hat{\lambda}_i + \frac{1}{4}\eta\|X\|^2$. Thus, this approximation is an outer product, $R^\dagger \bar{f}_i(CC^\dagger)R = \hat{v}_i\hat{v}_i^\dagger$, and we take the corresponding vector to be our eigenvector estimate: $\|\hat{v}_i\| \leq \sqrt{1+\varepsilon/2} \leq 1+\varepsilon/4$, so

$$\begin{aligned}
\varepsilon/2 &\geq \|(\hat{v}_i\hat{v}_i^\dagger - v_iv_i^\dagger)v_i\| && \text{by definition} \\
&= \|\langle \hat{v}_i, v_i\rangle \hat{v}_i - v_i\| && \text{by } \|v_i\|^2 = 1 \\
&\geq \|\hat{v}_i - v_i\| - (\langle \hat{v}_i, v_i\rangle - 1)\|\hat{v}_i\| && \text{by triangle inequality} \\
&\geq \|\hat{v}_i - v_i\| - (\|\hat{v}_i\|\|v_i\| - 1)\|u\| && \text{by Cauchy–Schwarz} \\
&\geq \|\hat{v}_i - v_i\| - (1+\varepsilon/4-1)(1+\varepsilon/4) && \text{by } \|\hat{v}_i\| \leq 1+\varepsilon/4 \\
&\geq \|\hat{v}_i - v_i\| - \varepsilon/2,
\end{aligned}$$

which is the desired bound. By choosing failure probability δ/k, the bound can hold true for all k with probability $\geq 1-\delta$.

Finally, we can get access to $\hat{v}_i = R^\dagger \bar{v}_i$, where $\bar{v}_i \in \mathbb{C}^r$ satisfies $\bar{v}_i^\dagger \bar{v}_i = \bar{f}_i(CC^\dagger)$. Since $\|\bar{v}_i^\dagger\| \leq \sqrt{\max_x \bar{f}_i(x)} \lesssim \lambda_i^{-\frac{1}{2}}$, using Lemmas 4.5 and 4.6, we have $\mathrm{SQ}_\phi(\hat{v}_i)$ with

$$\phi = r\frac{\sum_{s=1}^r |\hat{v}_i(s)|^2\|R(s,\cdot)\|^2}{\|R^\dagger \bar{v}_i\|^2} = r\frac{\sum_{s=1}^r |\hat{v}_i(s)|^2\|X\|_F^2}{\|R^\dagger \bar{v}_i\|^2 r} = \frac{\|\hat{v}_i\|^2\|X\|_F^2}{\|R^\dagger \bar{v}_i\|^2} \lesssim \frac{\|X\|_F^2}{\lambda_i(1-\varepsilon)^2} \lesssim \frac{\|X\|_F^2}{\lambda_i},$$

so $\overline{\boldsymbol{sq}}_\phi(\hat{v}_i) = \phi\, \boldsymbol{sq}_\phi(v) \log\frac{1}{\delta} \lesssim \frac{\|X\|_F^2}{\lambda_i} r \log\frac{1}{\delta}$. □

8.4 Matrix inversion and principal component regression

The low-rank matrix inversion algorithms given by Gilyén, Lloyd, and Tang [GLT18] and Chia, Lin, and Wang [CLW18] dequantize Harrow, Hassidim, and Lloyd's quantum matrix inversion algorithm (HHL) [HHL09] in the regime where the input matrix is *low-rank* instead of sparse. The corresponding quantum algorithm in this regime is given by Chakraborty, Gilyén, and Jeffery [CGJ19], among others. Since sparse matrix inversion is BQP-complete, it is unlikely that one can efficiently dequantize it. However, the variant of low-rank (non-sparse) matrix inversion appears often in quantum machine learning [Pra14; WZP18; RML14; CD16; RL18], making it an influential primitive in its own right.

Using our framework, we can elegantly derive the low-rank matrix inversion algorithm in a manner similar to prior quantum-inspired work [CGLLTW20]. Moreover, we can also handle the approximately low-rank regime and only invert the matrix on a well-conditioned subspace, solving principal component regression—for more discussion see [GSLW19]. Namely, we can find a thresholded pseudoinverse of an input matrix:

Definition 8.11 ($A^+_{\sigma,\eta}$). We define $A^+_{\sigma,\eta}$ to be any singular value transform of A satisfying:

$$A^+_{\sigma,\eta} := \operatorname{tinv}^{(\mathrm{SV})}_{\sigma,\eta}(A) \qquad \operatorname{tinv}_{\sigma,\eta}(\lambda)\begin{cases} = 1/\lambda & \lambda \geq \sigma \\ = 0 & \lambda < \sigma(1-\eta) \\ \in [0, \sigma^{-1}] & \text{otherwise} \end{cases}. \tag{57}$$

This definition is analogous to $A_{\sigma,\eta}$ in Section 8.1: it is A^+ for singular vectors with value $\geq \sigma$, zero for singular vectors with value $\leq \sigma(1-\eta)$, and a linear interpolation between the two in between.

Problem 8.12. Given $\mathrm{SQ}_\varphi(A) \in \mathbb{C}^{m\times n}, \mathrm{Q}(b) \in \mathbb{C}^m$, with probability $\geq 1-\delta$, get $\mathrm{SQ}_\phi(\hat{x})$ such that $\|\hat{x} - x^*\| \leq \varepsilon\|A\|^{-1}\|b\|$, where $x^* := A^+_{\sigma,\eta}b$.

The quantum algorithm obtains a state close to $|f(A)b\rangle$, where $f(x)$ is a polynomial approximation to $1/x$ in the interval $[-1, -1/\kappa] \cup [1/\kappa, 1]$ [GSLW19, Theorem 41]. Formally, this polynomial is the Chebyshev truncation of $(1-(1-x^2)^b)/x$ for $b = \lceil \kappa^2 \log(\kappa/\varepsilon) \rceil$, multiplied by the rectangle function from Lemma 8.3 to keep the truncation bounded.

Lemma 8.13 (Polynomial approximations of $1/x$, [GSLW19, Lemma 40], following [CKS17]). Let $\kappa > 1$ and $0 < \varepsilon < \frac{1}{2}$. There is an odd polynomial $p(x)$ of degree $\mathcal{O}(\kappa \log(\frac{\kappa}{\varepsilon}))$ with the properties that

- $|p(x) - 1/x| \leq \varepsilon$ for $x \in [-1, -1/\kappa] \cup [1/\kappa, 1]$;
- $|p(x)| = \mathcal{O}(\kappa \log \frac{\kappa}{\varepsilon})$.

We can solve Problem 8.12 by invoking Theorem 6.1 with the polynomial from Lemma 8.13.

Corollary 8.14 (Dequantizing linear regression). *Given a matrix $A \in \mathbb{C}^{m\times n}$ such that $0.01 \leq \|A\| \leq 1$; a vector $b \in \mathbb{C}^m$; and parameters $\varepsilon, 1/\kappa$ between 0 and 1, with $\mathcal{O}(\operatorname{nnz}(A) + \operatorname{nnz}(b))$ time pre-processing. Then there exists an algorithm that outputs a vector y such that $\|y - p(A)b\| \leq \varepsilon\|b\|/\kappa$ with probability at least 0.9, where p is the polynomial from Lemma 8.13 with parameters κ, ε. Further, the running time to compute a description of y is*

$$\widetilde{\mathcal{O}}\Big(\frac{\kappa^{11}\|A\|_{\mathrm{F}}^4}{\varepsilon^2}\Big),$$

and from this description we can output a sample from y in $\widetilde{\mathcal{O}}(\frac{\kappa^{10}\|A\|_{\mathrm{F}}^4\|b\|^2}{\varepsilon^2\|y\|^2})$ time.

For a result with a better η dependence, we can show the following.

Corollary 8.15. *For $0 < \varepsilon \lesssim \frac{\|A\|^2}{\sigma^2}$ and $\eta \leq 0.99$, we can solve Problem 8.12 in $\widetilde{\mathcal{O}}(\frac{\varphi^6 K^3\kappa^{11}}{\eta^6\varepsilon^6}\log^3\frac{1}{\delta})$ time to give $\mathrm{SQ}_\phi(\hat{x})$ for $\overline{\boldsymbol{sq}}_\phi(\hat{x}) = \widetilde{\mathcal{O}}(\frac{\varphi^4 K^2\kappa^5}{\eta^2\varepsilon^2}\frac{\|x^*\|^2}{\|\hat{x}\|^2}\log^2\frac{1}{\delta})$.*

This should be compared to [GLT18], which applies only to strictly rank-k A with $\varphi = 1$ and gets the incomparable runtime of $\widetilde{\mathcal{O}}(\frac{K^3\kappa^8 k^6}{\eta^6\varepsilon^6}\log^3\frac{1}{\delta})$. The corresponding quantum algorithm using block-encodings takes $\mathcal{O}(\|A\|_{\mathrm{F}}/\sigma)$ time, up to $\operatorname{polylog}(m,n)$ factors, to get this result for constant η [GSLW19, Theorem 41].

If we further assume that $\varepsilon < 0.99$ and b is in the image of A, then $\overline{\boldsymbol{sq}}_\phi(\hat{x})$ can be simplified, since $\|\hat{x}\| \geq \|x^*\| - \varepsilon\|A\|^{-1}\|b\| \geq (1-\varepsilon)\|x^*\|$, so $\frac{\|x^*\|}{\|\hat{x}\|} \leq 100$. However, this algorithm also works for larger ε; namely, if we only require that $\|\hat{x} - x^*\| \leq \varepsilon\sigma^{-1}\|b\|$ (a "worst-case" error bound), then this algorithm works with runtime smaller by a factor of κ^3 (and $\overline{\boldsymbol{sq}}_\phi(\hat{x})$ smaller by a factor of κ).

The algorithm comes from rewriting $A^+_{\sigma,\eta}b = \iota(A^\dagger A)A^\dagger b$ for ι a function encoding a thresholded inverse. Namely, $\iota(x) = 1/x$ for $x \geq \sigma^2$, $\iota(x) = 0$ for $x \leq (1-\eta)^2\sigma^2$, and is a linear interpolation between the endpoints for $x \in [(1-\eta)^2\sigma^2, \sigma^2]$. 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.

Proof. We will solve our problem for $x^* = A^+_{\sigma,\eta}b = \iota(A^\dagger A)A^\dagger b$ where

$$\iota(x) := \begin{cases} 0 & x < \sigma^2(1-\eta)^2 \\ \frac{1}{(2\eta-\eta^2)\sigma^4}(x - \sigma^2(1-\eta)^2) & \sigma^2(1-\eta)^2 \leq x < \sigma^2 \\ \frac{1}{x} & \sigma^2 \leq x \end{cases}.$$

So, if we can estimate $\iota(A^\dagger A)$ such that $\|\iota(A^\dagger A) - R^\dagger \bar{\iota}(CC^\dagger)R\| \leq \frac{\varepsilon}{\|A\|^2}$, then as desired,

$$\|A^+_{\sigma,\eta}b - R^\dagger \bar{\iota}(CC^\dagger)RA^\dagger b\| \leq \frac{\varepsilon}{\|A\|}\|b\| \leq \varepsilon\|A^+_{\sigma,\eta}b\|.$$

By Theorem 7.1 with $L = \frac{1}{(2\eta-\eta^2)\sigma^4}$ and $\bar{L} = \frac{1}{(1-\eta)^2(2\eta-\eta^2)\sigma^6}$, we can find such R and C with

$$r = \widetilde{\mathcal{O}}(\varphi^2 \frac{\|A\|^2\|A\|_F^2}{(2\eta-\eta^2)^2\sigma^8\frac{\varepsilon^2}{\|A\|^4}} \log\frac{1}{\delta}) = \widetilde{\mathcal{O}}(\frac{\varphi^2 K\kappa^3}{\eta^2\varepsilon^2}\log\frac{1}{\delta})$$
$$c = \widetilde{\mathcal{O}}(\varphi^2 \frac{\|A\|^6\|A\|_F^2}{(1-\eta)^4(2\eta-\eta^2)^2\sigma^{12}\frac{\varepsilon^2}{\|A\|^4}} \log\frac{1}{\delta}) = \widetilde{\mathcal{O}}(\frac{\varphi^2 K\kappa^5}{\eta^2\varepsilon^2}\log\frac{1}{\delta}).$$

Computing the SVD of a matrix of this size dominates the runtime, giving the complexity in the theorem statement. Next, we would like to further approximate $R^\dagger\bar{\iota}(CC^\dagger)RA^\dagger b$. We will do this by estimating $RA^\dagger b$ by some vector u to $\varepsilon\sigma^3\|A\|^{-1}\|b\| = \varepsilon\|A\|_F^2\|b\|K^{-1}\kappa^{-\frac{1}{2}}$ error, since then, using the bounds from Lemma 7.2,

$$\begin{aligned}\|R^\dagger\bar{\iota}(CC^\dagger)RA^\dagger b - R^\dagger\bar{\iota}(CC^\dagger)u\| &\leq \left\|R^\dagger\sqrt{\bar{\iota}(CC^\dagger)}\right\|\left\|\sqrt{\bar{\iota}(CC^\dagger)}\right\|\|RA^\dagger b - u\| \\ &\lesssim \sqrt{\sigma^{-2} + \tfrac{\varepsilon}{\|A\|^2}}\sigma^{-2}(\varepsilon\sigma^3\|A\|^{-1}\|b\|) \lesssim \varepsilon\|A\|^{-1}\|b\|.\end{aligned}$$

We use Remark 5.18 to estimate $u(i) = R(i,\cdot)A^\dagger b$, for all $i \in [r]$, to $\varepsilon\|R(i,\cdot)\|\|A\|_F\|b\|K^{-1}\kappa^{-\frac{1}{2}}$ error, with probability $\geq 1 - \delta/r$. This takes $\mathcal{O}(\varphi\frac{K^2\kappa}{\varepsilon^2}\log\frac{r}{\delta})$ samples for each of the r entries. This implies that $\hat{x} := R^\dagger\bar{\iota}(CC^\dagger)u$ has the desired error and failure probability. Finally, we can use

Lemmas 4.5 and 4.6 with matrix $R^\dagger$ and vector $\bar{\iota}(CC^\dagger)u$ to get $\mathrm{SQ}_\phi(\hat{x})$ for

$$\begin{aligned}
\phi &= \varphi r \frac{\sum_{s=1}^{r} |[\bar{\iota}(CC^\dagger)u](s)|^2 \|R(s,\cdot)\|^2}{\|\hat{x}\|^2} \\
&= \varphi^2 \frac{\|\bar{\iota}(CC^\dagger)u\|^2 \|A\|_{\mathrm{F}}^2}{\|\hat{x}\|^2} && \text{by } \|R(s,\cdot)\| \leq \|A\|_{\mathrm{F}}\sqrt{\varphi/r} \\
&\leq \varphi^2 \frac{(\|\bar{\iota}(CC^\dagger)R\|\|A^\dagger\|\|b\| + \|\bar{\iota}(CC^\dagger)\|\|RA^\dagger b - u\|)^2 \|A\|_{\mathrm{F}}^2}{\|\hat{x}\|^2} && \text{by linear algebra} \\
&\lesssim \varphi^2 \frac{(\sigma^{-3}\|A\|\|b\| + \sigma^{-4}\varepsilon\sigma^3\|b\|/\|A\|)^2 \|A\|_{\mathrm{F}}^2}{\|\hat{x}\|^2} && \text{by prior bounds} \\
&\lesssim \varphi^2 \frac{\sigma^{-6}\|A\|^2\|b\|^2\|A\|_{\mathrm{F}}^2}{\|\hat{x}\|^2} && \text{by } \varepsilon \lesssim \|A\|^2/\sigma^2 \\
&\leq \varphi^2 K\kappa^2 \frac{\|x^*\|^2}{\|\hat{x}\|^2}, && \text{by } \|A\|^{-1}\|b\| \leq \|x^*\|
\end{aligned}$$

so $\overline{\boldsymbol{sq}}_\phi(\hat{x}) = \phi\, \boldsymbol{sq}_\phi(\hat{x}) \log\frac{1}{\delta} = \mathcal{O}(r\varphi^2 K\kappa^2 \frac{\|x^*\|^2}{\|\hat{x}\|^2} \log\frac{1}{\delta})$. □

8.5 Support vector machines

In this section, we use our framework to dequantize Rebentrost, Mohseni, and Lloyd's quantum support vector machine [RML14], which was previously noted to be possible by Ding, Bao, and Huang [DBH22]. Mathematically, the support vector machine is a simple machine learning model attempting to label points in $\mathbb{R}^m$ as $+1$ or -1. Given input data points $x_1, \dots, x_m \in \mathbb{R}^n$ and their corresponding labels $y \in \{\pm 1\}^m$. Let $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ be the specification of hyperplanes separating these points. It is possible that no such hyperplane satisfies all the constraints. To resolve this, we add a slack vector $e \in \mathbb{R}^m$ such that $e(j) \geq 0$ for $j \in [m]$. We want to minimize the squared norm of the residuals:

$$\begin{aligned}
\min_{w,b} \quad & \frac{1}{2}\frac{\|w\|^2}{2} + \frac{\gamma}{2}\|e\|^2 \\
\text{s.t.} \quad & y(i)(w^T x_i + b) = 1 - e(i), \quad \forall i \in [m].
\end{aligned}$$

The dual of this problem is to maximize over the Karush-Kuhn-Tucker multipliers of a Lagrange function, taking partial derivatives of which yields the linear system

$$\begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & XX^T + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}, \tag{58}$$

where $\vec{1}$ is the all-ones vector and $X = \{x_1, \dots, x_m\} \in \mathbb{C}^{m \times n}$. Call the above $m+1 \times m+1$ matrix F, and $\hat{F} := F/\operatorname{tr}(F)$.

The quantum algorithm, given X and y in QRAM, outputs a quantum state $|\hat{F}^+_{\lambda,0.01}[\begin{smallmatrix} 0 \\ y \end{smallmatrix}]\rangle$ (Definition 8.11) in $\widetilde{\mathcal{O}}(\frac{1}{\lambda^3\varepsilon^3} \operatorname{polylog}(mn))$ time. The quantum-inspired analogue is as follows.

Problem 8.16. Given $\mathrm{SQ}(X) \in \mathbb{R}^{m\times n}$ and $\mathrm{SQ}(y) \in \mathbb{R}^m$, for $\|\hat{F}\| \le 1$, output $\mathrm{SQ}_\phi(v) \in \mathbb{R}^{m+1}$ such that $\|\hat{x} - \hat{F}^+_{\lambda,\eta}[\begin{smallmatrix} 0 \\ y \end{smallmatrix}]\| \le \varepsilon \|\hat{F}^+_{\lambda,\eta}[\begin{smallmatrix} 0 \\ y \end{smallmatrix}]\|$ with probability $\ge 1 - \delta$.

Note that we must assume $\|\hat{F}\| \le 1$; the quantum algorithm makes the same assumption[28]. Another dequantization was reported in [DBH22], which, assuming X is strictly low-rank (with minimum singular value σ), outputs a description of $(XX^T)^+ y$ that can be used to classify points. This can be done neatly in our framework: express $(XX^T)^+$ (or, more generally, $(XX^T)^+_{\sigma,\eta}$) as $Xf(X^TX)X^T$ for the appropriate choice of f. Then, use Theorem 7.1 to approximate $f(X^TX) \approx R^TZR$ and use Lemma 5.4 to approximate $XR^T \approx CW^T$. This gives an approximate "CUC" decomposition of the desired matrix, since $Xf(X^TX)X^T \approx XR^TZRX^T \approx CW^TZWC^T$, which we can use for whatever purpose we like.

For our solution to Problem 8.16, though, we simply reduce to matrix inversion as described in Section 8.4: we first get $\mathrm{SQ}_\phi(\hat{F})$, and then we apply Corollary 8.15 to complete. Section VI.C of [DBH22] claims to dequantize this version, but gives no correctness bounds[29] or runtime bounds (beyond arguing it is polynomial in the desired parameters).

Corollary 8.17. *For $0 < \varepsilon \lesssim 1$ and $\eta \le 0.99$, we can solve Problem 8.16 in $\widetilde{\mathcal{O}}(\lambda^{-28}\eta^{-6}\varepsilon^{-6}\log^3\frac{1}{\delta})$ time, where we get $\mathrm{SQ}_\phi(v)$ for $\overline{\mathbf{sq}}_\phi(v) = \widetilde{\mathcal{O}}(\lambda^{-14}\eta^{-2}\varepsilon^{-4}\log^2(\frac{1}{\delta})\log(\frac{m}{\delta}))$.*

The runtimes in the statement are not particularly tight, but we chose the form to mirror the runtime of the QSVM algorithm, which similarly depends polynomially on $\frac{1}{\lambda}$ and $\frac{1}{\eta}$.

[28]The algorithm as written in [RML14] assumes that $\|F\| \le 1$; we confirmed with an author that this is a typo.

[29]The correctness of this dequantization is unclear, since the approximations performed in this section incur significant errors.

Proof. Consider constructing $\mathrm{SQ}_\varphi(K) \in \mathbb{C}^{m\times m}$ as follows. To query an entry $K(i,j)$, we estimate $X(i,\cdot)X(j,\cdot)^T$ to $\varepsilon\|X(i,\cdot)\|\|X(j,\cdot)\|$ error. We define $K(i,j)$ to be this estimate. Using Lemma 5.17, we can do this in $\mathcal{O}(\frac{1}{\varepsilon^2}\log\frac{q}{\delta})$ time. q here refers to the number of times the query oracle is used, so in total the subsequent algorithm will only have an errant query with probability $\geq 1-\delta$. (q will not appear in the runtime because it's folded into a polylog term.) Then, we can take $\tilde{K} := xx^T$, where $x \in \mathbb{R}^m$ is the vector of row norms of X, since by Cauchy–Schwarz,

$$K(i,j) \leq X(i,\cdot)X(j,\cdot)^T + \varepsilon\|X(i,\cdot)\|\|X(j,\cdot)\| \leq (1+\varepsilon)\|X(i,\cdot)\|\|X(j,\cdot)\| = \tilde{K}(i,j).$$

Since we have $\mathrm{SQ}(x)$ from $\mathrm{SQ}(X)$, we have $\mathrm{SQ}(\tilde{K})$ with $\boldsymbol{sq}(\tilde{K}) = \mathcal{O}(1)$ by Lemma 4.8. $\|\tilde{K}\|_\mathrm{F}^2 = (1+\varepsilon)^2\|X\|_\mathrm{F}^4$, so we have $\mathrm{SQ}_\varphi(K)$ for $\varphi = (1+\varepsilon)^2\frac{\|X\|_\mathrm{F}^4}{\|K\|_\mathrm{F}^2}$. We can trivially get $\mathrm{SQ}(L)$ for $L := \left[\begin{smallmatrix} 0 & \vec{1}^T \\ \vec{1} & \gamma^{-1}I \end{smallmatrix}\right]$ with $\boldsymbol{sq}(L) = \mathcal{O}(1)$. Our approximation to $\hat{F}$ is

$$M := \frac{1}{\mathrm{tr}(F)}\Big(L + \left[\begin{smallmatrix} 0 & \vec{0}^T \\ \vec{0} & K \end{smallmatrix}\right]\Big); \qquad \|M - \hat{F}\| \leq \frac{1}{\mathrm{tr}(F)}\|K - XX^T\|_\mathrm{F} \leq \frac{1}{\mathrm{tr}(F)}\varepsilon\|X\|_\mathrm{F}^2 \leq \varepsilon.$$

Using Lemma 4.9, we have $\mathrm{SQ}_{\varphi'}(M)$ with

$$\varphi' = \frac{2((1+\varepsilon)^2\frac{\|X\|_\mathrm{F}^4}{\|K\|_\mathrm{F}^2}\|K\|_\mathrm{F}^2 + \|L\|_\mathrm{F}^2)}{\mathrm{tr}(F)^2\|M\|_\mathrm{F}^2} \lesssim \frac{\|X\|_\mathrm{F}^4 + \gamma^{-2}m + 2m}{(\|X\|_\mathrm{F}^2 + m\gamma^{-1})^2\|M\|_\mathrm{F}^2} \lesssim \frac{1}{\|M\|_\mathrm{F}^2},$$

where the last inequality uses that $\mathrm{tr}(F) \geq \sqrt{m}$, which follows from $\|\hat{F}\| \leq 1$:

$$1 = \|\hat{F}\|\|\left[\begin{smallmatrix} 0 \\ \vec{1}/\sqrt{m} \end{smallmatrix}\right]\| \geq \|\hat{F}\left[\begin{smallmatrix} 0 \\ \vec{1}/\sqrt{m} \end{smallmatrix}\right]\| \geq \frac{\sqrt{m}}{\mathrm{tr}(F)}.$$

Note that we can compute $\mathrm{tr}(F)$ given $\mathrm{SQ}(X)$. So, applying Corollary 8.15, we can get the desired $\mathrm{SQ}_\phi(v)$ in runtime

$$\widetilde{\mathcal{O}}\Big(\frac{\varphi^6\|M\|_\mathrm{F}^6\|M\|^{22}}{\lambda^{28}\eta^6\varepsilon^6}\log^3\frac{1}{\delta}\Big) \lesssim \widetilde{\mathcal{O}}\Big(\frac{\|M\|^{22}}{\|M\|_\mathrm{F}^6\lambda^{28}\eta^6\varepsilon^6}\log^3\frac{1}{\delta}\Big) \lesssim \widetilde{\mathcal{O}}\Big(\frac{1}{\lambda^{28}\eta^6\varepsilon^6}\log^3\frac{1}{\delta}\Big).$$

Here, we used that $\|M\| \leq \|M\|_\mathrm{F} \lesssim 1$, which we know since $\varphi' \geq 1$ (by our definition of oversampling and query access). That $\mathrm{Q}(M) = \mathcal{O}(\frac{1}{\varepsilon^2}\log\frac{q}{\delta})$ does not affect the runtime, since the dominating cost is still the SVD. On the other hand, this does come into play for the runtime for sampling:

$$\overline{\boldsymbol{sq}}_{\phi}(v) = \widetilde{\mathcal{O}}\Big(\frac{\varphi^4\|M\|_{\mathrm{F}}^4\|M\|^{10}}{\eta^2\varepsilon^2}\log^2\big(\frac{1}{\delta}\big)\frac{1}{\varepsilon^2}\log\big(\frac{m}{\delta}\big)\Big).$$

We take $q = m$ to guarantee that all future queries will be correct with probability $\geq 1-\delta$. $\square$

The normalization used by the quantum and quantum-inspired SVM algorithms means that these algorithms fail when X has too small Frobenius norm, since then the singular values from XX^T are all filtered out. Below, we describe an alternative method that relies less on normalization assumptions, instead simply computing F^+. This is possible if we depend on $\|X\|_{\mathrm{F}}^2\gamma$ in the runtime. Recall from Eq. (58) that we regularize by adding $\gamma^{-1}I$, so γ^{-1} acts as a singular value lower bound and $\|X\|_{\mathrm{F}}^2\gamma$ implicitly constrains.

Corollary 8.18. *Given* $\mathrm{SQ}(X^T)$ *and* $\mathrm{SQ}(y)$*, with probability* $\geq 1-\delta$*, we can output a real number* $\hat{b}$ *such that* $|b-\hat{b}| \leq \varepsilon(1+b)$ *and* $\mathrm{SQ}_{\phi}(\hat{\alpha})$ *such that* $\|\hat{\alpha}-\alpha\| \leq \varepsilon\gamma\|y\|$*, where* α *and* b *come from Eq. (58). Our algorithm runs in* $\widetilde{\mathcal{O}}(\|X\|_{\mathrm{F}}^6\|X\|^{16}\gamma^{11}\varepsilon^{-6}\log^3\frac{1}{\delta})$ *time, with* $\overline{\boldsymbol{sq}}_{\phi}(\hat{\alpha}) = \widetilde{\mathcal{O}}(\|X\|_{\mathrm{F}}^4\|X\|^6\gamma^5\frac{\gamma^2 m}{\|\hat{\alpha}\|^2}\varepsilon^{-4}\log^2\frac{1}{\delta})$*. Note that when* $\gamma^{-1/2}$ *is chosen to be sufficiently large (e.g.* $O(\|X\|_{\mathrm{F}})$*) and* $\|\alpha\| = \Omega(\gamma\|y\|)$*, this runtime is dimension-independent.*

Proof. Denote $\sigma^2 := \gamma^{-1}$, and redefine $X \leftarrow X^T$ (so we have $\mathrm{SQ}(X)$ instead of $\mathrm{SQ}(X^T)$). By the block matrix inversion formula[30] we know that

$$\begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & M \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{1}{\vec{1}^T M^{-1}\vec{1}} & \frac{\vec{1}^T M^{-1}}{\vec{1}^T M^{-1}\vec{1}} \\ \frac{M^{-1}\vec{1}}{\vec{1}^T M^{-1}\vec{1}} & M^{-1} - \frac{M^{-1}\vec{1}\vec{1}^T M^{-1}}{\vec{1}^T M^{-1}\vec{1}} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & M \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ y \end{bmatrix} = \begin{bmatrix} \frac{\vec{1}^T M^{-1} y}{\vec{1}^T M^{-1}\vec{1}} \\ M^{-1}\Big(y - \frac{\vec{1}^T M^{-1} y}{\vec{1}^T M^{-1}\vec{1}}\vec{1}\Big) \end{bmatrix}.$$

So, we have reduced the problem of inverting the modified matrix to just inverting M^{-1} where $M = X^T X + \sigma^{-2} I$. M is invertible because $M \succeq \sigma^2 I$. Note that $M^{-1} = f(X^T X)$, where

$$f(\lambda) := \frac{1}{\lambda + \sigma^2}.$$

[30] In a more general setting, we would use the Sherman-Morrison inversion formula, or the analogous formula for functions of matrices subject to rank-one perturbations.

So, by Theorem 7.1, we can find $R^\dagger \bar{f}(CC^\dagger)R$ such that $\|R^\dagger \bar{f}(CC^\dagger)R + \frac{1}{\sigma^2}I - f(X^TX)\| \leq \varepsilon\sigma^{-2}$, where (because $L = \sigma^{-4}$, $\bar{L} = \sigma^{-6}$)

$$r = \widetilde{\mathcal{O}}\Big(\frac{L^2\|A\|^2\|A\|_{\mathrm{F}}^2\sigma^4}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{K\kappa}{\varepsilon^2}\log\frac{1}{\delta}\Big),$$
$$c = \widetilde{\mathcal{O}}\Big(\frac{\bar{L}^2\|A\|^6\|A\|_{\mathrm{F}}^2\sigma^4}{\varepsilon^2}\log\frac{1}{\delta}\Big) = \widetilde{\mathcal{O}}\Big(\frac{K\kappa^3}{\varepsilon^2}\log\frac{1}{\delta}\Big).$$

So, the runtime for estimating this is $\widetilde{\mathcal{O}}(\frac{K^3\kappa^5}{\varepsilon^6}\log^3\frac{1}{\delta})$. We further approximate using Lemma 5.7: we find $r_1 \approx R^\dagger\vec{1}$, $r_y \approx R^\dagger\vec{y}$, and $\gamma \approx \vec{1}^\dagger y$ in $O(r\frac{K}{\varepsilon^2}\log\frac{1}{\delta})$ time (for the first two) and $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ time (for the last one) such that the following bounds hold:

$$\|R^\dagger\vec{1} - r_1\| \leq \varepsilon\sqrt{m}\sigma \qquad \|R^\dagger y - r_y\| \leq \varepsilon\sqrt{m}\sigma \qquad |\vec{1}^\dagger y - \gamma| \leq \varepsilon m \tag{59}$$

Via Lemma 7.2, we observe the following additional bounds:

$$\|M^{-1}\| \leq \sigma^{-2} \qquad \|R^\dagger(\bar{f}(CC^\dagger))^{1/2}\| \leq (1+\varepsilon)\sigma^{-1} \qquad \|(\bar{f}(CC^\dagger))^{1/2}\| \leq \sigma^{-2} \tag{60}$$

Now, we compute what the subsequent errors are for replacing M^{-1} with $N := R^\dagger ZR + \frac{1}{\sigma^2}I$, where $Z := \bar{f}(CC^\dagger)$.

$$\begin{aligned}
\frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}} &= \frac{\vec{1}^\dagger(R^\dagger ZR+\sigma^{-2}I)y \pm \|\vec{1}\|\|y\|\|R^\dagger ZR+\sigma^{-2}I-M^{-1}\|}{\vec{1}^\dagger(R^\dagger ZR+\sigma^{-2}I)\vec{1} \pm \|\vec{1}\|^2\|R^\dagger ZR+\sigma^{-2}I-M^{-1}\|} \\
&= \frac{\vec{1}^\dagger R^\dagger ZRy + \sigma^{-2}\vec{1}^\dagger y \pm \varepsilon\sigma^{-2}m}{\vec{1}^\dagger R^\dagger ZR\vec{1} + \sigma^{-2}\vec{1}^\dagger\vec{1} \pm \varepsilon\sigma^{-2}m} && \text{by SVT bound} \\
&= \frac{\vec{1}^\dagger R^\dagger Zr_y \pm \|\vec{1}^\dagger R^\dagger Z\|\|Ry-r_y\| + \sigma^{-2}\gamma \pm \sigma^{-2}|\gamma - \vec{1}^\dagger y| \pm \varepsilon\sigma^{-2}m}{\vec{1}^\dagger R^\dagger Zr_1 \pm \|\vec{1}^\dagger R^\dagger Z\|\|R\vec{1}-r_1\| + (1\pm\varepsilon)\sigma^{-2}m} \\
&= \frac{\vec{1}^\dagger R^\dagger Zr_y \pm (\sqrt{m}(1+\varepsilon)\sigma^{-3})(\varepsilon\sigma\sqrt{m}) + \sigma^{-2}\gamma \pm 2\varepsilon\sigma^{-2}m}{\vec{1}^\dagger R^\dagger Zr_1 \pm (\sqrt{m}(1+\varepsilon)\sigma^{-3})(\varepsilon\sigma\sqrt{m}) + \sigma^{-2}m \pm \varepsilon\sigma^{-2}m} && \text{by Eqs. (59) and (60)} \\
&= \frac{r_1^\dagger Zr_y \pm \|R\vec{1}-r_1\|\|Zr_y\| + \sigma^{-2}\gamma \pm \mathcal{O}(\varepsilon\sigma^{-2}m)}{r_1^\dagger Zr_1 \pm \|R\vec{1}-r_1\|\|Zr_1\| + \sigma^{-2}m \pm \mathcal{O}(\varepsilon\sigma^{-2}m)} \\
&= \frac{r_1^\dagger Zr_y \pm \varepsilon\sigma\sqrt{m}(\|ZRy\| + \|Z\|\|Ry-r_y\|) + \sigma^{-2}\gamma \pm \mathcal{O}(\varepsilon\sigma^{-2}m)}{r_1^\dagger Zr_1 \pm \varepsilon\sigma\sqrt{m}(\|ZR\vec{1}\| + \|Z\|\|R\vec{1}-r_1\|) + \sigma^{-2}m \pm \mathcal{O}(\varepsilon\sigma^{-2}m)} && \text{by Eq. (59)} \\
&= \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma \pm \mathcal{O}(\varepsilon\sigma^{-2}m)}{r_1^\dagger Zr_1 + \sigma^{-2}m \pm \mathcal{O}(\varepsilon\sigma^{-2}m)} && \text{by Eqs. (59) and (60)} \\
&= \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}(1\pm\mathcal{O}(\varepsilon)) \pm \mathcal{O}(\varepsilon). && \text{by } r_1^\dagger Zr_1 \geq 0
\end{aligned}$$

We will approximate the output vector as

$$M^{-1}y - \frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}}M^{-1}\vec{1} \approx R^\dagger Zr_y + \sigma^{-2}y - \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}(R^\dagger Zr_1 + \sigma^{-2}\vec{1}).$$

To analyze this, we first note that

$$\begin{aligned}
\|M^{-1}y - R^\dagger Zr_y + \sigma^{-2}y\| &\leq \|M^{-1} - R^\dagger ZR - \sigma^{-2}I\|\|y\| + \|R^\dagger Z\|\|Ry - r_y\| \\
&\leq \varepsilon\sigma^{-2}\sqrt{m} + (1+\varepsilon)\sigma^{-3}\varepsilon\sigma\sqrt{m} \lesssim \varepsilon\sigma^{-2}\sqrt{m}
\end{aligned}$$

and analogously, $\|M^{-1}\vec{1} - R^\dagger Zr_1 + \sigma^{-2}\vec{1}\| \lesssim \varepsilon\sigma^{-2}\sqrt{m}$. We also use that

$$\frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}} \leq \frac{\|\vec{1}M^{-1/2}\|\|M^{-1/2}y\|}{\|M^{-1/2}\vec{1}\|^2} = \frac{\|M^{-1/2}y\|}{\|M^{-1/2}\vec{1}\|} \leq \frac{\|X\|}{\sigma}. \tag{61}$$

With these bounds, we can conclude that (continuing to use Eqs. (59) and (60))

$$\begin{aligned}
&\Big\|M^{-1}\Big(y - \frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}}\vec{1}\Big) - \Big(R^\dagger Zr_y + \sigma^{-2}y - \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}(R^\dagger Zr_1 + \sigma^{-2}\vec{1})\Big)\Big\| \\
&\leq \|M^{-1}y - R^\dagger Zr_y + \sigma^{-2}y\| + \Big\|\frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}}M^{-1}\vec{1} - \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}(R^\dagger Zr_1 + \sigma^{-2}\vec{1})\Big)\Big\| \\
&\leq \varepsilon\sigma^{-2}\sqrt{m} + \frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}}\|M^{-1}\vec{1} - R^\dagger Zr_1 - \sigma^{-2}\vec{1}\| + \Big|\frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}} - \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}\Big|\|R^\dagger Zr_1 + \sigma^{-2}\vec{1}\| \\
&\lesssim \Big(1 + \frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}}\Big)\varepsilon\sigma^{-2}\sqrt{m} + \varepsilon\Big(1 + \frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}}\Big)\|R^\dagger Zr_1 + \sigma^{-2}\vec{1}\| \\
&= \varepsilon\Big(1 + \frac{\vec{1}^\dagger M^{-1}y}{\vec{1}^\dagger M^{-1}\vec{1}}\Big)\Big(\sigma^{-2}\sqrt{m} + \|R^\dagger Zr_1 + \sigma^{-2}\vec{1}\|\Big) \\
&\lesssim \varepsilon\frac{\|X\|}{\sigma}\Big(\sigma^{-2}\sqrt{m} + \|M^{-1}\vec{1}\| + \|(R^\dagger ZR + \sigma^{-2}I - M^{-1})\vec{1}\| + \|R^\dagger Zr_1 - R^\dagger ZR\vec{1}\|\Big) \qquad \text{by Eq. (61)} \\
&\lesssim \varepsilon\frac{\|X\|}{\sigma}\Big(\sigma^{-2}\sqrt{m} + \sigma^{-2}\sqrt{m} + \varepsilon\sigma^{-2}\sqrt{m} + \varepsilon\sigma^{-2}\sqrt{m}\Big) \\
&\lesssim \varepsilon\frac{\|X\|}{\sigma}\sigma^{-2}\sqrt{m}.
\end{aligned}$$

So, by rescaling ε down by $\frac{\|X\|}{\sigma}$, it suffices to sample from

$$\hat{\alpha} := R^\dagger Z\Big(r_y - \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}r_1\Big) - \sigma^{-2}\Big(y - \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}\vec{1}\Big).$$

To gain sampling and query access to the output, we consider this as a matrix-vector product, where the matrix is $(R^\dagger \mid y \mid \vec{1})$ and the vector is the corresponding coefficients in the linear combination. Then, by Lemmas 4.5 and 4.6, we can get $\mathrm{SQ}_\phi(\hat{\alpha})$ for

$$\begin{aligned}
\phi &= (r+2)\left(\frac{\|X\|_{\mathrm{F}}^2}{r}\Big\|Z\Big(r_y - \frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}r_1\Big)\Big\|^2 + \sigma^{-4}\Big(\|y\|^2 + \Big(\frac{r_1^\dagger Zr_y + \sigma^{-2}\gamma}{r_1^\dagger Zr_1 + \sigma^{-2}m}\Big)^2\|\vec{1}\|^2\Big)\right)\|\hat{\alpha}\|^{-2} \\
&\lesssim \Big(\|X\|_{\mathrm{F}}^2\frac{\|X\|^2}{\sigma^2}\sigma^{-6}m + r\sigma^{-4}\frac{\|X\|^2}{\sigma^2}m\Big)\|\hat{\alpha}\|^{-2} \lesssim \Big(\frac{\|X\|_{\mathrm{F}}^2}{\sigma^2} + r\Big)\frac{\|X\|^2}{\sigma^2}\frac{\sigma^{-4}m}{\|\hat{\alpha}\|^2}
\end{aligned}$$

so $\overline{\boldsymbol{sq}}_\phi(\hat{\alpha}) = \phi\,\boldsymbol{sq}_\phi(\hat{\alpha})\log\frac{1}{\delta} = \mathcal{O}(r(\frac{\|X\|_{\mathrm{F}}^2}{\sigma^2} + r)\frac{\|X\|^2}{\sigma^2}\frac{\sigma^{-4}m}{\|\hat{\alpha}\|^2}\log\frac{1}{\delta})$. □

Notice that $\varepsilon\gamma\|y\|$ is the right notion, since γ is an upper bound on the spectral norm of the inverse of the matrix in Eq. (58). We assume $\mathrm{SQ}(X^T)$ instead of $\mathrm{SQ}(X)$ for convenience, though both are possible via the observation that $f(XX^T) = X\bar{f}(X^TX)X^T$.

8.6 Hamiltonian simulation

The problem of simulating the dynamics of quantum systems was the original motivation for quantum computers proposed by Feynman [Fey82]. Specifically, given a Hamiltonian H, a quantum state $|\psi\rangle$, a time $t > 0$, and a desired error $\varepsilon > 0$, we ask to prepare a quantum state $|\psi_t\rangle$ such that

$$\||\psi_t\rangle - e^{iHt}|\psi\rangle\| \leq \varepsilon.$$

This problem, known as Hamiltonian simulation, sees wide application, including in quantum physics and quantum chemistry. A rich literature has developed on quantum algorithms for Hamiltonian simulation [Llo96; AT03; BCK15], with an optimal quantum algorithm for simulating sparse Hamiltonians given in [LC17]. In this subsection, we apply our framework to develop classical algorithms for Hamiltonian simulation. Specifically, we ask:

Problem 8.19. Consider a Hermitian matrix $H \in \mathbb{C}^{n\times n}$, a unit vector $b \in \mathbb{C}^n$, and error parameters $\varepsilon, \delta > 0$. Given $\mathrm{SQ}(H)$ and $\mathrm{SQ}(b)$, output $\mathrm{SQ}_\phi(\hat{b})$ with probability $\geq 1 - \delta$ for some $\hat{b} \in \mathbb{C}^n$ satisfying $\|\hat{b} - e^{iH}b\| \leq \varepsilon$.

We can solve this problem as a corollary of Theorem 6.1 upon choosing a polynomial approximation to $e^{\mathrm{i}x} = \cos(x) + \mathrm{i}\sin(x)$. We use the one the quantum algorithm uses.

Lemma 8.20 (Polynomial approximation to trigonometric functions, [GSLW19, Lemma 57]). Given $t \in \mathbb{R}$ and $\varepsilon \in (0, 1/e)$, let $r = \Theta(t + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)})$. Then, the following polynomials $c(x)$ (even with degree $2r$) and $s(x)$ (odd with degree $2r+1$),

$$c(x) = J_0(t) - 2\sum_{i\in[1,r]}(-1)^i J_{2i}(t)T_{2i}(x)$$

$$s(x) = 2\sum_{i\in[0,r]}(-1)^i J_{2i+1}(t)T_{2i+1}(x),$$

satisfy that $\|\cos(tx) - c(x)\|_{\sup} \leq \varepsilon$ and $\|\sin(tx) - s(x)\|_{\sup} \leq \varepsilon$. Here, $J_i(x)$ is the i-th Bessel function of the first kind [DLMF, (10.2.2)].

Corollary 8.21 (Dequantizing Hamiltonian simulation). *Given a Hermitian Hamiltonian $H \in \mathbb{C}^{n\times n}$ such that $0.01 \leq \|H\| \leq 1$; a vector $b \in \mathbb{C}^n$; a time $t > 0$; and $0 < \varepsilon < 1$, with $\mathcal{O}(\mathrm{nnz}(A) + \mathrm{nnz}(b))$ pre-processing. Then there exists an algorithm that outputs a vector y such that $\|y - e^{\mathrm{i}Ht}b\| \leq \varepsilon\|b\|$ with probability ≥ 0.9. Further, the running time to compute such a description of y is*

$$\widetilde{\mathcal{O}}\Big(\frac{t^{11}\|H\|_{\mathrm{F}}^4}{\varepsilon^2}\Big),$$

and from this description we can output a sample from y in $\widetilde{\mathcal{O}}(\frac{t^8\|H\|_{\mathrm{F}}^4}{\varepsilon^2\|y\|^2})$ time.

Proof. Let $c(x)$ and $s(x)$ be the polynomials from Lemma 8.20. We apply Theorem 6.1 to get descriptions of c and s such that $\|y_c - c(H)b\| \leq \varepsilon\|b\|$ and $\|y_s - s(H)b\| \leq \varepsilon\|b\|$. Then

$$e^{\mathrm{i}Ht}b = \cos(Ht)b + \mathrm{i}\sin(Ht)b \approx_{\varepsilon\|b\|} c(H)b + \mathrm{i}s(H)b \approx_{\varepsilon\|b\|} y_c + \mathrm{i}y_s.$$

This gives us a description $\mathcal{O}(\varepsilon)$-close to $e^{\mathrm{i}Ht}$. Using Corollary 4.10, we can get a sample from this output in the time described, by combining the two descriptions of y_c and y_s. $\square$

We give two more algorithms that are fundamentally the same, but operate in different regimes: the first works for low-rank H, and the second for arbitrary H.

Corollary 8.22. *Suppose H has minimum singular value σ and $\varepsilon < \min(0.5, \sigma)$. We can solve Problem 8.19 in $\widetilde{\mathcal{O}}(\frac{\|H\|_{\mathrm{F}}^6\|H\|^{16}}{\sigma^{16}\varepsilon^6}\log^3\frac{1}{\delta})$ time, giving $\mathrm{SQ}_\phi(\hat{b})$ with $\overline{\mathbf{sq}}_\phi(\hat{b}) = \widetilde{\mathcal{O}}(\frac{\|H\|_{\mathrm{F}}^4\|H\|^8}{\sigma^8\varepsilon^4}\log^3\frac{1}{\delta})$.*

This runtime is dimensionless in a certain sense. The natural error bound to require is that $\|\hat{b} - e^{iH}b\| \leq \varepsilon\|H\|$, since $|\frac{d}{dx}(e^{-\mathrm{i}\|H\|x})| = \|H\|$. So, if we rescale ε to $\varepsilon\|H\|$, the runtime is $\widetilde{\mathcal{O}}(\frac{\|H\|_{\mathrm{F}}^6\|H\|^{10}}{\sigma^{16}\varepsilon^6}\log^3\frac{1}{\delta})$, which is dimensionless. The runtime of the algorithm in the following corollary does not have this property, so its scaling with $\|H\|$ is worse, despite being faster for, say, $\|H\| = 1$.

Corollary 8.23. *For $\varepsilon < \min(0.5, \|H\|^3)$, we can solve Problem 8.19 in $\widetilde{\mathcal{O}}(\|H\|^{16}\|H\|_{\mathrm{F}}^6\varepsilon^{-6}\log^3\frac{1}{\delta})$ time, giving $\mathrm{SQ}_\phi(\hat{b})$ with $\overline{\mathbf{sq}}_\phi(\hat{b}) = \widetilde{\mathcal{O}}(\|H\|^8\|H\|_{\mathrm{F}}^4\varepsilon^{-4}\log^3\frac{1}{\delta})$.*

Our strategy proceeds as follows: consider a generic function $f(x)$ and Hermitian H. We can write $f(x)$ as a sum of an even function $a(x) \coloneqq \frac{1}{2}(f(x) + f(-x))$ and an odd function $b(x) \coloneqq \frac{1}{2}(f(x) - f(-x))$. For the even function, we can use Theorem 7.1 to approximate it via the function $f_a(x) \coloneqq a(\sqrt{x})$; the odd function can be written as H times an even function, which we approximate using Theorem 7.1 for $f_b(x) \coloneqq b(\sqrt{x})/\sqrt{x}$. In other words, $f(H) = f_a(H^\dagger H) + f_b(H^\dagger H)H$. Since $|a'(x)|, |b'(x)| \le |f'(x)|$, the Lipschitz constants don't blow up by splitting f into even and odd parts.

Now, we specialize to Hamiltonian simulation. We first rewrite the problem, using the function $\operatorname{sinc}(x) \coloneqq \sin(x)/x$.

$$e^{iH}b = \cos(H)b + i \cdot \operatorname{sinc}(H)Hb = f_{\cos}(H^\dagger H)b + f_{\operatorname{sinc}}(H^\dagger H)Hb,$$

where $f_{\cos}(\lambda) \coloneqq \cos(\sqrt{\lambda})$ and $f_{\operatorname{sinc}}(\lambda) \coloneqq i \cdot \operatorname{sinc}(\sqrt{\lambda})$. When applying Theorem 7.1 on $f_{\cos}$ and f_{sinc}, we will use the following bounds on the smoothness of $f_{\cos}$ and f_{sinc}.

$$\begin{aligned}
|f'_{\cos}(x)| &= \left|\frac{\sin(\sqrt{x})}{2\sqrt{x}}\right| \le \min\left(\frac{1}{2}, \frac{1}{2\sqrt{x}}\right) \\
|\bar{f}'_{\cos}(x)| &= \left|\frac{2 - 2\cos(\sqrt{x}) - \sqrt{x}\sin(\sqrt{x})}{2x^2}\right| \le \min\left(\frac{1}{24}, \frac{5}{2x^{3/2}}\right) \\
|f'_{\operatorname{sinc}}(x)| &= \left|\frac{\sqrt{x}\cos(\sqrt{x}) - \sin(\sqrt{x})}{2x^{3/2}}\right| \le \min\left(\frac{1}{4}, \frac{1}{x}\right) \\
|\bar{f}'_{\operatorname{sinc}}(x)| &= \left|\frac{2\sqrt{x} + \sqrt{x}\cos(\sqrt{x}) - 3\sin(\sqrt{x})}{2x^{5/2}}\right| \le \min\left(\frac{1}{60}, \frac{3}{x^2}\right)
\end{aligned}$$

We separate these bounds into the case where $x \ge 1$, which we use when we assume H has a minimum singular value, and the case where $x < 1$, which we use for arbitrary H.

Proof of Corollary 8.23. Using the Lipschitz bounds above with Theorem 7.1, we can find $R_{\cos} \in \mathbb{C}^{r_{\cos} \times n}, C_{\cos} \in \mathbb{C}^{r_{\cos} \times c_{\cos}}, R_{\operatorname{sinc}} \in \mathbb{C}^{r_{\operatorname{sinc}} \times n}, C_{\operatorname{sinc}} \in \mathbb{C}^{r_{\operatorname{sinc}} \times c_{\operatorname{sinc}}}$ such that

$$\|R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)R_{\cos} + I - f_{\cos}(H^\dagger H)\| \le \varepsilon \tag{62}$$

$$\|R_{\operatorname{sinc}}^\dagger \bar{f}_{\operatorname{sinc}}(C_{\operatorname{sinc}}C_{\operatorname{sinc}}^\dagger)R_{\operatorname{sinc}} + i \cdot I - f_{\operatorname{sinc}}(H^\dagger H)\| \le \frac{\varepsilon}{\|H\|} \tag{63}$$

where, using that our Lipschitz constants are all bounded by constants,

$$r_{\cos} = \widetilde{\mathcal{O}}\Big(\|H\|_F^2\|H\|^2\varepsilon^{-2}\log\frac{1}{\delta}\Big) \qquad c_{\cos} = \widetilde{\mathcal{O}}\Big(\|H\|_F^2\|H\|^6\varepsilon^{-2}\log\frac{1}{\delta}\Big)$$
$$r_{\mathrm{sinc}} = \widetilde{\mathcal{O}}\Big(\|H\|_F^2\|H\|^4\varepsilon^{-2}\log\frac{1}{\delta}\Big) \qquad c_{\mathrm{sinc}} = \widetilde{\mathcal{O}}\Big(\|H\|_F^2\|H\|^8\varepsilon^{-2}\log\frac{1}{\delta}\Big).$$

As a consequence,

$$\Big\|e^{iH}b - \Big(R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)R_{\cos}b + b + R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)R_{\mathrm{sinc}}Hb + iHb\Big)\Big\| \lesssim \varepsilon.$$

Note that, by Lemma 7.2, $\|R_{\cos}\| \lesssim \|H\|$, $\|\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)\| \lesssim 1$, and $\|R_{\cos}^\dagger\sqrt{\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)}\| \lesssim 1$; the same bounds hold for the sinc analogues. We now approximate using Lemma 5.7 four times.

1. We approximate $R_{\cos}b \approx u$ to $\varepsilon\|b\|$ error, requiring $\mathcal{O}(\|H\|_F^2\varepsilon^{-2}\log\frac{1}{\delta})$ samples.
2. We approximate $R_{\mathrm{sinc}}H \approx WC$ to ε error, requiring $\mathcal{O}(\|H\|_F^4\varepsilon^{-2}\log\frac{1}{\delta})$ samples.
3. We approximate $Cb \approx v$ to $\varepsilon\|H\|_F^{-1}\|b\|$ error, requiring $\mathcal{O}(\|H\|_F^4\varepsilon^{-2}\log\frac{1}{\delta})$ samples.
4. We approximate $Hb \approx R^\dagger w$ to $\varepsilon\|b\|$ accuracy, requiring $r := \mathcal{O}(\|H\|_F^2\varepsilon^{-2}\log\frac{1}{\delta})$ samples.

Our output will be

$$\hat{b} := R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)u + b + R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)Wv + iR^\dagger w,$$

which is close to $e^{iH}b$ because

$$\begin{aligned}
&\Big\|\hat{b} - \Big(R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)R_{\cos}b + b + R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)R_{\mathrm{sinc}}Hb + iHb\Big)\Big\| \\
&\leq \|R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)(u - R_{\cos}b)\| + \|R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)(R_{\mathrm{sinc}}H - WC)b\| \\
&\qquad + \|R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)W(Cb - v)\| + \|iR^\dagger w - iHb\| \\
&\lesssim \|u - R_{\cos}b\| + \|R_{\mathrm{sinc}}H - WC\|\|b\| + \|H\|_F\|Cb - v\| + \|R^\dagger w - Hb\| \leq 4\varepsilon\|b\|.
\end{aligned}$$

Now, we have expressed $\hat{b}$ as a linear combination of a small number of vectors, all of which we have sampling and query access to. We can complete using Lemmas 4.5 and 4.6, where the matrix is the concatenation $(R_{\cos}^\dagger \mid b \mid R_{\mathrm{sinc}}^\dagger \mid i \cdot R^\dagger)$, and the vector is the concatenation $(\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)u \mid 1 \mid \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)Wv \mid w)$. The length of this vector is $r_{\cos} + 1 + r_{\mathrm{sinc}} + r \lesssim$

r_{sinc}. We get $\mathrm{SQ}_\phi(\hat{b})$ where

$$\begin{aligned}
\phi &\lesssim r_{\mathrm{sinc}}\Big(\frac{\|H\|_{\mathrm{F}}^2}{r_{\cos}}\|\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)u\|^2 + \|b\|^2 + \frac{\|H\|_{\mathrm{F}}^2}{r_{\mathrm{sinc}}}\|\bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)Wv\|^2 + \frac{\|H\|_{\mathrm{F}}^2}{r}\|w\|^2\Big)\|\hat{b}\|^{-2} \\
&\lesssim \Big(\frac{r_{\mathrm{sinc}}}{r_{\cos}}\|H\|_{\mathrm{F}}^2(1+\varepsilon)^2\|b\|^2 + r_{\mathrm{sinc}}\|b\|^2 + \|H\|_{\mathrm{F}}^2(1+\varepsilon)^2\|b\|^2 + \frac{r_{\mathrm{sinc}}}{r}\|H\|_{\mathrm{F}}^2\|b\|^2\Big)\|b\|^{-2} \\
&= \widetilde{\mathcal{O}}(\|H\|_{\mathrm{F}}^2\|H\|^2 + r_{\mathrm{sinc}} + \|H\|_{\mathrm{F}}^2 + \|H\|_{\mathrm{F}}^2\|H\|^4) = \widetilde{\mathcal{O}}(r_{\mathrm{sinc}}).
\end{aligned}$$

In the second inequality, we use the same bounds for proving $\|\hat{b} - e^{iH}b\| \leq \varepsilon$, repurposed to argue that all approximations are sufficiently close to the values they are estimating, up to relative error. So, $\overline{\boldsymbol{sq}}_\phi(\hat{b}) = \widetilde{\mathcal{O}}(r_{\mathrm{sinc}}^2 \log\frac{1}{\delta})$. □

Proof of Corollary 8.22. Our approach is the same, though with different parameters. For Theorem 7.1, we use that in the interval $[\sigma^2/2, \infty)$, $f_{\cos}$ has Lipschitz constants of $L = O(1/\sigma)$ and $\bar{L} = O(1/\sigma^3)$ and f_{sinc} has $L = O(1/\sigma^2)$ and $\bar{L} = O(1/\sigma^4)$. So, if we take

$$\begin{aligned}
r_{\cos} &= \widetilde{\mathcal{O}}\Big(\|H\|^2\frac{\|H\|_{\mathrm{F}}^2}{\sigma^2}\varepsilon^{-2}\log\frac{1}{\delta}\Big) & c_{\cos} &= \widetilde{\mathcal{O}}\Big(\|H\|^2\frac{\|H\|_{\mathrm{F}}^2\|H\|^4}{\sigma^6}\varepsilon^{-2}\log\frac{1}{\delta}\Big) \\
r_{\mathrm{sinc}} &= \widetilde{\mathcal{O}}\Big(\|H\|^2\frac{\|H\|_{\mathrm{F}}^2\|H\|^2}{\sigma^4}\varepsilon^{-2}\log\frac{1}{\delta}\Big) & c_{\mathrm{sinc}} &= \widetilde{\mathcal{O}}\Big(\|H\|^2\frac{\|H\|_{\mathrm{F}}^2\|H\|^6}{\sigma^8}\varepsilon^{-2}\log\frac{1}{\delta}\Big),
\end{aligned}$$

all the conditions of Theorem 7.1 are satisfied: in particular, $\sigma^2/2 > \bar{\varepsilon}$ in both cases, up to rescaling ε by a constant factor:

$$\begin{aligned}
\bar{\varepsilon}_{\cos} &\lesssim \|H\|\|H\|_{\mathrm{F}}\frac{\varepsilon\sigma}{\|H\|\|H\|_{\mathrm{F}}} = \varepsilon\sigma \leq \sigma^2 \\
\bar{\varepsilon}_{\mathrm{sinc}} &\lesssim \|H\|\|H\|_{\mathrm{F}}\frac{\varepsilon\sigma^2}{\|H\|^2\|H\|_{\mathrm{F}}} = \varepsilon\sigma^2\|H\|^{-1} \leq \sigma^2
\end{aligned}$$

Here, we used our initial assumption that $\varepsilon \leq \sigma$. So, the bounds Eqs. (62) and (63) hold. Note that, by Lemma 7.2, $\|R_{\cos}\| \lesssim \|H\|$, $\|\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)\| \lesssim \sigma^{-2}$, and $\|R_{\cos}^\dagger\sqrt{\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)}\| \leq 1$; the same bounds hold for the sinc analogues. We now approximate using Lemma 5.7 four times.

1. We approximate $R_{\cos}b \approx u$ to $\varepsilon\sigma\|b\|$ error, requiring $\mathcal{O}(\|H\|_{\mathrm{F}}^2\sigma^{-2}\varepsilon^{-2}\log\frac{1}{\delta})$ samples.
2. We approximate $R_{\mathrm{sinc}}H \approx WC$ to $\varepsilon\sigma$ error, requiring $\mathcal{O}(\|H\|_{\mathrm{F}}^4\sigma^{-2}\varepsilon^{-2}\log\frac{1}{\delta})$ samples.
3. We approximate $Cb \approx v$ to $\varepsilon\sigma\|H\|_{\mathrm{F}}^{-1}\|b\|$ error, requiring $\mathcal{O}(\|H\|_{\mathrm{F}}^4\sigma^{-2}\varepsilon^{-2}\log\frac{1}{\delta})$ samples.

4. We approximate $Hb \approx R^\dagger w$ to $\varepsilon\|b\|$ accuracy, requiring $r := \mathcal{O}(\|H\|_F^2\varepsilon^{-2}\log\frac{1}{\delta})$ samples. Our output will be

$$\hat{b} := R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)u + b + R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)Wv + iR^\dagger w,$$

which is close to $e^{iH}b$ by the argument

$$\begin{aligned}
&\Big\|\hat{b} - \Big(R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)R_{\cos}b + b + R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)R_{\mathrm{sinc}}Hb + iHb\Big)\Big\| \\
&\le \|R_{\cos}^\dagger \bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)(u - R_{\cos}b)\| + \|R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)(R_{\mathrm{sinc}}H - WC)b\| \\
&\qquad + \|R_{\mathrm{sinc}}^\dagger \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)W(Cb - v)\| + \|iR^\dagger w - iHb\| \\
&\lesssim \sigma^{-1}\|u - R_{\cos}b\| + \sigma^{-1}\|R_{\mathrm{sinc}}H - WC\|\|b\| + \sigma^{-1}\|H\|_F\|Cb - v\| + \|R^\dagger w - Hb\| \le 4\varepsilon\|b\|
\end{aligned}$$

Now, we have expressed $\hat{b}$ as a linear combination of a small number of vectors, all of which we have sampling and query access to. We can complete using Lemmas 4.5 and 4.6, where the matrix is the concatenation $(R_{\cos}^\dagger \mid b \mid R_{\mathrm{sinc}}^\dagger \mid i\cdot R^\dagger)$, and the vector is the concatenation $(\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)u \mid 1 \mid \bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)Wv \mid w)$. The length of this vector is $r_{\cos} + 1 + r_{\mathrm{sinc}} + r \lesssim r_{\mathrm{sinc}}$. We get $\mathrm{SQ}_\phi(\hat{b})$ where

$$\begin{aligned}
\phi &\lesssim r_{\mathrm{sinc}}\Big(\frac{\|H\|_F^2}{r_{\cos}}\|\bar{f}_{\cos}(C_{\cos}C_{\cos}^\dagger)u\|^2 + \|b\|^2 + \frac{\|H\|_F^2}{r_{\mathrm{sinc}}}\|\bar{f}_{\mathrm{sinc}}(C_{\mathrm{sinc}}C_{\mathrm{sinc}}^\dagger)Wv\|^2 + \frac{\|H\|_F^2}{r}\|w\|^2\Big)\|\hat{b}\|^{-2} \\
&\lesssim \Big(\frac{r_{\mathrm{sinc}}}{r_{\cos}}\|H\|_F^2\sigma^{-2}\|b\|^2 + r_{\mathrm{sinc}}\|b\|^2 + \|H\|_F^2\sigma^{-2}\|b\|^2 + \frac{r_{\mathrm{sinc}}}{r}\|H\|_F^2\|b\|^2\Big)\|b\|^{-2} \\
&= \widetilde{\mathcal{O}}(\|H\|_F^2\|H\|^2\sigma^{-4} + r_{\mathrm{sinc}} + \|H\|_F^2\sigma^{-2} + \|H\|^4\sigma^{-4}\|H\|_F^2) = \widetilde{\mathcal{O}}(r_{\mathrm{sinc}} + \frac{t^2\|H\|_F^2}{\sigma^4}).
\end{aligned}$$

So, $\overline{\boldsymbol{sq}}_\phi(\hat{b}) = \widetilde{\mathcal{O}}(r_{\mathrm{sinc}}(r_{\mathrm{sinc}} + \|H\|_F^2\|H\|^2\sigma^{-4})\log\frac{1}{\delta})$. Since $\varepsilon < \sigma$, the r_{sinc}^2 term dominates. □

Remark 8.24. In the case where H is not low-rank, we could still run a modified version of Corollary 8.22 to compute a modified "$\exp_{\sigma,\eta}(iH)$" where singular values below σ are smoothly thresholded away. Following the same logic as Definition 8.11, we could redefine $f_{\cos}$ such that $f_{\cos}(x) = 1$ for $x < \sigma^2(1-\eta)$, $f_{\cos}(x) = \cos(\sqrt{\lambda})$ for $x \ge \sigma^2$, and is a linear interpolation between the endpoints for the x in between (and f_{sinc} similarly). These functions have the same Lipschitz constants as their originals, up to factors of $\frac{1}{\eta}$, and give the desired behavior of "smoothing away" small singular values (though we do keep the 0th and 1st order terms of the exponential).

Remark 8.25. Our result generalizes those of Ref. [RWCRPS20], which achieves essentially the same result only in the much easier regime where H and b are sparse. They achieve a significant speedup due to these assumptions: note that when H is sparse, and a subsample of rows R is taken, $RR^{\dagger}$ can be computed in time independent of dimension; so, we only need to take a subsample of rows, and not of columns. More corners can be cut from our algorithm in this fashion. In summary, though our algorithm is significantly slower, their sparsity assumptions are essential for their fast runtime, and our framework can identify where these tradeoffs occur.

8.7 Semidefinite program solving

A recent line of inquiry in quantum computing [BS17; AGGW20; BKLLSW19; AG19] focuses on finding quantum speedups for *semidefinite programs* (SDPs), a central topic in the theory of convex optimization with applications in algorithms design, operations research, and approximation algorithms. Chia, Li, Lin, and Wang [CLLW20] first noticed that quantum-inspired algorithms could dequantize these quantum algorithms in certain regimes. We improve on their result, giving an algorithm which is as general as the quantum algorithms, if the input is given classically (e.g., in a data-structure in RAM). Our goal is to solve the ε-feasibility problem; solving an SDP reduces by binary search to solving $\log(1/\varepsilon)$ instances of this feasibility problem.

Problem 8.26 (SDP ε-feasibility). Given an $\varepsilon > 0$, m real numbers $b_1, \dots, b_m \in \mathbb{R}$, and Hermitian $n \times n$ matrices $\mathrm{SQ}(A^{(1)}), \dots, \mathrm{SQ}(A^{(m)})$ such that $-I \preceq A^{(i)} \preceq I$ for all $i \in [m]$, we define $\mathcal{S}_\varepsilon$ as the set of all X satisfying[31]

$$\begin{aligned} \operatorname{tr}[A^{(i)}X] &\leq b_i + \varepsilon \quad \forall\, i \in [m]; \\ X &\succeq 0; \\ \operatorname{tr}[X] &= 1. \end{aligned}$$

If $\mathcal{S}_\varepsilon = \emptyset$, output "infeasible". If $\mathcal{S}_0 \neq \emptyset$, output an $X \in \mathcal{S}_\varepsilon$. (If neither condition holds, either output is acceptable.)

Corollary 8.27. *Let $F \geq \max_{j\in[m]}(\|A^{(j)}\|_{\mathrm{F}})$, and suppose[32] $F = \Omega(1)$. Then we can solve Problem 8.26 with success probability $\geq 1-\delta$ in cost*

$$\widetilde{\mathcal{O}}\Big(\Big(\frac{F^{18}}{\varepsilon^{40}}\log^{20}(n)\,\boldsymbol{sq}(A) + \frac{F^{22}}{\varepsilon^{46}}\log^{23}(n) + m\frac{F^{8}}{\varepsilon^{18}}\log^{8}(n)\,\boldsymbol{q}(A) + m\frac{F^{14}}{\varepsilon^{28}}\log^{13}(n)\Big)\log^3\frac{1}{\delta}\Big),$$

providing sampling and query access to a solution.

Assuming $\boldsymbol{sq}(A) = \widetilde{\mathcal{O}}(1)$, this runtime is $\widetilde{\mathcal{O}}\Big(\frac{F^{22}}{\varepsilon^{46}}\log^{23}(n) + m\frac{F^{14}}{\varepsilon^{28}}\log^{13}(n)\Big)$. For the same feasibility problem, the previous quantum-inspired SDP solver [CLLW20] proved a complexity bound $\widetilde{\mathcal{O}}(mr^{57}\varepsilon^{-92}\log^{37}(n))$, assuming that the constraint matrices have rank at most r. Since the rank constraint implies that $\|A^{(\cdot)}\|_{\mathrm{F}} \leq \sqrt{r}$, under this assumption our algorithm has complexity $\widetilde{\mathcal{O}}(r^{11}\varepsilon^{-46}\log^{23}(n) + mr^{7}\varepsilon^{-28}\log^{13}(n))$. So, our new algorithm both solves a more general problem and also greatly improves the runtime. The paper with the current best runtime for SDP solving does not discuss this precise model, but if we use the runtime they achieve in quantum state input model, making reasonable substitutions of $\gamma \to \frac{1}{\varepsilon}$ and $B \to F^2$, the corresponding quantum runtime is $\widetilde{\mathcal{O}}(\frac{F^7}{\varepsilon^{7.5}} + \frac{\sqrt{m}F^2}{\varepsilon^4})$, up to polylog$(n)$ factors.

Like prior work on quantum algorithms for SDP-solving, we use the matrix multiplicative weights (MMW) framework [AK16; Kal07] to solve Problem 8.26. Corollary 8.27 immediately follows from running the algorithm this framework admits (Algorithm 3), where we solve an instance of the problem described in Lemma 8.28 with precision $\theta = \varepsilon/4$ in each of the

[31] For simplicity, we assume here that X is normalized to have trace 1. This can be relaxed; for an example, see [AGGW20].

[32] Because of the normalization assumption that $\|A^{(\cdot)}\| \leq 1$, F is effectively a dimensionless "stable rank"-type constant, normalized by $\max_i \|A^{(i)}\|$.

$\mathcal{O}(\log(n)/\varepsilon^2)$ iterations.

Algorithm 3 (MMW based feasibility testing algorithm for SDPs).

```
Set $X_1 := \frac{I_n}{n}$, and the number of iterations $T := \frac{16\log n}{\varepsilon^2}$.
for $t = 1, \dots, T$:
    1. find a $j_t \in [m]$ such that $\mathrm{tr}[A^{(j_t)}X_t] > b_{j_t} + \frac{\varepsilon}{2}$
    2.     or conclude correctly that $\mathrm{tr}[A^{(j_t)}X_t] \leq b_{j_t} + \varepsilon$ for all $j \in [m]$
    3. if a $j_t \in [m]$ is found then
    4.     $X_{t+1} := \exp[-\frac{\varepsilon}{4}\sum_{i=1}^{t} A^{(j_i)}]/\,\mathrm{tr}[\exp[-\frac{\varepsilon}{4}\sum_{i=1}^{t} A^{(j_i)}]]$
    5. else conclude that $X_t \in \mathcal{S}_\varepsilon$
    6.     return $X_t$
If no solution found, conclude that the SDP is infeasible and terminate the algorithm
```

MMW works as a zero-sum game with two players, where the first player wants to provide an $X \in \mathcal{S}_\varepsilon$, and the second player wants to find a violation for any proposed X, i.e., a $j \in [m]$ such that $\mathrm{tr}[A^{(j)}X] > b_j + \varepsilon$. At the t^{th} round of the game, if the second player points out a violation j_t for the current solution X_t, the first player proposes a new solution

$$X_{t+1} \propto \exp[-\varepsilon(A^{(j_1)} + \cdots + A^{(j_t)})].$$

Solutions of this form are also known as *Gibbs states*. It is known that MMW solves the SDP ε-feasibility problem in $\mathcal{O}(\frac{\log n}{\varepsilon^2})$ iterations; a proof can be found, e.g., in the work of Brandão, Kalev, Li, Lin, Svore, and Wu [BKLLSW19, Theorem 3] or in Lee, Raghavendra and Steurer [LRS15, Lemma 4.6].

Our task is to execute Line 1 and Line 2 of 3, for an implicitly defined matrix with the form given in Line 4.

Lemma 8.28 ("Efficient" trace estimation). Consider the setting described in Corollary 8.27. Given $\theta \in (0,1]$, $t \leq \frac{\log(n)}{\theta^2}$ and $j_i \in [m]$ for $i \in [t]$, defining $H := \exp[-\theta\sum_{i=1}^{t} A^{(j_i)}]$, we can estimate $\operatorname{tr}(A^{(i)}H)/\operatorname{tr}(H)$ with success probability $\geq 1-\delta$ for all $i \in [m]$ to precision θ in cost

$$\widetilde{\mathcal{O}}\Big(\Big[\frac{F^{18}}{\theta^{38}}\log^{19}(n)\,\boldsymbol{sq}(A)+\frac{F^{22}}{\theta^{44}}\log^{22}(n)+m\frac{F^{8}}{\theta^{16}}\log^{7}(n)\,\boldsymbol{q}(A)+m\frac{F^{14}}{\theta^{26}}\log^{12}(n)\Big]\log^{3}\frac{1}{\delta}+\frac{\log(n)}{\theta^{2}}\,\boldsymbol{n}(A)\Big),$$

where $\boldsymbol{sq}(A)=\max_{j\in[m]}\boldsymbol{sq}(A^{(j)})$, and $\boldsymbol{s}(A)$, $\boldsymbol{q}(A)$, $\boldsymbol{n}(A)$ are defined analogously.

To estimate $\operatorname{tr}[A^{(i)}H]$, we first notice that we have $\mathrm{SQ}_{\phi}(\theta\sum_{i=1}^{t}A^{(j_i)})$, since it is a linear combination of matrices that we have sampling and query access to (Lemma 4.9). Then, we can find approximations of the Gibbs state by applying eigenvalue transformation (Theorem 7.8) according to the exponential function to get $\exp[-\theta\sum_{i=1}^{t}A^{(j_i)}]$ as an RUR decomposition. Then the estimation of $\operatorname{tr}[A^{(i)}H]$ can be performed by usual techniques (namely, Remark 5.18).

In order to understand how precisely we need to approximate the matrix in Line 4 we prove the following lemmas. Our first lemma will show that, to estimate $\operatorname{tr}(A^{(i)}H)/\operatorname{tr}(H)$ to θ precision, it suffices to estimate both $\operatorname{tr}(A^{(i)}H)$ and $\operatorname{tr}(H)$ to $\frac{1}{3}\theta\operatorname{tr}(H)$ precision.

Lemma 8.29. Suppose that $\theta\in[0,1]$ and $a,\tilde{a},Z,\tilde{Z}$ are such that $|a|\leq Z$, $|a-\tilde{a}|\leq\frac{\theta}{3}Z$, and $|Z-\tilde{Z}|\leq\frac{\theta}{3}Z$, then

$$\left|\frac{\tilde{a}}{\tilde{Z}}-\frac{a}{Z}\right|\leq\theta.$$

Proof.

$$\left|\frac{\tilde{a}}{\tilde{Z}}-\frac{a}{Z}\right|=\left|\frac{\tilde{a}Z}{Z\tilde{Z}}-\frac{a\tilde{Z}}{Z\tilde{Z}}\right|\leq\left|\frac{\tilde{a}Z-aZ}{Z\tilde{Z}}\right|+\left|\frac{aZ-a\tilde{Z}}{Z\tilde{Z}}\right|\leq\frac{3}{2Z}|\tilde{a}-a|+\frac{3a}{2Z^{2}}|Z-\tilde{Z}|\leq\frac{1}{2}\theta+\frac{1}{2}\theta\leq\theta.\ \square$$

Next, we will prove that the approximations we will use to $\operatorname{tr}(A^{(i)}H)$ and $\operatorname{tr}(H)$ suffice. We introduce some useful properties of matrix norms. For a matrix $A\in\mathbb{C}^{m\times n}$ and $p\in[1,\infty]$, we denote by $\|A\|_p$ the *Schatten p-norm*, which is the ℓ^p-norm of the singular values $(\sum_i\sigma_i^p(A))^{1/p}$. In particular, $\|A\|_{\mathrm{F}}=\|A\|_2$ and $\|A\|_{\mathrm{Op}}=\|A\|_\infty$. We recall some useful inequalities [Bha97, Section IV.2]. *Hölder's inequality* states that for all $B\in\mathbb{C}^{n\times k}$ and $r,p,q\in(0,\infty]$ such that $\frac{1}{p}+\frac{1}{q}=\frac{1}{r}$, we have $\|AB\|_r\leq\|A\|_p\|B\|_q$. The *trace-norm inequality* states that if $n=m$, then $|\operatorname{tr}(A)|\leq\|A\|_1$.

Lemma 8.30 (Perturbations of the partition function). For all Hermitian matrices $H, \tilde{H} \in \mathbb{C}^{n\times n}$,

$$\left|\operatorname{tr}(e^{\tilde{H}}) - \operatorname{tr}(e^{H})\right| \leq \|e^{\tilde{H}} - e^{H}\|_1 \leq \left(e^{\|\tilde{H}-H\|} - 1\right)\operatorname{tr}(e^{H}).$$

Proof. We will use the following formula introduced by [KS48; Fey51] (see also [Bel97, Page 181]):

$$\frac{d}{dt}e^{M(t)} = \int_0^1 e^{yM(t)}\frac{dM(t)}{dt}e^{(1-y)M(t)}dy. \tag{64}$$

Let $A \in \mathbb{C}^{n\times n}$ with $\|A\| \leq 1$, we define the function $g_A(t) := \operatorname{tr}\left(Ae^{H+t(\tilde{H}-H)}\right)$, and observe that

$$\begin{aligned}
g_A'(t) =& \frac{d}{dt}\operatorname{tr}\left(Ae^{H+t(\tilde{H}-H)}\right) && \text{by definiton}\\
=& \operatorname{tr}\left(A\frac{d}{dt}e^{H+t(\tilde{H}-H)}\right) && \text{by linearity of trace}\\
=& \operatorname{tr}\left(A\int_0^1 e^{y[H+t(\tilde{H}-H)]}(\tilde{H}-H)e^{(1-y)[H+t(\tilde{H}-H)]}dy\right) && \text{by Eq. (64)}\\
=& \int_0^1 \operatorname{tr}\left(Ae^{y[H+t(\tilde{H}-H)]}(\tilde{H}-H)e^{(1-y)[H+t(\tilde{H}-H)]}\right)dy && \text{by linearity of trace}^{33}\\
\leq& \int_0^1 \|Ae^{y[H+t(\tilde{H}-H)]}(\tilde{H}-H)e^{(1-y)[H+t(\tilde{H}-H)]}\|_1 dy && \text{by trace-norm inequality}\\
\leq& \int_0^1 \|Ae^{y[H+t(\tilde{H}-H)]}\|_{\frac{1}{y}}\|(\tilde{H}-H)e^{(1-y)[H+t(\tilde{H}-H)]}\|_{\frac{1}{1-y}} dy && \text{by Hölder's inequality}\\
\leq& \int_0^1 \|A\|\|e^{y[H+t(\tilde{H}-H)]}\|_{\frac{1}{y}}\|\tilde{H}-H\|\|e^{(1-y)[H+t(\tilde{H}-H)]}\|_{\frac{1}{1-y}} dy && \text{by Hölder's inequality}\\
\leq& \|\tilde{H}-H\|\int_0^1 \|e^{y[H+t(\tilde{H}-H)]}\|_{\frac{1}{y}}\|e^{(1-y)[H+t(\tilde{H}-H)]}\|_{\frac{1}{1-y}} dy && \text{since } \|A\| \leq 1\\
=& \|\tilde{H}-H\|\|e^{H+t(\tilde{H}-H)}\|_1.
\end{aligned} \tag{65}$$

Now we consider $z(t) := g_I(t) = \operatorname{tr}\left(e^{H+t(\tilde{H}-H)}\right)$. From Eq. (65) we have $z'(t) \leq \|\tilde{H}-H\|z(t)$. Using Grönwall's differential inequality, we can conclude that $z(t) \leq z(0)e^{t\|\tilde{H}-H\|}$ for every $t \in [0,\infty)$.

Finally, we use the fact that there exists a matrix A of operator norm at most 1 such that $\left\|e^{\tilde{H}} - e^{H}\right\|_1 = \operatorname{tr}(A(e^{\tilde{H}} - e^{H}))$ (take, e.g., $\operatorname{sgn}(e^{\tilde{H}} - e^{H})$). We finish the proof by observing that

[33] Note that in case $A = I$, by the cyclicity of trace, this equation implies that $\frac{d}{dt}\operatorname{tr}(e^{H(t)}) = \operatorname{tr}(e^{H(t)}\frac{d}{dt}H(t))$.

for such an A, $\|e^{\tilde{H}} - e^H\|_1 = \operatorname{tr}(Ae^{\tilde{H}}) - \operatorname{tr}(Ae^H) = g_A(1) - g_A(0) = \int_0^1 g_A'(t)dt$ and

$$\int_0^1 g_A'(t)dt \overset{(65)}{\leq} \int_0^1 \|\tilde{H} - H\| z(t)dt \leq z(0)\int_0^1 \|\tilde{H} - H\| e^{t\|\tilde{H}-H\|}dt = \operatorname{tr}(e^H)\Big(e^{\|\tilde{H}-H\|} - 1\Big). \qquad \square$$

The bound in the above lemma is tight, as shown by the example $\tilde{H} := H + \varepsilon I$. Before proving the following lemma, we observe that for any Hermitian matrix $H \in \mathbb{C}^{n\times n}$ with $\|H\|_F^2 \leq \frac{n}{4}$, we have by Hölder's inequality that

$$\operatorname{tr}(e^H) = n + \operatorname{tr}(e^H - I) = n + \sum_i (e^{\lambda_i} - 1) \geq n + \sum_i \lambda_i = n + \operatorname{tr}(H) \geq n - \sqrt{n}\|H\|_F \geq n/2. \quad (66)$$

Lemma 8.31. Consider a Hermitian matrix $H \in \mathbb{C}^{n\times n}$ such that $\|H\|_F^2 \leq \frac{n}{4}$. Let H have an approximate eigendecomposition in the following sense: for $r \leq n$, suppose we have a diagonal matrix $D \in \mathbb{R}^{r\times r}$ and $\widetilde{U} \in \mathbb{C}^{r\times n}$ that satisfy $\|\widetilde{U}\widetilde{U}^\dagger - I\| \leq \delta$ and $\|H - \widetilde{U}^\dagger D\widetilde{U}\| \leq \varepsilon$ for $\varepsilon \leq \frac{1}{2}$ and $\delta \leq \min(\frac{\varepsilon}{4(\|H\|+\varepsilon)}, \frac{\varepsilon}{2})$. Then we have

$$|(\operatorname{tr}(e^D) + n - r) - \operatorname{tr}(e^H)| \leq 2(e-1)\varepsilon \operatorname{tr}(e^H), \quad (67)$$

and, moreover, for all $A \in \mathbb{C}^{n\times n}$ we have

$$|\operatorname{tr}(A\widetilde{U}^\dagger(e^D - I)\widetilde{U}) + \operatorname{tr}(A) - \operatorname{tr}(Ae^H)| \lesssim \varepsilon\|A\| \operatorname{tr}(e^H).$$

Proof. First, recall that, by Lemma 3.5, there is unitary U such that $\|\widetilde{U} - U\| \leq \delta$. Consequently, also using facts from Lemma 3.5, along with bounds on δ,

$$\|H - U^\dagger DU\| \leq \|H - \widetilde{U}^\dagger D\widetilde{U}\| + \delta\frac{2-\delta}{(1-\delta)^2}\|\widetilde{U}^\dagger D\widetilde{U}\| \leq \varepsilon + 4\delta(\|H\| + \varepsilon) \leq 2\varepsilon. \quad (68)$$

By Lemma 8.30 we have

$$\left\|e^{U^\dagger DU} - e^H\right\|_1 \leq (e^{2\varepsilon} - 1)\operatorname{tr}(e^H) \leq 2(e-1)\varepsilon \operatorname{tr}(e^H),$$

and since $e^{U^\dagger DU} = U^\dagger(e^D - I)U + I$, by the linearity of trace, the trace-norm inequality, and

Hölder's inequality,

$$\begin{aligned}
&|\mathrm{tr}(AU^\dagger(e^D - I)U) + \mathrm{tr}(A) - \mathrm{tr}(Ae^H)| \\
&\qquad = |\mathrm{tr}(A(e^{U^\dagger DU} - e^H))| \leq \|A\|\|e^{U^\dagger DU} - e^H\|_1 \leq 2(e-1)\|A\|\varepsilon\,\mathrm{tr}(e^H). \quad (69)
\end{aligned}$$

In particular, setting $A = I$, we get the first desired bound

$$|(\mathrm{tr}(e^D) + n - r) - \mathrm{tr}(e^H)| = |\mathrm{tr}(U^\dagger(e^D - I)U + I) - \mathrm{tr}(e^H)| \leq 2(e-1)\varepsilon\,\mathrm{tr}(e^H).$$

Note that the two identity matrices in the equation above refer to identities of two different sizes. Now, if we show that $\mathrm{tr}(AU^\dagger(e^D - I)U) - \mathrm{tr}(A\widetilde{U}^\dagger(e^D - I)\widetilde{U})$ is sufficiently small, then the second desired bound follows by Eq. (69) and triangle inequality.

$$\begin{aligned}
&|\,\mathrm{tr}(AU^\dagger(e^D - I)U) - \mathrm{tr}(A\widetilde{U}^\dagger(e^D - I)\widetilde{U})| \\
&= |\,\mathrm{tr}((UAU^\dagger - \widetilde{U}A\widetilde{U}^\dagger)(e^D - I))| \\
&\leq \|UAU^\dagger - \widetilde{U}A\widetilde{U}^\dagger\|\|e^D - I\|_1 && \text{by trace-norm and Hölder's inequality} \\
&\leq (2\delta + \delta^2)\|A\|\|e^D - I\|_1 && \text{by Lemma 3.5} \\
&\leq 2\varepsilon\|A\|\|e^D - I\|_1 && \text{by assumption that } \delta \leq \varepsilon/2 \\
&\leq 2\varepsilon\|A\|\big(\mathrm{tr}(e^D) + r\big) && \text{by triangle inequality} \\
&\lesssim \varepsilon\|A\|\,\mathrm{tr}(e^H). && \text{by Eqs. (66) and (67)}
\end{aligned}$$

□

Now we are ready to devise our upper bound on the trace estimation subroutine.

Proof of Lemma 8.28. By Lemma 8.29, it suffices to find estimates of $\mathrm{tr}(e^H)$ and $\mathrm{tr}(Ae^H)$ for all $A = A^{(i)}$, to $\frac{\theta}{3}\,\mathrm{tr}(e^H)$ additive precision. Recall from the statement that $H := -\theta\sum_{i=1}^t A^{(j_i)}$. By triangle inequality, $\|H\|_{\mathrm{F}} \leq \frac{F}{\theta}\log(n)$. Because H is a linear combination of matrices, by Lemma 4.9, after paying $\frac{\log(n)}{\theta^2}\,\boldsymbol{n}(A)$ cost, we can obtain $\mathrm{SQ}_\phi(H)$ for $\phi \leq \frac{F^2\log^2(n)}{\theta^2\|H\|_{\mathrm{F}}^2}$ with $\boldsymbol{q}(H) = \boldsymbol{q}_\phi(H) \leq \frac{\log(n)}{\theta^2}\,\boldsymbol{q}(A)$ and $\boldsymbol{s}_\phi(H) = \boldsymbol{s}(A)$.

If $\frac{F}{\theta}\log(n) > \sqrt{n}/18$, then we simply compute the sum H by querying all matrix elements of every $A^{(j_i)}$ in the sum, costing $\mathcal{O}(tn^2\,\boldsymbol{q}(A))$. Then we compute e^H and its trace $\operatorname{tr}(e^H)$ all in time $\mathcal{O}(n^3)$ [PC99]. Finally, we compute all the traces $\operatorname{tr}(e^H A^{(m)})$ in time $\mathcal{O}(mn^2)$. The overall complexity is $\mathcal{O}(n^2(t\,\boldsymbol{q}(A)+n+m)) = \widetilde{\mathcal{O}}(\frac{F^6}{\theta^6}\,\boldsymbol{q}(A)\log^6(n) + m\frac{F^4}{\theta^4}\log^4(n))$.

If $\frac{F}{\theta}\log(n) \leq \sqrt{n}/18$ we do the following. Note that if $\|H\| \leq 1$, then $\operatorname{tr}(e^H) \geq n/e$ and $\operatorname{tr}(A^{(i)}e^H) \leq \|A^{(i)}\|_{\mathrm{F}}\|e^H\|_{\mathrm{F}} \leq Fe\sqrt{n}$, so $\operatorname{tr}(A^{(i)}e^H)/\operatorname{tr}(e^H) \leq e^2F/\sqrt{n} \leq \theta$, and outputting 0 as estimates is acceptable. We use Theorem 7.8 (with $f(x) = x$, so that $L = 1$, and choosing $\varepsilon := \Theta(\theta)$) to find a diagonal matrix $D \in \mathbb{R}^{s\times s}$ with $s = \widetilde{\mathcal{O}}(\phi^2\|H\|_{\mathrm{F}}^6/\varepsilon^6\log(1/\delta)) = \widetilde{\mathcal{O}}(F^6\theta^{-6}\log^6(n)\varepsilon^{-6}\log(1/\delta)) = \widetilde{\mathcal{O}}(F^6\theta^{-12}\log^6(n)\log(1/\delta))$ together with an approximate isometry $\widetilde{U} = N(SH) \in \mathbb{C}^{s\times n}$ such that $\|H - \widetilde{U}^\dagger D\widetilde{U}\| \leq \mathcal{O}(\varepsilon)$. If every diagonal element is less than $3/4$, then we conclude that $\|H\| \leq 1$, and return 0. Otherwise we have $\|H\| \geq 1/2$ and thus by Theorem 7.8 we have $\|\widetilde{U}\widetilde{U}^\dagger - I\| \lesssim \varepsilon^3\|H\|^{-3} \lesssim \frac{\varepsilon}{\|H\|+\varepsilon} + \varepsilon$ with probability at least $1-\frac{\delta}{2}$. As per Theorem 7.8, the cost of this is $\log^3(1/\delta)$ times at most

$$\begin{aligned}\widetilde{\mathcal{O}}\Big(\frac{\|H\|_{\mathrm{F}}^{18}}{\varepsilon^{18}}\phi^7\,\boldsymbol{sq}_\phi(H) + \frac{\|H\|_{\mathrm{F}}^{22}}{\varepsilon^{22}}\phi^6\Big) &= \widetilde{\mathcal{O}}\Big(\frac{\|H\|_{\mathrm{F}}^4}{\varepsilon^{18}}\frac{F^{14}}{\theta^{14}}\log^{14}(n)\,\boldsymbol{sq}_\phi(H) + \frac{\|H\|_{\mathrm{F}}^{10}}{\varepsilon^{22}}\frac{F^{12}}{\theta^{12}}\log^{12}(n)\Big)\\ &= \widetilde{\mathcal{O}}\Big(\frac{\|H\|_{\mathrm{F}}^4}{\varepsilon^{18}}\frac{F^{14}}{\theta^{16}}\log^{15}(n)\,\boldsymbol{sq}(A) + \frac{\|H\|_{\mathrm{F}}^{10}}{\varepsilon^{22}}\frac{F^{12}}{\theta^{12}}\log^{12}(n)\Big)\\ &= \widetilde{\mathcal{O}}\Big(\frac{1}{\varepsilon^{18}}\frac{F^{18}}{\theta^{20}}\log^{19}(n)\,\boldsymbol{sq}(A) + \frac{1}{\varepsilon^{22}}\frac{F^{22}}{\theta^{22}}\log^{22}(n)\Big)\\ &= \widetilde{\mathcal{O}}\Big(\frac{F^{18}}{\theta^{38}}\log^{19}(n)\,\boldsymbol{sq}(A) + \frac{F^{22}}{\theta^{44}}\log^{22}(n)\Big).\end{aligned}$$

By Lemma 8.31 we have[34] that $\operatorname{tr}(e^D)+(n-s)$ is a multiplicative $\frac{\theta}{3}$-approximation of $\operatorname{tr}(e^H)$ as desired, and for all $A = A^{(i)}$, $\operatorname{tr}((e^D-I)\widetilde{U}A\widetilde{U}^\dagger)+\operatorname{tr}(A)$ is an additive $(\frac{\theta}{9}\operatorname{tr}(e^H))$-approximation of $\operatorname{tr}(Ae^H)$. We can ignore the $\operatorname{tr}(A)$ in our approximation: by Eq. (66) we have

$$\operatorname{tr}(A) \leq \|A\|_{\mathrm{F}}\|I\|_{\mathrm{F}} \leq F\sqrt{n} \leq \theta n/18 \leq \theta\operatorname{tr}(e^H)/9,$$

so $|\operatorname{tr}((e^D-I)\widetilde{U}A\widetilde{U}^\dagger) - \operatorname{tr}(Ae^H)| \leq \frac{2\theta}{9}\operatorname{tr}(e^H))$. So, it suffices to compute an additive $(\frac{\theta}{9}\operatorname{tr}(e^H))$-approximation of $\operatorname{tr}((e^D-I)\widetilde{U}A\widetilde{U}^\dagger) = \operatorname{tr}(A\widetilde{U}^\dagger(e^D-I)\widetilde{U})$ to obtain the $(\frac{\theta}{3}\operatorname{tr}(e^H))$-approximation of $\operatorname{tr}(Ae^H)$ we seek.

We use Remark 5.18 to estimate $\operatorname{tr}(A\widetilde{U}^\dagger(e^D-I)\widetilde{U})$ to additive precision $(\frac{\theta}{9}\operatorname{tr}(e^H))$. Note

[34] In case applying Theorem 7.8 would result in $s > n$, we instead directly diagonalize H ensuring $s \leq n$.

that by Lemma 8.31 and Eq. (66) we have

$$\|\widetilde{U}^\dagger(e^D - I)\widetilde{U}\|_F \leq \|\widetilde{U}\|^2\|e^D - I\|_F \leq 2\|e^D - I\|_F \leq 2\|e^D - I\|_1 \lesssim \operatorname{tr}(e^H),$$

and since $s = \widetilde{\mathcal{O}}(F^6\theta^{-12}\log^6(n)\log(1/\delta))$ and $\boldsymbol{q}(H) \leq \frac{\log(n)}{\theta^2}\boldsymbol{q}(A)$, we also have

$$\begin{aligned}\boldsymbol{q}(\widetilde{U}^\dagger(e^D - I)\widetilde{U}) &= \boldsymbol{q}((SH)^\dagger N^\dagger(e^D - I)N(SH)) \\ &= \mathcal{O}(s \cdot \boldsymbol{q}(H) + s^2) \\ &= \widetilde{\mathcal{O}}\Big(F^6\theta^{-14}\log^7(n)\log(1/\delta)\,\boldsymbol{q}(A) + F^{12}\theta^{-24}\log^{12}(n)\log^2(1/\delta)\Big).\end{aligned}$$

Therefore, Remark 5.18 tells us that given SQ(A), a $(\frac{\theta}{9}\operatorname{tr}(e^H))$-approximation of $\operatorname{tr}(A\widetilde{U}^\dagger(e^D - I)\widetilde{U})$ can be computed with success probability at least $1 - \frac{\delta}{2m}$ in time

$$\mathcal{O}\Big(\frac{\|A\|_F^2}{\theta^2}\big(s\boldsymbol{q}(A) + s \cdot \boldsymbol{q}(H) + s^2\big)\log\frac{m}{\delta}\Big).$$

Since we do this for all $i \in [m]$, the overall complexity of obtaining the desired estimates $\operatorname{tr}(A^{(i)}e^H)$ with success probability at least $1 - \frac{\delta}{2}$ is m times

$$\widetilde{\mathcal{O}}\Big(\frac{F^8}{\theta^{16}}\log^7(n)\log(1/\delta)\log(m/\delta)\,\boldsymbol{q}(A) + \frac{F^{14}}{\theta^{26}}\log^{12}(n)\log^2(m/\delta)\log(m/\delta)\Big). \qquad \square$$

8.8 Discriminant analysis

Discriminant analysis is used for dimensionality reduction and classification over large data sets. Cong and Duan introduced a quantum algorithm to perform both with Fisher's linear discriminant analysis [CD16], a generalization of principal component analysis to data separated into classes.

The problem is as follows: given classified data, we wish to project our data onto a subspace that best explains between-class variance, while minimizing within-class variance. Suppose there are M input data points $\{x_i \in \mathbb{R}^N : 1 \leq i \leq M\}$ each belonging to one of k classes. Let μ_c denote the centroid (mean) of class $c \in [k]$, and $\bar{x}$ denote the centroid of all data points.

Following the notation of [CD16], let

$$S_B = \sum_{c=1}^{k} (\mu_c - \bar{x})(\mu_c - \bar{x})^T \text{ and } S_W = \sum_{c=1}^{k} \sum_{x \in c} (\mu_c - x)(\mu_c - x)^T.$$

denote the between-class and within-class scatter matrices of the dataset respectively. The original goal is to solve the generalized eigenvalue problem $S_B v_i = \lambda_i S_W v_i$ and output the top eigenvalues and eigenvectors; for dimensionality reduction using linear discriminant analysis, we would project onto these top eigenvectors. If S_W would be full-rank, this problem would be equivalent to finding the eigenvalues of $S_W^{-1} S_B$. However, this does not happen in general, and therefore various relaxations are considered in the literature [BHK97; Wel09]. For example, Welling [Wel09] considers the eigenvalue problem of

$$S_B^{\frac{1}{2}} S_W^{-1} S_B^{\frac{1}{2}}. \tag{70}$$

Cong and Duan further relax the problem, as they ignore small eigenvalues of S_W and S_B, and only compute approximate eigenvalues of Eq. (70) (after truncating eigenvalues), leading to inexact eigenvectors. We construct a classical analogue of their quantum algorithm.[35] Cong and Duan also describe a quantum algorithm for discriminant analysis classification; this algorithm does a matrix inversion procedure very similar to those described in Section 8.4 and Section 8.5, so for brevity we will skip dequantizing this algorithm.

To formally analyze this algorithm, we could, as in Section 8.3, assume the existence of an eigenvalue gap, so the eigenvectors are well-conditioned. However, let us instead use a different convention: if we can find diagonal D and an approximate isometry U such that $S_B^{\frac{1}{2}} S_W^{-1} S_B^{\frac{1}{2}} U \approx UD$, then we say we have found approximate eigenvalues and eigenvectors of $S_W^{+} S_B$.

[35]Analyzing whether or not the particular relaxation used in this and other quantum machine learning papers provides a meaningful output is unfortunately beyond the scope of our paper.

Problem 8.32 (Linear discriminant analysis). Consider the functions

$$\operatorname{sqrt}(x) = \begin{cases} 0 & x < \sigma^2/2 \\ 2x/\sigma - \sigma & \sigma^2/2 \le x < \sigma^2 \\ \sqrt{x} & x \ge \sigma^2 \end{cases} \qquad \operatorname{inv}(x) = \begin{cases} 0 & x < \sigma^2/2 \\ 2x/\sigma^4 - 1/\sigma^2 & \sigma^2/2 \le x < \sigma^2 \\ 1/x & x \ge \sigma^2 \end{cases}.$$

Given $\mathrm{SQ}(B, W) \in \mathbb{C}^{m\times n}$, with $S_W := W^\dagger W$ and $S_B := B^\dagger B$, find an α-approximate isometry $U \in \mathbb{C}^{n\times p}$ and diagonal $D \in \mathbb{C}^{p\times p}$ such that we have $\mathrm{SQ}_\phi(U(\cdot, i))$ for all i, $|D_{ii} - \lambda_i| \le \varepsilon\|B\|^2/\sigma^2$ for λ_i the eigenvalues of $\operatorname{sqrt}(S_B)\operatorname{inv}(S_W)\operatorname{sqrt}(S_B)$, and

$$\|\operatorname{sqrt}(S_B)\operatorname{inv}(S_W)\operatorname{sqrt}(S_B)U - UD\| \le \varepsilon\|\operatorname{sqrt}(S_B)\|^2\|\operatorname{inv}(S_W)\| \le \varepsilon\|B\|^2/\sigma^2.$$

The choice of error bound is natural, since $\|B\|^2/\sigma^2$ is essentially $\|\operatorname{sqrt}(S_B)\|^2\|\operatorname{inv}(S_W)\|$: we aim for additive error. The quantum algorithm achieves a runtime of $\widetilde{\mathcal{O}}(\frac{\|B\|_F^7}{\varepsilon^3\sigma^7} + \frac{\|W\|_F^7}{\varepsilon^3\sigma^7})$, up to polylog$(m, n)$ factors [CD16, Theorem 2].[36]

Corollary 8.33. *For $\varepsilon < \sigma/\|B\|$, we can solve Problem 8.32 in $\widetilde{\mathcal{O}}((\frac{\|B\|_F^6\|B\|^4}{\varepsilon^6\sigma^{10}} + \frac{\|W\|_F^6\|W\|^{10}}{\varepsilon^6\sigma^{16}})\log^3\frac{1}{\delta})$ time, with $\overline{\boldsymbol{sq}}_\phi(U(\cdot, i)) = \widetilde{\mathcal{O}}(\frac{\|B\|_F^4}{\varepsilon^2\sigma^4}\log^2\frac{1}{\delta})$.*

We prove this by using Theorem 7.1 to approximate $\operatorname{sqrt}(W^\dagger W)$ and $\operatorname{inv}(B^\dagger B)$ by RUR decompositions $R_W^\dagger U_W R_W$ and $R_B^\dagger U_B R_B$. Then, we use Lemma 5.7 to approximate $R_W R_B^\dagger$ by small submatrices $R'_W R'^\dagger_B$. This yields an approximate RUR decomposition of the matrix whose eigenvalues and vectors we want to find, $R_W^\dagger U R_W$ for $U = U_W R'_W R'^\dagger_B U_B R'_B R'^\dagger_W U_W$.

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), $(([C_W]_k)^+ R_W)^\dagger$ has singular values either close to zero or close to one (Lemma 5.22). 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

[36]This is the runtime of Step 2 of Algorithm 1. The normalization factor of $\max(\|B\|_F, \|S\|_F)$ is implicit there, κ_{eff} corresponds to $\frac{\max(\|B\|_F, \|S\|_F)}{\sigma^2}$, and the error bound the algorithm achieves is the one we describe here, since the authors must implicitly rescale the inverse and square root function by a cumulative factor of $\|B\|^2/\sigma^2$ to apply their Theorem 1.

sufficiently large and choosing C to be the same sketch used for U. Then, we can compute the eigendecomposition of the center $C_k^\dagger U C_k = VDV^\dagger$, which gives us an approximate eigendecomposition for $R_W^\dagger U R_W$: $(C_k^+ R_W)^\dagger V$ is an approximate isometry, so we choose its columns to be our eigenvectors, and our eigenvalues are the diagonal entries of D. We show that this has the approximation properties analogous to the quantum algorithm.

Proof. By Theorem 7.1, we can find R_B, C_B, R_W, C_W such that

$$\|\operatorname{sqrt}(B^\dagger B) - R_B^\dagger \overline{\operatorname{sqrt}}(C_B C_B^\dagger) R_B\| \leq \varepsilon \|B\|$$
$$\|\operatorname{inv}(W^\dagger W) - R_W^\dagger \overline{\operatorname{inv}}(C_W C_W^\dagger) R_W\| \leq \varepsilon/\sigma^2$$

with

$$r_B = \widetilde{\mathcal{O}}\Big(\frac{\|B\|_F^2}{\varepsilon^2\sigma^2}\log\frac{1}{\delta}\Big) \qquad c_B = \widetilde{\mathcal{O}}\Big(\frac{\|B\|^4\|B\|_F^2}{\varepsilon^2\sigma^6}\log\frac{1}{\delta}\Big)$$
$$r_W = \widetilde{\mathcal{O}}\Big(\frac{\|W\|^2\|W\|_F^2}{\varepsilon^2\sigma^4}\log\frac{1}{\delta}\Big) \qquad c_W = \widetilde{\mathcal{O}}\Big(\frac{\|W\|^6\|W\|_F^2}{\varepsilon^2\sigma^8}\log\frac{1}{\delta}\Big).$$

Let $Z_B := \overline{\operatorname{sqrt}}(C_B C_B^\dagger)$ and $Z_W := \overline{\operatorname{inv}}(C_W C_W^\dagger)$. These approximations suffice for us:

$$\begin{aligned}
&\|\operatorname{sqrt}(S_B)\operatorname{inv}(S_W)\operatorname{sqrt}(S_B) - R_B^\dagger Z_B R_B R_W^\dagger Z_W R_W R_B^\dagger Z_B R_B\| \\
&\qquad \leq \|\operatorname{sqrt}(S_B) - R_B^\dagger Z_B R_B\|\|\operatorname{inv}(S_W)\operatorname{sqrt}(S_B)\| \\
&\qquad + \|R_B^\dagger Z_B R_B\|\|\operatorname{inv}(S_W) - R_W^\dagger Z_W R_W\|\|\operatorname{sqrt}(S_B)\| \\
&\qquad\qquad + \|R_B^\dagger Z_B R_B R_W^\dagger Z_W R_W\|\|\operatorname{sqrt}(S_B) - R_B^\dagger Z_B R_B\|,
\end{aligned}$$

each of which is bounded by $\varepsilon\|B\|^2/\sigma^2$. Next, we approximate $\|R_B R_W^\dagger - R_B' R_W'^\dagger\|_F \leq \varepsilon\sigma^{3/2}\sqrt{\|B\|}$, since then

$$\begin{aligned}
&\|\bar{\Sigma}_B^{\frac{1}{2}}\bar{U}_B^\dagger R_B R_W^\dagger \bar{U}_W \bar{\Sigma}_W^{\frac{1}{2}} - \bar{\Sigma}_B^{\frac{1}{2}}\bar{U}_B^\dagger R'_B R_W'^\dagger \bar{U}_W \bar{\Sigma}_W^{\frac{1}{2}}\| \\
&\leq \|\bar{\Sigma}_B^{\frac{1}{2}}\bar{U}_B^\dagger\|\|R_B R_W^\dagger - R'_B R_W'^\dagger\|\|\bar{U}_W \bar{\Sigma}_W^{\frac{1}{2}}\| \\
&\leq \sigma^{-\frac{1}{2}}\|R_B R_W^\dagger - R'_B R_W'^\dagger\|\sigma^{-2} \\
&\leq \varepsilon\sqrt{\|B\|/\sigma^2},
\end{aligned}$$

and so

$$\|R_B^\dagger Z_B R_B R_W^\dagger Z_W R_W R_B^\dagger Z_B R_B - R_B^\dagger Z_B R'_B R_W'^\dagger Z_W R'_W R_B'^\dagger Z_B R_B\| \lesssim \varepsilon\|B\|^2/\sigma^2.$$

Now, we can compute $Z := Z_B R'_B R_W'^\dagger Z_W R'_W R_B'^\dagger Z_B$ and, using that $Z_B = Z_B[C_B]_{\frac{\sigma}{\sqrt{2}}}[C_B]^+_{\frac{\sigma}{\sqrt{2}}}$, rewrite

$$R_B^\dagger Z R_B = R_B^\dagger([C_B]^+_{\frac{\sigma}{\sqrt{2}}})^\dagger [C_B]^\dagger_{\frac{\sigma}{\sqrt{2}}} Z [C_B]_{\frac{\sigma}{\sqrt{2}}}[C_B]^+_{\frac{\sigma}{\sqrt{2}}} R_B.$$

By Lemma 5.22, $([C_B]^+_{\frac{\sigma}{\sqrt{2}}} R_B)^\dagger$ is an $\varepsilon\sigma/\|B\|$-approximate projective isometry[37] onto the image of $[C_B]^+_{\frac{\sigma}{\sqrt{2}}}$ (where we use that $\varepsilon < \sigma/\|B\|$). To turn this approximate projective isometry into an isometry, we compute the eigendecomposition $[C_B]^\dagger_{\frac{\sigma}{\sqrt{2}}} Z [C_B]_{\frac{\sigma}{\sqrt{2}}} = V\Sigma V^\dagger$, where we truncate so that V is full rank. Consequently, $U := R_B^\dagger([C_B]^+_{\frac{\sigma}{\sqrt{2}}})^\dagger V$ is full rank—the image of V is contained in the image of $[C_B]^+_{\frac{\sigma}{\sqrt{2}}}$—and thus is an $\varepsilon\sigma/\|B\|$-approximate isometry. So, our eigenvectors are U and our eigenvalues are $D := \Sigma$. This satisfies the desired bounds because

$$\begin{aligned}
&\|\operatorname{sqrt}(S_B)\operatorname{inv}(S_W)\operatorname{sqrt}(S_B)U - UD\| \\
&\leq \|\operatorname{sqrt}(S_B)\operatorname{inv}(S_W)\operatorname{sqrt}(S_B)U - UDU^\dagger U\| + \|UDU^\dagger U - UD\| \\
&\leq \varepsilon\frac{\|B\|^2}{\sigma^2}\|U\| + \|UD\|\|U^\dagger U - I\| \lesssim \varepsilon\frac{\|B\|^2}{\sigma^2}.
\end{aligned}$$

The eigenvalues are correct because, by the approximate isometry condition, $\|U - \tilde{U}\| \lesssim \varepsilon\frac{\sigma}{\|B\|}$

[37]We get more than we need here: an ε-approximate projective isometry would suffice for the subsequent arguments.

for $\tilde{U}$ an isometry, and so we can use Lemma 3.5 to conclude

$$\begin{aligned}
&\|\operatorname{sqrt}(S_B)\operatorname{inv}(S_W)\operatorname{sqrt}(S_B) - \tilde{U}D\tilde{U}^\dagger\| \\
&\leq \|\operatorname{sqrt}(S_B)\operatorname{inv}(S_W)\operatorname{sqrt}(S_B) - UDU^\dagger\| + \|UDU^\dagger - \tilde{U}D\tilde{U}^\dagger\| \\
&\lesssim \varepsilon\frac{\|B\|^2}{\sigma^2} + \varepsilon\frac{\sigma}{\|B\|}\|D\| \lesssim \varepsilon\frac{\|B\|^2}{\sigma^2}.
\end{aligned}$$

$\tilde{U}D\tilde{U}^\dagger$ is an eigendecomposition. Furthermore, this is an approximation of a Hermitian PSD matrices, where singular value error bounds align with eigenvalue error bounds. So, Weyl's inequality (Lemma 5.16) implies the desired bound $|D_{ii} - \lambda_i| \lesssim \varepsilon\frac{\|B\|^2}{\sigma^2}$ for λ_i the true eigenvalues.

We have $\mathrm{SQ}_\phi(U(\cdot,i))$ by Lemmas 4.5 and 4.6, since $U(\cdot,i) = R_B^\dagger([C_B]^+_{\frac{\sigma}{\sqrt{2}}})^\dagger V(\cdot,i)$. The runtime is $\overline{\boldsymbol{sq}}_\phi(U(\cdot,i)) = r_B\phi\log\frac{1}{\delta}$, where

$$\phi = r_B\frac{\sum_{j=1}^{r_B}\|R_B(j,\cdot)\|^2|[([C_B]^+_{\frac{\sigma}{\sqrt{2}}})^\dagger V(\cdot,i)](j)|^2}{\|U(\cdot,i)\|} \lesssim \|B\|_{\mathrm{F}}^2\|([C_B]^+_{\frac{\sigma}{\sqrt{2}}})^\dagger V(\cdot,i)\|^2 \lesssim \frac{\|B\|_{\mathrm{F}}^2}{\sigma^2}.$$

This gives the stated runtime. □

References

[DLMF] *NIST Digital Library of Mathematical Functions*. https://dlmf.nist.gov/, Release 1.1.10 of 2023-06-15. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds. URL: https://dlmf.nist.gov/ (page 136).

[AA18] Scott Aaronson and Andris Ambainis. "Forrelation: a problem that optimally separates quantum from classical computing". In: *SIAM Journal on Computing* 47.3 (Jan. 2018), pp. 982–1038. DOI: 10.1137/15m1050902. arXiv: 1411.5729 [quant-ph] (page 18).

[Aar15] Scott Aaronson. "Read the fine print". In: *Nature Physics* 11.4 (2015), pp. 291–293. DOI: 10.1038/nphys3272 (pages 1, 2).

[Aar22] Scott Aaronson. "How much structure is needed for huge quantum speedups?" Sept. 14, 2022. arXiv: 2209.06930 [quant-ph] (page 1).

[AC17] Scott Aaronson and Lijie Chen. "Complexity-Theoretic Foundations of Quantum Supremacy Experiments". In: *32nd Computational Complexity Conference (CCC 2017)*. Vol. 79. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 22:1–22:67. DOI: 10.4230/LIPIcs.CCC.2017.22 (page 18).

[ACQ22] Dorit Aharonov, Jordan Cotler, and Xiao-Liang Qi. "Quantum algorithmic measurement". In: *Nature Communications* 13.1 (Feb. 2022). DOI: 10.1038/s41467-021-27922-0. arXiv: 2101.04634 [quant-ph] (page 8).

[ADBL20] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. "Quantum-inspired algorithms in practice". In: *Quantum* 4 (Aug. 2020), p. 307. DOI: 10.22331/q-2020-08-13-307. arXiv: 1905.10415 [quant-ph] (page 20).

[AG19] Joran van Apeldoorn and András Gilyén. "Improvements in quantum SDP-solving with applications". In: Leibniz International Proceedings in Informatics (LIPIcs) 132 (2019). Ed. by Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, 99:1–99:15. ISSN: 1868-8969. DOI: 10.4230/LIPIcs.ICALP.2019.99. arXiv: 1804.05058 [quant-ph] (pages 15, 16, 141).

[AGGW20] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. "Quantum SDP-solvers: better upper and lower bounds". In: *Quantum* 4 (Feb. 2020), p. 230. DOI: 10.22331/q-2020-02-14-230. arXiv: 1705.01843 [quant-ph] (pages 141, 142).

[AK16] Sanjeev Arora and Satyen Kale. "A combinatorial, primal-dual approach to semidefinite programs". In: *Journal of the ACM* 63.2 (May 2016), pp. 1–35. DOI: 10.1145/2837020 (page 142).

[Akh+22] Ismail Yunus Akhalwaya, Shashanka Ubaru, Kenneth L. Clarkson, Mark S. Squillante, Vishnu Jejjala, Yang-Hui He, Kugendran Naidoo, Vasileios Kalantzis, and Lior Horesh. "Towards quantum advantage on noisy quantum computers". Sept. 19, 2022. DOI: 10.48550/ARXIV.2209.09371. arXiv: 2209.09371 [quant-ph] (page 18).

[AP10] A.B. Aleksandrov and V.V. Peller. "Operator Hölder–Zygmund functions". In: *Advances in Mathematics* 224.3 (June 2010), pp. 910–966. DOI: 10.1016/j.aim.2009.12.018. arXiv: 0907.3049 [math.FA] (page 104).

[AP11] A.B. Aleksandrov and V.V. Peller. "Estimates of operator moduli of continuity". In: *Journal of Functional Analysis* 261.10 (Nov. 2011), pp. 2741–2796. DOI: 10.1016/j.jfa.2011.07.009. arXiv: 1104.3553 [math.FA] (page 102).

[AT03] Dorit Aharonov and Amnon Ta-Shma. "Adiabatic quantum state generation and statistical zero knowledge". In: *Proceedings of the 35th ACM Symposium on the Theory of Computing*. ACM. New York, NY, USA: Association for Computing Machinery, 2003, pp. 20–29. DOI: 10.1145/780542.780546. arXiv: quant-ph/0301023 (page 135).

[BCK15] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. "Hamiltonian simulation with nearly optimal dependence on all parameters". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, Oct. 2015. DOI: 10.1109/focs.2015.54. arXiv: 1501.01715 [quant-ph] (page 135).

[BCWW01] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. "Quantum fingerprinting". In: *Physical Review Letters* 87.16 (Sept. 2001), p. 167902. DOI: 10.1103/physrevlett.87.167902. arXiv: quant-ph/0102001 [quant-ph] (page 5).

[Bel97] R. Bellman. *Introduction to matrix analysis*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997. DOI: 10.1137/1.9781611971170 (page 145).

[Ber+22] Dominic W. Berry, Yuan Su, Casper Gyurik, Robbie King, Joao Basso, Alexander Del Toro Barba, Abhishek Rajput, Nathan Wiebe, Vedran Dunjko, and Ryan Babbush. "Quantifying quantum advantage in topological data analysis". Sept. 27, 2022. DOI: 10.48550/ARXIV.2209.13581. arXiv: 2209.13581 [quant-ph] (page 18).

[Bha97] Rajendra Bhatia. *Matrix analysis*. Springer New York, 1997. DOI: 10.1007/978-1-4612-0653-8 (pages 49, 144).

[BHK97] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7 (1997), pp. 711–720. DOI: 10.1109/34.598228 (page 150).

[BJ99] Nader H. Bshouty and Jeffrey C. Jackson Jeffrey C.on. "Learning DNF over the uniform distribution using a quantum example oracle". In: *SIAM Journal on Computing* 28.3 (1999), pp. 1136–1153 (page 1).

[BKLLSW19] Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. "Quantum SDP solvers: large speed-ups, optimality, and applications to quantum learning". In: (2019). DOI: 10.4230/LIPICS.ICALP.2019.27. arXiv: 1710.02581 [quant-ph] (pages i, 15, 16, 141, 143).

[BS17] Fernando G.S.L. Brandao and Krysta M. Svore. "Quantum speed-ups for solving semidefinite programs". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, Oct. 2017. DOI: 10.1109/focs.2017.45. arXiv: 1609.05537 [quant-ph] (page 141).

[BT23] Ainesh Bakshi and Ewin Tang. "An improved classical singular value transformation for quantum machine learning". Mar. 2, 2023. arXiv: 2303.01492 [quant-ph] (page iii).

[CB20] Daan Camps and Roel Van Beeumen. "Approximate quantum circuit synthesis using block encodings". In: *Physical Review A* 102.5 (Nov. 2020), p. 052411. DOI: 10.1103/physreva.102.052411. arXiv: 2007.01417 [quant-ph] (page 13).

[CCHLW22] Nadiia Chepurko, Kenneth Clarkson, Lior Horesh, Honghao Lin, and David Woodruff. "Quantum-inspired algorithms from randomized numerical linear algebra". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 3879–3900. arXiv: 2011.04125 [cs.DS]. URL: https://proceedings.mlr.press/v162/chepurko22a.html (page 20).

[CD16] Iris Cong and Luming Duan. "Quantum discriminant analysis for dimensionality reduction and classification". In: *New Journal of Physics* 18.7 (July 2016), p. 073011. DOI: 10.1088/1367-2630/18/7/073011. arXiv: 1510.00113 [quant-ph] (pages 15, 16, 124, 149–151).

[CGJ19] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. "The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation". In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. LIPIcs. Schloss Dagstuhl, 2019. DOI: 10.4230/LIPIcs.ICALP.2019.33. arXiv: 1804.01973 [quant-ph] (pages 4, 15, 16, 21, 35, 117, 121, 122, 124).

[CGLLTW20] Nai-Hui Chia, András Gilyén, Han-Hsuan Lin, Seth Lloyd, Ewin Tang, and Chunhao Wang. "Quantum-inspired algorithms for solving low-rank linear equation systems with logarithmic dependence on the dimension". In: *31st International Symposium on Algorithms and Computation (ISAAC 2020)*. Vol. 181. Leibniz International Proceedings in Informatics (LIPIcs). Published version of [CLW18] and [GLT18]. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 47:1–47:17. ISBN: 978-3-95977-173-3. DOI: 10.4230/LIPIcs.ISAAC.2020.47 (pages 15, 125).

[CGLLTW22] Nai-Hui Chia, András Pal Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. "Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning". In: *Journal of the ACM* 69.5 (Oct. 2022), pp. 1–72. DOI: 10.1145/3549524. arXiv: 1910.06151 [cs.DS] (page iii).

[CHM21] Jordan Cotler, Hsin-Yuan Huang, and Jarrod R. McClean. "Revisiting dequantization and quantum advantage in learning tasks". 2021. DOI: 10.48550/ARXIV.2112.00811. arXiv: 2112.00811 [quant-ph] (page 17).

[Cil+18] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. "Quantum machine learning: a classical perspective". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (Jan. 2018), p. 20170551. DOI: 10.1098/rspa.2017.0551. arXiv: 1707.08561 (pages 1, 8).

[CJS13] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. "Preconditioned quantum linear system algorithm". In: *Physical Review Letters* 110.25 (June 2013), p. 250504. DOI: 10.1103/physrevlett.110.250504. arXiv: 1301.2340 [quant-ph].

[CKS17] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. "Quantum algorithm for systems of linear equations with exponentially improved dependence on precision". In: *SIAM Journal on Computing* 46.6 (2017), pp. 1920–1950. DOI: 10.1137/16M1087072 (page 125).

[CLLW20] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. "Quantum-Inspired Sublinear Algorithm for Solving Low-Rank Semidefinite Programming". In: *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*. LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. DOI: 10.4230/LIPIcs.MFCS.2020.23. arXiv: 1901.03254 [cs.DS] (pages iii, 15, 109, 141, 142).

[CLW18] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang. "Quantum-inspired sublinear classical algorithms for solving low-rank linear systems". 2018. arXiv: 1811.04852 [cs.DS] (pages iii, 124, 157).

[CW23] Yanlin Chen and Ronald de Wolf. “Quantum algorithms and lower bounds for linear regression with norm constraints”. In: *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Vol. 261. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 38:1–38:21. ISBN: 978-3-95977-278-5. DOI: 10.4230/LIPIcs.ICALP.2023.38. arXiv: 2110.13086 [quant-ph] (page 10).

[DBH22] Chen Ding, Tian-Yi Bao, and He-Liang Huang. “Quantum-inspired support vector machine”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (Dec. 2022), pp. 7210–7222. DOI: 10.1109/tnnls.2021.3084467. arXiv: 1906.08902 [cs.LG] (pages iii, 15, 128, 129).

[DH80] Philip J. Davis and Reuben Hersh. *The mathematical experience*. With an introduction by Gian-Carlo Rota. Birkhäuser, Boston, Massachusetts, 1980, pp. xv+440. ISBN: 3-7643-3018-X (page 17).

[DKM06] P. Drineas, R. Kannan, and M. Mahoney. “Fast Monte Carlo algorithms for matrices I: approximating matrix multiplication”. In: *SIAM Journal on Computing* 36.1 (Jan. 2006), pp. 132–157. DOI: 10.1137/s0097539704442684 (pages 8, 20, 40, 41, 43).

[DKR02] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. “Competitive recommendation systems”. In: *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC)*. New York, NY, USA: Association for Computing Machinery, 2002, pp. 82–90. DOI: 10.1145/509907.509922 (page 32).

[DKW18] Yogesh Dahiya, Dimitris Konomis, and David P Woodruff. “An empirical evaluation of sketching for numerical linear algebra”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1292–1300. DOI: 10.1145/3219819.3220098 (page 21).

[DM07] Petros Drineas and Michael W. Mahoney. “A randomized algorithm for a tensor-based generalization of the singular value decomposition”. In: *Linear Algebra and its Applications* 420.2-3 (2007), pp. 553–571. DOI: 10.1016/j.laa.2006.08.023 (page 19).

[DW20] Vedran Dunjko and Peter Wittek. “A non-review of Quantum Machine Learning: trends and explorations”. In: *Quantum Views* 4 (Mar. 2020), p. 32. DOI: 10.22331/qv-2020-03-17-32. URL: https://doi.org/10.22331/qv-2020-03-17-32 (page 1).

[Fey51] Richard P. Feynman. “An operator calculus having applications in quantum electrodynamics”. In: *Physical Review* 84 (1 1951), pp. 108–128. DOI: 10.1103/PhysRev.84.108 (page 145).

[Fey82] Richard P. Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. DOI: 10.1007/bf02650179 (page 135).

[FKV04] Alan Frieze, Ravi Kannan, and Santosh Vempala. “Fast Monte-Carlo algorithms for finding low-rank approximations”. In: *Journal of the ACM* 51.6 (Nov. 2004), pp. 1025–1041. DOI: 10.1145/1039488.1039494 (pages 19, 32, 39).

[GCD22] Casper Gyurik, Chris Cade, and Vedran Dunjko. “Towards quantum advantage via topological data analysis”. In: *Quantum* 6 (Nov. 2022), p. 855. DOI: 10.22331/q-2022-11-10-855. arXiv: 2005.02607 [quant-ph] (page 18).

[Gil10] Michael I. Gil. “Perturbations of functions of diagonalizable matrices”. In: *Electronic Journal of Linear Algebra* 20 (2010), pp. 303–313. DOI: 10.13001/1081-3810.1375 (page 102).

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. “Quantum random access memory”. In: *Physical Review Letters* 100.16 (2008), p. 160501. DOI: 10.1103/PhysRevLett.100.160501. arXiv: 0708.1879 (page 6).

[GLT18] András Gilyén, Seth Lloyd, and Ewin Tang. “Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension”. 2018. arXiv: 1811.04909 [cs.DS] (pages iii, 51, 124, 126, 157).

[GR02] Lov Grover and Terry Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions”. 2002. arXiv: 0208112 [quant-ph] (page 36).

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. “Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics”. In: *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*. ACM, June 2019, pp. 193–204. DOI: 10.1145/3313276.3316366. arXiv: 1806.01838 (pages i, 4, 9, 10, 13–16, 26, 32, 54–56, 99, 104, 115, 116, 125, 126, 135).

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. In: *Physical Review Letters* 103 (15 Oct. 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502 (pages 1, 3, 4, 18, 22, 37, 124).

[HKP21] Hsin-Yuan Huang, Richard Kueng, and John Preskill. “Information-theoretic bounds on quantum advantage in machine learning”. In: *Physical Review Letters* 126.19 (May 2021), p. 190505. DOI: 10.1103/physrevlett.126.190505. arXiv: 2101.02464 [quant-ph] (page 8).

[HKS11] Elad Hazan, Tomer Koren, and Nati Srebro. “Beating SGD: learning SVMs in sublinear time”. In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Red Hook, NY, USA: Curran Associates, Inc., 2011, pp. 1233–1241 (page 19).

[Hua+22] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, and Jarrod R. McClean. “Quantum advantage in learning from experiments”. In: *Science* 376.6598 (2022), pp. 1182–1186. DOI: 10.1126/science.abn7293 (page 17).

[JR23] Samuel Jaques and Arthur G. Rattew. “QRAM: A survey and critique”. May 17, 2023. arXiv: 2305.10310 [quant-ph] (page 6).

[JW06] Dominik Janzing and Pawel Wocjan. “Estimating diagonal entries of powers of sparse symmetric matrices is BQP-complete”. June 27, 2006. arXiv: quant-ph/0606229 [quant-ph] (page 4).

[Kal07] Satyen Kale. “Efficient algorithms using the multiplicative weights update method”. PhD thesis. Princeton University, 2007. URL: http://www.satyenkale.com/papers/thesis.pdf (page 142).

[KP17] Iordanis Kerenidis and Anupam Prakash. “Quantum recommendation systems”. In: *Proceedings of the* 8^{th} *Innovations in Theoretical Computer Science Conference (ITCS)*. 2017, 49:1–49:21. DOI: 10.4230/LIPIcs.ITCS.2017.49. arXiv: 1603.08675 (pages i, 1, 15, 16, 20, 36, 114, 115).

[KP20] Iordanis Kerenidis and Anupam Prakash. “Quantum gradient descent for linear systems and least squares”. In: *Physical Review A* 101.2 (2020), p. 022316. DOI: 10.1103/PhysRevA.101.022316. arXiv: 1704.04992 [quant-ph] (pages 21, 35).

[KPS21] Iordanis Kerenidis, Anupam Prakash, and Dániel Szilágyi. “Quantum algorithms for second-order cone programming and support vector machines”. In: *Quantum* 5 (Apr. 2021), p. 427. DOI: 10.22331/q-2021-04-08-427. arXiv: 1908.06720 [quant-ph] (page 10).

[KS48] Robert Karplus and Julian Schwinger. “A note on saturation in microwave spectroscopy”. In: *Physical Review* 73 (9 1948), pp. 1020–1026. DOI: 10.1103/PhysRev.73.1020 (page 145).

[KV17] Ravindran Kannan and Santosh Vempala. “Randomized algorithms in numerical linear algebra”. In: *Acta Numerica* 26 (2017), pp. 95–135. DOI: 10.1017/S0962492917000058 (pages 19, 45, 49).

[LC17] Guang Hao Low and Isaac L. Chuang. “Optimal Hamiltonian simulation by quantum signal processing”. In: *Physical Review Letters* 118.1 (Jan. 2017), p. 010501. DOI: 10.1103/PhysRevLett.118.010501. arXiv: 1606.02685 [quant-ph] (pages 4, 135).

[LGZ16] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. “Quantum algorithms for topological and geometric analysis of data”. In: *Nature Communications* 7.1 (Jan. 2016), p. 10138. DOI: 10.1038/ncomms10138. arXiv: 1408.3106 (pages 1, 18).

[Llo96] Seth Lloyd. “Universal quantum simulators”. In: *Science* 273.5278 (Aug. 1996), pp. 1073–1078. DOI: 10.1126/science.273.5278.1073 (page 135).

[LMR13] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. “Quantum algorithms for supervised and unsupervised machine learning”. 2013. arXiv: 1307.0411 [quant-ph] (pages 6, 15, 16, 119, 120).

[LMR14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. “Quantum principal component analysis”. In: *Nature Physics* 10.9 (July 2014), pp. 631–633. DOI: 10.1038/nphys3029. arXiv: 1307.0401 [quant-ph] (pages 15, 16, 121).

[LRS15] James R. Lee, Prasad Raghavendra, and David Steurer. “Lower bounds on the size of semidefinite programming relaxations”. In: *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*. ACM, June 2015. DOI: 10.1145/2746539.2746599. arXiv: 1411.6317 [cs.CC] (page 143).

[Mah11] Michael W. Mahoney. “Randomized algorithms for matrices and data”. In: *Foundations and Trends® in Machine Learning* 3.2 (2011), pp. 123–224. ISSN: 1935-8237. DOI: 10.1561/2200000035 (page 19).

[McD89] Colin McDiarmid. “On the method of bounded differences”. In: *Surveys in Combinatorics, 1989: Invited Papers at the Twelfth British Combinatorial Conference*. London Mathematical Society Lecture Note Series. Cambridge, England: Cambridge University Press, 1989, pp. 148–188. DOI: 10.1017/CBO9781107359949.008 (pages 41, 42).

[MH02] John C Mason and David C Handscomb. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002 (pages 27, 68, 72).

[MMD08] Michael W. Mahoney, Mauro Maggioni, and Petros Drineas. “Tensor-CUR decompositions for tensor-based data”. In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008), pp. 957–987. DOI: 10.1137/060665336 (page 19).

[MMS18] Cameron Musco, Christopher Musco, and Aaron Sidford. “Stability of the lanczos method for matrix function approximation”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, Jan. 2018, pp. 1605–1624. DOI: 10.1137/1.9781611975031.105 (page 72).

[MRTC21] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. “Grand unification of quantum algorithms”. In: *PRX Quantum* 2 (4 Dec. 2021), p. 040203. DOI: 10.1103/PRXQuantum.2.040203. arXiv: 2105.02859 [quant-ph] (page 4).

[MZ11] Avner Magen and Anastasios Zouzias. “Low rank matrix-valued chernoff bounds and approximate matrix multiplication”. In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM. 2011, pp. 1422–1436 (page 45).

[Oli79] J. Oliver. “Rounding error propagation in polynomial evaluation schemes”. In: *Journal of Computational and Applied Mathematics* 5.2 (1979), pp. 85–97. ISSN: 0377-0427. DOI: 10.1016/0771-050X(79)90002-0 (page 72).

[PC99] Victor Y. Pan and Zhao Q. Chen. “The complexity of the matrix eigenproblem”. In: *Proceedings of the thirty-first annual ACM symposium on Theory of Computing*. ACM, May 1999. DOI: 10.1145/301250.301389 (page 148).

[Pra14] Anupam Prakash. “Quantum algorithms for linear algebra and machine learning”. PhD thesis. University of California at Berkeley, 2014. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.pdf (pages 6, 8, 16, 35, 121, 124).

[Pre18] John Preskill. “Quantum computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), p. 79. DOI: 10.22331/q-2018-08-06-79. arXiv: 1801.00862 (pages i, 16, 115).

[RL18] Patrick Rebentrost and Seth Lloyd. “Quantum computational finance: quantum algorithm for portfolio optimization”. Nov. 9, 2018. DOI: 10.1145/3318041.3355465. arXiv: 1811.03975 [quant-ph] (page 124).

[RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. “Quantum support vector machine for big data classification”. In: *Physical Review Letters* 113.13 (13 Sept. 2014), p. 130503. DOI: 10.1103/PhysRevLett.113.130503. arXiv: 1307.0471 (pages 15, 16, 124, 128, 129).

[RSML18] Patrick Rebentrost, Adrian Steffens, Iman Marvian, and Seth Lloyd. "Quantum singular-value decomposition of nonsparse low-rank matrices". In: *Physical Review A* 97 (1 Jan. 2018), p. 012327. DOI: 10.1103/PhysRevA.97.012327.

[RSWPL19] Patrick Rebentrost, Maria Schuld, Leonard Wossnig, Francesco Petruccione, and Seth Lloyd. "Quantum gradient descent and Newton's method for constrained polynomial optimization". In: *New Journal of Physics* 21.7 (July 2019), p. 073023. DOI: 10.1088/1367-2630/ab2a9e. arXiv: 1612.01789 [quant-ph] (page 35).

[RV07] Mark Rudelson and Roman Vershynin. "Sampling from large matrices: an approach through geometric functional analysis". In: *Journal of the ACM* 54.4 (July 2007), 21–es. ISSN: 0004-5411. DOI: 10.1145/1255443.1255449. URL: https://doi.org/10.1145/1255443.1255449 (pages 44, 45).

[RWCRPS20] Alessandro Rudi, Leonard Wossnig, Carlo Ciliberto, Andrea Rocchetto, Massimiliano Pontil, and Simone Severini. "Approximating Hamiltonian dynamics with the Nyström method". In: *Quantum* 4 (Feb. 2020), p. 234. DOI: 10.22331/q-2020-02-20-234. arXiv: 1804.02484 [quant-ph] (page 141).

[Sch41] A. C. Schaeffer. "Inequalities of A. Markoff and S. Bernstein for polynomials and related functions". In: *Bull. Amer. Math. Soc.* 47 (1941), pp. 565–579. ISSN: 0002-9904. DOI: 10.1090/S0002-9904-1941-07510-5 (pages 58, 78).

[Sha04] Aleksei Shadrin. "Twelve proofs of the Markov inequality". In: *Approximation theory: a volume dedicated to Borislav Bojanov*. Prof. M. Drinov Acad. Publ. House, Sofia, 2004, pp. 233–298 (page 78).

[Sho97] Peter W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172. arXiv: quant-ph/9508027 [quant-ph] (page 1).

[SV14] Sushant Sachdeva and Nisheeth K. Vishnoi. "Faster algorithms via approximation theory". In: *Foundations and Trends in Theoretical Computer Science* 9.2 (2014), pp. 125–210. ISSN: 1551-305X. DOI: 10.1561/0400000065 (pages 27, 28).

[SWZ16] Zhao Song, David Woodruff, and Huan Zhang. "Sublinear time orthogonal tensor decomposition". In: *Advances in Neural Information Processing Systems 29*. Red Hook, NY, USA: Curran Associates, Inc., 2016, pp. 793–801 (page 19).

[Tan19] Ewin Tang. "A quantum-inspired classical algorithm for recommendation systems". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing - STOC 2019*. ACM Press, 2019, pp. 217–228. DOI: 10.1145/3313276.3316310. arXiv: 1807.04271 [cs.IR] (pages iii, 15, 20, 24, 31, 50, 114, 117).

[Tan21] Ewin Tang. "Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions". In: *Physical Review Letters* 127 (6 Aug. 2021), p. 060503. DOI: 10.1103/PhysRevLett.127.060503. arXiv: 1811.00414 [cs.IR] (pages iii, 15, 17, 119–122).

[Tan22] Ewin Tang. "Dequantizing algorithms to understand quantum advantage in machine learning". In: *Nature Reviews Physics* 4.11 (Sept. 2022), pp. 692–693. DOI: 10.1038/s42254-022-00511-w (page iii).

[Tao10] Terence Tao. *254a, Notes 3a: Eigenvalues and sums of Hermitian matrices.* https://terrytao.wordpress.com/2010/01/12/254a-notes-3a-eigenvalues-and-sums-of-hermitian-matrices/. 2010 (page 49).

[Tao13] Terence Tao. *Matrix identities as derivatives of determinant identities, 2013.* 2013. URL: https://terrytao.wordpress.com/2013/01/13/matrix-identities-as-derivatives-of-determinant-identities (page 63).

[Tre19] Lloyd N. Trefethen. *Approximation theory and approximation practice, extended edition.* Extended edition [of 3012510]. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2019, pp. xi+363. ISBN: 978-1-611975-93-2. DOI: 10.1137/1.9781611975949 (pages 27, 58, 66, 71).

[Tro15] Joel A. Tropp. "An introduction to matrix concentration inequalities". In: *Foundations and Trends® in Machine Learning* 8.1-2 (2015), pp. 1–230. DOI: 10.1561/2200000048. arXiv: 1501.01571 [math.PR] (page 47).

[Van11] Maarten Van den Nest. "Simulating quantum computers with probabilistic methods". In: *Quantum Information and Computation* 11.9&10 (Sept. 2011), pp. 784–812. ISSN: 1533-7146. DOI: 10.26421/qic11.9-10-5. arXiv: 0911.1624 [quant-ph] (pages 19, 37).

[Vos91] Michael D. Vose. "A linear algorithm for generating random numbers with a given distribution". In: *IEEE Transactions on Software Engineering* 17.9 (1991), pp. 972–975. DOI: 10.1109/32.92917 (page 35).

[WBL12] Nathan Wiebe, Daniel Braun, and Seth Lloyd. "Quantum algorithm for data fitting". In: *Physical Review Letters* 109.5 (Aug. 2012), p. 050505. DOI: 10.1103/physrevlett.109.050505.

[Wel09] Max Welling. "Fisher linear discriminant analysis". https://www.ics.uci.edu/~welling/teaching/273ASpring09/Fisher-LDA.pdf. 2009. URL: https://www.ics.uci.edu/~welling/teaching/273ASpring09/Fisher-LDA.pdf (page 150).

[Woo14] David P. Woodruff. "Sketching as a tool for numerical linear algebra". In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (2014), pp. 1–157. ISSN: 1551-305X. DOI: 10.1561/0400000060 (pages 8, 19).

[WZP18] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. "Quantum linear system algorithm for dense matrices". In: *Physical Review Letters* 120.5 (2018), p. 050502. DOI: 10.1103/PhysRevLett.120.050502. arXiv: 1704.06174 (pages 16, 35, 124).

[ZFF19] Zhikuan Zhao, Jack K. Fitzsimons, and Joseph F. Fitzsimons. "Quantum-assisted Gaussian process regression". In: *Physical Review A* 99 (5 May 2019), p. 052331. DOI: 10.1103/PhysRevA.99.052331. arXiv: 1512.03929 [quant-ph] (page 18).