Quantum Algorithms for Linear Algebra and Machine Learning.

Anupam Prakash

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-211
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.html

December 9, 2014

Quantum Algorithms for Linear Algebra and Machine Learning

by

Anupam Prakash

B.Tech., IIT Kharagpur 2008

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

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Umesh Vazirani, Chair
Professor Satish Rao
Professor Ashvin Vishwanath

Fall 2014

Quantum Algorithms for Linear Algebra and Machine Learning

Anupam Prakash

Abstract

Quantum Algorithms for Linear Algebra and Machine Learning

by

Anupam Prakash

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Umesh Vazirani, Chair

Most quantum algorithms offering speedups over classical algorithms are based on the three techniques of phase estimation, amplitude estimation and Hamiltonian simulation. In spite of the linear algebraic nature of the postulates of quantum mechanics, until recent work by Lloyd and coauthors (23; 22; 24) no quantum algorithms achieving speedups for linear algebra or machine learning had been proposed.

A quantum machine learning algorithm must address three issues: encoding of classical data into a succinct quantum representation, processing the quantum representation and extraction of classically useful information from the processed quantum state. In this dissertation, we make progress on all three aspects of the quantum machine learning problem and obtain quantum algorithms for low rank approximation and regularized least squares.

The oracle $QRAM$, the standard model studied in quantum query complexity, requires time $O(\sqrt{n})$ to encode vectors $v \in \mathbb{R}^n$ into quantum states. We propose simple hardware augmentations to the oracle $QRAM$, that enable vectors $v \in \mathbb{R}^n$ to be encoded in time $O(\log n)$, with pre-processing. The augmented $QRAM$ incurs minimal hardware overheads, the pre-processing can be parallelized and is a flexible model that allows storage of multiple vectors and matrices. It provides a framework for designing quantum algorithms for linear algebra and machine learning.

Using the augmented $QRAM$ for vector state preparation, we present two different algorithms for singular value estimation where given singular vector $|v\rangle$ for $A \in \mathbb{R}^{m \times n}$, the singular value σ_i is estimated within additive error $\epsilon \|A\|_F$. The first algorithm requires time $\widetilde{O}(1/\epsilon^3)$ and uses the approach for simulating $e^{-i\rho}$ in (23). However, the analysis (23) does not establish the coherence of outputs, we provide a qualitatively different analysis that uses the quantum Zeno effect to establish coherence and reveals the probabilistic nature of the simulation technique. The second algorithm has a running time $\widetilde{O}(1/\epsilon)$ and uses Jordan's lemma from linear algebra and the augmented $QRAM$ to implement reflections.

We use quantum singular value estimation to obtain algorithms for low rank approximation by column selection, the algorithms are based on importance sampling from the leverage score distribution. We obtain quadratic speedups for a large class of linear algebra algorithms that rely on importance sampling from the leverage score distribution including approximate least squares and

CX and CUR decompositions. Classical algorithms for these problems require time $O(mn \log n + poly(1/\epsilon))$, the quantum algorithms have running time $O(\sqrt{m}poly(1/\epsilon, k, \Delta))$ where k, Δ are the rank and spectral gap. The running time of the quantum CX decomposition algorithm does not depend on m, it is polynomial in problem parameters. We also provide quantum algorithms for ℓ_2 regularized regression problems, the quantum ridge regression algorithm requires time $\widetilde{O}(1/\mu^2\delta)$ to output a quantum state that is δ close to the solution, where μ is the regularization parameter.

Professor Umesh Vazirani
Dissertation Committee Chair

Contents

List of Algorithms

List of Figures

List of Tables

Acknowledgements

I would like to thank Umesh Vazirani for advice, encouragement and helpful discussions on various occasions, this dissertation would not have been written without his constant support. I am also thankful to Professors Satish Rao, Elchanan Mossel and Ashvin Vishwanath for being on my qualifying exam and thesis committees.

Finally, I want to thank Guoming Wang for helpful discussions, Zeph Landau for organizing the quantum reading group and Michael Mahoney for teaching 'Randomized algorithms for matrices and data' in Fall 2013.

Chapter 1

Introduction

Although there are a large number of problems for which quantum algorithms offer a speed up over classical algorithms, the techniques for designing quantum algorithms are rather limited in spite of intensive research over the last two decades. Most known quantum algorithms that obtain speedups over classical algorithms use the three techniques of phase estimation (20), amplitude estimation (4) and Hamiltonian simulation. Shor's factoring algorithm (36), Grover's search (12) and the adiabatic algorithm (8) are canonical examples of algorithms from these three paradigms and it remains a significant research challenge to find new quantum algorithms.

The postulates of quantum mechanics are linear algebraic in nature, so it might appear surprising that until recent work by Lloyd and co-authors (23; 22; 24) no quantum algorithms achieving speedups for linear algebra or machine learning problems had been proposed. These pre-prints claimed exponential quantum speedups for machine learning tasks like principal components analysis, ℓ_2 regularized SVM and k-means clustering, where the quantum algorithm requires time $O(\text{polylog}(n))$ and outputs a quantum state corresponding to the solution, as opposed to a classical algorithm that requires time $O(\text{poly}(n))$ and returns a vector or matrix as answer.

While these results were promising and established a framework for studying quantum algorithms for data analysis problems, they left several issues to be addressed. The exponential speedups were restricted to particular inputs, correctness proofs were not provided for some of the algorithms and the utility of generating quantum states as output was unclear. Addressing these outstanding issues was the main motivation for our work.

The difficulty of designing quantum algorithms for matrix and vector valued data can be appreciated by considering three problems that such an algorithm must address: (i) Encoding the classical matrix/vector valued input into a succinct quantum state. (ii) Processing the quantum encoding in a way that reveals useful structure. (iii) Extracting classically relevant information from the processed quantum state. In this dissertation we make progress on all three aspects of the quantum machine learning problem and present quantum algorithms for low rank approximation and regularized regression.

In chapter 2, we discuss the problem of encoding vectors and matrices into quantum states. The oracle $QRAM$, the standard memory model studied in quantum query complexity requires time $O(\sqrt{n})$ for encoding vectors $v \in \mathbb{R}^n$ into quantum states. The worst case time bound $O(\sqrt{n})$ is known to be tight by the search lower bounds in (2), and is the reason why the algorithms in (23; 22; 24) run in time $O(\text{polylog}(n))$ for a restricted class of inputs.

We propose changes to the memory organization of the oracle $QRAM$ and some simple hardware augmentations, that enable vectors $v \in \mathbb{R}^n$ to be encoded in time $O(\text{polylog}n)$, with pre-processing. The augmented $QRAM$ incurs minimal hardware overheads, the pre-processing can be parallelized and it is a flexible model that allows storage of multiple vectors and matrices of varying size. It provides an efficient solution to the problem of encoding data into quantum states and density matrices and is presented in chapter 2.

In chapter 3, we present two different quantum algorithms for the problem of singular value estimation where the singular values of $M \in \mathbb{R}^{m \times n}$ are estimated to error $\pm\epsilon \|M\|_F$ for M stored in the augmented $QRAM$. The algorithms require time $O(\text{poly}(1/\epsilon))$, polynomial in the accuracy as opposed to classical algorithms that require time polynomial in the matrix dimensions.

The first algorithm is based on the idea for simulating $e^{-i2\pi\rho}$ using an oracle for preparing copies of ρ that was proposed in (23). However, the correctness argument presented in (23) does not establish coherence of the outputs. We provide a qualitatively different and provably correct analysis based on the quantum Zeno effect, concentration bounds and approximate phase estimation. The second algorithm uses Jordan's lemma (18) to relate singular values of M to principal angles between subspaces associated with M, and estimates the angles using the augmented $QRAM$ to implement reflection in the subspaces.

In chapter 4, we use singular value estimation and amplitude amplification to obtain algorithms for sampling from the leverage score distribution in linear algebra and approximating the leverage scores. Importance sampling from the leverage score distribution has several applications to linear algebra including approximate least squares and low rank approximation by column selection. Classical algorithms for these problems require time $O(mn \log n + poly(1/\epsilon))$, the running time for the quantum algorithms if $O(\sqrt{m}poly(1/\epsilon, k, \Delta))$ where k, Δ is the rank and spectral gap. The running times for the different problems are summarized in table 4.1.

In chapter 5, we provide quantum algorithms for the ℓ_2 regularized regression problem and its generalization to polynomial kernels.The ridge regression algorithm requires time $\widetilde{O}(1/\mu^2\delta)$ and produces a quantum state that is δ close to the solution. The algorithms produces a quantum state as answer and are not directly comparable to classical algorithms that output regression coefficients, however we show that the quantum algorithms can be used for comparing models and selecting regularization parameters.

1.1 Preliminaries and notation

We introduce some linear algebra and quantum computing preliminaries and notational conventions that will be followed throughout this dissertation. $\mathbb{R}, \mathbb{C}, \mathbb{Z}, \mathbb{N}$ and $\mathbb{R}^+$ denote the real numbers, complex numbers, integers, natural numbers and positive real numbers. The running

time of algorithms is stated in the standard asymptotic notation, $O(f(n))$ indicates a running time upper bounded by $cf(n)$ for a fixed $c \in \mathbb{R}^+$ and sufficiently large $n \in \mathbb{N}$. The notation $\widetilde{O}()$ hides poly-logarithmic factors, that is $O(f(n)\text{polylog}(n))$ is represented by $\widetilde{O}(f(n))$.

1.1.1 Linear algebra preliminaries

We follow standard linear algebra notation, given a matrix $A \in \mathbb{R}^{m\times n}$, a_{ij} denotes the (i,j)-th entry of A, $\text{nnz}(A)$ is the number of non zero entries of A and $A^T \in \mathbb{R}^{n\times m}$ is the transpose of A. The i-th row of A is denoted by a_i while the j-th column is denoted by a^j. The set $\{1, 2, \cdots, n\}$ is denoted by $[n]$, the standard basis for $\mathbb{R}^n$ is denoted by $\{e_1, e_2, \cdots, e_n\}$. The ℓ_p norm of a vector $x \in \mathbb{R}^n$ is $|x|_p := (\sum_{i\in[n]} |x_i|^p)^{1/p}$ and $|x|_\infty = \max_{i\in[n]} |x_i|$.

The kernel of $A \in \mathbb{R}^{m\times n}$ is defined as $Ker(A) = \{x \in \mathbb{R}^n \mid Ax = 0\}$, the image of A is the column space of A and is denoted by $Col(A)$. Subspaces of $\mathbb{R}^n$ are denoted by calligraphic fonts, that is $\mathcal{A}, \mathcal{B}$ denote subspaces. The orthogonal complement of $\mathcal{A} \in \mathbb{R}^n$ is defined as $\mathcal{A}^\perp = \{x \in \mathbb{R}^n \mid a.x = 0 \; \forall \; a \in \mathcal{A}\}$. The direct sum of subspaces $\mathcal{A}, \mathcal{B} \in \mathbb{R}^n$ is defined as $\mathcal{A} \oplus \mathcal{B} = \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\}$.

The adjoint $A^* = \overline{A^T}$ for a complex matrix $A \in \mathbb{C}^{n\times n}$ is obtained by taking the element-wise complex conjugate of the transpose. The matrix A is normal if it commutes with the adjoint $AA^* = A^*A$, Hermitian if $A = A^*$ and unitary if $AA^* = A^*A = I$. A matrix $A \in \mathbb{C}^{n\times n}$ has n eigenvectors and eigenvalues $Av_i = \lambda_i v_i$ where $v_i \in \mathbb{C}^n$ and $\lambda_i \in \mathbb{C}$. The spectral theorem states that every normal operator has a spectral decomposition $A = \sum_{i\in[n]} \lambda_i v_i v_i^t$ where the eigenvectors v_i form an orthonormal basis for $\mathbb{C}^n$. The eigenvalues $\lambda_i \in \mathbb{R}$ for Hermitian matrices and $|\lambda_i| = 1$ for unitary matrices.

By the spectral theorem, a symmetric matrix $A \in \mathbb{R}^{n\times n}$ has spectral decomposition $A = \sum_{i\in[n]} \lambda_i v_i v_i^t$ with $v_i \in \mathbb{R}^n$ and $\lambda_i \in \mathbb{R}$. A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive semi definite if $x^t Ax \geq 0$ for all $x \in \mathbb{R}^n$, or equivalently if all the eigenvalues $\lambda_i \geq 0$. We use the notation $A \succeq 0$ to indicate that A is positive semi-definite. An operator $P \in \mathbb{R}^{n\times n}$ is a projector if $P^2 = P$, projectors are positive semi definite and have eigenvalues 0 or 1.

Real valued functions are extended to matrix variables using the spectral theorem, if $A \in \mathbb{C}^{n\times n}$ has spectral decomposition $A = \sum_i \lambda_i v_i v_i^t$ and $f : \mathbb{R} \to \mathbb{R}$, then $f(A) = \sum_i f(\lambda_i) v_i v_i^t$. Matrix exponentials e^{-iA} and square roots $\sqrt{A}$ for positive semi definite matrices are are defined in this manner.

Singular value decomposition: The singular value decomposition of $A \in \mathbb{R}^{m\times n}$ is a decomposition of the from $A = U\Sigma V^t$ where $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ are unitary and Σ is a diagonal matrix with positive entries. If r is the rank of A, the SVD can be expressed as:

$$A = \sum_{i\in[r]} \sigma_i u_i v_i^t \tag{1.1}$$

where the left and the right singular vectors u_i and v_i are the columns of U and V. The Moore Penrose pseudo-inverse is defined as $A^+ = V\Sigma^+ U^t$,

$$A^+ = \sum_{i\in[r]} \frac{1}{\sigma_i} v_i u_i^t \tag{1.2}$$

AA^+ is the projection onto the column space $Col(A)$ while A^+A is the projection onto the row space $Row(A)$. The condition number $\kappa(A) = \frac{\sigma_{max}}{\sigma_{min}}$, for a square matrix the condition number is $\frac{\lambda_{max}}{\lambda_{min}}$. The truncation of A to the space of the largest k singular values is denoted by A_k, that is $A_k = \sum_{i\in[k]} \sigma_i u_i v_i^t$, the threshold $\tau = \sigma_k^2 / \|A\|_F^2$. The spectral decomposition of AA^t and A^tA can be easily computed from the singular value decomposition of A,

$$AA^t = \sum_{i\in[r]} \sigma_i^2 u_i u_i^t$$
$$A^tA = \sum_{i\in[r]} \sigma_i^2 v_i v_i^t \tag{1.3}$$

The Schatten-p norm for a matrix $\|A\|_p$ is the ℓ_p norm of the vector of its singular values. The Frobenius norm $\|A\|_F^2 = \sum_{ij} a_{ij}^2 = \sum_i \sigma_i^2$, the trace norm $\|A\|_1 = \sum_i |\sigma_i|$ and the spectral norm $\|A\| = \sigma_{max}$. All Schatten-p norms are sub-multiplicative, that is $\|AB\|_p \leq \|A\|_p \|B\|_p$ and unitarily invariant, that is $\|UAV\|_p = \|A\|_p$ for unitaries U, V.

1.1.2 Quantum preliminaries

We use the standard bra-ket notation for representing quantum states. Consider a quantum system with n degrees of freedom. A (pure) quantum state for such a system is a unit vector in the Hilbert space $\mathbb{C}^n$ and is represented by $|v\rangle$. A mixed state is a probabilistic mixture of pure states and is represented by a density matrix, a mixture of states $|w_i\rangle$ with probability p_i is described by a density matrix $\rho = \sum_i p_i |w_i\rangle \langle w_i|$. A quantum register is a system that stores quantum information, the set of all mixed states for register X is denoted by $\mathcal{L}(X)$.

A matrix ρ is a density matrix iff. it is positive semidefinite and has trace 1. The spectrum of ρ consists of eigenvector eigenvalue pairs (v_i, λ_i), ρ represents a mixed state where $|v_i\rangle$ occurs with probability λ_i. Note that several mixtures of pure states may correspond to the same density matrix ρ, but the spectral decomposition provides a unique representation of ρ as a mixture of orthogonal states.

The evolution of an isolated (closed) quantum system is described by Schrödinger's equation,

$$i\hbar \frac{\partial |\phi(t)\rangle}{\partial t} = H |\phi(t)\rangle \tag{1.4}$$

The Hamiltonian H is a Hermitian matrix, if the Hamiltonian is time independent the solution to Schrodinger's equation is $|\phi(t)\rangle = e^{-iHt} |\phi(0)\rangle$, thus the evolution of a closed quantum system is described by a unitary operator $U = e^{-iHt}$. Under the action of U the pure state $|v\rangle$ evolves to $U|v\rangle$ while the density matrix ρ evolves to $U\rho U^*$.

A quantum measurement ($POVM$) on an n dimensional quantum system is a collection of positive operators $M_a \succeq 0$ such that $\sum_a M_a = I_n$, the probability of obtaining outcome a is $Tr(M_a\rho)$. The state of the system after the measurement is $\rho' = \sum_a \sqrt{M_a}\rho\sqrt{M_a}$. Measuring in a given orthogonal basis corresponds to a $POVM$ where the M_a are one dimensional projectors along the basis vectors. Different mixtures of pure states represented by the same density matrix

exhibit identical behavior under quantum measurements. The outcome of measurement M on state ρ is denoted by $M(\rho)$.

The state space for two copies of the quantum system is $C^n \otimes C^n$ where $\otimes$ denotes the tensor product. The density matrix ρ describes the state of the bipartite system, the reduced state of the first system is represented by ρ_1. The reduced state ρ_1 must describe outcomes of measurements on the first subsystem, that is $Tr(M \otimes I\rho) = Tr(M\rho_1)$ must hold for all measurements M. This requirement can be used to show that the reduced state is unique, and is is obtained by tracing out the second system,

$$\rho_1 = Tr_2(\rho) := \sum_{i,j,k,l \in [n]} \rho_{ij,kl}\delta_{jl} \left|i\right\rangle \left\langle k\right| \tag{1.5}$$

The delta function $\delta_{jl} = 1$ for $j = l$ and 0 otherwise. The partial trace is invariant under change of basis for the second system, physically the operation corresponds to discarding the second subsystem or measuring in some fixed basis.

The trace norm for a matrix is the Schatten-1 norm, that is $\left\|A\right\|_1 := Tr(\sqrt{A^*A})$ is the ℓ_1 norm of the vector of singular values of A. The trace norm for a Hermitian matrix is the sum of the absolute values of the eigenvalues. The trace norm is unitarily invariant, that is $\left\|UAV\right\|_1 = \left\|A\right\|_1$ for unitary operators U, V. The maximum probability of distinguishing density matrices ρ, σ using a quantum measurement is upper bounded by the trace norm, thus $\Pr[M(\rho) \neq M(\sigma)] \leq \left\|\rho - \sigma\right\|_1$ for all measurements.

Quantum Gates: A quantum system with 2 degrees of freedom is called a qubit. Quantum gates are local unitary operations that act on a small number of qubits. Circuits for quantum computation are constructed by composing quantum gates, we describe the action of some useful quantum gates. The Hadamard, controlled not ($CNOT$) and Toffoli ($CCNOT$) gates act on $1, 2$ and 3 qubits,

$$\begin{aligned} H\left|i\right\rangle &= \frac{1}{\sqrt{2}}(\left|0\right\rangle + (-1)^i \left|1\right\rangle) && \text{i} = 0,\, 1 \\ CNOT\left|a,b\right\rangle &= \left|a, a \oplus b\right\rangle && a, b \in \{0,1\} \\ CCNOT\left|a,b,c\right\rangle &= \left|a, b, (a \wedge b) \oplus c\right\rangle && a, b, c \in \{0,1\} \end{aligned} \tag{1.6}$$

The $\oplus$ and $\wedge$ operators represent the bitwise Xor and And operations. Adding Not gates to the input of the Toffoli gate, we obtain a gate that applies a bit flip to $\left|c\right\rangle$ conditioned on the values of $\left|a,b\right\rangle$. More generally, using Toffoli gates we can implement a qubit flip conditioned on the values of an arbitrary number of other qubits.

In addition to the above gates, we require phase and controlled rotation gates,

$$\begin{aligned} R_\theta(\left|0\right\rangle) &= \cos(\theta)\left|0\right\rangle + \sin(\theta)\left|1\right\rangle \\ R_\theta(\left|1\right\rangle) &= -\sin(\theta)\left|0\right\rangle + \cos(\theta)\left|1\right\rangle \\ Z_\theta\left|j\right\rangle &= e^{-i\theta j}\left|j\right\rangle \end{aligned} \tag{1.7}$$

These gates are used to encode real valued data into quantum states, the precision θ for these gates must equal the machine precision.

Chapter 2

Quantum Memory Models

In this chapter we address the problem of encoding vector and matrix valued data into quantum states. Such encodings can potentially achieve exponential compression as an n dimensional vector $v \in \mathbb{R}^n$ can be encoded into a quantum state with $O(\log n)$ qubits. We seek encoding algorithms with running time $O(\text{polylog}(n))$, that is polynomial in the encoding size. An encoding algorithm is constrained by the memory model used for storing the data, for example if $v \in \mathbb{R}^n$ is stored in a classical random access memory, an algorithm for any reasonable notion of encoding requires time $O(n)$ as it must access all the coordinates of v and each access requires time $O(1)$.

We formalize the problem of encoding vectors and matrix valued data into quantum states. Density matrix and vector state preparation defined below are fundamental primitives for succinctly encoding real valued data into quantum states, we seek a memory model that allows these tasks to be accomplished in time $\widetilde{O}(\log n)$ with pre-processing.

Definition 2.0.1. *Density matrix preparation: Given $A \in \mathbb{R}^{m \times n}$ stored in memory, create copies of the density matrix $\rho = AA^t/Tr(AA^t)$ and $\rho = A^tA/Tr(A^tA)$.*

Definition 2.0.2. *Vector state preparation: Given $x \in \mathbb{R}^N$ stored in memory create copies of the vector state $|x\rangle = \frac{1}{|x|}\sum_{i\in[N]} x_i |i\rangle$.*

Density matrix preparation reduces to vector state preparation, the choice of memory model is therefore guided by the resources required for vector state preparation.

A $O(\text{polylog}(n))$ time encoding algorithm is potentially achievable only if the memory model allows queries in quantum superposition. The oracle $QRAM$ (quantum random access memory) is the standard model for memory allowing queries in quantum superposition and has been studied extensively in the quantum algorithms and query complexity literature. The quantum query complexity literature establishes upper and lower bounds on the number of queries made to an oracle $QRAM$ for solving specific problems. Query complexity lower bounds (2) show the worst

case query complexity for vector state preparation 2.0.2 is $O(\sqrt{n})$, ruling out the possibility of a $O(\text{polylog}(n)))$ time encoding algorithm using the oracle $QRAM$.

Our solution to the encoding problem circumvents the search lower bounds by pre-processing vectors when loading them into memory. The pre-processing requires simple hardware augmentations to the oracle $QRAM$ and modifications to the memory organization. We propose the augmented $QRAM$ as a model for designing quantum algorithms for problems with matrix and vector valued inputs. The augmented $QRAM$ provides a framework for implementing vector state preparation and related primitives in time $\widetilde{O}(1)$, which we utilize in later chapters to design quantum algorithms for linear algebra and machine learning problems.

This chapter is organized as follows, in section 2.1 we discuss the known results on vector state and density matrix preparation. Amplitude amplification using the oracle $QRAM$ discussed in section 2.1.1 is the best known vector state preparation method. In section 2.1.2, we discuss the reduction of density matrix preparation to vector state preparation (23) and present some additional state preparation procedures that do not require an oracle $QRAM$.

In section 2.2 we present two special cases of vector state preparation that are used as components of the augmented $QRAM$. The first method extends amplitude amplification to create vector states $|x\rangle$ in time $O(\sqrt{\text{nnz}(x)})$ using a quantum key value map. The second method prepares vector states using an auxiliary quantum circuit, it is used for creating constant sized vector states in the augmented $QRAM$.

In section 2.3 we present the augmented $QRAM$, a quantum memory model for storing multiple vectors that can prepare vector state $|x\rangle$ in time $\widetilde{O}(1)$ with a pre processing overhead of $O(\text{nnz}(x))$. The augmented $QRAM$ is obtained by adding a controller to the oracle $QRAM$ and changing the memory organization, the augmentations are used to pre-process vectors while loading them into memory. The hardware and pre-processing overheads for the augmented $QRAM$ are small and the pre-processing can be parallelized. The augmented $QRAM$ is used for the quantum algorithms for singular value estimation in chapter 3.

Finally we present some applications of vector state preparation in section 2.4, including Fourier sampling for matrices and creating states corresponding to products of matrices and matrix vector products.

2.1 Preliminaries

In section 2.1.1, we discuss existing results on vector state preparation and introduce the oracle $QRAM$ model and the amplitude amplification algorithm which are extended to the augmented $QRAM$ in section 2.3. The oracle $QRAM$ is the standard model studied in the quantum query complexity literature (12; 2), architectures for the oracle $QRAM$ have been proposed in (10; 9). The amplitude amplification algorithm (4) is a generalization of Grover's search (12) and has been found to be useful in several contexts, here we use amplitude amplification for vector state preparation using the oracle $QRAM$.

In section 2.1.2, we discuss the reduction from density matrix preparation to vector state

preparation that was implicit in (23), in addition to the reduction we present a method for preparing density matrices corresponding to normalized graph Laplacians without using the oracle $QRAM$.

2.1.1 Oracle $QRAM$ and amplitude amplification

The oracle $QRAM$ with is a memory device capable of answering queries in quantum superposition, if the $QRAM$ has N memory cells with contents $x_i, i \in [N]$, it can achieve the following transformation:

$$\sum_{i\in[N]} \alpha_i |i\rangle \rightarrow \sum_{i\in[N]} \alpha_i |i, x_i\rangle \tag{2.1}$$

The oracle $QRAM$ is the memory model studied in the quantum query complexity literature, algorithms such as Grover's search and amplitude amplification use the the oracle $QRAM$. Physical architectures implementing an oracle $QRAM$ have been proposed but there are significant challenges to be overcome before such a device can be realized in practice.

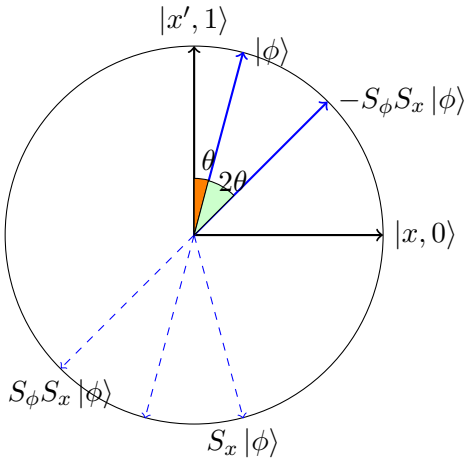
Possible physical realizations and architectures for the oracle $QRAM$ are discussed in detail in the papers (10; 9). A straightforward conversion of the classical RAM to the quantum setting achieves a query time $O(\log N)$ but requires creation of exponentially large quantum superpositions. The bucket brigade architecture proposed in (10) reduces the size of the quantum superpositions but requires query time $O(\log^2 N)$. We note that for all proposed $QRAM$ architectures, the query register is used to address memory and does not interact with the memory contents, that is a transformation of the form $|i\rangle \rightarrow |i \oplus x_i\rangle$ can not be achieved. The query time for the oracle $QRAM$ is $\widetilde{O}(\log n)$ for all proposed architectures.

The vector state $|x\rangle = \frac{1}{|x|}\sum_{i\in[N]} x_i |i\rangle$ can be generated by querying the $QRAM$ on a uniform superposition, appending an ancilla qubit and applying a rotation on it conditioned on the memory contents, post-selecting on the ancilla qubit being $|0\rangle$ and un-computing the memory contents by repeating the $QRAM$ query.

$$\frac{1}{\sqrt{N}}\sum_{i\in[N]} |i\rangle |x_i\rangle \rightarrow \frac{1}{\sqrt{N}}\sum_{i\in[N]} |i\rangle |x_i\rangle \left(\frac{x_i}{|x|_\infty}|0\rangle + \beta_i |1\rangle\right) \tag{2.2}$$

In the above equation $\beta_i = \left(1 - \left(\frac{x_i}{|x|_\infty}\right)^2\right)^{1/2}$, the choice of β_i ensures correct normalization. If $|x| = 1$ the probability of obtaining $|x\rangle$ is $\frac{1}{N|x|_\infty^2}$, the worst case time for preparing $|x\rangle$ using this procedure is $\widetilde{O}(N)$ for a basis vector e_i.

The time complexity can be improved to $\widetilde{O}(\sqrt{N})$ using amplitude amplification (4) as discussed next. Query complexity lower bounds for Grover's search show that some states require $O(\sqrt{N})$ queries to prepare, indicating that amplitude amplification is tight up to logarithmic factors and that without additional assumptions/pre processing $O(\sqrt{N})$ time is required for creating vector states using the oracle $QRAM$.



The composition of reflections about $|x,0\rangle$ and $|\phi\rangle = \cos(\theta)|x',1\rangle + \sin(\theta)|x,0\rangle$ is is a clockwise rotation by 2θ on the two dimensional subspace $Span(|x,0\rangle, |x',1\rangle)$.

Amplitude amplification boosts the probability of obtaining $|x,0\rangle$ by repeated applications of $U = -S_\phi S_x$ while amplitude estimation estimates θ by performing phase estimation on U.

Figure 2.1. An illustration of amplitude amplification and estimation.

Amplitude amplification

We describe the algorithm for creation of vector states using amplitude amplification, first note that the $QRAM$ can be used to implement the unitary transformation $U|0\rangle^{\log N+1} = |\phi\rangle$ where $|\phi\rangle$ is defined as,

$$|0\rangle^{\log N+1} \xrightarrow{FT_N} \frac{1}{\sqrt{N}}\sum_i |i\rangle|0\rangle \xrightarrow{QRAM} \frac{1}{\sqrt{N}}\sum_i |i\rangle\left(\frac{x_i}{|x|_\infty}|0\rangle + \beta_i|1\rangle\right) := |\phi\rangle \tag{2.3}$$

The state $|\phi\rangle$ can be decomposed as $|\phi\rangle = \sin(\theta)|x,0\rangle + \cos(\theta)|x',1\rangle$ where $\sin^2(\theta) = \frac{1}{N|x|_\infty^2}$ is the probability of obtaining $|x\rangle$. The transformations U and its inverse U^{-1} can be implemented in time $\widetilde{O}(\log n)$ using the $QRAM$.

The reflection in $|\phi\rangle$ is defined as $S_\phi|\phi\rangle = |\phi\rangle$ and $S_\phi|\phi^\perp\rangle = -|\phi^\perp\rangle$ where $|\phi^\perp\rangle$ is a state orthogonal to $|\phi\rangle$. It is implemented as $S_\phi = US_0U^{-1}$ where S_0 is a reflection in $|0\rangle^{\log N+1}$, that is $S_0|0\rangle^{\log N+1} = |0\rangle^{\log N+1}$ and $S_0|i\rangle = -|i\rangle$ for $i \neq |0^{\log N+1}\rangle$. Over the two dimensional subspace of states of the form $\sin(\theta)|x,0\rangle + \cos(\theta)|x',1\rangle$, the reflection S_x in $|x,0\rangle$ is a controlled phase flip conditioned on the ancillary qubit being 1. The reflections S_ϕ and S_x can therefore be implemented efficiently.

The product of the reflections $-S_\phi S_x$ acts as (a clockwise) rotation by an angle of 2θ on the subspace spanned by $|x,0\rangle, |x',1\rangle$ as in figure 2.1. The amplitude amplification algorithm starts with the state $|\phi\rangle$ and iteratively applies the operators S_x and $-S_\phi$. After k iterations of amplitude amplification,

$$(-S_\phi S_x)^k|\phi\rangle = \sin((2k+1)\theta)|x,0\rangle + \cos((2k+1)\theta)|x^\perp,1\rangle \tag{2.4}$$

Using the inequality $\theta \geq \sin(\theta) = \frac{1}{\sqrt{N}|x|_\infty}$, after $k = O(\sqrt{N}|x|_\infty)$ iterations of amplitude amplification, the probability of obtaining state $|x\rangle$ becomes a constant. Each iteration of amplitude amplification involves two calls to the oracle $QRAM$, the overall running time is $\widetilde{O}(\sqrt{N})$.

Claim 2.1.1. *The vector state $|x\rangle$ for $x \in \mathbb{R}^N$ can be generated in time $\widetilde{O}(\sqrt{N}|x|_\infty)$ using amplitude amplification.*

Amplitude amplification can be modified so that knowledge of the number of iterations is not required (4). We state a more general version of amplitude amplification as it is a fundamental primitive and will be used subsequently in numerous places. A precise statement may be found in (4), however the version stated below is sufficient for our purposes.

Theorem 2.1.2. *[Amplitude Amplification, (4)] If there is unitary operator U such that $U|0\rangle^l = |\phi\rangle = \sin(\theta)|x,0\rangle + \cos(\theta)|x',1\rangle$ then $|x\rangle$ can be generated in expected time $O(\frac{T(U)}{\sin(\theta)})$, where $T(U)$ is the time to implement U.*

Amplitude amplification can be made exact in the setting where the success probability $p = \sin^2(\theta)$ is known. This can be accomplished by computing $\overline{p} = \sin^2(\overline{\theta})$ where $\overline{\theta} < \theta$ is the largest angle less than θ such that $\frac{\pi/2}{\overline{\theta}} = 2k+1$ is an odd integer. Amplitude amplification can be made exact by appending an ancilla qubit, running the algorithm for k steps with the operator $\overline{U}$ defined below,

$$\overline{U}|0\rangle^{l+1} = |\overline{\phi}\rangle = (\sin(\theta)|x,0\rangle + \cos(\theta)|x',1\rangle) \otimes \left(\sqrt{\frac{\overline{p}}{p}}|0\rangle + \sqrt{1-\frac{\overline{p}}{p}}|1\rangle\right) \tag{2.5}$$

The reflection S_x is replaced by a reflection about $|00\rangle$, after k steps of amplitude amplification the state $|x,00\rangle$ is obtained.

Theorem 2.1.3. *[Exact Amplitude Amplification, (4)] If there is unitary operator U such that $U|0\rangle^l = |\phi\rangle = \sin(\theta)|x,0\rangle + \cos(\theta)|x',1\rangle$ and $\sin(\theta)$ is known then $|x\rangle$ can be generated in time $O(\frac{T(U)}{\sin(\theta)})$, where $T(U)$ is the time to implement U.*

Amplitude estimation

The amplitude estimation algorithm (4) works in the same setting as amplitude amplification and yields an estimate of the success probability $\sin^2(\theta)$. Amplitude estimation is not required for the augmented $QRAM$ or other algorithms in this chapter, however we introduce it here as it extends amplitude amplification. Amplitude estimation will be used later for several applications over chapters 3-5.

Suppose there is a unitary operator U such that $U|0\rangle^l = |\phi\rangle = \sin(\theta)|x,0\rangle + \cos(\theta)|x',1\rangle$. Amplitude estimation produces an estimate for the success probability $\sin(\theta)$ by applying phase estimation (discussed in section 3.1) to the two dimensional eigenspace of $-S_\phi S_x = -US_0U^{-1}S_x$ spanned by $|x,0\rangle$ and $|x',1\rangle$. Recall that $-S_\phi S_x$ is a rotation by an angle of 2θ on this subspace,

the action of $S_\phi S_x$ on this subspace is given by:

$$\begin{aligned} -S_\phi S_x \left|x,0\right\rangle &= \cos(2\theta)\left|x,0\right\rangle - \sin(2\theta)\left|x',1\right\rangle \\ -S_\phi S_x \left|x',1\right\rangle &= \sin(2\theta)\left|x,0\right\rangle + \cos(2\theta)\left|x',1\right\rangle \end{aligned} \tag{2.6}$$

The eigenvalues are $e^{\pm i2\theta}$ with eigenvectors:

$$-S_\phi S_x(\left|x,0\right\rangle \pm i\left|x',1\right\rangle) = e^{\pm i2\theta}(\left|x,0\right\rangle \pm i\left|x',1\right\rangle) \tag{2.7}$$

Phase estimation with $k = O(1/\epsilon)$ achieves an estimate of 2θ within additive error ϵ. As $|\theta - \overline{\theta}| \leq \epsilon \Rightarrow |\sin^2(\theta) - \sin^2(\theta')| \leq 2\sin(\theta)\cos(\theta)\epsilon + O(\epsilon^2)$ by the mean value theorem. Our applications use amplitude estimation to obtain relative error estimates of the success probability.

Theorem 2.1.4. *(Amplitude estimation, (4)) If there is unitary operator U such that $U\left|0\right\rangle^l = \left|\phi\right\rangle = sin(\theta)\left|x,0\right\rangle + cos(\theta)\left|x',1\right\rangle$, then $\sin^2(\theta)$ can be estimated to additive error $\epsilon\sin^2(\theta)$ in time $O(\frac{T(U)}{\epsilon\sin(\theta)})$.*

Our main application of amplitude estimation to obtain relative error estimates of the length of the projection of vector $\left|\phi\right\rangle$ onto the column space of matrix $M \in \mathbb{R}^{m\times n}$ stored in the augmented $QRAM$. The reflection S_ϕ is implemented using the augmented $QRAM$ to prepare the state $\left|\phi\right\rangle$, while the reflection S_x is requires a quantum singular value estimation algorithm 3.3.1.

2.1.2 Density matrix preparation

A density matrix is a positive semidefinite matrix having trace 1. A density matrix represents a mixed state in quantum information, a probabilistic mixture of states where state $\left|v_i\right\rangle$ occurs with probability p_i is represented by the density matrix $\rho = \sum_i p_i \left|v_i\right\rangle\left\langle v_i\right|$. The mixed state interpretation of a density matrix can be used to reduce density matrix preparation to vector state preparation.

Claim 2.1.5. *The density matrix $\rho = A^tA/Tr(A^tA)$ for $A \in \mathbb{R}^{m\times n}$ is a mixture of vector states $\left|a_i\right\rangle$ with probability $p_i = \frac{|a_i|^2}{|A|_F^2}$ for $i \in [m]$.*

Proof. We express the mixture of vector states $\left|a_i\right\rangle$ with probability p_i as a density matrix and show that it is equals ρ,

$$\begin{aligned} \sum_{i\in[m]} p_i \left|a_i\right\rangle\left\langle a_i\right| &= \sum_{i\in[m]} \frac{p_i}{|a_i|^2} \sum_{j,k\in[n]} a_{ij}a_{ik}\left|j\right\rangle\left\langle k\right| \\ &= \frac{1}{|A|_F^2} \sum_{j,k\in[n]} (A^tA)_{jk}\left|j\right\rangle\left\langle k\right| = \rho \end{aligned} \tag{2.8}$$

The second equality follows as $(A^tA)_{jk} = \sum_{i\in[m]} a_{ij}a_{ik}$ and $Tr(A^tA) = |A|_F^2$. Sampling $i \in [m]$ according to probabilities p_i can either be accomplished classically or by creating the vector state $|p\rangle = \sum_{i\in[m]} \sqrt{p_i}\,|i\rangle$ and measuring in the standard basis. □

Applying claim 2.1.5 to A^t, $\rho = AA^t/Tr(AA^t)$ can be prepared as a mixture of states $|a^j\rangle$ with probabilities $p_j = \frac{|a^j|^2}{|A|_F^2}$ for $j \in [n]$. Density matrices $A^tA/Tr(A^tA)$ and $AA^t/Tr(AA^t)$ can also be generated by tracing out the first and second register from the superposition,

$$|A\rangle = \frac{1}{\sqrt{|A|_F}} \sum_{i\in[m],j\in[n]} a_{ij}\,|i\rangle\,|j\rangle \tag{2.9}$$

this follows as the register being traced out samples from the probability distribution in claim 2.1.5. Density matrix preparation therefore reduces to vector state preparation, either using claim (2.1.5) or by creating the vector state in equation (2.9) and tracing out the appropriate sub system.

The mixed state interpretation of density matrices can be used to prepare some classes of density matrices without using an oracle $QRAM$. This is particularly useful for density matrices corresponding to graph Laplacians and sums of density matrices.

Graph Laplacians

A graph $G(V, E)$ with n vertices and m edges has vertex set $V = [n]$ and edges set $E \subseteq (V \times V)$ with $|E| = m$. The Laplacian matrix $L(G) := BB^t$ where $B \in \mathbb{R}^{n\times m}$ is the vertex edge incidence matrix with the edges oriented arbitrarily,

$$B_{v,e} = \begin{cases} -1 & \text{if } e = (v, u) \\ 1 & \text{if } e = (u, v) \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

The sparsity and uniform norms of the columns of B can be utilized to prepare the normalized Laplacian in time $O(\log n)$ for a graph stored in classical memory, access to an oracle $QRAM$ is not required.

Claim 2.1.6. *Given a graph $G(V, E)$ with n vertices and m edges stored as an edge list, the normalized graph Laplacian $L(G)/Tr(L(G))$ can be prepared in time $O(\log n)$.*

Proof. Each column of B has exactly two non zero entries and all columns have equal norm, claim 2.1.5 with $A = B^t$ shows that $L(G)/Tr(L(G))$ is a mixture of states $\frac{1}{\sqrt{2}}(|u\rangle - |v\rangle)$ with probability $1/m$ for $(u, v) \in E(G)$. To create a copy of the normalized Laplacian, sample an edge (u, v)

uniformly at random and apply the following quantum operations,

$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\,|u,v\rangle\,|0\rangle^{\log n} \xrightarrow{CCNOT} \frac{1}{\sqrt{2}}\,|0u\rangle - |1v\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}\,|u\rangle - |v\rangle \tag{2.11}$$

The first step applies two $CCNOT$ operations on all the $\log n$ qubits in register 3, the first gate is a bit flip conditioned on $(0, u_i)$ while the second is conditioned on $(1, v_i)$ for $i \in [\log n]$. The second step applies a Hadamard gate to the first qubit measures in the standard basis, obtaining the desired state with probability $1/2$. The time required is $O(\log m)$ for sampling an edge and $O(\log n)$ for applying the required $CCNOT$ gates. □

Sums of density matrices

The probabilistic interpretation of density matrices can be used to prepare $\mu\rho + (1-\mu)\sigma$ for $\mu \in [0,1]$ if density matrices ρ, σ can be prepared.

Claim 2.1.7. *Selecting ρ with probability μ and σ with probability $(1-\mu)$ prepares the density matrix $\mu\rho + (1-\mu)\sigma$.*

The density matrix I/n corresponds to the completely mixed state $I = \frac{1}{n}\sum_i |v_i\rangle\langle v_i|$ and can be prepared efficiently by selecting $|i\rangle$ with probability $1/n$. Thus, density matrices of the form $\mu\rho + (1-\mu)I/n$ can be prepared, this will be utilized by quantum algorithms for ridge regression and page rank in chapter 4.

2.2 Vector state preparation

In this section we present algorithms for two special cases of vector state preparation, these are used as as subroutines in the augmented $QRAM$ insert and query algorithms. The first special case considers sparse vectors and provides an algorithm for generating $|x\rangle$ in time $O(\sqrt{\text{nnz}(x)})$. The extension requires a quantum key value map implemented using classical data structures in combination with the oracle $QRAM$. The second special case considers vector states having constant dimension, such states are generated in constant time using an auxiliary quantum circuit in section 2.2.2.

2.2.1 Sparse vector states

Consider the problem of preparing vector state $|v\rangle = \sum_{i\in[n]} v_i\,|i\rangle$ where $v \in \mathbb{R}^n$ is a sparse vector with $\text{nnz}(v)$ non zero entries. Amplitude amplification requires time $\widetilde{O}(\sqrt{n})$ to generate the

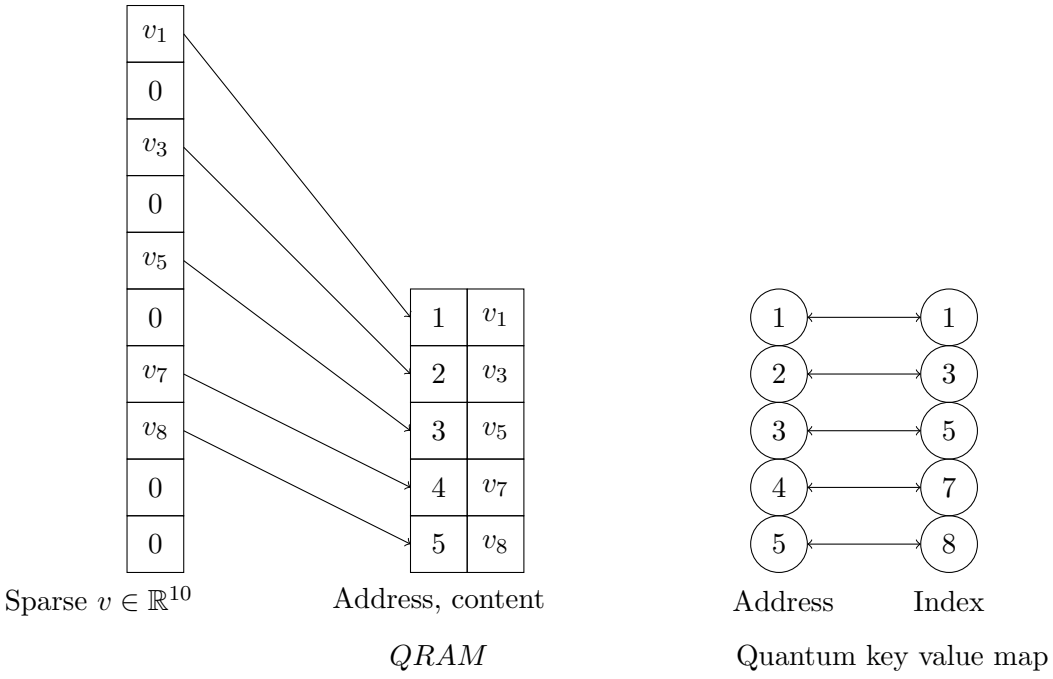


Figure 2.2. Sparse vector state preparation using a quantum key value map.

vector state $|v\rangle$ for $v \in \mathbb{R}^n$, we show that $|v\rangle$ can be prepared in time $\widetilde{O}(\sqrt{\text{nnz}(v)})$. Our solution reduces sparse vector state preparation to amplitude amplification on $nnz(v)$ coordinates with the help of a quantum key value map.

The reduction is illustrated in figure 2.2 where v is a sparse vector whose non zero entries $v_{t_i}, i \in [\text{nnz}(v)]$ are stored in contiguous memory locations addressed by integers in $[\text{nnz}(v)]$. A quantum key value map is maintained between pairs $(K_i, V_i), i \in [\text{nnz}(v)]$, where $K_i = i$ and $V_i = t_i$ is the index of the entry of v stored at address K_i. The key value map allows queries in quantum superposition,

$$\sum_{i\in[n]} \alpha_i |K_i\rangle \leftrightarrow \sum_{i\in[n]} \alpha_i |V_i\rangle \tag{2.12}$$

Claim 2.2.2 shows that the key value map can be set up in time $O(\text{nnz}(v))$ and the query in (2.12) can be implemented in time $\widetilde{O}(1)$. The state $|v\rangle$ is generated by first using amplitude amplification to create the state $|v'\rangle = \sum_{i\in[\text{nnz}(v)]} v_{t_i} |i\rangle$ where j are the indices of the non zero entries of v in time $O(\sqrt{\text{nnz}(v)})$. The state $|v'\rangle$ is converted to $|v\rangle$ using the quantum key value map (2.12), the choice of the key value pairs ensures that $|v'\rangle$ is converted to $\sum_{i\in[\text{nnz}(v)]} v_{t_i} |t_i\rangle = |v\rangle$.

We formalize the above reduction in the following claim and then provide an implementation for the key value map in claim 2.2.2. The key value map is the conceptual primitive that enables sparse vector state preparation, and will be used as a component of the augmented $QRAM$.

Claim 2.2.1. *If $x \in \mathbb{R}^n$ has $nnz(x)$ non zero entries, then k copies of $|x\rangle$ can be prepared in time $\widetilde{O}(nnz(x) + k\sqrt{nnz(x)}|x|_\infty)$ time using an oracle $QRAM$ and a quantum key value map in* (2.2.2).

Proof. Let $t_i, i \in [\text{nnz}(x)]$ denote the indices of the non zero coordinates of $x \in \mathbb{R}^n$, the non zero coordinates of x are stored at addresses $[X_0, X_0 + \text{nnz}(x)]$ in the $QRAM$. The address $X_0 + i$ stores entry x_{t_i}, the offset X_0 indicates the starting location for storing x in the $QRAM$. A quantum key value map is set up as in claim 2.2.2 to map $(X_0 + i) \leftrightarrow t_i$ in time $O(\text{nnz}(x))$. Given the key value map, the state $|x\rangle$ is prepared in time $\sqrt{\text{nnz}(x)}|x|_\infty$ as follows,

$$\begin{aligned}\frac{1}{\sqrt{k}}\sum_{i\in[k]}|i\rangle &\xrightarrow{QRAM} \frac{1}{\sqrt{k}}\sum_{i\in[k]}|i + X_0, x_{t_i}\rangle \\ &\xrightarrow{2.1.1} \frac{1}{\sqrt{k}}\sum_{i\in[k]} x_{t_i}|X_0 + i\rangle \xrightarrow{2.2.2} \frac{1}{\sqrt{k}}\sum_{i\in[k]} x_{t_i}|t_i\rangle \end{aligned} \tag{2.13}$$

□

Claim 2.2.2 describes the implementation of the quantum key value map 2.12, the key value map requires some classical data structures in addition to the oracle $QRAM$. The additional classical components are a circuit for computing a hash function, a bitmap Z to keep track of the memory usage for the $QRAM$ and a list L for resolving hash collisions.

The construction of the key value map is illustrated in figure 2.3, memory addresses for storing key value pairs are determined using a hash function h, the constant c is an upper bound on the maximum number of hash collisions. If there were no hash collisions, storing the key value pair K, V at memory address $h(V), h(K)$ would suffice to implement the key value map. Collisions are resolved by a quantum analog of chaining in classical hash tables.

The key K is stored at memory addresses $f(V) = ch(V) + o$ where the offset $o \in [c-1]$ selects the first unoccupied memory address following $ch(V)$, if the offset is non zero the entry $(V, f(V))$ is stored in the collision list L. Similarly the value V is stored at memory address $f(K)$, the bitmap Z keeps track of the memory usage and helps compute the offsets. The function $f(K)$ is computable by a quantum circuit that outputs $f(K)$ if $K \in L$ and otherwise outputs $ch(K)$, the list L is hard-wired into the circuit computing $f(K)$.

Collisions are addressed using a list instead of memory queries as the function $f(K)$ needs to be computed (and erased) in quantum superposition, and this solution is simpler that making $QRAM$ queries for computing $f(K)$. The size of the list L is expected to be small if the hash function is collision resistant, the augmented $QRAM$ uses a truncated version of the SHA hash functions. Claim 2.2.2 is proved for general key value pairs, the domain from which the keys and values are drawn will be clear from the application.

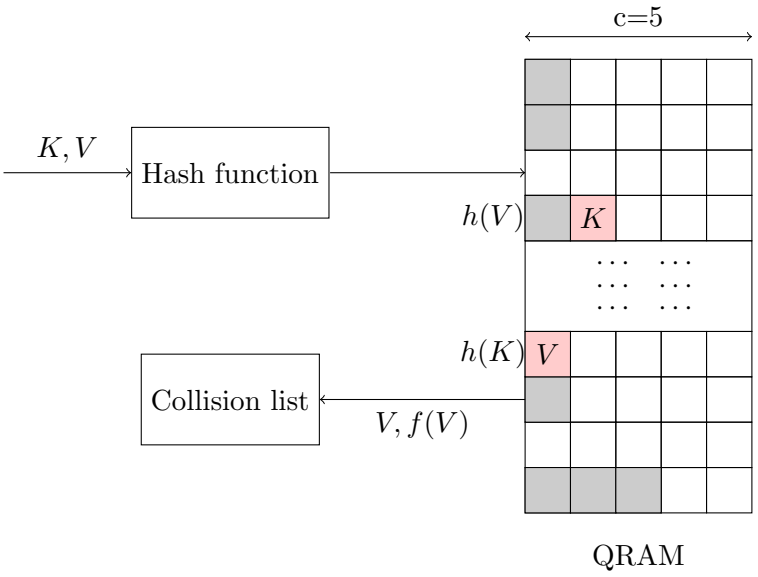


Figure 2.3. The quantum key value map.

Claim 2.2.2. *Let $(K_i, V_i), i \in [n]$ be a set of key value pairs, a bijective quantum key value map can be set up in time $\widetilde{O}(n)$ such that the mappings,*

$$\sum_{i\in[n]} \alpha_i |K_i\rangle \leftrightarrow \sum_{i\in[n]} \alpha_i |V_i\rangle \tag{2.14}$$

can be implemented in time $\widetilde{O}(1)$ using the QRAM.

Proof. Let $h(x)$ be a $\widetilde{O}(1)$ computable hash function that maps keys and values to $[N/c]$, where N is the size of the address space allocated for the key value map and $c \in \mathbb{N}$ is an upper bound on the number of hash collisions. The following algorithm sets up the quantum key value map, the bitmap Z keeps track of memory usage for the $QRAM$, that is $z_i = 1$ if address i is occupied and 0 otherwise.

Algorithm 2.2.1 Quantum key value map

Require: Key value pairs (K_i, V_i) for $i \in [n]$, hash function h, auxiliary data structures bitmap Z and list L.

1: Repeat steps 2-4 for $i \in [n]$.

2: Compute memory addresses $f(x) := ch(x) + o$ where the offset $o \in [c-1]$ is the first unoccupied memory location after $ch(x)$ for $x = K_i, V_i$. The offset $o \in [c-1]$ is computed using Z.

3: If $o > 0$ store the key value pair $(x, f(x))$ in the collision list L.

4: Store V_i at memory location $f(K_i)$ and K_i at memory location $f(V_i)$. Update Z.

5: The address $f(x)$ is subsequently computed as: If $x \in L$ then $f(x) = L(x)$ else $f(x) = ch(x)$.

Note that the collision list L facilitates the computation of $f(x)$ by storing all collisions. After setting up the key value map as above, the quantum key value mapping is achieved as follows,

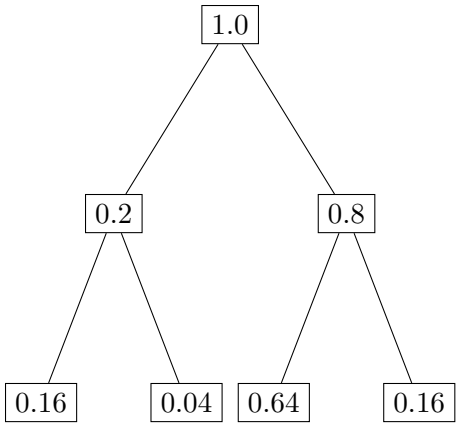
$$\begin{aligned}
\sum_{i\in[n]} \alpha_i |K_i\rangle &\xrightarrow{f(K)} \sum_{i\in[n]} \alpha_i |K_i, f(K_i)\rangle \\
&\xrightarrow{QRAM} \sum_{i\in[n]} \alpha_i |K_i, f(K_i), V_i\rangle \\
&\xrightarrow{f(K),f(V)} \sum_{i\in[n]} \alpha_i |K_i, f(V_i), V_i\rangle \\
&\xrightarrow{QRAM} \sum_{i\in[n]} \alpha_i |f(V_i), V_i\rangle \\
&\xrightarrow{f(V)} \sum_{i\in[n]} \alpha_i |V_i\rangle
\end{aligned} \tag{2.15}$$

Each step involves querying the $QRAM$ or the computation of $f(x)$ in superposition, thus the transformation can be achieved in time $\widetilde{O}(\log N)$.

□

2.2.2 Constant size vector states

We show that the vector state $|x\rangle$ for $x \in \mathbb{R}^b$ can be created in time $\widetilde{O}(\log b)$ using a specialized quantum circuit of size $O(b)$ and pre computed amplitudes. The method is useful for creating constant sized superpositions and is illustrated for a 4 dimensional state $|\phi\rangle$ in figure 2.4.



Let $|\phi\rangle = 0.4\,|00\rangle + 0.2\,|01\rangle + 0.8\,|10\rangle + 0.4|11\rangle$.

- Rotation: $|00\rangle \to \sqrt{0.2}\,|00\rangle + \sqrt{0.8}\,|10\rangle$
- Conditional rotation on qubit 2:
  $\sqrt{0.2}\,|00\rangle + \sqrt{0.8}\,|10\rangle \to$
  $0.4\,|00\rangle + 0.2\,|01\rangle + \sqrt{0.8}\,|10\rangle$
- Conditional rotation on qubit 2:
  $0.4\,|00\rangle + 0.2\,|01\rangle + \sqrt{0.8}\,|10\rangle \to |\phi\rangle$

Figure 2.4. Constant size vector state preparation for 4-dimensional state $|\phi\rangle$.

A complete binary tree on $2^{\lceil \log b \rceil}$ nodes is constructed where the internal nodes store the sum of squared amplitudes of $|\phi\rangle$, as shown in the figure. The amplitudes are then used to apply a sequence of conditional rotations to the initial state $|0\rangle^{\lceil \log b \rceil}$ to obtain $|\phi\rangle$. The circuit for constructing $|\phi\rangle$ is a sequence of controlled rotation gates, where the amplitudes are used to compute the rotation angles. Claim 2.2.3 is the generalization of the method used for preparing $|\phi\rangle$ in figure 2.4.

Claim 2.2.3. *Given $x \in \mathbb{R}^b$, k copies of state $|x'\rangle$ such that $|\,|x'\rangle - |x\rangle\,|^2 = O(1/poly(b))$ can be prepared in time $\widetilde{O}(b + k \log b)$ using an auxiliary quantum circuit of size $O(b)$.*

Proof. Assume that $b = 2^l$ so that $|x\rangle$ is a state on l qubits, the l qubits are denoted by $|x_i\rangle, i \in [l]$. The state $|x'\rangle$ is generated by starting with $|0\rangle^l$ and applying rotations to $|x_{t+1}\rangle$ conditioned on the value of the first t qubits $|x_{1:t}\rangle$. For $k \in \{0,1\}^t, i \in \{0,1\}$ and $t \in [0, l-1]$, define,

$$l_{ki} := \Pr[|x_{t+1}\rangle = i \mid |x_{1:t}\rangle = k] = \sum_{j \in [2^l], j_{1:t+1} = ki} \frac{x_j^2}{|x|^2} \tag{2.16}$$

where the probabilities are with respect to measurement in the standard basis. The rotations applied to $|x_{t+1}\rangle$ are $|0\rangle \to \sqrt{l_{k0}}\,|0\rangle + \sqrt{l_{k1}}\,|1\rangle$, the sign is included for rotations applied to the l-th qubit $|0\rangle \to sgn(x_{k0})\sqrt{l_{k0}}\,|0\rangle + sgn(x_{k1})\sqrt{l_{k1}}\,|1\rangle$. The rotation angles $\theta_k = \arccos(\sqrt{l_{k0}})$ are computed by looking up a trigonometric table, the l_{ki} are accurate to additive error $\epsilon = 1/\widetilde{O}(poly(b))$.

Consider the state $|x'\rangle$ generated by this process, the probability that $|x'\rangle$ when measured in

the standard basis yields outcome y is,

$$\begin{aligned}\Pr[|x'\rangle = y] &= \left(\prod_{i\in[0,l-1]} \sqrt{l_{y_{1:i+1}}} \pm \epsilon\right)^2 \\ &= (1 + O(l\epsilon) + O(\epsilon^2)) \prod_{i\in[0,l-1]} \Pr[|x_{i+1}\rangle = y_{i+1} \mid |x_{1:t}\rangle = y_{1:t}] \\ &= (1 + O(l\epsilon)) \Pr[|x\rangle = y] + O(\epsilon^2) \end{aligned} \tag{2.17}$$

The statistical distance $|\,|x'\rangle - |x\rangle\,|^2$ is a constant for precision $\epsilon = \widetilde{O}(1/\sqrt{b})$ and is $O(1/poly(b))$ for $\epsilon = \widetilde{O}(1/poly(b))$.

Given amplitudes l_{ki} the circuit generating $|x'\rangle$ is a series of controlled rotation gates, the circuit size is $O(b)$ and the depth is $O(\log b)$. The amplitudes are computed using the relations $l_{ki} = x_{ki}^2/|x|^2$ if $|k| = l - 1$ and $l_{ki} = l_{ki0} + l_{ki1}$ otherwise, these relations follow from (2.16). The amplitudes can be computed in time $O(b)$ by computing them bottom up with k decreasing from $(l-1)$ to 0. □

2.3 Augmented $QRAM$

The augmented $QRAM$ is a quantum memory device that stores vectors $v \in \mathbb{R}^n$ and can be used to prepare the vector state $|v\rangle = \frac{1}{\sqrt{|v|}} \sum_{i\in[n]} v_i |i\rangle$ in time $\widetilde{O}(1)$. It can store several vectors of different dimensions, if $v_i \in \mathbb{R}^n$ is the i-th vector stored in an augmented $QRAM$ and $l = \lceil \log n \rceil$ is the number of qubits in $|v_i\rangle$, then the following transformation can be implemented as a unitary operator in time $\widetilde{O}(1)$,

$$|i, 0^l\rangle \to |i, v_i\rangle \tag{2.18}$$

The speedup in vector state preparation is achieved by pre processing the vectors before loading them into memory, the pre-processing is entirely classical.

The starting point for the construction of the augmented $QRAM$ is the observation that given a standard classical RAM, additional data structures can be introduced and the memory organization modified to facilitate computation. Such augmentations are straightforward to implement and do not require additional hardware, we refer to the memory model obtained by augmenting the RAM to facilitate vector state preparation as the augmented RAM.

The construction of the augmented $QRAM$ from the augmented RAM is completely analogous to the construction of an oracle $QRAM$ from a RAM as illustrated in figure 2.5. The obstacles

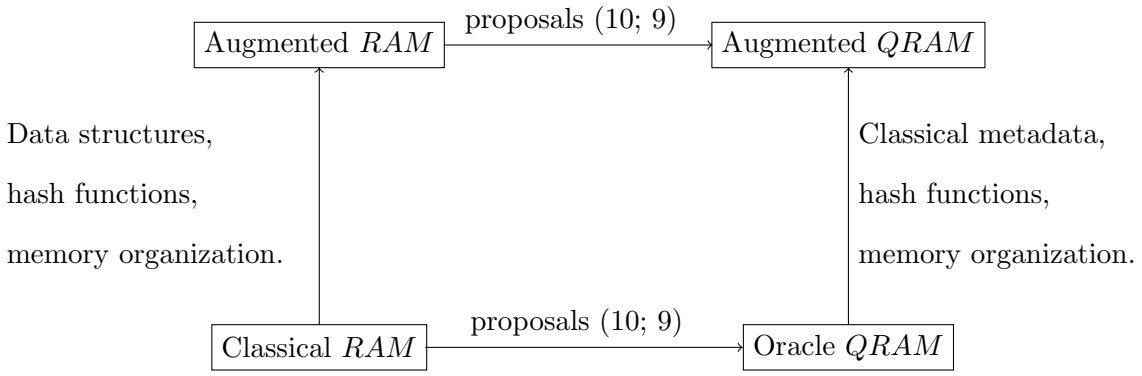


Figure 2.5. A conceptual view of the augmented $QRAM$.

towards the realization of the augmented $QRAM$ as a physical device appear to be the same as the obstacles towards the construction of an oracle $QRAM$ (10; 9).

This section is organized as follows, the components and organization of the augmented $QRAM$ are described in section 2.3.1. The augmentations are classical components added to the RAM to facilitate vector state preparation, the translation to the quantum setting can be achieved following the proposals in (10; 9) to construct the oracle $QRAM$. We provide algorithms for inserting a vector into memory and implementing the query (2.18) in section 2.3.2. In section 2.3.3, we show that the pre-processing for the augmented $QRAM$ can be parallelized. We summarize and compare the different vector state and density matrix preparation methods presented in this chapter in section 2.3.3.

The oracle $QRAM$ provides a framework for studying quantum query complexity, where the complexity of a problem is measured by the number of queries made to the $QRAM$ for solving it. The literature on quantum query complexity is extensive, with both algorithms and lower bounds known for several problems. The augmented $QRAM$ provides a framework for studying quantum algorithms for problems with vector and matrix valued inputs, in this dissertation we provide some examples of problems for which this framework provides speedups.

2.3.1 Augmented $QRAM$ organization

Figure 2.6 illustrates the memory organization for the augmented $QRAM$, for the case where three vectors $v_i \in \mathbb{R}^8, i \in [3]$ are stored in memory. The memory is organized into bins, bin B_k stores all vector entries in the interval $(2^{-k}, 2^{-k+1}]$, for example bin B_2 stores vector entries in the interval $(0.25, 0.5]$. The entries of a vector v_i are stored in chunks of consecutive locations in the bins, the offset o_{ik} is the starting location for the entries of vector v_i in B_k, for example $o_{32} = 3$ in figure 2.6.

The address space of the augmented $QRAM$ is partitioned into three blocks, the first block stores vector entries and is organized into bins. The second block maintains a key value map 2.2.2 to converts bin locations to vector entries, key value maps are maintained for all bins, figure 2.6

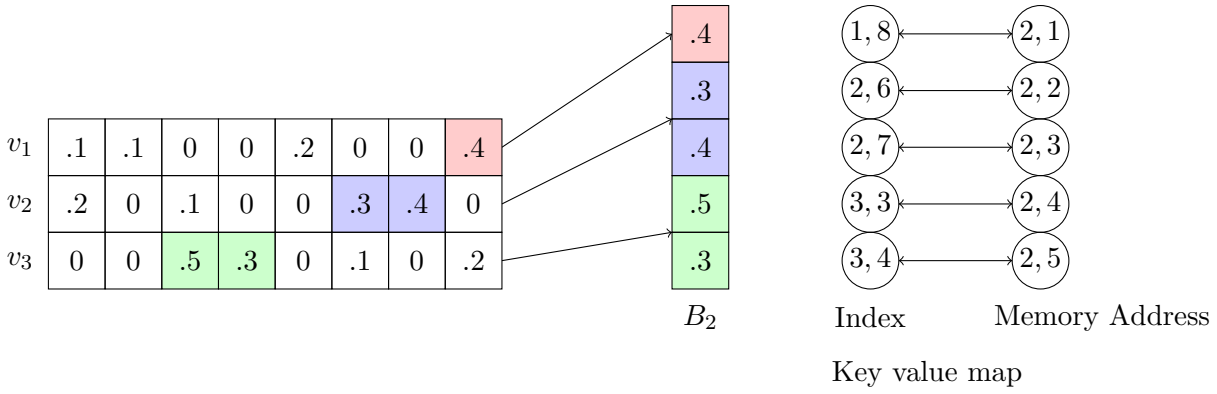


Figure 2.6. Augmented $QRAM$ memory organization

shows the key value map for bin B_2. The third block stores metadata about vectors stored in the augmented $QRAM$. The partitioning of the address space can be implemented logically or by using separate physical devices. The idea for vector state preparation using the augmented $QRAM$ is a generalization of sparse vector state preparation: first create a weighted superposition over bins, perform amplitude amplification over bins this in constant time (2.3) as bin entries are bounded and finally use the key value map to map bin addresses to vector coordinates, the details are described in algorithm 2.3.2.

Figure 2.7 illustrates all the components of the augmented $QRAM$, for the same set of three vectors $v_i, i \in [3]$ as figure 2.6. The $QRAM$ controller streams over the entries of the vectors v_i in order and assigns them to the correct bins, given vector entry x the bin B_k such that $x \in B_k$ can be computed easily from the binary representation of x. The controller also sets up a key value map between $(i, j) \leftrightarrow (k, t)$ where v_{ij} is the t-th entry in B_k, that is it sets up the key value maps as illustrated figure 2.6 for all bins.

The controller maintains some additional metadata for the vectors stored in the augmented $QRAM$. Let m be the number of vectors stored in the augmented $QRAM$, $m = 3$ in figure 2.7. The offset matrix $O \in \mathbb{R}^{m \times b}$ has entries o_{ik} equal to the starting location for v_i in B_k, and is shown in figure 2.7. Some more classical metadata is stored by the controller, the count matrix $C \in \mathbb{R}^{m \times b}$ has entries c_{ik} counting the number of entries of v_i in bin B_k and can be derived from the offset matrix. The success probability vector $p \in \mathbb{R}^m$ has entries $p_i = \sum_{k \in [b]} 2^{-2k} c_{ik}$, the norm vector $q \in \mathbb{R}^m$ has entries $q_i = |v_i|$. The space requirements for maintaining the metadata are $O(m)$ as the number of bins is a constant, the metadata is stored in the third block of the augmented $QRAM$.

The pre-processing performed by the $QRAM$ controller can be done at the hardware level if the hash function for the key value map is implemented in hardware. The hash function also needs to be computable by a quantum circuit for setting up key value maps as in claim 2.2.2. Single chip implementations of the SHA-384 and SHA-512 family of hash functions have been proposed in (28). The address space for a $QRAM$ is much smaller than 384 bits, however the hardware implementations in (28) can be used as a black box truncating the number of bits in the hash function to the match the capacity of the $QRAM$. The single chip implementation can also be used as a blueprint for a quantum circuit implementation of the SHA hash functions.

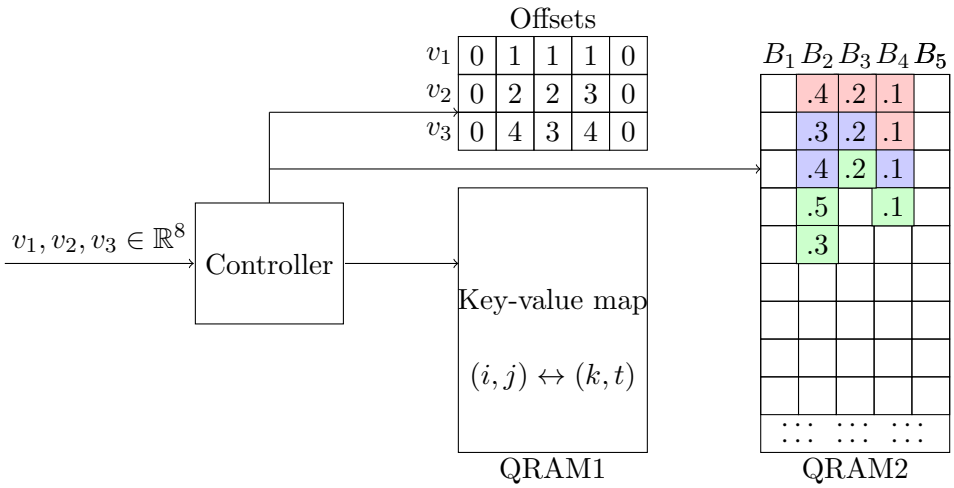


Figure 2.7. Components of the augmented $QRAM$.

The SHA family of hash functions are believed to be collision resistant, that is it is computationally hard to find keys such that $f(k_1) = f(k_2)$. Our algorithms do not require collision resistance as collisions are resolved using the an associative array D. Collisions are rare for SHA hash function, so the size of D is expected to be small and the bound c on the number of hash collisions in claim 2.2.2 can be chosen to be a small constant.

2.3.2 Insertion and query algorithms

We present algorithms for inserting a vector into the augmented $QRAM$ and for implementing the queries in (2.18). Inserting x into the augmented $QRAM$ requires pre-processing x, the pre-processing overhead is $O(\text{nnz}(x))$, in section 2.3.3 we show that the insertion algorithm can be parallelized and the pre-processing overhead reduced to $O(\text{nnz}(x)/p)$ if p parallel units are available. The insertion algorithm normalizes the vectors x being stored in the augmented $QRAM$ to have unit norm, this is without loss of generality as the notion of vector states 2.0.2 is independent of the norm. The normalization of vectors to unit length can also be performed classically as a part of dataset preparation. The norms for the vectors stored in the augmented $QRAM$ are stored as metadata in vector $q \in \mathbb{R}^m$.

Algorithm 2.3.1 Augmented $QRAM$ insertion algorithm

Require: x is the $N+1$ st vector inserted into the augmented $QRAM$, count matrix C and offset matrix O are stored as metadata.

1: Compute offsets for x in B_k as $o_{N+1,k} = o_{N,k} + c_{N,k}$. Initialize $c(k) = 0$ for $k \in [b]$, the $c(k)$ are running counts of the number of elements of x in B_k.

2: Compute $q(N+1) = |x|$, repeat steps 3-4 streaming over the non zero entries x_j of x.

3: For each x_j compute k such that $x_j/q(N+1) \in B_k = [2^{-k}, 2^{-k-1}]$ and index $t = o_{N+1,k} + c(k)$. Increment count $c(k)$.

4: Set up a quantum key value map as in claim 2.2.2 between $(N+1, j) \leftrightarrow (k, t)$, store x_j at memory location (k, t) in the first memory block.

5: Update the count matrix setting $c_{N+1,k} = c(k)$, compute $p(N+1) = \sum_{k\in[b]} 2^{-2k} c(k)$.

Vector state preparation using the augmented $QRAM$ utilizes the quantum key value map set up in step 4 of the insertion algorithm and the circuit in claim 2.2.3 for preparing weighted superpositions over bins. The algorithm for implementing the query $|i, 0\rangle \rightarrow |i, x\rangle$ where x is the i-th vector stored in the augmented $QRAM$ is presented next. We establish correctness and bound the running time in claim 2.3.1.

Algorithm 2.3.2 Augmented $QRAM$ vector state preparation

Require: $x \in \mathbb{R}^n$ is the i-th vector stored in the augmented $QRAM$.

1: Using counts $c(k) = c_{ik}$, create the weighted superposition $\frac{1}{\sqrt{\sum_{k\in[b]} 2^{-2k}c(k)}} \sum_{k\in[b]} 2^{-k}\sqrt{c(k)}\,|k\rangle$ over bins using the circuit from claim 2.2.3.

2: Create the superposition $\frac{1}{\sqrt{\sum_{k\in[b]} 2^{-2k}c(k)}} \sum_{k\in[b],t\in[c(k)]} 2^{-k}\,|k,t\rangle$, a superposition over the entries in each bin, use exact amplitude amplification 2.1.3 to avoid measurements.

3: Query the $QRAM$, append an ancilla and apply a conditional rotation,

$$\frac{1}{\sqrt{\sum_{k\in[b]} 2^{-2k}c(k)}} \sum_{k\in[b],t\in c(k)} 2^{-k}\,|k,t\rangle \left(\frac{x_j}{2^{-k}}\,|0\rangle + \left(1 - \frac{x_j^2}{2^{-2k}}\right)^{1/2} |1\rangle\right) \tag{2.19}$$

4: Using exact amplitude amplification 2.1.3 on (2.19) with success probability $p(i) = \sum_{k\in[b]} 2^{-2k}c(k)$ stored as metadata, obtain $\sum_{k\in[b],t\in[c(k)]} x_j\,|k,t\rangle$.

5: Apply the quantum key value map set up by the insertion algorithm 2.3.2 to convert $\sum_{k\in[b],t\in[c(k)]} x_j\,|k,t\rangle \to \sum_{x_j\neq 0} x_j\,|i,j\rangle = |i,x\rangle$.

Claim 2.3.1. *Algorithm* 2.3.2 *implements* $|i,0\rangle \to |i,x\rangle\,, x \in \mathbb{R}^N$ *for* $|x| = 1$ *in time* $\widetilde{O}(\log N)$.

Proof. The output of the algorithm 2.3.2 has amplitude of $|i,j\rangle$ proportional to x_j up to a global scaling factor, so the state generated is $|i,x\rangle$. As each element in B_k is at least 2^{-k-1} and $|x| = 1$,

$$\forall k \in [b], \quad \sum_{x_j\in B_k} x_j^2 \geq c(k)2^{-2k-2} \Rightarrow 4 \geq \sum_{k\in[b]} 2^{-2k}c(k) \Rightarrow \frac{1}{\sum_k 2^{-2k}c(k)} \geq \frac{1}{4} \tag{2.20}$$

the probability of measuring $|0\rangle$ in (2.19) is at least $1/4$, thus the number of steps required for exact amplitude amplification in step 4 is constant. Steps 1,2 require time $O(1)$ for creating the weighted superposition over bins as the number of bins b is a constant. The running time is $\widetilde{O}(\log N)$ as applying the key value mapping and querying the $QRAM$ requires time $\widetilde{O}(\log N)$. □

Algorithm 2.3.2 can be implemented as a unitary, that is without using any measurements. Measurements in step 2 can be avoided by setting up superpositions $\frac{1}{\sqrt{c(k)}} \sum_{t\in[c(k)]} |t\rangle$ using the value of $c(k)$ and exact amplitude amplification 2.1.3. The unitary implementation of algorithm

2.3.2 allows implementation of queries in superposition,

$$\sum_i \alpha_i |i, 0\rangle \to \sum_i \alpha_i |i, v_i\rangle \tag{2.21}$$

The unitary for the query algorithm uses the metadata stored in the count matrix C and success probability vector p as inputs, the queries in superposition (2.21) can be implemented if the metadata is stored in quantum memory. Queries of the above kind are used in the quantum singular value estimation algorithm in chapter 3, storing the metadata in classical memory suffices for all other applications of the augmented $QRAM$.

2.3.3 Parallelized augmented $QRAM$

The augmented $QRAM$ uses does not incur a large hardware overhead, however the pre-processing overhead for inserting x is $O(\text{nnz}(x))$. We show that the pre-processing overhead can be reduced to $O(\text{nnz}(x)/p)$ by splitting x into p chunks and processing them in parallel. Such a solution can be readily adapted for existing parallel devices like GPUs and multi-core processors.

The insertion algorithm 2.3.2 computes the norm of x and a hash function to set up a bijective key value map between $(N, j) \leftrightarrow (k, t)$ where x is the N-th vector being stored and x_j is the t-th entry in B_k. The quantum key value map can be parallelized as the hash function can be computed locally and classical data structures used in the key value map can be made concurrent. The bin k can be computed from x_i using a comparison and the norm $|x|$ can be easily computed in parallel.

The only non trivial step in parallelizing the insertion algorithm is computing the index t given x_j. The index computation appears to be an inherently sequential task as elements before $x_{1:i-1}$ must be examined to determine the index of x_j. However, index computation can be expressed as computing the prefixes/cumulative sums an associative operator and the parallel prefix algorithm of (21) can be used. The claim below is stated for the case of dense vectors $x \in \mathbb{R}^n$ with $\text{nnz}(x) = n$, the proof can be adapted for the case of sparse evictors replacing n by $\text{nnz}(x)$.

Claim 2.3.2. *For $x \in \mathbb{R}^n$, the mapping $(N, j) \leftrightarrow (k, t)$ where N is an arbitrary label, $x_j \in B_k$ and t is the index of x_j in B_k can be computed in time $O(nnz(x)/p)$ with p parallel processing units.*

Proof. Define vectors $v_j \in \mathbb{Z}^b$ for $j \in [n]$ such that $v_j = e_k$ if $x_j \in B_k$, and let $+$ be the associative vector addition operator over $\mathbb{Z}^b$. The index t is the k-th coordinate of the cumulative sum $\sum_{i \in [j]} v_i$, it therefore suffices to compute all cumulative sums in time $O(n/p)$. The cumulative sums are computed using the parallel prefix algorithm (21). The parallel prefix algorithm with $n/2$ processors computes cumulative sums in time $O(\log n)$ as described below.

Algorithm 2.3.3 Parallel prefix algorithm (21)

Require: Vector $v_1, v_2, \cdots, v_n$ associative operation $+$, $n/2$ processors, assumes $n = 2^l$ is a power of 2. Outputs $w_1, w_2, \cdots, w_n$, the cumulative sums of the v_i in time $O(\log n)$.

1: Up pass: For $d = 1, 2, \cdots, \log(n)$ and $i \in [n/2^d]$, $v_{2^d i} = v_{2^d i} + v_{2^d i - 2^{d-1}}$.

2: Store v_n in auxiliary variable S and set $v_n = 0$.

3: Down pass: For $d = \log(n), \cdots, 1$ and $i \in [n/2^d]$, set $t = v_{2^d i - 2^{d-1}}$, $v_{2^d i - 2^{d-1}} = v_{2^d i}$ and $v_{2^d i} = v_{2^d i} + t$.

4: Output $w_i = v_{i+1}$ for $i \in [n-1]$ and $w_n = S$.

It is straightforward to adapt the parallel prefix algorithm to compute cumulative sums in time $O(n/p + \log p)$ if p processors are available. Each processor is assigned a chunk of size n/p, processor i computes the sums S_i of the elements assigned to it in time $O(n/p)$. The parallel prefix algorithm is used to compute the cumulative sums C_i of the S_i, given C_i processor i computes cumulative sums for the elements assigned to it with initial value C_i in time $O(n/p)$.

□

Summary and Discussion

The table below summarizes the performance and extra resources required for the different methods we proposed for vector state preparation. The running time for creating k copies of the vector state is stated, to separate the pre-processing overheads from the time required for state creation.

Method	**Running time**	**Extra Resources**
Amplitude amplification	$\widetilde{O}(k\sqrt{N}\|x\|_\infty)$	None
Amplitude amplification	$\widetilde{O}(\text{nnz}(x) + k\sqrt{\text{nnz}(x)}\|x\|_\infty)$	Key value map
Specialized circuit	$\widetilde{O}(N + k)$	Quantum circuit size $O(N)$
Augmented $QRAM$	$\widetilde{O}(\text{nnz}(x) + k)$	Metadata, key value map
Parallel Augmented $QRAM$	$\widetilde{O}(\text{nnz}(x)/p + k)$	p parallel units

Table 2.1. Comparison of vector state preparation algorithms for creating k copies of $|x\rangle$ for $x \in \mathbb{R}^N$.

The augmented $QRAM$ is the preferred solution for vector state preparation as hardware

and pre-processing overheads incurred are small. The parallelized augmented $QRAM$ is feasible from the implementation point of view as all the augmentations over the oracle $QRAM$ can be implemented with currently available hardware. However the other methods are useful in certain settings. Amplitude amplification is useful for operating on data that not stored explicitly in the $QRAM$, but is derived using quantum operations from data stored in the $QRAM$. Some such applications are presented in section 2.4.

The augmented $QRAM$ can be used for density matrix preparation as described in section 2.1.2. Density matrix $\rho = A^tA/Tr(A^tA)$ is prepared from A stored in the $QRAM$ by sampling $i \in [m]$ with probability $p_i = |a_i|^2/|A|_F^2$ and generating $|a_i\rangle$. The sampling can be done classically or by storing the vector $p \in \mathbb{R}^m$ in the augmented $QRAM$ and measuring $|p\rangle$ in the standard basis.

The running time and resources used by different density matrix preparation methods are summarized in table 2.2. The expectations are with respect to the squared norm distribution $p_i = |a_i|^2/|A|_F^2$ on the rows. The parallelized augmented $QRAM$ is the preferred method for density matrix preparation, except for the special cases of graph Laplacians that can be prepared without a $QRAM$.

Method	**Expected running time**	**Extra Resources**
Amplitude amplification	$k\widetilde{O}(\sqrt{n}\|p\|_\infty + E_{j\sim p}[\sqrt{m}\|a_i\|_\infty])$	$QRAM$
Graph Laplacian	$\widetilde{O}(k \log n)$	None
Augmented $QRAM$	$\widetilde{O}(\text{nnz}(A) + k)$	Metadata, key value map
Parallel Augmented $QRAM$	$\widetilde{O}(\text{nnz}(A)/p + k)$	p parallel units

Table 2.2. Comparison of density matrix preparation algorithms for creating k copies of $\rho = A^tA/Tr(A^tA)$ for $A \in \mathbb{R}^{m\times n}$.

2.4 Operations on matrix states

In this section we discuss quantum operations that can be performed on matrix states $|A\rangle = \sum_{i,j} a_{ij} |i,j\rangle$, such operations are potentially useful for designing linear algebra algorithms. We present algorithms for two operations, left and right multiplication by a unitary implementable by a quantum circuit in section 2.4.1 and computation of states corresponding to matrix produces or matrix vector products in section 2.4.2.

2.4.1 Multiplication by unitary operators

Given a matrix state $|A\rangle = \sum_{i\in[m],j\in[n]} a_{ij} |i,j\rangle$ and unitaries U or V that can be implemented as a quantum circuit, the following claim provides a method for generating states $|UA\rangle$ and $|AV^t\rangle$.

Claim 2.4.1. *For $A \in \mathbb{R}^{m\times n}$ and unitaries $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$, states $|UA\rangle$ and $|AV^t\rangle$ can be*

generated by applying U, V to the first and second register of $|A\rangle$,

$$|UA\rangle = (U \otimes I)|A\rangle, |AV^t\rangle = (I \otimes V)|A\rangle \tag{2.22}$$

Proof. As $U|i\rangle = \sum_{k\in[m]} u_{ki}|k\rangle$ we have,

$$\begin{aligned}(U \otimes I)|A\rangle &= \sum_{i\in[m],j\in[n]} a_{ij} U|i\rangle|j\rangle = \sum_{k\in[m],j\in[n]} (u_{k.}a_{.j})|k,j\rangle \\ &= \sum_{k\in[m],j\in[n]} (UA)_{kj}|k,j\rangle = |UA\rangle \end{aligned} \tag{2.23}$$

Similarly as $V|j\rangle = \sum_{k\in[n]} v_{kj}|k\rangle$,

$$\begin{aligned}(I \otimes V)|A\rangle &= \sum_{i\in[m],j\in[n]} a_{ij}|i\rangle V|j\rangle = \sum_{i\in[m],k\in[n]} (a_{i.}v_{k.})|i,k\rangle \\ &= \sum_{k\in[m],j\in[n]} (AV^t)_{ik}|i,k\rangle = |AV^t\rangle \end{aligned} \tag{2.24}$$

□

The above claim and the following fact show that the ability to create $|A\rangle$ and implement unitary U can be used to sample rows or columns of UA or AV^t with probabilities proportional to the squared Euclidean norms.

Fact 2.4.2. *The probability of obtaining outcome i/j if the first/second register of $|A\rangle$ is measured in the standard basis is:*

$$\Pr[i] = \frac{|a_i|^2}{|A|_F^2}, \Pr[j] = \frac{|a^j|^2}{|A|_F^2} \tag{2.25}$$

Corollary 2.4.3. *There is a quantum algorithm that runs in time $\widetilde{O}(T(U))$ where $T(U)$ is the time to implement U and produces samples from the distribution,*

$$\Pr[i] = \frac{|(UA)_i|^2}{|A|_F^2}, \Pr[j] = \frac{|(UA)^j|^2}{|A|_F^2} \tag{2.26}$$

In particular, quantum algorithms can perform Fourier sampling on the rows/columns of A by choosing the unitary U to be the discrete Fourier transform.

2.4.2 States corresponding to matrix products

Suppose $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$ are stored in a $QRAM$, the memory is organized as an an augmented $QRAM$ for multiple matrices/vectors. We present a method for creating $|AD_uB\rangle$ where D_u is a diagonal matrix with $u \in \mathbb{R}^n$ on the diagonal using amplitude amplification. Other methods are not applicable as the entries of AD_uB are not stored in the $QRAM$ but are generated from matrices A and B.

Claim 2.4.4. *If $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$ are stored in the $QRAM$, state $|AD_uB\rangle$ for $u \in \mathbb{R}^n$ can be created in time $\widetilde{O}(\frac{\sqrt{mnp}\max_{ijk} a_{ij}b_{jk}}{|AD_uB|_F})$.*

Proof. The matrix product AB is a sum of outer products of the columns of A with the rows of B,

$$AB = \sum_{t\in[n]} a^t \otimes b_t \tag{2.27}$$

More generally for $u \in \mathbb{R}^n$, $AD_uB = \sum_{t\in[n]} u_t a^t \otimes b_t$, the matrix product is a special case for $u = \frac{1}{\sqrt{n}}\vec{1}$. Creating a uniform superposition over three registers $|i,j,k\rangle$ and querying the $QRAM$ twice on registers $1,2$ and $2,3$ generate the state,

$$|\phi\rangle = \frac{1}{\sqrt{mnp}} \sum_{i\in[m],j\in[n],k\in[p]} |i,j,k,a_{ij}b_{jk}\rangle \left(\frac{a_{ij}b_{jk}}{\max_{ijk} a_{ij}b_{jk}}|0\rangle + \beta|1\rangle\right) \tag{2.28}$$

Erase the auxiliary register using the $QRAM$, and post select on $|0\rangle$ and register j being in the state $|u\rangle$. There is a unitary transformation such that $U|0\rangle^l = |\phi\rangle$ and the reflections involved can be implemented easily, thus amplitude amplification can be used to create a copy of the state $|AD_uB\rangle$ in time $\widetilde{O}(\frac{\sqrt{mnp}\max_{ijk} a_{ij}b_{jk}}{|AD_uB|_F})$. Using the same procedure the matrix product $|AB\rangle$ can be created in time $\widetilde{O}(\frac{n\sqrt{mp}\max_{ijk} a_{ij}b_{jk}}{|AB|_F})$. □

The matrix product preparation procedure is particularly useful for matrices where all entries are bounded in magnitude, if A is a density matrix ρ with bounded entries $a_{ij} \leq O(\frac{1}{\sqrt{\text{nnz}(A)}})$, the state $|A^2\rangle$ requires time $O(1/\sum_i \lambda_i^2)$ to prepare. A useful corollary of the matrix product preparation is the creating the state $|Ax\rangle$ where $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$ is k-sparse.

Corollary 2.4.5. *The state $|Ax\rangle$ for $A \in \mathbb{R}^{m\times n}$ and k-sparse $x \in \mathbb{R}^n$ can be generated in time $\widetilde{O}(\frac{k\sqrt{m}\max_{ij} a_{ij}x_j}{|Ax|_2})$.*

Chapter 3

Quantum singular value estimation

In this chapter, we present quantum algorithms that utilize the ability to create vector states and density matrices from classical data stored in the augmented $QRAM$. Given $M \in \mathbb{R}^{m \times n}$ with singular value decomposition $M = \sum_i \sigma_i u_i v_i^t$ stored in the augmented $QRAM$, we present two algorithms that given $|v_i\rangle$ or $|u_i\rangle$ estimate σ_i within additive error $\epsilon \|M\|_F$ in time $\widetilde{O}(poly(1/\epsilon))$. These algorithms will be used to obtain quantum algorithms for linear algebra and machine learning problems in chapters 4 and 5.

We begin by defining the singular value estimation problem and its quantum analog.

Definition 3.0.6. *Singular value estimation (M, ϵ): Suppose $M \in \mathbb{R}^{m \times n}$ has singular value decomposition $M = \sum_i \sigma_i u_i v_i^t$, given vector v_i (respectively u_i), output estimate $\overline{\sigma_i} \in [\sigma_i \pm \epsilon \|M\|_F]$.*

The classical singular value estimation problem has access to the entries of the singular vector v_i, the quantum analog of the singular value estimation problem uses $|v_i\rangle$ as input, the output for both cases and an additive error $\epsilon \|M\|_F$ estimate for σ_i.

Definition 3.0.7. *Quantum singular value estimation (M, ϵ): Suppose $M \in \mathbb{R}^{m \times n}$ has singular value decomposition $M = \sum_i \sigma_i u_i v_i^t$, given vector $|v_i\rangle$ (respectively $|u_i\rangle$) output estimate $\overline{\sigma_i} \in [\sigma_i \pm \epsilon \|M\|_F]$.*

Generating inputs $|v_i\rangle$ or $|u_i\rangle$ without knowing the singular value decomposition of M is a difficult task, however quantum singular value estimation can be invoked with a mixed state as input. Quantum spectral sampling defined below is a useful special case of singular value estimation using the density matrix $\rho = MM^t/Tr(MM^t)$ as input. The density matrix ρ represents a mixture of states $|u_i\rangle$ with probability σ_i^2. Copies of ρ can be generated from M stored in the augmented $QRAM$ as described in chapter 2.

More generally, the eigenvalues of a density matrix in $\mathbb{R}^{m\times m}$ define a probability distribution on $[m]$ as they are positive and sum to 1. The quantum spectral sampling problem is to sample an eigenvector of from this distribution and obtain an estimate of the sampling probability.

Definition 3.0.8. *Quantum spectral sampling (ρ, ϵ): Given copies of density matrix ρ with spectral decomposition $\rho = \sum_{i\in[m]} \lambda_i |v_i\rangle \langle v_i|$, sample $(|v\rangle, \overline{\lambda})$ where $\Pr[|v\rangle = |v_i\rangle] = \lambda_i$ and $\overline{\lambda} \in \lambda_i \pm \epsilon$.*

Spectral sampling is used to obtain low rank approximation algorithms in chapter 4. Another application of quantum singular value estimation is computing projections of a given vector state $|v\rangle$ onto the column space of M. The projection algorithm is used to obtain algorithms for ℓ_2 regularized least squares and generalizations in chapter 5.

We provide two algorithms for quantum singular value estimation, both approaches use phase estimation (20) for unitary operators simulated using the augmented $QRAM$. The first algorithm requires an oracle for generating independent copies of $\rho = M^tM/Tr(M^tM)$ for approximately simulating the unitary operator $e^{-i2\pi\rho}$. The algorithm for simulating $e^{-i2\pi\rho}$ using copies of ρ was suggested in (23) but the analysis presented there does not establish coherence which is necessary for phase estimation. We provide a provably correct analysis that uses the quantum Zeno effect to establish coherence, and the Chernoff bounds and approximate phase estimation to bound the approximation error. This is in contrast to existing Hamiltonian simulation techniques that rely on Suzuki-Trotter expansions (3; 23). The running time for the the algorithm is $\widetilde{O}(1/\epsilon^3)$.

Our second algorithm for singular value estimation achieves an improved running time of $\widetilde{O}(1/\epsilon)$, it relies on a conceptual connection between the singular values of M and principal angles between subspaces associated with M. The connection is established using Jordan's lemma relating the eigenspaces of two projectors (27; 39). The second algorithm 3.3.1 is faster and simpler to implement than algorithm 3.2.1 and is therefore preferred for most of the applications in later chapters.

This chapter is organized as follows, in section 3.1 we discuss quantum concepts and techniques used in our analyses. In section 3.2 we describe the spectral sampling and simulation algorithm from (23) and discuss the problems with the analysis. In section 3.2.2, we provide a provably correct analysis for quantum spectral sampling using the quantum Zeno effect and approximate phase estimation. We also provide an efficient implementation for the algorithm without using sparse Hamiltonian simulation techniques and discuss some extensions. In section 3.3.2 we present the second algorithm for quantum singular value estimation using Jordan's lemma.

3.1 Preliminaries

Basic quantum notation and the notions of states, density matrices and the partial trace were introduced in section 1.1.2. In this section we introduce additional quantum concepts and techniques that are required for our analyses.

The completely mixed state is represented by the density matrix I/n, it may be viewed as a

uniform mixture over any orthonormal basis $|v_i\rangle, i \in [n]$ for $\mathbb{C}^n$,

$$I_n = \frac{1}{n} \sum_{i \in [n]} |v_i\rangle \langle v_i| \tag{3.1}$$

The swap operator $S := \sum_{i,j \in [n]} |i,j\rangle \langle j,i|$, acts on two n qubit registers and swaps their contents, that is $S|ab\rangle = |ba\rangle$. The swap operator is an involution, that is $S^2 = I$, the following identity holds for all involution operators:

$$e^{iSt} = \cos t I + i \sin t S \tag{3.2}$$

The identity can be established using the series expansion of e^{iSt} and is analogous to the statement $e^{i\theta} = \cos(\theta) + i\sin(\theta)$.

Phase estimation

Phase estimation (20) is a quantum algorithm that on input $|v\rangle$ an eigenvector of unitary operator U estimates the eigenvalue to additive error ϵ, given a black box oracle for implementing U^k for $k = O(1/\epsilon)$. The black box is implemented by a quantum circuit in most applications of phase estimation, but the spectral sampling algorithm 3.2.1 implements the black box approximately. We describe the phase estimation circuit as the analysis in section 3.2.2 requires showing that phase estimation succeeds with an the approximate implementation of the black box.

The Fourier transform mod N is the unitary operator:

$$FT_N(|j\rangle) = \frac{1}{\sqrt{N}} \sum_{k \in [N]} \omega^{jk} |k\rangle \tag{3.3}$$

where $\omega = e^{2\pi i/N}$ is the N-th root of unity. The Fourier transform mod 2 is an application of the Hadamard gate $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The Fourier transform FT_N can be implemented efficiently as a quantum circuit for all N.

The controlled unitary operator CU uses l control qubits to control the application of U on $|\psi\rangle$, that is $CU(|k\rangle \otimes |\psi\rangle) = |k\rangle \otimes U^k |\psi\rangle$. Let U be a unitary operator and $|v\rangle$ be an eigenvector of U with eigenvalue $e^{i\theta}$, the phase estimation algorithm for estimating θ is the following:

$$\begin{aligned}
|0\rangle |v\rangle &\xrightarrow{H^{\otimes l}} \frac{1}{\sqrt{2^l}} \sum_{k \in [2^l]} |k\rangle |v\rangle \\
&\xrightarrow{CU} \frac{1}{\sqrt{2^l}} \sum_{k \in [2^l]} |k\rangle U^k |v\rangle \\
&= \frac{1}{\sqrt{2^l}} \sum_{k \in [2^l]} e^{i\theta k} |k\rangle |v\rangle \\
&\xrightarrow{(FT_{2^l})^{-1}} |\overline{\theta}\rangle |v\rangle
\end{aligned} \tag{3.4}$$

The analysis of phase estimation shows that if $\theta \in \frac{2\pi}{2^l}[j, j+1]$, then the phase estimation circuit outputs j or $j+1$ with constant probability, thus obtaining an additive error ϵ estimate of θ for $k = O(1/\epsilon)$. The following series sum and trigonometric identities will be used for the analysis in section 3.2.2.

Fact 3.1.1. $\sum_{t\in[n]} t^2 = \frac{n.(n+1).(2n+1)}{6} \leq \frac{n^3}{2.95}$ *for* $n > 100$.

Fact 3.1.2. *The sum of sines and cosines of angles in an arithmetic progression are given by:*

$$\sum_{k\in[0,N-1]} \sin(k\theta) = \frac{\sin(N\theta/2)}{\sin(\theta/2)} \sin((N-1)\theta/2)$$
$$\sum_{k\in[0,N-1]} \cos(k\theta) = \frac{\sin(N\theta/2)}{\sin(\theta/2)} \cos((N-1)\theta/2) \tag{3.5}$$

The following fact follows from the Chernoff bounds:

Fact 3.1.3. *Consider a coin flipping experiment where p_i is the probability of outcome i and define $\delta := p_i - p_j$ where (i, j) are the two most frequent outcomes. The probability that i is the most frequent outcome over $O(\log n/\delta^2)$ independent trials is at least $1 - 1/poly(n)$.*

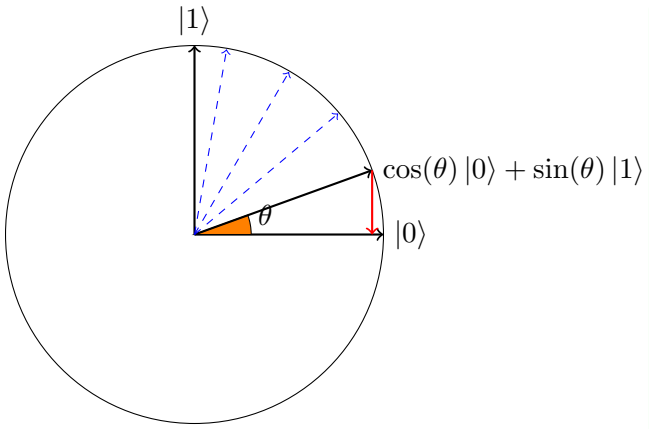
Quantum Zeno effect

The philosopher Zeno posed the arrow paradox asserting that an arrow can never be in motion as at every instant of time its position is measurable and therefore unchanging. If time is composed of instants, and the arrow is at rest for all instants then how can the arrow be in motion? The quantum Zeno effect was first introduced in (30), it refers to the fact that the state of a quantum system remains invariant if it is measured continuously.

A discrete version of the quantum Zeno effect shows that if a quantum system in state $|\phi\rangle$ evolves according to $e^{-iHt/k}|\phi\rangle$ and is measured at discrete time intervals in a basis containing $|\phi\rangle$, the state of the system remains invariant with high probability. The trade off between the error probability and the rate of measurement is illustrated in the example below, if k measurements are made, the probability of the state of the system changing is $O(1/k)$.

The following experiment illustrates the discrete version of the quantum Zeno effect, suppose $|0\rangle$ is rotated to $|1\rangle$ over k steps where each step is a rotation by an angle of $\pi/2k$, that is the rotation gate $R_{\pi/2k}$ is applied k times.

In the first scenario, the qubit is measured in the standard basis after applying k rotations, the outcome of the measurement is 1 as the state of the qubit is $|1\rangle$ after k rotations. In the second scenario, the qubit is measured in the standard basis after each rotation. The probability of obtaining outcome 1 on the first measurement is $\sin^2(\pi/2k) \leq \pi^2/4k^2$, if the measurement outcome is 0 the state collapses to $|0\rangle$ and the same argument can be applied for the next rotation. The



The blue arrows represent a process where $k = 4$ rotations by $\theta = \pi/9$ are applied to $|0\rangle$ followed by measurement in the standard basis. The red arrow represents a process where a measurement is applied after each rotation.

The probability of measuring $|1\rangle$ is 0.96 for process 1. and at most 0.46 for process 2.

Figure 3.1. An illustration of the quantum Zeno effect.

probability of obtaining outcome 1 over the k steps of the process is at most $\pi^2/4k$ by the union bound. The probability of obtaining measurement outcome 1 over k steps in the second scenario can be made arbitrarily small by increasing the number of steps.

3.2 Quantum spectral sampling

An algorithm for simulating $e^{-i2\pi\rho}$ using independent copies of density matrix ρ was proposed in (23), where it is referred to as quantum self analysis. While such an algorithm can be used for singular value estimation, algorithm 3.2.1 is presented for the special case of quantum spectral sampling. The analysis in (23) does not establish the correctness of the algorithm , in particular it does not show that the simulator preserves coherence, this is discussed in section 3.2.1. Our main result theorem 3.2.1 provides a provably correct analysis for algorithm 3.2.1 and bounds the failure probability.

Algorithm 3.2.1 Quantum spectral sampling (23)

Require: Oracle for producing copies of $\rho \in \mathbb{R}^{n\times n}$, precision ϵ, the spectrum of ρ consists of eigenvector eigenvalue pairs $(|v_j\rangle, \lambda_j), j \in [n]$.

1: Simulate the action of $U = e^{-2\pi i\rho}$ on $|\phi\rangle = |\phi_0\rangle$ by iterating

$$|\phi_{t+1}\rangle = Tr_2(e^{-2\pi iS/k}(|\phi_t\rangle\langle\phi_t| \otimes \rho)e^{2\pi iS/k}) \tag{3.6}$$

for $k = \frac{200\log(n\log(1/\epsilon)/\epsilon)}{\epsilon^2}$ steps, output $U_k|\phi\rangle = |\phi_k\rangle$.

2: Perform phase estimation with U replaced by U_k and precision ϵ on input $\sigma = \rho$. Repeat phase estimation $O(\log n)$ times and select the most frequently observed estimate $\overline{\lambda_j}$ to obtain a sample:

$$\sum_{j\in[n]} \lambda_j |v_j\rangle\langle v_j| \otimes |\overline{\lambda_j}\rangle\langle\overline{\lambda_j}| \tag{3.7}$$

Theorem 3.2.1. *Algorithm* 3.2.1 *runs in time* $\widetilde{O}(T(\rho)/\epsilon^3)$ *where* $T(\rho)$ *is the time to prepare* ρ *and returns a sample* $(|v\rangle, \overline{\lambda})$ *with* $\Pr[|v\rangle = |v_j\rangle] = \lambda_j$ *and* $\overline{\lambda} \in \lambda_j \pm \epsilon$ *with probability at least* $1 - O(\epsilon)$.

The identity matrix I/n can be viewed as a mixture of states $|v_i\rangle, i \in [n]$ where the orthonormal basis v_i extends the singular vector basis for A if A is rank deficient. Thus, using the completely mixed state I/n as input in step 2 of algorithm 3.2.1 yields a uniform sample from the eigenvectors.

Corollary 3.2.2. *Algorithm* 3.2.1 *with input* I/n *in step 2 produces a sample* $(|v\rangle, \overline{\lambda})$ *where* $\Pr[|v\rangle = |v_j\rangle] = 1/n$, *with the same running time and guarantees on* $\overline{\lambda}$ *as in theorem 3.1.*

The simulator in step 1 of algorithm 3.2.1 introduces an independent copy of ρ for each iteration, applies the swap operator $e^{-iS2\pi/k}$ and traces out/measures the second system. The simulator can not be described by a unitary operator acting on the first subsystem, it is an example of an open quantum system described by the Kraus operator formalism (33). Establishing correctness algorithm 3.2.1 requires that the unitary operator $U = e^{-i2\pi\rho}$ be simulated coherently, that is if the input $\sigma = |\phi\rangle\langle\phi|$ where $|\phi\rangle$ is a superposition over the eigenstates of ρ, the output should be the coherent superposition $U|\phi\rangle$ as opposed to a probabilistic mixture.

It is not clear a priori that the simulator in algorithm 3.2.1 preserves coherence, in fact the analysis in (23) shows that the output of the simulator is close to the density matrix $U\sigma U^*$ and thus does not address the issue of coherence. Our main contribution is to show that algorithm 3.2.1 approximately simulates U coherently with high probability, thus justifying the use of the simulator for phase estimation.

Before proving theorem 3.2.1, we try to formalize the correctness argument for algorithm 3.2.1 that was implicit in (23) and discuss why such an approach does not seem to work.

3.2.1 The need for coherence

The following identity can be proved by expanding the series $e^{-iS2\pi/k}$ up to second order terms and computing the partial traces for individual terms on the left hand side, note that $[\sigma, \rho] = \sigma\rho - \rho\sigma$ denotes the commutator.

$$Tr_2(e^{-iS2\pi/k}(\sigma \otimes \rho)e^{iS2\pi/k}) = \sigma - i(2\pi/k)[\sigma, \rho] + (4\pi^2/k^2)(\sigma - \rho) + O(1/k^3) \tag{3.8}$$

The identity (3.8) is compared to the Suzuki Trotter expansion of $e^{-i\rho 2\pi/k}\sigma e^{i\rho 2\pi/k}$ up to second order terms,

$$e^{-i\rho 2\pi/k}\sigma e^{i\rho 2\pi/k} = \sigma - i(2\pi/k)[\sigma, \rho] + (2\pi^2/k^2)[\rho, [\sigma, \rho]] + O(1/k^3) \tag{3.9}$$

The following claim formalizes the correctness argument for algorithm 3.2.1 implicit in (23), we argue that the claim is not sufficient for justifying the phase estimation step in algorithm 3.2.1.

Claim 3.2.3. *Let $\sigma_0 = \sigma$ and σ_l be the state obtained after l iterations of step 1 of algorithm* 3.2.1 *and $\overline{\sigma_l} = e^{-i2\pi\rho l/k}\sigma e^{i2\pi\rho l/k}$, then*

$$\|\sigma_l - \overline{\sigma}_l\|_1 \leq \frac{16\pi^2 l}{k^2} \tag{3.10}$$

Proof. The equality of the first order terms in the identities (3.8) and (3.9) yields the following bound for all $j \in [l]$,

$$\left\|\sigma_j - e_1^{-i2\pi\rho/k}\sigma_{j-1}e^{i2\pi\rho/k}\right\|_1 \leq \frac{4\pi^2}{k^2}\left(\rho - \sigma - \frac{1}{2}[\rho, [\sigma, \rho]]\right) \leq \frac{16\pi^2}{k^2} \tag{3.11}$$

The last inequality follows from the triangle inequality and sub multiplicativity for the trace norm, that is $\|\sigma\| = 1$ and $\|\sigma\rho\sigma\|_1 \leq 1$. Denoting the unitary $e^{-i2\pi\rho/k}$ by U and applying the triangle inequality,

$$\begin{aligned}\|\sigma_l - \overline{\sigma}_l\|_1 &\leq \sum_{j\in[l]}\left\|U^{l-j}(\sigma_j - U\sigma_{j-1}U^*)(U^{l-j})^*\right\|_1 \\ &= \sum_{j\in[l]}\|\sigma_j - U\sigma_{j-1}U^*\|_1 \leq \frac{16\pi^2 l}{k^2}\end{aligned} \tag{3.12}$$

The second equality follows from the unitary invariance of the trace norm, the last inequality follows from (3.11). □

Phase estimation requires $O(1/\epsilon)$ iterations to achieve an additive error of ϵ, with the choice of $k = O(1/\epsilon^2)$ in algorithm 3.2.1, step 1 is repeated $l = O(k/\epsilon) = O(1/\epsilon^3)$ times. It is tempting to conclude that algorithm 3.2.1 produces correct results with probability $1 - O(\epsilon)$ as claim 3.2.3 shows that the trace norm error is $O(l/k^2) = O(\epsilon)$.

However, the above argument does not directly establish the correctness of algorithm 3.2.1. Phase estimation with $U = e^{-i2\pi\rho}$ applies a unitary operation over the control register and an input register (3.4), the unitary acting on the combined system is $\overline{U} = \sum_{j\in[1/\epsilon]} |j\rangle\langle j| \otimes e^{-i2\pi\rho j}$. There are two possible ways of using claim 3.2.3 to obtain an algorithm that simulates the unitary $\overline{U}$ on the combined system.

The first approach is to express the unitary $\overline{U}$ in the form $e^{i\overline{\rho}}$ for a density matrix $\overline{\rho}$ of the form $\overline{\rho} = \sigma' \otimes \rho$ for some density matrix σ' on the control register that can be prepared easily. However, there does not seem to be a simple expression for $\overline{\rho}$ of this form, in particular the choice $\overline{\rho} = I/n \otimes \rho$ does not work. The second approach for simulating $\overline{U}$ is to swap independent copies of ρ with the input register as in algorithm 3.2.1. Claim 3.2.3 does not establish coherence of the outputs for this approach, for the setting of phase estimation it does not show that $\sum_{k\in[2^l]} |k, v_j\rangle$ coherently evolves to $\sum_{k\in[2^l]} e^{-2\pi\lambda_j k} |k, v_j\rangle$.

3.2.2 Overview of analysis

Our analysis of the quantum spectral sampling algorithm 3.2.1 splits into two steps, in section 3.2.3 we analyze the simulator U_k and show that if $|\phi\rangle$ is a coherent superposition, then $U_k |\phi\rangle$ is coherent and approximates $U |\phi\rangle$ with high probability. The quantum Zeno effect is used to establish coherence while the Chernoff bounds are used to show that $U_k |\phi\rangle \approx U |\phi\rangle$.

In section 3.2.4 we show that the simulator U_k can be used instead of U for phase estimation. Instead of the state $\sum_{k\in[2^l]} e^{-i2\pi\lambda_j k} |k, v_j\rangle$ obtained using U for for phase estimation, with high probability algorithm 3.2.1 produces the coherent superposition $\sum_{k\in[2^l]} e^{-i2\pi\lambda_j(1\pm\delta)k} |k, v_j\rangle$, where the relative phases $e^{-i2\pi\lambda_j(1\pm\delta)k}$ are approximately correct. Finally we show that the approximate implementation suffices for phase estimation with a small increase in error probability.

The analysis of algorithm 3.2.1 is not restricted to quantum spectral sampling, the algorithm can also be used for singular value estimation and solving linear systems in ρ in the sense described in (15). The running time for algorithm 3.2.1 is $\widetilde{O}(1/\epsilon^3)$, the running time is improved to $\widetilde{O}(1/\epsilon)$ using a different algorithm based on Jordan's lemma in section 3.3.2.

Before presenting the proofs, we provide a high level overview using a simple two dimensional example to illustrate all the important ideas involved in the analysis.

Illustrative example

The following two dimensional example illustrates all the important ideas in the analysis. Let ρ be a rank 2 density matrix with spectral decomposition $\rho = \lambda |v_1\rangle\langle v_1| + (1-\lambda) |v_2\rangle\langle v_2|$. Let $|\phi\rangle = \alpha |v_1\rangle + \beta |v_2\rangle$ be the input state, on which we want to simulate the action of $e^{-i2\pi\rho}$. We

argue that algorithm 3.2.1 maintains coherence with high probability and simulates the action of $U = e^{-i2\pi\rho}$ approximately in the following sense,

$$\begin{aligned} U(\alpha\,|v_1\rangle + \beta\,|v_2\rangle) &= \alpha e^{-i2\pi\lambda}\,|v_1\rangle + \beta e^{-i2\pi(1-\lambda)}\,|v_2\rangle \\ U_k(\alpha\,|v_1\rangle + \beta\,|v_2\rangle) &= \alpha e^{-i2\pi\lambda(1\pm\delta)}\,|v_1\rangle + \beta e^{-i2\pi(1-\lambda)(1\pm\delta)}\,|v_2\rangle \end{aligned} \tag{3.13}$$

The action of U_k is described by the above equation with probability $1 - O(1/k)$. The output $U_k\,|\phi\rangle$ is incoherent with probability $O(1/k)$, if coherence is lost the output can be arbitrary. Equation (3.13) shows that $U\,|\phi\rangle \approx U_k\,|\phi\rangle$ with high probability, a similar argument shows that $U^t\,|\phi\rangle \approx U_k^t\,|\phi\rangle$ for sufficiently large k, thus U can be replaced by U_k in a phase estimation circuit. This overview establishes equation 3.13 for the two dimensional example $(\rho, |\phi\rangle)$, it glosses over the technical details, the complete analysis is contained in sections 3.2.3 and 3.2.4.

Consider the first iteration of the simulator (3.6) on input $|\phi_0\rangle = |\phi\rangle$. The state $|\phi_1\rangle$ depends on the contents of the second register, it equals $|\psi_1\rangle = Tr_2(e^{-i2\pi S/k}\,|\phi\rangle\,|v_1\rangle)$ with probability λ and $|\psi_2\rangle = Tr_2(e^{-i2\pi S/k}\,|\phi\rangle\,|v_2\rangle)$ with probability $(1-\lambda)$. Using the decomposition of the swap operator (3.2) the states $|\psi_i\rangle$ can be computed explicitly,

$$|\psi_1\rangle = Tr_2(\alpha e^{-i2\pi/k}\,|v_1v_1\rangle + \beta\cos(2\pi/k)\,|v_2v_1\rangle - i\beta\sin(2\pi/k)\,|v_1v_2\rangle) \tag{3.14}$$

If the second register when traced out/measured in the spectral basis for ρ remains in state $|v_1\rangle$, $|\psi_1\rangle = (\alpha e^{-i2\pi/k}\,|v_1\rangle + \beta\cos(2\pi/k)\,|v_2\rangle)$. Coherence is maintained, a phase $e^{-i2\pi/k}$ is added to the $|v_1\rangle$ component of the superposition, the amplitude of the $|v_2\rangle$ component decays by $\cos(2\pi/k)$ and the state is renormalized to have norm 1. However, if the state of the second register changes to $|v_2\rangle$, coherence is lost and the superposition collapses with $|\psi_1\rangle = |v_1\rangle$.

The probability of the second register being in state $|v_2\rangle$ in equation (3.14) is at most $\beta^2\sin^2(2\pi/k) \leq 4\pi^2/k^2$ as $\sin(\theta) \leq \theta$ for all $\theta > 0$. A similar argument applies to $|\psi_2\rangle$ and shows that $|\psi_2\rangle = (\alpha\cos(2\pi/k)\,|v_1\rangle + \beta e^{-i2\pi/k}\,|v_2\rangle)$ with probability at least $1 - 4\pi^2/k^2$.

Thus with probability at least $1 - 4\pi^2/k^2$, the effect of a single step of the simulator is to flip a coin that selects (v_1, v_2) with probabilities $(\lambda, 1-\lambda)$ and add a phase of $e^{-i2\pi/k}$ to the corresponding component of the input. Let X_1, X_2 be random variables counting the number of times (v_1, v_2) get selected over k such coin flips. The effect of k steps of the simulator, assuming that the state of the second register remains invariant over each step is,

$$|\phi_k\rangle = \frac{1}{\sqrt{p}}\left(\alpha\cos(2\pi/k)^{X_2}e^{-2\pi iX_1/k}\,|v_1\rangle + \beta\cos(2\pi/k)^{X_1}e^{-2\pi iX_2/k}\,|v_2\rangle\right) \tag{3.15}$$

where p is the probability that the system evolves as in the above equation (3.15). The probability of losing coherence over a single iteration is at most $\frac{4\pi^2}{k^2}$, by the union bound the simulator preserves coherence and evolution is described by equation (3.15) with probability at least $1 - 4\pi^2/k$. This bound on the probability of maintaining coherence is an instance of the quantum Zeno effect and is central to the analysis.

Let $|\phi_k\rangle = \alpha' e^{-2\pi iX_1/k}\,|v_1\rangle + \beta' e^{-2\pi iX_2/k}\,|v_2\rangle$, computing the ratio of the squared amplitudes bounds the effect of renormalization of amplitudes,

$$\frac{1}{1-4\pi^2/k} \geq \frac{1}{p} \geq \frac{(\alpha')^2}{\alpha^2} \geq (1-\sin^2(2\pi/k))^{k/2} \geq 1 - \frac{2\pi^2}{k} \tag{3.16}$$

where the last inequality follows from $(1-x)^n \geq 1-nx$ for $x > 0$. The effect of renormalization of amplitudes is $O(1/k)$ by equation (3.16) and is negligible compared to the other sources of error in the final analysis.

The renormalization of amplitudes can therefore be ignored in equation (3.15). By the Chernoff bounds the random variable X_j are concentrated around its expected value $\lambda_j k$, that with high probability the output state is of the form $U_k |\phi\rangle = \alpha e^{-i2\pi\lambda(1\pm\delta)} |v_1\rangle + \beta e^{-i2\pi(1-\lambda)(1\pm\delta)} |v_2\rangle$ for a suitable value of δ, establishing equation (3.13).

3.2.3 Coherence using the Zeno effect

The simulator U_k in step 1 of algorithm 3.2.1 on input $|\phi_0\rangle = |\phi\rangle$ applies the operation,

$$|\phi_{t+1}\rangle = Tr_2(e^{-i2\pi S/k}(|\phi_t\rangle \langle\phi_t| \otimes \rho)e^{i2\pi S/k}) \tag{3.17}$$

for k iterations with independent copies of ρ and outputs $|\phi_k\rangle$. Claim 3.2.4 shows that for input $|\phi\rangle = \sum_i \alpha_i |v_i\rangle$, the simulator U_k approximately simulates the action of $U = e^{-i2\pi\rho}$ coherently, that is with high probability the output $|\phi_k\rangle$ is a coherent superposition close to $U|\phi\rangle$.

Claim 3.2.4. *Let $|\phi\rangle = \sum_j \alpha_j |v_j\rangle$, the output of the simulator $U_k|\phi\rangle$ is coherent and close to $U|\phi\rangle$ with high probability,*

$$\Pr\left[U_k|\phi\rangle = \sum_{j\in[n]} e^{-i2\pi\lambda_j(1\pm\delta_j)}\beta_j |v_j\rangle\right] \geq 1 - \left(\frac{4\pi^2}{k} + \sum_{j\in[n]} e^{-2\pi\delta_j^2\lambda_j k/3}\right) \tag{3.18}$$

Further, for evolution described by the above equation (3.18), $\sum_j |\alpha_j^2 - \beta_j^2| = O(1/k)$.

Proof. The simulator (3.35) is analyzed by treating the independent copies of ρ as samples from a distribution over eigenstates $|v_j\rangle, j \in [n]$ where $|v_j\rangle$ is sampled with probability λ_j. Suppose the t-th copy of ρ is in state $|v_l\rangle$, and the input state $|\phi_t\rangle = \sum_{j\in[n]} \gamma_j |v_j\rangle$ is coherent, then the effect of the simulator (3.35) over the t-th iteration is described by,

$$|\phi_{t+1}\rangle = Tr_2\left((\cos(2\pi/k)I - i\sin(2\pi/k)S)(\gamma_l |v_l v_l\rangle + \sum_{j\neq l}\gamma_j |v_j v_l\rangle)\right) \tag{3.19}$$

using the decomposition for $e^{-i2\pi S/k}$ from equation (3.2). If both registers are in the state $|v_l\rangle$ the swap operator acts as identity,

$$|\phi_{t+1}\rangle = Tr_2\left(e^{-i2\pi/k}\gamma_l |v_l v_l\rangle + \sum_{j\neq l}\cos(2\pi/k)\gamma_j |v_j v_l\rangle - \sum_{j\neq l} i\sin(2\pi/k)\gamma_j |v_l v_j\rangle\right) \tag{3.20}$$

Conditioned on the event that the register being traced out remains in the state $|v_l\rangle$ when measured in the spectral basis for ρ, the output of the t-th iteration of the simulator is,

$$|\phi_{t+1}\rangle = \frac{1}{\sqrt{N}} \left(e^{-i2\pi/k} \gamma_l |v_l\rangle + \cos(2\pi/k) \sum_{j \neq l} \gamma_j |v_j\rangle \right) \tag{3.21}$$

where N is a normalizing constant. The probability that the second register is in not in the state $|v_l\rangle$ is at most $\sin^2(2\pi/k) \sum_{j \neq l} \gamma_j^2 \leq 4\pi^2/k^2$. Thus if the input state $|\phi_t\rangle$ is coherent, with probability at least $1 - 4\pi^2/k^2$ coherence is maintained and the next iteration of the simulator evolves according to (3.21). By the union bound, coherence is maintained over k steps starting with $|\phi\rangle$ with probability at least $1 - 4\pi^2/k$.

Note that if the measurement outcome for the second register was orthogonal to $|v_l\rangle$, coherence would be lost and $|\phi_{t+1}\rangle$ would collapse to $|v_l\rangle$. Conceptually, coherence is maintained due to the quantum Zeno effect (30; 32). Iteration t applies a small rotation $U = e^{-i2\pi S/k}$ to $|\phi_t, v_l\rangle$ and measures the second register. Coherence is lost the if second register is measured to be in a state orthogonal to $|v_l\rangle$, the probability of losing coherence over k steps is controlled by decreasing the rotation angle, or equivalently increasing k. The error probability over k steps of the simulator decays as $O(1/k)$ by the quantum Zeno effect.

Let Z_{jt} for $t \in [k]$ be a random variable that is $2\pi/k$ with probability λ_j and 0 otherwise and let $X_j = \sum_{t \in [k]} Z_{jt}$. It follows that with probability at least $1 - 4\pi^2/k$ the final state $U_k(|\phi\rangle)$ is in a coherent superposition,

$$U_k |\phi\rangle = \frac{1}{\sqrt{N}} \left(\sum_{j \in [n]} e^{-iX_j \lambda_j} \cos(2\pi/k)^{k - X_j} \alpha_j |v_j\rangle \right) \tag{3.22}$$

By the Chernoff bounds applied to k flips of a coin with bias λ_j the random variables X_j are concentrated around their expected value,

$$\Pr[X_j \notin 2\pi(1 \pm \delta)\lambda_j] \leq e^{-2\pi\delta^2 \lambda_j k/3} \tag{3.23}$$

Equations (3.22) and (3.23) show that the output $U_k |\phi\rangle$ is of the form $\sum_{j \in [n]} e^{-i2\pi\lambda_j(1 \pm \delta_j)} \beta_j |v_j\rangle$ described by equation (3.18) with probability at least $1 - \left(\frac{4\pi^2}{k} + \sum_{j \in [n]} e^{-2\pi\delta_j^2 \lambda_j k/3} \right)$. Establishing

a bound on $\sum_{j\in[n]} |\alpha_j^2 - \beta_j^2|$ conditioned on evolution according to (3.22) completes the proof of the claim.

Deferring all the measurements made on the k auxiliary registers to the end, the normalization factor N in equation (3.22) is equal to the probability that $U_k |\phi\rangle$ evolves according to (3.22). Thus,

$$\frac{1}{1-4\pi^2/k} \geq \frac{1}{N} \geq \frac{\beta_j^2}{\alpha_j^2} \geq \cos(2\pi/k)^{2k} \geq \left(1 - \frac{4\pi^2}{k^2}\right)^k \geq \left(1 - \frac{4\pi^2}{k}\right) \tag{3.24}$$

where the last inequality follows as $(1-x)^n > (1-nx)$ for $x \geq 0$. The above inequalities imply that for all j, $|\alpha_j^2 - \beta_j^2| \leq \frac{4\pi^2}{k} \max(\alpha_j^2, \beta_j^2)$. summing up over all $j \in [n]$ it follows that $\sum_{j\in[n]} |\alpha_j^2 - \beta_j^2| = O(1/k)$.

□

3.2.4 Approximate phase estimation

Claim 3.2.4 shows that the simulator in algorithm 3.2.1 preserves coherence and approximately simulates $e^{-2\pi i\rho}$ with high probability. A similar argument establishes coherence and closeness to for the setting of phase estimation where controlled U_k operations are applied. We analyze phase estimation for the setting where the unitary $U = e^{-i2\pi\rho}$ is replaced by the simulator U_k. Let l be the number of control qubits in the phase estimation circuit, so that the precision $\epsilon = 1/2^l$.

Claim 3.2.5. *If unitary $U = e^{-i2\pi\rho}$ is replaced by the simulator U_k in a phase estimation circuit with precision $\epsilon = 1/2^l, l \geq 7$ and $k = \frac{200\log(n\log(1/\epsilon)/\epsilon)}{\epsilon^2}$, then on input $|\phi, 0\rangle = \sum_{j\in[n]} \alpha_j |v_j, 0\rangle$, output $\sum_{j\in[n]} \alpha_j |v_j, \overline{\lambda_j}\rangle$ such that $|\frac{\overline{\lambda_j}}{2^l} - \lambda_j| \leq \epsilon$ is obtained with probability at least $(1 - O(\epsilon \log n))$ by repeating phase estimation $O(\log n)$ times and choosing the most frequent estimate.*

Proof. If the controlled $U = e^{-i2\pi\rho}$ operator were implemented exactly the state produced after the first step of phase estimation is a coherent superposition,

$$|0, v_j\rangle \xrightarrow{FT, C-U} \frac{1}{\sqrt{2^l}} \sum_{t\in[2^l], j\in[n]} \alpha_j e^{-i2\pi\lambda_j t} |t, v_j\rangle := |x\rangle \tag{3.25}$$

If U is replaced by U_k, we show that coherence is maintained and the output state $|\overline{x}\rangle$ is close to $|x\rangle$. The argument before equation (3.21) shows that coherence is maintained if the state of all the $k.2^l$

registers containing independent copies of ρ remains invariant after application of $e^{-i2\pi S/k}$. By the Zeno effect the probability of maintaining coherence over $2^l k$ steps is at least $1 - 4\pi^2 2^l/k \geq 1 - \epsilon/5$.

The state $|\overline{x}\rangle$ after applying the Fourier transform and a controlled U_k operation is,

$$|0, v_j\rangle \xrightarrow{FT, C-U_k} \frac{1}{\sqrt{2^l}} \sum_{t \in [2^l]} |t\rangle U_k^t |\phi\rangle := |\overline{x}\rangle \tag{3.26}$$

Coherent evolution is described by (3.18), the states $U_k^t |\phi\rangle$ are close to $U^t |\phi\rangle$ with probabilities bounded by claim 3.2.4.

$$\Pr\left[U_k^t |\phi\rangle = \sum_{j \in [n]} e^{-i2\pi\lambda_j t(1 \pm \delta_{jt})} \beta_{jt} |v_j\rangle, |\delta_{jt}| \leq \delta_j \ \ \forall \ \ t \in [2^l]\right] \geq 1 - \epsilon/5 - l \sum_{j \in [n]} e^{-2\pi\delta_j^2 \lambda_j k/3} \tag{3.27}$$

Controlled U_k^t operators for $t \in [2^l]$ are implemented by composing U_k^t for $t = 2^r, r \in [l]$, thus to ensure that $|\delta_{jt}| \leq \delta_j$ for all $t \in [2^l]$, it suffices to show that $|\delta_{jt}| \leq \delta_j$ for $t = 2^r, r \in [l]$.

Choosing δ_j such that $\lambda_j \delta_j = \frac{\epsilon}{20}$, our choice of $k = \frac{200 \log(nl/\epsilon)}{\epsilon^2} = \frac{l}{2(\lambda_j \delta_j)^2}$ can be used to bound the error probability in (3.27),

$$e^{-2\pi\delta_j^2 \lambda_j k/3} = e^{-\frac{2\pi \log(nl/\epsilon)}{6\lambda_j}} \leq e^{-\log(nl/\epsilon)} = \frac{\epsilon}{nl} \tag{3.28}$$

Thus equation (3.27) can be simplified to,

$$\Pr\left[U_k^t |\phi\rangle = \sum_{j \in [n]} e^{-i2\pi\lambda_j t(1 \pm \frac{\epsilon}{20\lambda_j})} \beta_{jt} |v_j\rangle, \forall t \in [2^l]\right] \geq 1 - 1.2\epsilon \tag{3.29}$$

The outputs of the exact and approximate phase estimation circuits are samples from $FT_{2^l}^{-1}(|x\rangle)$ and $FT_{2^l}^{-1}(|\overline{x}\rangle)$. As the Fourier transform is a unitary operator, the statistical distance between the distributions $FT_{2^l}^{-1}(|\overline{x}\rangle)$ and $FT_{2^l}^{-1}(|x\rangle)$ is equal to the squared ℓ_2 distance $||x\rangle - |\overline{x}\rangle|_2^2$ where $|x\rangle, |\overline{x}\rangle$ are the exact and approximate states defined in equations (3.25), (3.26).

In order to bound the squared ℓ_2 distance we introduce the intermediate state,

$$|x'\rangle := \frac{1}{\sqrt{2^l}} \sum_{t \in [2^l], j \in [n]} \alpha_j e^{-i2\pi\lambda_j t(1 \pm \frac{\epsilon}{20\lambda_j})} |t, v_j\rangle \tag{3.30}$$

The squared ℓ_2 distance between $|x'\rangle$ and $|\overline{x}\rangle$ is bounded by,

$$\begin{aligned}\left||x'\rangle - |\overline{x}\rangle\right|_2^2 &= \frac{1}{2^l}\sum_{t\in[2^l],j\in[n]} |\alpha_j^2 - \beta_{jt}^2| \\ &= O(1/k) = O(\epsilon^2) \end{aligned} \tag{3.31}$$

using claim 3.2.4. The squared ℓ_2 distance between $|x'\rangle$ and $|x\rangle$ is bounded by,

$$\begin{aligned}\left||x'\rangle - |x\rangle\right|_2^2 &= \frac{1}{2^l}\sum_{t\in[2^l],j\in[n]} \alpha_j^2 \left|(1 - e^{-i\frac{2\pi\epsilon t}{20}})\right|^2 \\ &\leq \frac{1}{2^l}\sum_{t\in[2^l]} \left(\frac{2\pi\epsilon t}{20}\right)^2 \\ &\leq \frac{0.1\epsilon^2}{2^l}\frac{2^{3l}}{2.95} \leq 0.035 \end{aligned} \tag{3.32}$$

The first inequality follows as $2\sin(\theta/2) < \theta$ while the last inequality follows from fact 3.1.1 for $l \geq 7$. As $l \geq 7$, it follows from equations (3.31) and (3.32) that $||\overline{x}\rangle - |x\rangle|_2^2 \leq 0.04$.

The analysis of phase estimation in claim 3.2.6 shows that $FT_{2^l}^{-1}(|x\rangle)$ outputs $\sum_{j\in[n]} \alpha_j |v_j, \overline{\lambda_j}\rangle$ where for all $j \in [n]$, $|\frac{\overline{\lambda_j}}{2^l} - \lambda_j| \leq \epsilon$ with probability at least $1 - poly(n)$. Claim 3.2.6 continues to hold for $FT_{2^l}^{-1}(|x\rangle)$ with success probability 0.76 instead of 0.8 in (3.34). The probability of maintaining coherence over $O(\log n)$ repetitions of phase estimation required for claim 3.2.6 is $1 - O(\epsilon \log n)$, the claim follows. □

The following claim follows from the analysis of phase estimation (20), it is stated in a convenient form for our analysis of algorithm 3.2.1 and is included for completeness. The proof of claim 3.2.6 continues to hold for phase estimation with approximate states generated using the simulator, this is used in the proof of claim 3.2.5.

Claim 3.2.6. *Phase estimation: Repeating phase estimation $O(\log n)$ times with input $|\phi\rangle = \sum_{j\in[n]} \alpha_j |v_j\rangle$, unitary $U = e^{-i2\pi\rho}$ and precision ϵ, and selecting the most frequent estimate yields $\sum_{j\in[n]} \alpha_j |v_j, \lambda'_j\rangle$ where $|\frac{\lambda'_j}{2^l} - \lambda_j| \leq \epsilon$ for all $j \in [n]$ with probability at least $1 - 1/poly(n)$.*

Proof. Using the linearity of phase estimation we first consider the input $|\phi\rangle = |v_j\rangle$, the intermediate state $|x_j\rangle = \sum_{t\in[2^l]} e^{-i2\pi\lambda_j t} |t, v_j\rangle$ is obtained after applying the Fourier transform and the controlled

unitary U. The probability that the phase estimation circuit outputs k for $k \in [2^l]$ is,

$$\Pr\left[FT_{2^l}^{-1}(|x_j\rangle) = k\right] = \frac{\left|\sum_{t\in[2^l]} e^{i2\pi t(\lambda_j - \frac{k}{2^l})}\right|^2}{2^{2l}}$$
$$= \frac{\left|\sum_{t\in[2^l]} \cos(2\pi t(\lambda_j - \frac{k}{2^l}))\right|^2 + \left|\sum_{t\in[2^l]} \sin(2\pi t(\lambda_j - \frac{k}{2^l}))\right|^2}{2^{2l}} \tag{3.33}$$

Suppose $\lambda'_j/2^l$ is the best estimate for λ_j so that $(\lambda_j - \frac{\lambda'_j}{2^l}) = \frac{\alpha}{2^{l+1}}$ for $\alpha \in [0,1]$. Using fact 3.1.2 for $\theta := 2\pi(\lambda_j - \frac{\lambda'_j}{2^l}) = \frac{\pi\alpha}{2^l}$ to estimate (3.33),

$$\Pr\left[FT_{2^l}^{-1}(|x_j\rangle) = \lambda'_j\right] = \frac{\sin^2(\theta 2^{l-1})}{2^{2l}\sin^2(\theta/2)}$$
$$\geq \frac{\sin^2(\pi\alpha/2)}{\pi^2\alpha^2/4} \geq \frac{4}{\pi^2} \approx 0.4 \tag{3.34}$$

The last inequality follows from the fact that the function $\sin(x)/x$ tends to 1 for $x = 0$, is decreasing in the interval $[0, \pi]$ and attains the value $2/\pi$ at $x = \pi/2$. The probability of measuring one of the end points of the interval $\lambda_j \in \frac{1}{2^l}[\lambda'_j, \lambda'_j + 1]$ is is at least $f(\alpha) + f(2-\alpha)$ where $f(x) = \frac{\sin^2(\pi\alpha/2)}{\pi^2\alpha^2/4}$, this is a decreasing function of α in the interval $[0,1]$ and attains value 0.8 for $\alpha = 1$ as illustrated in figure 3.1a. The measured value is either λ'_j or $\lambda'_j \pm 1$ with probability at least 0.8.

The most frequent measurement outcome is either λ'_j or $\lambda'_j \pm 1$, the measurement probabilities decay fast with α as illustrated in figure 3.1b. The gap δ in the statement of fact 3.1.3 can therefore be taken to be a constant and repeating the process $O(\log n)$ times yields an estimate $|\frac{\lambda'_j}{2^l} - \lambda_j| \leq \epsilon$ with probability $1 - 1/n^2$. From linearity of phase estimation and the union bound it follows that for input $|\phi\rangle$ the output is $\sum_{j\in[n]} \alpha_j |v_j, \lambda'_j\rangle$ where $|\frac{\lambda'_j}{2^l} - \lambda_j| \leq \epsilon$ for all $j \in [n]$ with probability $1 - 1/n$.

□

The proof of theorem 3.2.1 follows easily from claim 3.2.5, specialized to the case $|\phi\rangle = |v_j\rangle$. The number of independent copies of ρ used for 3.2.1 is $\widetilde{O}(k2^l) = \widetilde{O}(1/\epsilon^3)$.

Claim 3.2.7 shows that each iteration of the simulator in algorithm 3.2.1 can be implemented by a simple quantum circuit without using techniques for sparse Hamiltonian simulation as suggested in (23). Implementing algorithm 3.2.1 using claim 3.2.7, the running time is $\widetilde{O}(T(\rho)/\epsilon^3)$.

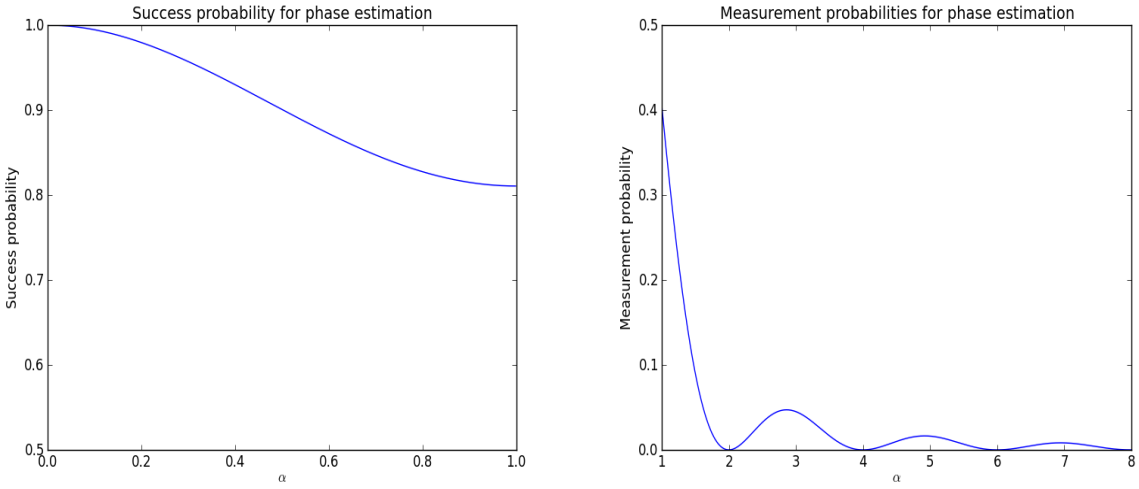


Figure 3.2. (a) Success probability for phase estimation as a function of α. (b) Decaying measurement probabilities for phase estimation.

3.2.5 Implementing spectral sampling

Recall that the simulator U_k is a quantum circuit that on input $|\phi_0\rangle$ applies the operation,

$$|\phi_{t+1}\rangle = Tr_2(e^{-i2\pi S/k}(|\phi_t\rangle \langle\phi_t| \otimes \rho)e^{i2\pi S/k}) \tag{3.35}$$

for k iterations and outputs $|\phi_k\rangle$. While the simulator can be implemented using sparse Hamiltonian simulation techniques (3; 41) as observed in (23), the special structure of the swap operator can be used to implement the simulator using a simple quantum circuit.

Claim 3.2.7. *The simulator $U_k |\psi\rangle$ for $|\psi\rangle \in \mathbb{C}^n \otimes \mathbb{C}^n$ can be implemented in time $O(kT(\rho))$.*

Proof. The swap operator S has eigenvalues ± 1, the $+1$ eigenspace consists of vectors of the form $|ab\rangle + |ba\rangle, |aa\rangle, a, b \in [n]$ while the -1 eigenspace consists of vectors of the form $|ab\rangle - |ba\rangle, a, b \in [n]$. Let $|\psi\rangle = \alpha |\psi_+\rangle + \beta |\psi_-\rangle$ be the decomposition of the input state in the spectral basis of the swap operator, $e^{-i2\pi S/k} |\psi\rangle$ can be obtained using the following sequence of operations,

$$\begin{aligned}
\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle &\xrightarrow{C-Swap} \frac{\alpha}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi_+\rangle + \frac{\beta}{\sqrt{2}}(|0\rangle - |1\rangle)|\psi_-\rangle \\
&\xrightarrow{H,C-Phase} \frac{\alpha}{\sqrt{2}}|0\rangle e^{-i2\pi/k}|\psi_+\rangle + \frac{\beta}{\sqrt{2}}|1\rangle e^{i2\pi/k}|\psi_-\rangle \\
&\xrightarrow{H,C-Swap} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)(\alpha e^{-i2\pi/k}|\psi_+\rangle + \beta e^{i2\pi/k}|\psi_-\rangle) \\
&\xrightarrow{H} |0\rangle e^{-i2\pi S/k}|\psi\rangle
\end{aligned} \tag{3.36}$$

A single iteration (3.35) of the simulator can be implemented in time $O(T(\rho))$, thus U_k can be implemented in time $O(kT(\rho))$. □

Difference of density matrices

A weighted sum of density matrices $\mu\rho + (1-\mu)\sigma$ is a density matrix and can be simulated using algorithm 3.2.1. The weighted difference of density matrices $\mu\rho - (1-\mu)\sigma$ is not positive semidefinite, but a variant of algorithm 3.2.1 can be used to simulate it given oracles for producing ρ, σ. The modified algorithm is the following: with probability μ perform operation $|\phi_{t+1}\rangle = Tr_2(e^{-i2\pi S/k}(|\phi_t\rangle\langle\phi_t| \otimes \rho)e^{i2\pi S/k})$, with probability $1-\mu$ perform operation $|\phi_{t+1}\rangle = Tr_2(e^{i2\pi S/k}(|\phi_t\rangle\langle\phi_t| \otimes \sigma)e^{-i2\pi S/k})$. The operations are implemented using claim 3.2.7 and correctness follows from the analysis of algorithm 3.2.1.

3.3 Quantum singular value estimation

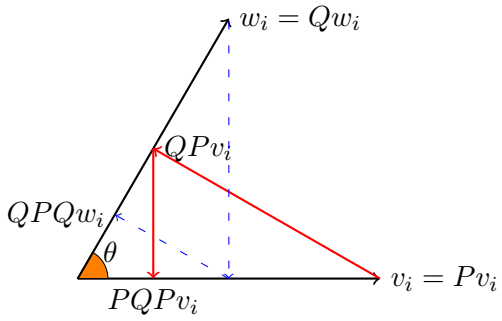
In this section, we present another algorithm 3.3.1 for quantum singular value estimation for $M \in \mathbb{R}^{m\times n}$ with $\|M\|_F = 1$ stored in the augmented $QRAM$. The normalization $\|M\|_F = 1$ is analogous to using $\rho = M^tM/Tr(M^tM)$ in algorithm 3.2.1, both algorithms rely on phase estimation and achieve a relative error $\pm\epsilon\|M\|_F$ for singular value estimation.

The running time for algorithm 3.3.1 is $\widetilde{O}(1/\epsilon)$ improving upon the $\widetilde{O}(1/\epsilon^3)$ time required for algorithm 3.2.1. The algorithm is applicable for all $M \in \mathbb{R}^{m\times n}$ stored in the augmented $QRAM$. Quantum singular value estimation can be used to implement spectral sampling and projection onto the column space of M, applications of the algorithm to linear algebra and machine learning are described in chapters 4 and 5. Algorithm 3.3.1 is preferred over algorithm 3.2.1 for most applications as it is faster and simpler to implement.

The main idea used in algorithm 3.3.1 is the connection between singular values of M and principal angles between certain subspaces associated with M. The connection is established using Jordan's lemma that describes the relation between the eigenspaces of two projectors, and is discussed in section 3.3.2.

Singular value estimation thus reduces to estimating the principal angles between subspaces associated with M. In section 3.3.3 we show that reflections in these subspaces can be implemented efficiently using the augmented $QRAM$. Phase estimation applied to a product of these reflections is used to estimate the angles between the subspaces, and thus the singular values of M in algorithm 3.3.1.

The ideas used for the quantum singular value estimation algorithm are well known in the quantum computing literature, variants of Jordan's lemma has been used in (39; 27) while using the product of reflection operators to estimate angles is the well known amplitude estimation algorithm (4). We also note that Wang (40) recently used these ideas to estimate effective resistances for graphs.



The two dimensional invariant subspaces in Jordan's lemma are spanned by eigenvectors (v_i, w_i) for PQP, QPQ with eigenvalue $\lambda_i = \cos^2(\theta)$. The projectors P and Q map back and forth between v_i and w_i, the angle θ is the principal angle corresponding to this subspace.

Figure 3.3. The two dimensional invariant subspaces in Jordan's lemma.

3.3.1 Jordan's lemma

Jordan's lemma 3.3.1 shows that any two projectors $P, Q \in \mathbb{R}^{d\times d}$ can be simultaneously block diagonalized with blocks of dimension at most 2. It is a fundamental fact in linear algebra and was first discovered by Jordan in 1875 (18).

The lemma has been rediscovered several times and has found numerous applications in the quantum computing literature including gap amplification for QMA (27), accelerating quantum walks (39), verification of quantum computers (34) and the design of span programs (1). In the linear algebra literature, Jordan's lemma is used to define principal angles between subspaces. Principal angles have recently been used for analyzing the power method for computing the largest k singular values (14) and in statistical techniques like canonical correlation analysis (17).

Lemma 3.3.1. ***Jordan's lemma (18)*** *If $P, Q \in \mathbb{R}^{d\times d}$ are projectors, then there is a decomposition of $\mathbb{R}^d$ into a direct sum of subspaces of dimension at most 2 that are invariant under P and Q.*

We require an explicit description of the two dimensional subspaces in the decomposition guaranteed by Jordan's lemma. The following claim reveals the structure of the two dimensional invariant subspaces in Jordan's lemma, the invariant subspaces are geometric quantities and are independent of the bases used for representing P and Q.

Lemma 3.3.2. *If $P, Q \in \mathbb{R}^{d\times d}$ are projectors onto subspaces $\mathcal{P}, \mathcal{Q}$ and $PQP = \sum_{\lambda_i>0} \lambda_i v_i v_i^t$ is the spectral decomposition for PQP, then $w_i := Qv_i/|Qv_i|_2$ is an eigenvector for QPQ with eigenvalue λ_i and the subspace $Span(v_i, w_i)$ is invariant under the action of P and Q.*

Proof. The matrix PQP is positive semidefinite as it can be factorized as $(PQ)(PQ)^t$, thus all the eigenvalues $\lambda_i \geq 0$. Let k be the number of non zero eigenvalues (rank) of PQP. If $PQPv_i = \lambda_i v_i$

and $\lambda_i \neq 0$, then left multiplying by P shows that $v_i = Pv_i$, that is v_i is in the range of P. There is a bijection between non zero eigenvectors of PQP and QPQ, for all $i \in [k]$,

$$QPQ(Qv_i) = QPQv_i = QPQ(Pv_i) = Q(PQPv_i) = \lambda_i Qv_i \tag{3.37}$$

thus Qv_i is an eigenvector for QPQ with eigenvalue λ_i, the eigenvector $w_i = Qv_i/|Qv_i|_2$ is the unit vector in direction Qv_i.

The subspace $Span(v_i, w_i) = Span(v_i, Qv_i)$ is invariant under the action of P and Q as $Pv_i = v_i, QQv_i = Qv_i$, and $PQv_i = PQPv_i = \lambda_i v_i$. The action of P and Q on $Span(v_i, w_i)$ is illustrated in figure 3.3.2. Note that these subspace can be one dimensional if $v_i \in \mathcal{P} \cap \mathcal{Q}$ and are two dimensional otherwise.

□

The principal/canonical angles between subspaces $\mathcal{P}, \mathcal{Q} \in \mathbb{R}^d$ are geometric quantities that quantify the overlap between the subspaces, they are defined recursively as follows in the linear algebra literature.

Definition 3.3.3. *Given subspaces $\mathcal{P}, \mathcal{Q} \in \mathbb{R}^d$ there is a sequence of principal vectors $v_i \in \mathcal{P}, w_i \in \mathcal{Q}, i \in [k]$ and angles θ_i such that,*

$$\cos(\theta_i) = \max_{v \in \mathcal{P}, w \in \mathcal{Q}} (\langle v|w\rangle \mid |v| = |w| = 1, v \perp v_j, w \perp w_j, j \in [i-1]) \tag{3.38}$$

Note that the above definition is basis independent, thus the principal angles are geometric quantities that do not depend on the bases chosen for $\mathcal{P}, \mathcal{Q}$. Jordan's lemma yields an algorithm for computing the principal vectors and angles, given orthonormal bases for $\mathcal{P}, \mathcal{Q}$.

Definition 3.3.4. *The pairs of principal vectors for subspaces $\mathcal{P}, \mathcal{Q}$ are $(v_i, Qv_i), i \in [k]$ where v_i are as in claim* 3.3.2. *The principal angles are,*

$$\cos(\theta_i) = \frac{\langle v_i|Qv_i\rangle}{|Qv_i|} = \sqrt{\langle v_i|PQPv_i\rangle} = \sqrt{\lambda_i} \tag{3.39}$$

The equivalence of the two notions of principal angles in definitions 3.3.4 and 3.3.3 will be clear from claim 3.3.6 relating principal angles to the singular value decomposition.

3.3.2 Singular values and principal angles

In this section, we show that the singular values of $M \in \mathbb{R}^{m\times n}$ with $\|M\|_F = 1$ are equal to the principal angles between sub-spaces $\mathcal{P}, \mathcal{Q} \in \mathbb{R}^{mn}$ associated with M. The subspaces $\mathcal{P}, \mathcal{Q}$ are defined to be $Col(A), Col(B)$, where $M = A^tB$ is a factorization for M given by the following claim.

Claim 3.3.5. *For $M \in \mathbb{R}^{m\times n}$ with $\|M\|_F = 1$, there is a factorization $M = A^tB$ with $A \in \mathbb{R}^{mn\times m}, B \in \mathbb{R}^{mn\times n}$ such that $A^tA = I_m$ and $B^tB = I_n$.*

Proof. Let $p \in \mathbb{R}^m$ be the vector with coordinates $p_i = |m_i|$, note that p is a unit vector as $\|M\|_F = 1$. Define $A \in \mathbb{R}^{mn\times m}$ to have column vectors $a^i = |i, m_i\rangle$ for $i \in [m]$ and $B \in \mathbb{R}^{mn\times n}$ have column vectors $b^j = |p, j\rangle$ for $j \in [n]$. The columns of A and B are orthonormal by definition and thus $A^tA = I_m$ and $B^tB = I_n$. The factorization of $M = A^tB$ follows as,

$$(A^tB)_{ij} = \langle i, m_i|p, j\rangle = |m_i|\frac{m_{ij}}{|m_i|} = m_{ij} \tag{3.40}$$

□

The following claim shows that the singular values of M are equal to the principal angles between subspaces $\mathcal{P}, \mathcal{Q}$, thus reducing singular value estimation to principal angle estimation.

Claim 3.3.6. *If θ_i are the principal angles between $\mathcal{P} = Col(A), \mathcal{Q} = Col(B)$, A and B have orthonormal columns and $M = A^tB$ has singular value decomposition $M = \sum_i \sigma_i u_i v_i^t$, then $\cos(\theta_i) = \sigma_i$.*

Proof. As the columns of A are orthonormal, the projector P onto $Col(A)$ is,

$$P = AA^t = \sum_{i\in[m]} |a^i\rangle \langle a^i| \tag{3.41}$$

Similarly $Q = BB^t$ is the projector onto the $Col(B)$. Let $M = A^tB = \sum_i \sigma_i u_i v_i^t$ be the singular value decomposition for M. The principal angles between $Col(A)$ and $Col(B)$ can be computed in terms of the spectrum of PQP using claim 3.3.2. The eigenvectors of PQP are Au_i with eigenvalues σ_i^2,

$$PQPAu_i = AMM^tA^tAu_i = AMM^tu_i = \sigma_i^2 Au_i \tag{3.42}$$

where $A^tA = I$ by the orthonormality of the columns of A. The eigenvectors of QPQ are $QAu_i = BM^tu_i = \sigma_i Bv_i$ with eigenvalues σ_i^2. The principal vector pairs are (Au_i, Bv_i) and the principal angles are $\cos(\theta_i) = \sqrt{\sigma_i^2} = \sigma_i$ by definition (3.3.4).

□

3.3.3 Singular value estimation

The singular value estimation algorithm 3.3.1 uses the augmented $QRAM$ to implement the reflections $(2P-I)$ and $(2Q-I)$ about the subspaces $\mathcal{P}, \mathcal{Q}$, the product of the reflections $U = (2P-I)(2Q-I)$ is a unitary operator. The two dimensional subspaces of principal vectors $Span(Au_i, Bv_i)$ are invariant under the action of U, with U acting as a clockwise rotation by an angle of $2\theta_i$ on these subspaces. Phase estimation with unitary U and precision ϵ on input $|Au_i\rangle$ produces an estimate $|\overline{\theta_i}| \in [2\theta_i \pm \epsilon]$, the singular value σ_i for M is estimated as $\overline{\sigma_i} = \cos(\overline{\theta_i})$.

Given $M \in \mathbb{R}^{m\times n}$ stored in an augmented $QRAM$, the following claim shows that multiplications $|x\rangle \to |Ax\rangle$ and the reflections $2P - I$ and $2Q - I$ can be implemented efficiently.

Claim 3.3.7. *If $M \in \mathbb{R}^{m\times n}$ is stored in an augmented $QRAM$ and has factorization $M = A^tB$ as in claim* 3.3.5 *then multiplications $|x\rangle \to |Ax\rangle, |Bx\rangle$ and reflections in $Col(A)$ and $Col(B)$ can be implemented in time $\widetilde{O}(1)$.*

Proof. The columns of A and B are orthonormal, so $AA^t = I_n$ and $BB^t = I_m$. The columns of A are $|a^i\rangle = |i, m_i\rangle$ for $i \in [m]$ can be prepared using the m_i stored in the augmented $QRAM$. Multiplication by A can be implemented as a unitary using algorithm 2.3.2 to query the augmented $QRAM$ in superposition as in (2.21),

$$|x, 0^{\log n}\rangle \to \sum_{i\in[m]} x_i |i, m_i\rangle = |Ax\rangle \tag{3.43}$$

The reflection in $Col(A)$ is implemented as UR_0U^{-1} where U is the unitary implementing multiplication by A and R_0 is the reflection in the state $|0\rangle^{\log m}$. Multiplication by B is simpler, it maps $|0, x\rangle \to |p, x\rangle$ independent of $|x\rangle$ using the augmented $QRAM$ 2.3.2 to prepare $|p\rangle$ in an auxiliary register, reflection in $Col(B)$ is UR_0U^{-1} where U is the unitary operator for multiplication by B. □

Algorithm 3.3.1 Quantum singular value estimation

Require: $M \in \mathbb{R}^{m \times n}$ with $\|M\|_F = 1$ stored in augmented $QRAM$, singular vector $|u_i\rangle \in \mathbb{R}^m$, M has factorization $M = A^t B$ as in claim 3.3.5.

1: Append an auxiliary register and use claim 3.3.7 to obtain $|Au_i\rangle$.

2: Perform phase estimation to additive error ϵ with input $|Au_i\rangle$ and unitary operator $U = (2P - I)(2Q - I)$ where P and Q are projections onto $Col(A), Col(B)$ to obtain $|Au_i, \overline{2\theta_i}\rangle$.

U^k is implemented using the augmented $QRAM$ as in claim 3.3.7.

3: Repeat step 2 $O(\log n)$ times and select the most frequent estimate for $\overline{2\theta_i}$.

4: Output $\overline{\sigma_i} = \cos(\overline{\theta_i})$ as an estimate for σ.

Phase estimation guarantees that $|\overline{\theta_i} - \theta_i| \leq \epsilon$ as $\sigma_i = \cos(\theta)$ the error in estimating σ_i is,

$$|\overline{\sigma_i} - \sigma_i| \leq \sin(\theta_i \pm \epsilon)|\overline{\theta_i} - \theta_i| \leq \epsilon\sqrt{1 - \sigma_i^2} \leq \epsilon \tag{3.44}$$

Algorithm 3.3.1 therefore produces an additive error ϵ estimate of the singular value in time $O(1/\epsilon)$. The algorithm is implemented using unitary operators, it is therefore coherent and extends to superpositions over singular vectors. Success probability for step 2 of algorithm 3.3.1 is 0.8 by the analysis of phase estimation, it is boosted to $1-1/poly(n)$ by repeating $O(\log n)$ times and choosing the most frequent estimate. We have proved the following theorem,

Theorem 3.3.8. *For $M \in \mathbb{R}^{m \times n}$ with $\|M\|_F = 1$ stored in the augmented $QRAM$, algorithm 3.3.1 has running time $\widetilde{O}(1/\epsilon)$ and outputs $|u_i, \overline{\sigma_i}\rangle$ with $\overline{\sigma_i} \in [\sigma_i \pm \epsilon]$ with probability at least $1-1/poly(n)$.*

Algorithm 3.3.1 improves upon the running time of algorithm 3.2.1 and is preferred over it for most applications. Spectral sampling can be implemented using singular value estimation as discussed earlier. Another important application of singular value estimation is the computation of projections onto the column space of M which we describe next.

3.3.4 Quantum projections

The projection of a state $|v\rangle$ onto the column space $Col(M)$ is MM^+v, algorithm 3.3.2 computes the quantum state $|MM^+v\rangle$ corresponding to the projection and estimates the length of the projection. The projection is computed using quantum singular value estimation and amplitude amplification, the length of the projection is estimated using amplitude estimation. The running time for the quantum projection algorithm 3.3.2 is $\widetilde{O}(1/\sigma_{min}|MM^+v|)$, polynomial in the spectral gap and the length of the projection.

Algorithm 3.3.2 Quantum projection onto $Col(M)$.

Require: $(M \in \mathbb{R}^{m\times n}, v \in \mathbb{R}^m, \|M\|_F = 1)$ stored in $QRAM$, outputs $|MM^+v\rangle$ and additive error $\delta|MM^+v|^2$ estimate for $|MM^+v|^2$, σ_{min} is the smallest non zero singular value for M.

1: Reflection in $Col(M)$: To reflect $|\phi\rangle = \sum_i \alpha_i |u_i\rangle$ in $Col(M)$ use quantum singular value estimation 3.3.1 with precision $\epsilon = O(\sigma_{min}(M))$ to obtain $\sum_{i\in[m]} b_i |v_i, \overline{\sigma_i}\rangle$.

If $\overline{\sigma_i} \neq 0$, apply a phase flip to register 1 and erase register 2.

2: Obtain $|MM^+v\rangle$ using amplitude amplification 2.1.1 performing reflection in $|v\rangle$ using the augmented $QRAM$ and reflection in $Col(M)$ as in step 1.

3: Estimate $|MM^+v|^2$ to additive error $\delta|MM^+v|^2$ using amplitude estimation 2.1.1 implementing reflections as in step 2.

The algorithm succeeds if all the reflections in $Col(M)$ are implemented correctly, by theorem 3.3.8 the reflections are correct with probability $1 - 1/poly(n)$. The number of iterations required for amplitude amplification is $O(1/|MM^+v|)$ while the number of iterations for amplitude estimation is $O(1/|MM^+v|\delta)$, yielding the following running time guarantees for algorithm 3.3.2.

Theorem 3.3.9. *Algorithm* 3.3.2 *outputs* $|MM^+v\rangle$ *in time* $\widetilde{O}(1/\sigma_{min}|MM^+v|)$ *and an additive error* $\delta|MM^+v|^2$ *estimate for* $|MM^+v|^2$ *in time* $\widetilde{O}(1/\sigma_{min}|MM^+v|\delta)$ *with probability* $1 - O(1/|MM^+v|\delta poly(n))$.

The running time of the quantum projection algorithm is depends on the spectral gap $O(1/\sigma_{min})$ and the length of the projection $|MM^+v|$. The spectral gap is controlled using ℓ_2 regularization as in chapter 5 while relative error approximations of the length of the projection are used for estimating leverage scores in chapter 4.

Chapter 4

Linear Algebra Algorithms

In this chapter, we present quantum algorithms for low rank matrix approximation based on importance sampling from the leverage score distribution for matrices $A \in \mathbb{R}^{m \times n}$ stored in the augmented $QRAM$. Low rank approximation by column sampling is an application of quantum machine learning where the output is not a quantum state but a sample from $[n]$, thus providing an elegant solution to the problem of extracting classically useful information from a quantum state.

An (approximate) sampler from the leverage score distribution can be used obtain algorithms for low rank approximation by column/row selection (6). Such an approximations is called a CX decomposition in the linear algebra literature. There are no known classical algorithms for sampling from the leverage score distribution faster than algorithms that approximate all the leverage scores, the fastest known algorithms for approximating leverage scores (25) require time polynomial in matrix dimensions. The quantum CX decomposition algorithm has running time polynomial in problem parameters as opposed to classical algorithms that are polynomial in matrix dimensions.

Importance sampling algorithms according to the leverage score distribution has found several applications in the linear algebra literature including CUR decompositions (6), approximate least squares (7) and graph sparsification (38). Importance sampling requires relative error ϵ approximations for the leverage scores, we provide a quantum algorithm that achieves a quadratic speedup for relative error approximation of the leverage scores.

Our results are summarized in table 3.1, the input to the problem is matrix $A \in \mathbb{R}^{m \times n}$ with $\|A\|_F = 1$ stored in the augmented $QRAM$, the problems are parametrized by a rank parameter k, the threshold $\tau = \sigma_k^2$ is the k-th largest singular value. The condition number $\kappa(A_k A_k^t) = \sigma_1^2/\sigma_k^2$, $\eta = \sum_{i \in [k]} \sigma_i^2$ is the fraction of the mass of the singular value spectrum in the space spanned by the largest k singular vectors while $\Delta_k = \sigma_{k+1}^2 - \sigma_k^2$ is the spectral gap at the k-th singular value. The running times of all the algorithms are polynomial in these problem parameters, the cases where this constitutes a speedup over classical algorithms will be discussed subsequently.

Problem	Quantum Algorithm	Classical Algorithm
Approximate leverage score sampler	$\widetilde{O}(\kappa(A_k A_k^t)/\eta\tau)$	$O(mn\log n)$
CX decomposition	$\widetilde{O}(\kappa(A_k A_k^t)/\tau\epsilon^2)$	$O(mn\log n + k/\epsilon^2)$
Exact leverage score sampler	$\widetilde{O}(\sqrt{m/k}/\Delta_k)$	$O(mnk)$
Relative error ϵ approximation	$\widetilde{O}(\sqrt{m/k}/\epsilon\Delta_k)$	$O(mn\log n)$
Quadratic form approximation	$\widetilde{O}(\sqrt{mk}/\epsilon^3\Delta_k)$	$O(mn\log n + k/\epsilon^2)$
Approximate least squares	$\widetilde{O}(\sqrt{mk}/\epsilon^3\Delta_k)$	$O(mn\log n + k/\epsilon^2)$
CUR decomposition	$\widetilde{O}(\sqrt{mk}/\epsilon^5\Delta_k)$	$O(mn\log n + k/\epsilon^4)$

Table 4.1. Running times for quantum and classical importance sampling algorithms.

This chapter is organized as follows: section 4.1 recalls the notion of statistical leverage scores, section 4.2 presents quantum algorithms for leverage score sampling and CX decomposition while section 4.3 presents algorithms for leverage score approximation and the other applications of importance sampling using leverage scores listed in table 4.1.

4.1 Leverage Scores

Basic linear algebra notation and the singular value decomposition were introduced in section 1.1. In this section, we recall the notion of statistical leverage scores required for the quantum algorithms for low rank approximation.

Let $A \in \mathbb{R}^{m\times n}$ have singular value decomposition $A = U\Sigma V^t$ so that $A = \sum_{i\in[r]} \sigma_i u_i v_i^t$. The optimal rank k approximation to A is $A_k = \sum_{i\in[k]} \sigma_i u_i v_i^t$, the approximation is optimal in the sense $A_k = \text{argmin}_{rank(A')=k} \|A - A'\|$ for any unitarily invariant matrix norm. The row and column leverage scores with respect to the rank k approximation for A are defined as follows,

$$\ell_k(i) = \frac{1}{k}\sum_{t\in[k]} u_{it}^2 = \frac{1}{k}|A_k A_k^+ e_i|^2$$
$$\ell^k(j) = \frac{1}{k}\sum_{t\in[k]} v_{jt}^2 = \frac{1}{k}|A_k^+ A_k e_j|^2 \tag{4.1}$$

The row leverage score $\ell_k(i)$ is the squared ℓ_2 norm of the row i of U_k, the truncation of U to the first k columns. Similarly the column leverage score $\ell^k(j)$ is the squared ℓ_2 norm of row j of V_k, the truncation of V to the first k columns. The scores are normalized so that ℓ_k and ℓ^k are probability distributions on $[m]$ and $[n]$, the leverage score distribution for A is denoted by $\ell_k(A)$.

The columns of U_k are orthonormal and span the column space of A_k, thus $k\ell_k(i)$ is the squared length of the projection of e_i onto $Col(A_k)$. Similarly, $k\ell^k(j)$ is the projection of e_j onto the $Row(A_k)$. Importance sampling from the leverage score distribution has been used to obtain algorithms for low rank matrix approximation (6), approximate least squares (7) and graph sparsification (38).

The β approximate leverage score distribution has sampling probabilities $p_i \geq \beta \ell_k(i)$. Sampling from the β approximate leverage score distribution suffices for the algorithms mentioned above, the number of samples required for relative error $(1+\epsilon)$ approximation scales as $O(k \log k/\beta\epsilon^2)$ where k is the rank parameter.

Our quantum algorithms use the interpretation of leverage scores as the squared norms of the rows of U_k for approximately sampling from the leverage score distribution and the interpretation of the leverage score as the length of the projection onto the column/row space for leverage score approximation.

4.2 Sampling based algorithms

4.2.1 Leverage score sampling

Algorithm 4.2.1 samples approximately from the leverage score distribution $\ell_k(A)$ for matrix $A \in \mathbb{R}^{m \times n}$ stored in the augmented $QRAM$. The algorithm is parametrized by a threshold parameter $\tau \in [0,1]$, it uses spectral sampling to select singular vectors having singular values greater than τ followed by a rejection sampling step to approximately sample from $\ell_k(A)$.

Algorithm 4.2.1 Approximate leverage score sampler

Require: $A \in \mathbb{R}^{m \times n}$ stored in augmented $QRAM$, $\rho = AA^t/Tr(AA^t)$, threshold $\tau \in [0,1]$ and k such that $\tau \in [\lambda_k(\rho), \lambda_{k+1}(\rho)]$.

Ensure: Samples from $\beta = 0.95\tau k$ approximate leverage score distribution $\ell_k(A)$.

1: Perform spectral sampling to obtain $|u_i, \overline{\lambda_i}\rangle$ by running algorithm 3.3.1 with input ρ and $\epsilon = \tau/20$. Discard the second register and repeat if $\overline{\lambda_i} \leq (\tau - \epsilon)$.

2: Measure the first register in the standard basis to obtain $s \in [m]$, with probability $\min(1, \tau/\overline{\lambda_i})$ output s, otherwise discard and repeat step 1.

The rank k is not a parameter for the algorithm but is specified indirectly by the choice of $\tau \in [0,1]$, the threshold τ can be chosen empirically by repeating the algorithm $O(\log n)$ times, and selecting τ so that the success probability in step 1 is a constant. The same algorithm with $\rho = A^tA/Tr(A^tA)$ yields approximate samples from the distribution $\ell^k(A)$.

Theorem 4.2.1. *Algorithm* 4.2.1 *samples from the the* $\beta = 0.95\tau k$ *approximate leverage score distribution* $\ell_k(A)$ *and has expected running time* $\widetilde{O}(\lambda_{max}(\rho)/\eta\tau^2)$ *where* $\eta = \sum_{i \in [k]} \sigma_i^2/|A|_F^2$.

Proof. Let $A = \sum_i \sigma_i u_i v_i^t$ be the SVD for A, then the input ρ to the spectral sampling algorithm

is a mixture of $|u_i\rangle$ with probability $\lambda_i(\rho) = \sigma_i^2/|A|_F^2$. The acceptance probability for step 1 is at least $\eta(1 - 1/poly(n))$ by theorem 3.3.8, while the acceptance probability for step 2 is at least $\tau/\lambda_{max}(\rho)$. The expected number of trials required for getting an output from algorithm 4.2.1 is therefore $O(\lambda_{max}(\rho)/\eta\tau)$.

Denoting the first register of the sample $|u_i, \overline{\lambda_i}\rangle$ by $|\phi\rangle$ and conditioning on $\overline{\lambda_i} \in [\lambda_i \pm \epsilon]$, the probability that algorithm 4.2.1 outputs $j \in [m]$ can be bounded by,

$$\begin{aligned}
\Pr[s = j] &\geq (1 - 1/poly(n)) \sum_{\lambda_i \geq \tau} \Pr[s = j \mid |\phi\rangle = |u_i\rangle] \Pr[|\phi\rangle = |u_i\rangle] \\
&\geq (1 - 1/poly(n)) \sum_{\lambda_i \geq \tau} \frac{\tau}{\overline{\lambda_i}} u_{ij}^2 \lambda_i \\
&\geq (1 - 1/poly(n)) \frac{20}{21} \tau k \ell_k(j) = \beta \ell_k(j) \qquad (4.2)
\end{aligned}$$

where $\beta \approx 0.95\tau k$, the last step follows as the approximation error $\epsilon = \tau/20$. Algorithm 4.2.1 therefore produces a single sample from the β approximate leverage score distribution in expected time $\widetilde{O}(\lambda_{max}(\rho)/\eta\tau^2)$. □

4.2.2 CX decomposition

Columns sampled according to the approximate column leverage score distribution $\ell^k(A)$ achieve relative error low rank approximations as demonstrated in (6). Factorizations $A = CX$ approximating A by a subset of columns are called CX decompositions in the linear algebra literature.

The CX decomposition algorithm in (6) computes $C = ASD$ where S and D are sampling and rescaling matrices for the β approximate leverage score distribution $\ell^k(A)$. The analysis establishes the guarantee $\|A - CC^+A\|_F \leq (1 + \epsilon) \|A - A_k\|_F$ with constant probability, if $k = O(k \log k/\beta\epsilon^2)$ columns are selected. As CC^+ is the projection onto the column space of A it is independent of rescaling the columns of C, and the result continues to hold for matrix $C = AS$ that samples columns from A.

Theorem 4.2.2 (CX decomposition, (6))**.** *Let $A \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{m \times c}$ be a matrix consisting of $c = O(k \log k/\beta\epsilon^2)$ columns of A where each column is sampled from a probability distribution on $[n]$ with $p_j \geq \beta \ell^k(j)$, then with probability at least 0.7,*

$$\left\|A - CC^+A\right\|_F \leq (1 + \epsilon) \|A - A_k\|_F \qquad (4.3)$$

Leverage score sampling 4.2.1 can be used to construct CX decompositions for matrix $A \in \mathbb{R}^{m\times n}$ stored in the augmented $QRAM$. The rank k for the CX decomposition is chosen so that the largest k singular vectors account for a constant fraction of the Frobenius norm, that is $\sum_{i\in[k]} \lambda_k \geq \eta$ for a constant η where λ_k are eigenvalues for $\rho = A^tA/Tr(A^tA)$.

A rank k, ϵ approximate CX approximation (4.3) for A can be constructed in time $\widetilde{O}(\lambda_{max}(\rho)k\log k/\eta\beta\epsilon^2\tau^2) = \widetilde{O}(\lambda_{max}(\rho)/\tau^3\epsilon^2)$ using the leverage score sampler, as $\beta = O(\tau k)$ by (4.2) and η is a constant. We have the following theorem.

Theorem 4.2.3. *There is an $\widetilde{O}(\kappa(A_kA_k^t)/\eta\tau^2\epsilon^2)$ time quantum algorithm that finds $C \in \mathbb{R}^{c\times n}$ consisting of $c = O(k\log k/\beta\epsilon^2)$ columns of A such that with probability at least 0.7,*

$$\|A - A_k\|_F \leq \left\|A - CC^+A\right\|_F \leq (1+\epsilon)\|A - A_k\|_F \tag{4.4}$$

Note that the running time of the CX decomposition algorithm is $\Omega(\lambda_{max}(\rho)k^3/\epsilon^2)$ as $\tau k < 1$. If τk is a small constant the running time is polynomial in the rank. Classical algorithms for computing the CX decomposition require time $SVD(A,k)$, the time required to compute the largest k singular vectors of A. The article (13) reviews the best known deterministic and randomized classical algorithms for $SVD(A,k)$ which require time $O(mnk)$ and $O(mn\log n)$.

The running time for the classical and quantum CX decomposition algorithms when the singular value spectrum exhibits a power law decay are compared in section 4.2.3. The comparison indicates that for the regime $m = O(n)$ the quantum algorithm achieves a speedup only if the singular value spectrum of A decays fast. If $m \gg n$, the quantum algorithm achieves a speedup over the classical CX decomposition algorithm for all cases.

Theorem 4.2.3 yields a fast quantum algorithm for selecting rows/columns from a large matrix that approximate the subspace A_k spanned by the first k principal components. Projection onto A_k is used in several machine learning algorithms like spectral clustering and principal components regression. The column space of C is a good approximation to A_k and can be used as a proxy for A_k for these algorithms. The quantum part of the algorithm identifies significant columns faster than any classical method, this reduces dimensions of the dataset which can then be processed classically. Interpretability is another advantage of CX decompositions, as selecting columns corresponds to identifying important features.

The CX decomposition is the only algorithm that can be implemented using the leverage score sampler alone, while importance sampling according to leverage scores has found several applications in linear algebra, these algorithms require relative estimates of the sampling probability. Using the interpretation of leverage scores as projections onto A_k, we provide quantum algorithms for relative error approximations to leverage scores in section 4.3.

4.2.3 Comparison with classical algorithms

Our algorithms for leverage score sampling and CX decomposition have running time polynomial in problem parameters η, τ and k, a priori it is not clear how these algorithms compare

to classical CX decomposition algorithms. We compare our quantum algorithm to the classical CX decomposition algorithm, for matrices where the singular law spectrum exhibits a power law decay, real world datasets often exhibit power law decay. The comparison clarifies the nature of the quantum speedup for CX decomposition.

Power law decay

Consider a matrix $M \in \mathbb{R}^{m\times n}$ whose singular values exhibit a power law decay with exponent α, that is $\sigma_k^2 \propto 1/k^\alpha$. Power law decay of the singular value spectrum has been observed empirically for several real world datasets, this can be proved rigorously for special cases, for example (29) showed that eigenvalues of graphs with a power law degree distribution follow a power law. We compute the parameters for the quantum CX decomposition algorithm where the singular value spectrum follows a power law with exponent α.

Normalizing so that the Frobenius norm $\|M\|_F = 1$ is 1 and approximating using the integral $\int_0^x t^\alpha dt = x^{\alpha+1}/(1+\alpha)$,

$$1 = \sum_{k\in[n]} \sigma_k^2 = \sum_{k\in[n]} \frac{C}{k^\alpha} = O(n^{1-\alpha}) \tag{4.5}$$

The singular values are $\sigma_k^2 = O(\frac{1}{n^{1-\alpha}k^\alpha})$, all the parameters in the running time of the CX decomposition algorithm can be computed given the singular values. The condition number $\kappa(A_kA_k^t) = \frac{\sigma_1^2}{\sigma_k^2} = k^\alpha$, the threshold $\tau = O(1/n^{1-\alpha}k^\alpha)$ and $\eta = O(k^{1-\alpha}/n^{1-\alpha})$. For η to capture a constant fraction of the mass Frobenius norm, $k = O(n^{1-\alpha})$. The running time for the leverage score sampler is,

$$\frac{\kappa(A_kA_k^t)}{\eta\tau} = k^{3\alpha-1}n^{2-2\alpha} = n^{(1-\alpha)(3\alpha+1)} \tag{4.6}$$

The CX decomposition algorithm invokes the leverage score sampler $O(k\log k/\epsilon^2)$ times and has running time $O(n^{(1-\alpha)(3\alpha+2)})$, assuming ϵ to be a constant. The spectral gap $\Delta_k = \sigma_{k+1}^2 - \sigma_k^2 = O(1/n^{1-\alpha}k^{1+\alpha})$ taking derivatives.

Figure 4.1 (a) compares the running times of the classical and quantum algorithms for power law exponents $\alpha \in [0,1]$ for $m = O(n)$. The algorithms have running time $O(n^\beta)$, the running time exponent β for is compared for different values of α. The blue line is the classical randomized algorithm that requires time $O(mn\log n) = O(n^2)$. The red curve is the exponent for the quantum leverage score sampler while the green curve is the exponent for the quantum CX decomposition algorithm. The figure indicates that the quantum algorithm achieves speedups for $\alpha > 0.4$. Note that the performance of the quantum algorithm improves if $m \gg n$, the quantum CX decomposition algorithm achieves a speed up for all values of α if $m = O(n^2)$.

The singular value spectrum of many real world datasets can be modeled by power laws, for example the exponent α for the web graph is close to 0.5 (29). We empirically compute α for for a term document matrix where documents are randomly sampled wikipedia articles and terms are commonly occurring words (with stop words removed). Figure 4.1 (b) plots $\log(\sigma_k^2)$ vs $\log k$ for

such a matrix and shows that the singular value spectrum can be modeled by a power law with $\alpha \approx 0.5$. The quantum CX decomposition has running time $O(n^{1.75})$ for $\alpha = 1/2$ and thus offers a speedup over the classical algorithm with running time $O(mn) = O(n^2)$.

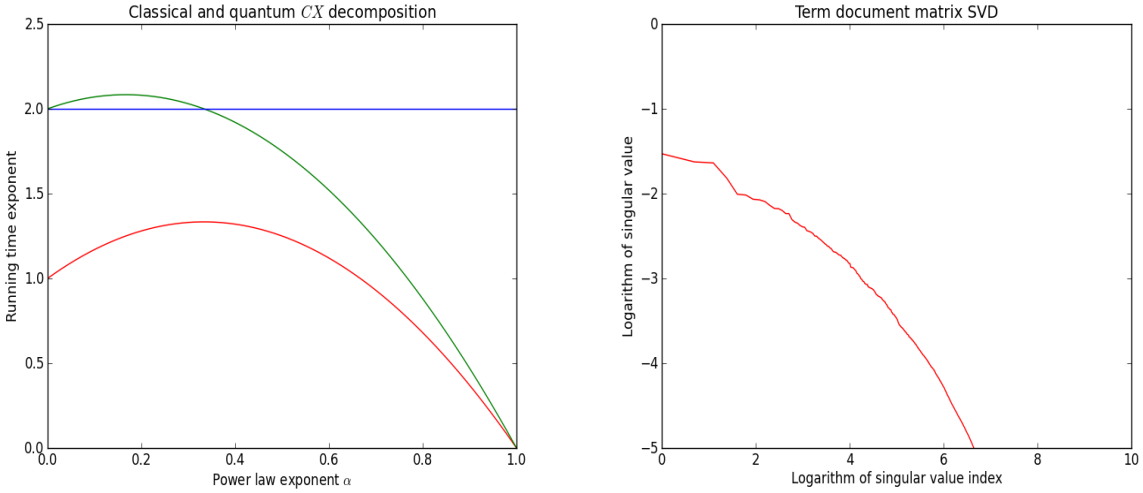


Figure 4.1. (a) Quantum and classical CX decomposition for power law decay of singular value spectrum. (b) SVD for term document matrix exhibiting power law decay.

4.3 Importance sampling

Importance sampling according to the leverage score distribution has found several applications in the linear algebra literature including approximate least squares, CUR decompositions and approximation of quadratic forms. The importance sampling algorithms sample rows from $\ell_k(A)$ and use the sampling probabilities for rescaling the sampled rows, they therefore require computing the leverage scores. In section 4.3.1, we present a quantum algorithm for sampling exactly from the leverage score distribution and approximating the leverage scores to relative error ϵ. In section 4.3.2 we show that approximate leverage scores can be used for the linear algebra applications and obtain quantum algorithms with running times as in table 4.1 for algorithms involving importance sampling from the leverage score distribution.

4.3.1 Leverage score approximation

Leverage scores can be approximated using the quantum projection algorithm 3.3.2 as $k\ell_k(i)$ is the projection of e_i onto $Col(A_k)$ by (4.1). The rank k is specified implicitly using a threshold $\tau \in [0, 1]$ as in algorithm 4.2.1. Algorithm 4.3.1 samples exactly from the distribution $\ell_k(A)$ and provides a relative error estimate for the corresponding leverage score.

Algorithm 4.3.1 Leverage score sampling and relative error approximation.

Require: $A \in \mathbb{R}^{m\times n}$ stored in augmented $QRAM$ with $\|A\|_F = 1$, $\tau \in [0,1]$ and k such that $\tau \in [\sigma_k(A), \sigma_{k+1}(A)]$, precision $\Delta_k = (\sigma_{k+1}(A) - \sigma_k(A))$ and relative error ϵ.

1: Let $v \in \mathbb{R}^m$ be a random unit vector, prepare $|v\rangle = \sum_{i\in[m]} \alpha_i |u_i, \overline{\sigma_i}\rangle$ using algorithm 3.3.1 with precision Δ_k, post-select on $\overline{\sigma_i} \geq \tau$ to obtain $|A_k A_k^+ v\rangle$, use amplitude estimation to estimate k/m within relative error ϵ.

2: Measure $|A_k A_k^+ v\rangle$ in the standard basis to obtain sample $s \sim \ell_k(A)$.

3: Estimate $k\ell_k(s)$, the squared length of the projection of e_s onto $Col(A_k)$ to relative error ϵ using amplitude estimation as in algorithm 3.3.2, divide by estimated value of k from step 1 to approximate $\ell_k(s)$.

Theorem 4.3.1. *Algorithm* 4.3.1 *samples* $s \sim \ell_k(A)$ *exactly in expected time* $\widetilde{O}(\sqrt{m/k}/\Delta_k)$ *and obtains a relative error* 2ϵ *estimate for* $\ell_k(s)$ *in expected time* $\widetilde{O}(\sqrt{m/k}/\epsilon\Delta_k)$.

Proof. A random unit vector $v \in \mathbb{R}^m$ is obtained by sampling coordinates from the normal distribution $N(0,1)$ and scaling v to have unit length. Let $v = \sum_i \alpha_i u_i$ in the left singular basis of A, the state $|v\rangle$ can be prepared in time $\widetilde{O}(1)$ if the coordinates of v are stored in a $QRAM$. The projection $|A_k A_k^+ v\rangle$ onto $Col(A_k)$ is a random vector in $Col(A_k)$, measuring $|A_k A_k^+ v\rangle$ in the standard basis samples from $\ell_k(A)$,

$$\begin{aligned}\Pr[|A_k A_k^+ v)\rangle = i] &= \frac{m}{k} E_v \left[\left(\sum_{j\in[k]} \alpha_j |u_j\rangle \right)_i^2 \right] \\ &= \frac{m}{k} E_v \left[\sum_j \alpha_j^2 u_{ji}^2 \right] = \ell_k(i) \end{aligned} \tag{4.7}$$

The m/k factor arises due to renormalization of the vector state to have norm 1 and the second equality follows as $E[\alpha_j \alpha_{j'}] = \frac{\delta_{jj'}}{m}$. The expected squared length of the projection of $|v\rangle$ onto $Col(A_k)$ is k/m, thus the expected running time for step 1 is $\widetilde{O}(\sqrt{m/k}/\Delta_k)$ for $|A_k^+ A_k v\rangle$ and $\widetilde{O}(\sqrt{m/k}/\Delta_k\epsilon)$ for estimating k to relative error ϵ. Note that step 1 is algorithm 3.3.2 with precision $\Delta_k = (\sigma_{k+1} - \sigma_k)$, the choice of precision ensures that the reflections in $Col(A_k)$ are implemented correctly.

Amplitude estimation estimates $k\ell_k(s)$ to relative error ϵ in time $O(1/\sqrt{k\ell_k(s)}\Delta_k\epsilon)$. As $i \sim \ell_k(A)$, the expected running time can be computed using,

$$E_{s\sim\ell_k(A)}[\frac{1}{\sqrt{k\ell_k(s)}}] = \sum_{i\in[m]} \frac{\ell_k(i)}{\sqrt{k\ell_k(i)}} = \frac{1}{\sqrt{k}} \sum_{i\in[m]} \sqrt{\ell_k(i)} \leq \sqrt{\frac{m}{k}} \tag{4.8}$$

The last inequality is a consequence of Cauchy Schwartz, as $\sum_i \sqrt{\ell_k(i)} \leq \sqrt{m}\sqrt{\sum_{i\in[m]} \ell_k(i)} = \sqrt{m}$, thus a relative error ϵ estimate for $k\ell_k(i)$ is obtained in time $O(\sqrt{m/k}/\Delta_k\epsilon)$. Dividing by the relative error ϵ estimate for k obtained in step 2, the leverage score $\ell_k(s)$ is approximated to relative error 2ϵ.

□

Computing the leverage scores exactly in the classical setting requires time $SVD(A,k)$ which is $O(mnk)$ and $O(mn\log n)$ for the best known deterministic and randomized algorithms (13). Relative error approximations for leverage scores can be computed in time $O(mn\log n)$ using random projections (25). The quantum approximate leverage score sampler 4.3.1 has running time $O(\sqrt{m/k}/\epsilon\Delta_k)$ and thus achieves a polynomial speedup over the classical algorithms at the expense of introducing dependence on the spectral gap. The quantum algorithm 4.3.1 is particularly useful for the exact rank k case where the spectral gap Δ_k is expected to be $1/poly(k)$.

We next show that relative error approximations of the leverage scores suffice for most of the linear algebra applications of importance sampling. The importance sampling algorithms can therefore use algorithm 4.3.1 as a black box yielding quantum algorithms for quadratic form approximation, approximate least squares and approximate least squares with running times as presented 4.1.

4.3.2 Importance sampling algorithms

Importance sampling from the leverage score distribution has several applications to linear algebra (6; 7) , these importance sampling algorithms require samples from $\ell_k(A)$ and the values of the corresponding leverage scores. The importance sampling algorithms use exact sampling probabilities, we show that the analyses can be adapted to use approximate sampling probabilities.

We first show that quadratic forms can be approximated using approximate leverage scores, followed by approximate least squares and the CUR decompositions. The proofs contained in this section are adaptations of the analyses (6; 7) using relative error approximations of leverage scores instead of exact values, they are included for completeness and the reader's convenience.

Approximating quadratic forms

Sampling $O(k\log k/\epsilon^2)$ rows from $A \in \mathbb{R}^{m\times n}$ according to $\ell_k(A)$ and rescaling the rows using relative error ϵ estimates for $\ell_k(i)$ suffices to approximate the quadratic form $x^t A_k A_k^t x$ within error

$1 \pm 2\epsilon$. Approximation of the quadratic form will subsequently be used to obtain results about approximate least squares and CUR decompositions. The approximation relies on Rudelson's sampling lemma,

Lemma 4.3.2. *[Rudelson's sampling lemma (35)] Let $Y = \{y_1, y_2, \cdots, y_m\}$ be a set of vectors in $\mathbb{R}^n$ such that $|y_i|_2 \leq M$ and w be a distribution on Y such that $\left\|E_{y\sim w}[yy^t]\right\|_2 \leq 1$. If $w_i, i \in [d]$ are independent samples from w then there is an absolute constant C such that,*

$$E\left[\left\|\frac{1}{d}\sum w_i w_i^t - E_{y\sim w}[yy^t]\right\|_2\right] \leq MC\sqrt{\frac{\log d}{d}} \tag{4.9}$$

Further, a concentration bound holds,

$$\Pr\left[\left\|\frac{1}{d}\sum w_i w_i^t - E_{y\sim w}[yy^t]\right\|_2 \geq t\right] \leq 2e^{-t^2 d/M^2C^2\log d} \tag{4.10}$$

Rudelson's sampling lemma can be used the quadratic form $x^t A_k A_k^t x$, the next claim is an adaptation of the argument in (38) for the case of low rank approximation, using approximate leverage scores instead of exact effective resistances.

Claim 4.3.3. *Let A_k be the optimal rank k approximation for $A \in \mathbb{R}^{m\times n}$ and Π be the projector onto $Col(A_k)$. If S is a diagonal matrix with $S_{ii} = \frac{n_i}{dp_i}$ where $p_i \in (1\pm\epsilon)\ell_k(i)$ and n_i is the number of times i is sampled over $d = O(k\log k\log m/\epsilon^2)$ draws from $\ell_k(A)$, then with probability at least $1 - 1/poly(m)$,*

$$\left\|A_k^t S A_k - A_k^t A_k\right\|_2 = \|\Pi S\Pi - \Pi\Pi\|_2 \leq 2\epsilon \tag{4.11}$$

Proof. It suffices to establish a spectral norm bound of the form $\|\Pi S\Pi - \Pi\Pi\|_2 \leq 2\epsilon$, as $x^t A_k^t A_k x$ can be written as $y^t y$ where $y = A_k x \in Col(A_k)$.

$$\frac{x^t A_k S A_k^t x}{x^t A_k A_k^t x} \in (1\pm 2\epsilon) \Leftrightarrow \frac{y^t \Pi S \Pi y}{y^t \Pi\Pi y} \in (1\pm 2\epsilon) \tag{4.12}$$

Note that $|\pi_i|^2 = k\ell_k(i)$ as Π is a symmetric matrix and the column π^i is the projection of e_i onto A_k . The spectral norm bound is established using Rudelson's sampling lemma to $Y = \left\{y_i = \frac{\pi^i}{\sqrt{\ell_k(i)}}, i \in [m]\right\}$ the set of columns of Π normalized to have length $\sqrt{k}$, let w be the

distribution on Y such that $\Pr_w[y_i] = \ell_k(i)$ and $w_i, i \in [d]$ be samples from w,

$$\Pi S \Pi = \sum_{i \in [m]} \frac{n_i}{dp_i} \pi_i \pi_i^t = \frac{(1 \pm \epsilon)}{d} \sum_{i \in [d]} w_i w_i^t$$
$$\Pi \Pi = \sum_{i \in [m]} \pi_i^t \pi = E_{y \sim w}[yy^t] \tag{4.13}$$

Applying Rudelson's sampling lemma 4.3.2 with $M = \sqrt{k}$,

$$E[\|\Pi S \Pi - \Pi \Pi\|_2] \leq O\left(\sqrt{\frac{k \log d}{d}}\right) + \epsilon \left\| \frac{1}{d} \sum_{i \in [d]} w_i w_i^t \right\|_2 \tag{4.14}$$

The concentration bound (4.10) shows that for $d = O(k \log k \log m/\epsilon^2)$ the spectral norm $\|\Pi S \Pi - \Pi \Pi\|_2 \leq 2\epsilon$ with probability at least $1 - 1/poly(m)$ and the claim follows. □

In addition to the above result, approximating least squares by importance sampling according to leverage scores requires that the sampling matrix S preserve matrix vector products, this will be follow from the following result about approximate matrix multiplication from (5),

Theorem 4.3.4. *(5) If $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{m \times p}$, p is a distribution on $[n]$ with $p_i \geq \beta \frac{|a^i|^2}{|A|_F^2}$ and $S \in \mathbb{R}^{m \times m}$ is a diagonal matrix with $S_{ii} = \frac{n_i}{dp_i}$ where n_i is the number of times i is sampled in d draws from p,*

$$E[\|AB - ASB\|_F^2] \leq \frac{1}{\beta d} \|A\|_F^2 \|B\|_F^2 \tag{4.15}$$

Approximate least squares

The least squares problem $Ax = b$ where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ has solution given by,

$$x_{opt} = \text{argmin}_{x \in \mathbb{R}^n} |Ax - b|_2 = A^+ b \tag{4.16}$$

Consider the sketched least squares problem $Z^t Ax = Z^t b$, where the sketched matrix $Z^t A \in \mathbb{R}^{r \times n}$ is obtained by sampling and rescaling the rows of A. Let $s_j \sim \ell_k(A), j \in [r]$ be samples from the leverage score distribution and $p_i \in (1 \pm \epsilon)\ell_k(i), i \in [m]$ be relative error estimates for the leverage scores, the entries of $Z \in \mathbb{R}^{m \times r}$ are,

$$Z_{ij} = \frac{\delta_{is_j}}{\sqrt{rp_i}} \tag{4.17}$$

Note that Z can be constructed using the quantum algorithm 4.3.1 to sample and approximate leverage scores. The matrix Z can be factorized as $Z = DS$ where $S \in \mathbb{R}^{m \times r}$ is a sampling matrix

with $S_{ij} = \delta_{is_j}$ and $D \in \mathbb{R}^{m\times m}$ is a diagonal rescaling matrix with entries $D_{ii} = \frac{1}{\sqrt{rp_i}}$. The solution to the sketched least squares problem is,

$$\overline{x_{opt}} = \text{argmin}_{x\in\mathbb{R}^n}|Z^tAx - Z^tb|_2 = (Z^tA)^+Z^tb \tag{4.18}$$

The sketched problem can be solved in time $poly(r, n)$ instead of $poly(m, n)$ time required for the original problem, the following claim shows that with high probability the solution to the sketched problem approximates the least squares solution (4.16).

Claim 4.3.5. *Let $Ax = b$ be a least squares problem with $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ such that $rank(A) = k$, let $r = O(k \log k/\epsilon^2)$ and $Z \in \mathbb{R}^{m\times r}$ be a sketching matrix as in* (4.17)*, then with probability at least* 0.75*,*

$$|A\overline{x_{opt}} - b|_2 \leq (1 + 7\epsilon)|Ax_{opt} - b|_2 \tag{4.19}$$

Proof. It suffices to show that $|A(x_{opt} - \overline{x_{opt}})|_2 \leq 7\epsilon|Ax_{opt} - b|_2$ as by the triangle inequality,

$$|A\overline{x_{opt}} - b|_2 \leq |Ax_{opt} - b|_2 + |A(x_{opt} - \overline{x_{opt}})|_2 \tag{4.20}$$

Let $A = U\Sigma V^t$ be the singular value decomposition for A and $b = Ax_{opt}+b^\perp$ where $b^\perp$ is orthogonal to $Col(A)$, the sketched least squares problem (4.18) is equivalent to,

$$\overline{x_{opt}} = \text{argmin}_{x\in\mathbb{R}^n}|Z^tA(x - x_{opt}) - Z^tb^\perp| \tag{4.21}$$

As $Col(U) = Col(A)$, the above least squares problem can be reformulated as an optimization problem over $\mathbb{R}^k$,

$$y_{opt} = \text{argmin}_{y\in\mathbb{R}^k}|Z^tUy - Z^tb^\perp| = (Z^tU)^+Z^tb^\perp \tag{4.22}$$

The equivalence of the two formulations (4.21) and (4.22) of the sketched least squares problem implies that $A(\overline{x_{opt}} - x_{opt}) = Uy_{opt}$. Multiplication by U acts as an isometry, that is $|Uy_{opt}| = |y_{opt}|$ as columns of $U^tU = I_k$,

$$\begin{aligned}
|A(\overline{x_{opt}} - x_{opt})|_2 &= |Uy_{opt}|_2 = |y_{opt}|_2 \\
&= |(Z^tU)^+Z^tb^\perp|_2 \\
&\leq |U^tZZ^tb^\perp|_2 + \left\|(Z^tU)^+ - U^tZ)\right\|_2 |Z^tb^\perp|_2
\end{aligned} \tag{4.23}$$

The terms $|Z^t b^\perp|_2$, $|UZ^t Z b^\perp|_2$ and $\left\|(Z^t U)^+ - UZ^t\right\|_2$ are bounded separately to complete the proof. The expected value for $|Z^t x|_2$ for $x \in \mathbb{R}^m$ can be computed using the entries of the sampling matrix Z (4.17),

$$\begin{aligned} E[|Z^t x|_2^2] &= E\left[\sum_{j\in[r]} ((z^j)^t . x)^2\right] \\ &= r \sum_{i\in[m]} \frac{\ell_k(i) x_i^2}{r p_i} \in (1 \pm \epsilon)|x|_2^2 \end{aligned} \tag{4.24}$$

By Markov's inequality,

$$\Pr\left[|Z^t b^\perp|_2 \leq \frac{(1+\epsilon/2)|b^\perp|_2}{\sqrt{\delta}}\right] \geq 1 - \delta \tag{4.25}$$

The term $|U^t Z Z^t b^\perp|_2$ is bounded by invoking theorem 4.3.4 with $A = U^t$ and $B = b^\perp$. Note that the distribution p in the statement of 4.3.4 is the $(1-\epsilon)$ approximate leverage score distribution, as $p_i \geq (1-\epsilon)\ell_k(i)$,

$$\begin{aligned} E[|U^t b^\perp - U^t Z Z^\perp b^\perp|_2^2] &= E[|U^t Z Z^\perp b^\perp|_2^2] \\ &\leq \frac{1}{(1-\epsilon)r} \left\|U^t\right\|_F^2 |b^\perp|_2 \\ &\leq (1+\epsilon)\frac{k}{r}|b^\perp|^2 \leq \epsilon^2(1+\epsilon)|b^\perp|^2 \end{aligned} \tag{4.26}$$

where the first equality uses $U^t b^\perp = 0$ and the last inequality follows as $r = O(k \log k/\epsilon^2)$. An application of Markov's inequality yields,

$$\Pr\left[|U^t Z Z^\perp b^\perp|_2 \leq \frac{\epsilon(1+\epsilon/2)}{\sqrt{\delta}}|b^\perp|_2\right] \geq 1 - \delta \tag{4.27}$$

In order to bound $\left\|(Z^t U)^+ - UZ^t\right\|_2$, it suffices to show that the minimum singular values of $U^t Z$ is at least $1-\epsilon$. Let $\Pi = UU^t$ be the projector onto the column space of U,

$$\begin{aligned} \sigma_{min}^2(U^t Z) &= \lambda_{min}(U^t Z Z^t U) = \lambda_{min}((U^t U)U^t Z Z^t U)) \\ &= \lambda_{min}(UU^t Z Z^t U U^t) = \lambda_{min}(\Pi Z Z^t \Pi) \end{aligned} \tag{4.28}$$

The rank of $Z^t U$ is at most k as it is obtained by sampling from rows of a rank k matrix. Claim 4.3.3 shows that with probability at least $1 - 1/poly(m)$, $Z^t U$ has rank k and establishes a bound

on the $\left\|(Z^tU)\right\|_2^2$,

$$\left\|\Pi ZZ^t\Pi - \Pi\Pi\right\|_2 \leq 2\epsilon \Rightarrow \lambda_{min}(\Pi ZZ^t\Pi) \leq 1 - 2\epsilon \tag{4.29}$$

Thus $\sigma_{min}(Z^tU) \leq \sqrt{1-2\epsilon} \leq 1 - \epsilon$ and the spectral norm $\left\|(Z^tU)^+ - UZ^t\right\|_2$ can be bounded as,

$$\left\|(Z^tU)^+ - UZ^t\right\|_2 = \frac{1}{\sigma_{min}(UZ^t)} - \sigma_{min}(UZ^t) \leq 2\epsilon \tag{4.30}$$

Choosing $\delta = 0.25$ so that $1/\sqrt{\delta} = 2$ and plugging in the bounds (4.25),(4.27),(4.30) in equation (4.23), with probability at least $1 - \delta = 0.75$ we obtain,

$$|A(\overline{x_{opt}} - x_{opt})|_2 \leq 2\epsilon(1 + \epsilon/2)|b^{\perp}|_2 + (2 + \epsilon)2\epsilon|b^{\perp}|_2 \leq (1 + 7\epsilon)|b^{\perp}|_2 \tag{4.31}$$

The claim follows. □

CUR decomposition

A factorization of the form $A = CUR$ where C and R are obtained by sampling c columns and r rows from A and $U \in \mathbb{R}^{c\times r}$ such that $\|A - CUR\| \leq (1 \pm \epsilon) \|A - A_k\|$ for a suitable norm, is called a CUR decomposition. Like the CX decomposition, a CUR decomposition is an interpretable low rank approximation for A. The CUR decomposition algorithm (6) is obtained by first computing the CX decomposition followed by approximate least squares to compute U and R. The algorithm 4.3.2 uses exact sampling probabilities for computing U and R, however estimates of the sampling probabilities can be used as in shown claim 4.3.5.

The correctness of algorithm 4.3.2 follows from the approximate least squares 4.3.5 and the CX decomposition,

$$\begin{aligned}\|A - CUR\|_F = \left\|A - C\overline{X}_{opt}\right\|_F &\leq (1 + 7\epsilon) \left\|A - CC^+A\right\|_F \\ &\leq (1 + 9\epsilon) \|A - A_k\|_F\end{aligned} \tag{4.32}$$

The matrices R and C consist of rows and columns of A, the matrix $U = (D_RS_RC)^+D_R$ is a weighted pseudo-inverse of the intersection of R and C. The Moore Penrose pseudo-inverse is not multiplicative, that is $(AB)^+ \neq B^+A^+$ in general, multiplicativity holds for the cases where (i) A has orthonormal columns. (ii) B has orthonormal rows. (iii) A has full column rank and B has full row rank. Multiplicativity does not hold for $(D_RS_RC)^+$ so U does not simplify to $(S_RC)^+$, therefore knowledge of the sampling probabilities is required for computing U in the CUR decomposition.

Discussion

The importance sampling algorithms using the leverage score distribution including approximation of quadratic forms, approximate least squares and CUR decompositions can be implemented

Algorithm 4.3.2 CUR decomposition (6)

1: Compute a CX decomposition C for A with $c = O(k \log k/\epsilon^2)$ columns.

2: Approximate the least squares problem $\min_X \|A - CX\|_F$ to obtain $\overline{X}_{opt} = (Z^t C)^+ Z^t A$ where $Z \in \mathbb{R}^{m \times r}$ where $r = O(c \log c/\epsilon^2)$ and Z has entries given by equation (4.17).

3: Using the factorization $Z = DS$ obtain $U = (DS^t C)^+ D$ and $R = S^t A$, with probability at least 0.5 the CUR decomposition satisfies,

$$\|A - CUR\|_F \leq (1 + 9\epsilon) \|A - A_k\|_F \tag{4.33}$$

using relative error ϵ approximations to the leverage scores. The running times of the quantum importance sampling algorithms stated in table 4.1 follow from the analyses. Classically leverage scores can be approximated in time $O(mn \log n)$ (25), the quantum algorithms achieve a quadratic speedup with respect to the matrix dimensions but have a polynomial dependence on the spectral gap.

We note that relative error approximations of the leverage scores appears to be necessary for importance sampling algorithms. The running time for our algorithms is constrained by the precision required for relative error estimates of $\ell_k(i)$ which are $O(k/m)$ in expectation. The spectral gap $1/\Delta_k$ may be polynomial in the matrix dimensions in general so the quantum algorithms do not achieve a speedup for all matrices. However, the quantum algorithms achieve a speedup for the special case of rank k matrices where the spectral gap is $1/poly(k)$. The dependence on ϵ is similar for both quantum and classical algorithms, the quantum algorithms pick up an additional factor of $O(1/\epsilon)$ for relative error ϵ approximations of the leverage scores.

Chapter 5

Machine Learning Algorithms

In this chapter, we present quantum algorithms for machine learning problems generalizing least squares with ℓ_2 regularization. Our applications include ridge regression, page rank vectors and polynomial kernels, and are different from those previously considered in the literature (23; 22; 24; 42).

The algorithms presented in this chapter produce quantum states as answers in contrast to classical algorithms that output vectors or matrices. The coordinates of $v \in \mathbb{R}^n$ are not directly accessible given a vector state $|v\rangle$, extracting classically useful information from $|v\rangle$ requires an amplitude estimation/Grover search like operation and requires time $O(\sqrt{n})$. We show that the vector states can be used to estimate training and test errors. The quantum algorithms can therefore be used to compare different models and select the regularization parameter.

This chapter is organized as follows, in section 5.1 introduces some quantum primitives are used by our algorithms, while section 5.2 presents quantum algorithms for ridge regression, page rank and polynomial kernels.

5.1 Preliminaries

Given an oracle that maps $|i, 0^{\log n}\rangle \rightarrow |i, x_i\rangle, i \in [0,1]$ for $x_0, x_1 \in \mathbb{R}^n$ the vector state $|\alpha x_0 + \beta x_1\rangle$ where $\alpha^2 + \beta^2 = 1$ can be prepared as follows,

$$(\alpha |0, x_0\rangle + \beta |1, x_1\rangle) \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle |\alpha x_0 + \beta x_1\rangle + |1\rangle |\alpha x_0 - \beta x_1\rangle) \tag{5.1}$$

and post selecting on the first qubit being in state $|0\rangle$. The state $|\alpha x_0 + \beta x_1\rangle$ is obtained with probability $|\alpha x_0 + \beta x_1|^2/2$, this method is a variant of the the swap test (11) and is well known in the literature. The oracle $|i, 0^{\log n}\rangle \rightarrow |i, x_i\rangle, i \in [0,1]$ can be instantiated with a $QRAM$ or by an algorithm like 3.3.2 that produces a vector state as output.

The standard swap test uses $\alpha = \beta = 1/2$ and the probability of measuring $|0\rangle$ is $(1+\langle x_0|x_1\rangle^2)/2$, the test can be used to estimate the inner product between vector states,

Claim 5.1.1. *The inner product between vector states $|v\rangle, |w\rangle$ can be estimated to additive error ϵ by performing the swap test on $O(\log n/\epsilon^2)$ copies of the states.*

The following geometric fact is used to bound the error in the analysis of the quantum ridge regression algorithm, the proof is a straightforward calculation.

Fact 5.1.2. *For all $u, v \in \mathbb{R}^n$ such that $u.v \geq 0$,*

$$\left| \frac{u}{|u|_2} - \frac{v}{|v|_2} \right|_2^2 \leq \frac{|u-v|_2^2}{\min(|u|_2^2, |v|_2^2)} \tag{5.2}$$

Proof. Wlog assume that $|u|_2^2 = (1+\delta)|v|_2^2$ for $\delta \geq 0$ and let θ be the angle between u and v, note that $\cos(\theta) \geq 0$ as $u.v \geq 0$.

$$\begin{aligned} \frac{|u-v|_2^2}{\min(|u|_2^2, |v|_2^2)} &= 2 + \delta - 2\sqrt{1+\delta}\cos(\theta) \\ &\geq 2 - 2\cos(\theta) + \delta(1-\cos(\theta)) \\ &> 2 - 2\cos(\theta) = \left| \frac{u}{|u|_2} - \frac{v}{|v|_2} \right|_2^2 \end{aligned} \tag{5.3}$$

where the first inequality follows as $\sqrt{1+\delta} \leq 1 + \delta/2$ for all $\delta > 0$ and the second as $\cos(\theta) \in [0, 1]$ as $u.v \geq 0$. □

5.2 Regression with ℓ_2 regularization

We present quantum algorithms for ridge regression and its generalization to polynomial kernels. We also discuss the special case of page rank vectors for undirected graphs which can be viewed as an ℓ_2 regularized linear system over the graph Laplacian (26). The output of the quantum algorithm is a vector state corresponding to the solution and a priori its usefulness is not clear. We show that the quantum algorithm can be used to select the regularization parameter and to compare the performance of different models.

5.2.1 Ridge regression

The classical regression problem is to find an estimator or regression function that predicts a response variable $b \in \mathbb{R}$ as a function of input variables $a \in \mathbb{R}^n$. The input to the regression

problem is a training set $(A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m)$ consisting of examples $(a_i \in \mathbb{R}^n, b_i \in \mathbb{R}), i \in [m]$. Linear regression assumes that the response is a linear function of the input variables and seeks $\theta \in \mathbb{R}^n$ that minimizes the least squares loss function,

$$L(x) = \frac{1}{m}\sum_{i\in[m]}(a_i.x - b_i)^2 = \frac{1}{m}|Ax - b|^2 \tag{5.4}$$

Setting the gradient $\frac{\partial L(x)}{\partial x}$ to 0 in (5.4), the solution θ to the linear regression problem satisfies $A^t(A\theta - b) = 0$ yielding the closed form solution $\theta = (A^tA)^+A^tb$.

Linear regression can be solved in time $O(mn+n^3)$ using direct methods like Gaussian elimination and in time $O(mn\kappa)$ using iterative methods like stochastic gradient descent, iterative methods are preferred for large datasets. Regularization adds an additional penalty term to the least squares objective function (5.4) to handle ill conditioned problems. The ℓ_2-regularized least squares (ridge regression) with regularization parameter μ minimizes the following objective function,

$$L(\mu, x) = \frac{1}{m}\sum_{i\in[m]}(a_i^t.x - b_i)^2 + \mu|x|^2 = \frac{1}{m}|Ax - b|^2 + \mu|x|^2 \tag{5.5}$$

Setting the gradient $\frac{\partial L(\mu,x)}{\partial x}$ to 0 in (5.4), the solution $\theta(\mu)$ to the ridge regression problem satisfies $A^t(A\theta(\mu) - b) + m\mu\theta(\mu) = 0$ yielding a closed form solution $\theta(\mu) = (A^tA + m\mu I)^{-1}A^tb$, note that $A^tA + m\mu I$ is a full rank matrix and thus invertble.

Let $A = \sum_i \sigma_i u_i u_i^t$ be the singular value decomposition of A, then the matrix $A^tA + m\mu I$ has spectral decomposition $\sum_i(\sigma_i^2 + m\mu)v_iv_i^t$. Decomposing $y = A^tb = \sum_i \alpha_i u_i$ as a linear combination of the left singular vectors of A, the ridge regression solution (5.5) can be expressed as,

$$\theta(\mu) = (A^tA + m\mu I)^{-1}A^tb = \sum_i \frac{\alpha_i}{m\mu + \sigma_i(A)^2}v_i \tag{5.6}$$

The above expression for $\theta(\mu)$ in terms of the singular vectors of A can be used to verify that,

$$\theta(\mu) = A^t(AA^t + m\mu I)^{-1}b \tag{5.7}$$

this alternate expression will useful for polynomial kernels. Ridge regression is computationally easier than least squares as the matrix to be inverted has a smaller condition number. From the Bayesian perspective, ridge regression imposes a Gaussian prior on the least squares solution. The book (16) is a comprehensive reference for regression algorithms and regularization methods.

The regularization parameter μ for ridge regression is typically selected by cross validation, the data set is randomly partitioned into a training set A and a validation set V. The ridge regression problem is solved on the training set to obtain $\theta(\mu)$. The training error is $|A\theta(\mu) - b|^2$ and the validation error is $|V\theta(\mu) - b_V|^2$, where b_V denotes the response vector for the validation set. Model selection involves computing the solutions (5.6) for different values of μ and selecting the model that minimizes the validation error.

There is a tradeoff involved in the choice of the regularization parameter μ. Overfitting occurs for models with small values of μ, that is the training error is small but the validation error is large and the model does not generalize well. Models with a large values of μ under-fit and have large training and validation errors. More sophisticated model selection criteria have been proposed in the statistics literature, however model selection based on the validation error is a reasonable choice.

Quantum ridge regression

Algorithm 5.2.1 Quantum ridge regression

Require: $(A, b, y = A^t b)$ with $\|A\|_F^2 = m$ stored in $QRAM$, regularization parameter $\mu \in [0, 1]$, validation set V, matrices $M = \frac{(A,-b)}{\|(A,-b)\|_F}$ and $M_V = \frac{(V,-b_V)}{\|(V,-b_V)\|_F}$, precision $\delta < 0.4$.

Ensure: Outputs $|\overline{\theta(\mu)}\rangle$ such that $\| |\overline{\theta(\mu)}\rangle - |\theta(\mu)\rangle \|_2 \leq \delta$ and additive error δ estimate of the validation error.

1: Let $|y\rangle = \sum_i \alpha_i |v_i\rangle$ where v_i are the right singular vectors of A, obtain $\sum_i \alpha_i |v_i\rangle |\overline{\sigma_i}\rangle$ where $\overline{\sigma_i} \in [\sigma_i(A)/\|A\|_F \pm \epsilon]$ using singular value estimation 3.3.1 with precision $\epsilon = \mu\delta/2$.

2: Append an ancilla qubit and apply the conditional rotation,

$$\sum_i \alpha_i |v_i\rangle |\overline{\sigma_i}\rangle \left(\frac{\mu}{\overline{\sigma_i}^2 + \mu} |0\rangle + \left(1 - \frac{\mu^2}{(\overline{\sigma_i}^2 + \mu)^2}\right)^{1/2} |1\rangle \right) \tag{5.8}$$

post select on $|0\rangle$ using amplitude amplification and erase $\overline{\sigma_i}$ to obtain $|z\rangle = |\overline{\theta(\mu)}\rangle$.

3: Estimate p, the probability of obtaining $|0\rangle$ in (5.8) to relative error δ^2 using amplitude estimation, this requires time $\widetilde{O}(1/\delta^3\mu^2)$ as $p \geq \mu^2/4$.

4: Prepare the linear combination $|z'\rangle = \frac{\sqrt{p}}{\sqrt{1+p}} |\overline{\theta(\mu)}\rangle + \frac{1}{\sqrt{1+p}} |(n+1)\rangle$ as in (5.1).

5: Let $|z'\rangle = \sum_i \beta_i |w_i\rangle$ where w_i are the right singular vectors of M_V, obtain $\sum_i \beta_i |w_i, \overline{\sigma_i}\rangle$ using singular value estimation 3.3.1 for M_V with precision δ.

Append an ancilla qubit and apply the conditional rotation,

$$\sum_i \beta_i |w_i\rangle |\overline{\sigma_i}\rangle \left(\overline{\sigma_i} |0\rangle + \left(1 - \overline{\sigma_i}^2\right)^{1/2} |1\rangle \right) \tag{5.9}$$

Use amplitude estimation to obtain an additive error δ estimate of the probability of obtaining outcome $|0\rangle$ when the ancilla qubit is measured.

The ridge regression solution (5.6) is not scale invariant, that is if A, b are scaled to $A/c, b/c$ the solution $\theta(\mu)$ does not scale to $\theta(\mu)/c$, therefore a suitable normalization is made while preparing the data set A. Algorithm 5.2.1 assumes that the dataset is normalized so that $\|A\|_F^2 = m$, this is a reasonable choice of normalization as it corresponds to scaling each feature vector $a_i \in \mathbb{R}^n$ to have unit length. The notation (A, b) in algorithm 5.2.1 represents concatenation, it denotes the matrix obtained by appending $b \in \mathbb{R}^m$ to the columns of A.

Claim 5.2.1. *Algorithm* 5.2.1 *outputs* $|\overline{\theta(\mu)}\rangle$ *such that* $\| |\theta(\mu)\rangle - |\overline{\theta(\mu)}\rangle \|_2^2 \leq 2\delta^2$ *in time* $\widetilde{O}(1/\mu^2\delta)$

and an additive error $O(\delta)$ estimate of the normalized validation error $\eta(\mu)^2 = \frac{|V\theta(\mu)-b_V|_2^2}{(|\theta(\mu)|+1)\|(V,-b_V)\|_F^2}$ in time $\widetilde{O}(1/\mu^2\delta^3)$.

Proof. Note that $|\theta(\mu)\rangle = |m\theta(\mu)\rangle = \frac{1}{\sqrt{\theta(\mu)}}\sum_i \frac{\alpha_i}{\mu+\sigma_i^2}|v_i\rangle$ by (5.6) where $\sigma_i = \frac{\sigma_i(A)}{\|A\|_F}$ are the normalized singular values of A, estimated in step 1 of algorithm 5.2.1. The squared distance between $|\overline{\theta(\mu)}\rangle$ and $|\theta(\mu)\rangle$ can be bounded using fact 5.1.2, the assumption $\delta < 0.4$ and the estimate $\overline{\sigma}_i^2 \in [\sigma_i^2 \pm 2\epsilon]$ guaranteed by the singular value estimation algorithm,

$$\begin{aligned} |\,|\theta(\mu)\rangle - |\overline{\theta(\mu)}\rangle\,|_2^2 &\leq \frac{1}{m^2 \min\left(|\theta(\mu)|^2, |\overline{\theta(\mu)}|^2\right)} \sum_i \left(\frac{\alpha_i}{\sigma_i^2+\mu} - \frac{\alpha_i}{\sigma_i^2+\mu\pm 2\epsilon}\right)^2 \\ &\leq \left(\frac{2\epsilon}{\mu(1-2\epsilon/\mu)}\right)^2 \leq \delta^2(1+\delta)^2 < 2\delta^2 \end{aligned} \tag{5.10}$$

The probability of measuring $|0\rangle$ in step 2 is at least $(\frac{\mu}{1+\mu})^2 \geq \mu^2/4$, so amplitude amplification requires $O(1/\mu)$ iterations to prepare $|\overline{\theta(\mu)}\rangle$. The running time is $\widetilde{O}(1/\mu^2\delta)$ as each iteration invokes the singular value estimation algorithm with precision $O(\mu\delta)$.

Let $z = \frac{1}{\sqrt{|\theta(\mu)|+1}}(\theta(\mu), 1)$ be the unit vector in the direction $\frac{(\theta(\mu),1)}{|\theta(\mu)|}$, the squared norm $|M_V z|_2^2$ equals the normalized validation error $\eta(\mu)^2$. Algorithm 5.2.1 approximates $\eta(\mu)^2$ by estimating $|M_V z'|_2^2$ where z' is the unit vector in the direction $\left(\frac{\overline{\theta(\mu)}}{|\overline{\theta(\mu)}|}, \frac{1}{\sqrt{p}}\right)$. The bound $|z-z'|^2 \leq O(\delta^2)$ follows from fact 5.1.2 and equation (5.10) and can be used to bound $|M_V z'|_2^2 - |M_V z|^2$,

$$|M_V z'|_2^2 - |M_V z|^2 \leq (|M_V z| + |M_V z'|)\sigma_{max}(M_V)\delta = O(\delta) \tag{5.11}$$

where the final equality follows as $\sigma_{max}(M_V) \leq 1$ and $|M_V z| \leq 1$ for all $z \in \mathbb{R}^n$.

If $z' = \sum_i \beta_i w_i$ be the decomposition of z' as a linear combination of the singular vectors of M_V, then $|M_V z'|_2^2 = \sum_i \beta_i^2\sigma_i^2$. The probability of measuring $|0\rangle$ in step 5 of algorithm 5.2.1 $\sum_i \beta_i^2\overline{\sigma_i}^2$ approximates $|M_V z'|_2^2$,

$$\left|\sum_i \beta_i^2\overline{\sigma_i}^2 - |M_V z'|_2^2\right| \leq 2\delta \tag{5.12}$$

the inequality follows as $\overline{\sigma}_i \in \sigma_i \pm \delta$ and $\sum_i \beta_i^2 = 1$. Algorithm 5.2.1 estimates $\sum_i \beta_i^2\overline{\sigma_i}^2$ within additive error δ, the output estimate is within $\pm 3\delta$ of $|M_V z'|_2^2$ by (5.12). It follows from (5.11)

that the additive error in estimating $\eta(\mu)^2$ is $O(\delta)$, amplitude estimation in step 3 is the most time consuming step in the algorithm and requires time $\widetilde{O}(1/\delta^3\mu^2)$. □

The quantum ridge regression algorithm can be used to select the regularization parameter μ by estimating the normalized validation error $\eta(\mu)$ and selecting μ for which the error is minimum. The value $\eta(\mu)$ lies in the interval $[\sigma_{min}(M_V), \sigma_{max}(M_V)]$, relative error estimates of $\eta(\mu)$ are obtained in worst case time $\widetilde{O}(1/\mu^2\sigma^3_{min})$. The actual time may requirements may be smaller and the quantum algorithm may achieve a speedup over classical algorithms over several computations of ridge regression solutions for different values of μ.

We note that most of the computational effort in ridge regression goes towards computing models for different values of μ and selecting μ achieving the best tradeoffs. The quantum algorithm thus complements classical algorithms providing a fast method for comparing models with different values of μ, having selected μ classical algorithms can be used to compute the coordinates of the solution $\theta(\mu) \in \mathbb{R}^n$.

5.2.2 Pagerank

The computation of the page rank vector (31) for undirected graphs can be regarded as an instance of ℓ_2 regularized linear system in the Laplacian matrix of the graph as shown in (26). A quantum state corresponding to the page rank vector can therefore be computed using methods similar to the ridge regression algorithm 5.2.1.

We recall the notion of page rank vector and its interpretation (26) as an ℓ_2 regularized linear system on the normalized Laplacian matrix. Given a graph $G(V, E)$ the adjacency matrix $A \in \mathbb{R}^{n\times n}$ has entries $a_{ij} = 1$ if $i \sim j$ and 0 otherwise, the diagonal matrix $D \in \mathbb{R}^{n\times n}$ has entries $d_{ii} = deg(i)$ where the degree $deg(i)$ is the number of edges incident to vertex i. The lazy random walk matrix is defined as $Z := (I + AD^{-1})/2$.

The pagerank vector $pr(\alpha, s)$ is the stationary distribution for a random walk that with probability α teleports to seed distribution s and with probability $(1 - \alpha)$ carries out a step of the lazy random walk Z, that is:

$$pr(\alpha, s) = \alpha s + (1 - \alpha)Zpr(\alpha, s) \tag{5.13}$$

The teleportation constant α is analogous to a regularization parameter, walks starting at s with length $O(1/\alpha)$ contribute to the pagerank vector. Computing the pagerank is equivalent to solving a linear system over a regularized Laplacian,

$$\begin{aligned} (I - (1 - \alpha)Z)pr(\alpha, s) &= \alpha s \\ \Rightarrow \left(\frac{1+\alpha}{2}I - \frac{1-\alpha}{2}AD^{-1}\right) pr(\alpha, s) &= \alpha s \\ \Rightarrow \left(\frac{1+\alpha}{1-\alpha}I - AD^{-1}\right) pr(\alpha, s) &= \frac{2\alpha}{1-\alpha}s \end{aligned} \tag{5.14}$$

The combinatorial Laplacian $L = D - A$ is normalized to $D^{-1/2}LD^{-1/2}$. Substituting $\beta = \frac{2\alpha}{1-\alpha}$ in equation (5.14), the page rank vector can be expressed in terms of the normalized Laplacian,

$$\begin{aligned} pr(\alpha, s) &= \beta((1+\beta)I - AD^{-1})^{-1}s \\ &= \beta(\beta I + D^{-1/2}LD^{-1/2})^{-1}s \end{aligned} \tag{5.15}$$

Thus computing pagerank vectors is equivalent to solving a linear system, comparing with the ridge regression problem (5.6), the regularization parameter $\mu = \frac{\beta}{Tr(D^{-1/2}LD^{-1/2})} = \frac{\beta}{n}$, the ridge regression algorithm therefore yields a quantum algorithm for computing the vector state corresponding to the approximate page rank vector.

Claim 5.2.2. *There is a quantum algorithm with running time $\widetilde{O}(n^2/\beta^2\delta)$ that outputs a vector state $|\overline{pr(\alpha, s)}\rangle$ such that $|\,|pr(\alpha, s)\rangle - |\overline{pr(\alpha, s)}\rangle\,|_2 \leq \delta$.*

Page rank vectors can be approximated classically in time $\widetilde{O}(m \log n)$ using fast Laplacian linear system solvers (19) or iterative methods so the quantum algorithm does not constitute a speedup over classical algorithms. The page rank example illustrates that the quantum ridge regression algorithm does not achieve a speed up for all cases. Speedups are achieved when the regularization parameter $\mu = c\,\|A\|_F$.

5.2.3 Polynomial Kernels

Linear regression assumes that the relationship between the input variables and the response is linear, that is the response y is a linear combination of the features $x_i, i \in [n]$. Kernels generalize the linear model expressing the response as a linear combination of features in an implicit N dimensional feature space specified by the mapping $\phi : \mathbb{R}^n \to \mathbb{R}^N$. The kernel $k(x, x') = \langle\phi(x)|\phi(x')\rangle$ corresponds to the inner product in the implicit feature space, algorithms that can be specified in terms of the inner product can be kernelized replacing the inner product $\langle x|y\rangle$ by $k(x, y)$.

The quantum singular value estimation algorithms 3.2.1 and 3.3.1 are not specified in terms of inner products, so these algorithms can not be kernelized in the classical sense. However, algorithm 3.3.1 can be used to compute models corresponding to the degree d polynomial kernels $k(x, x') = \langle x|x'\rangle^d$. More generally a degree d polynomial defines a valid kernel $k(x, x')$ if and only if it is a linear combination of Legendre polynomials with positive coefficients (37). We restrict our discussion to polynomial kernels of the form $k(x, x') = \langle x|x'\rangle^d$ whose feature spaces correspond to all degree d monomials.

Ridge regression can be generalized to non linear feature spaces replacing inner products in the loss function (5.5) by the inner product in the expanded feature space,

$$L(\mu, x) = \frac{1}{m}\sum_{i\in[m]}(\phi(a_i).x - b_i)^2 + \mu|x|^2 \tag{5.16}$$

The kernel matrix $K \in \mathbb{R}^{m\times m}$ has entries $K_{ij} = k(a_i, a_j)$, and the feature matrix $\Phi \in \mathbb{R}^{m\times N}$ has rows $\phi(x_i)$ so that $K = \Phi\Phi^t$. The solution to kernel regression is given by $\theta(\mu) = \text{argmin}_{x\in\mathbb{R}^N} L(\mu, x)$

and is analogous to the ridge regression solution (5.7),

$$\theta(\mu) = \Phi^t(K + m\mu I)^{-1}b \tag{5.17}$$

Density matrices $K/Tr(K)$ can be prepared efficiently using the augmented $QRAM$ for kernels of the form $k(x, y) = \langle x|y\rangle^d$.

Claim 5.2.3. *Let $A \in \mathbb{R}^{m\times n}$ be stored in an augmented $QRAM$ and $K \in \mathbb{R}^{m\times m}$ have entries $K_{ij} = \langle a_i|a_j\rangle^d$, the density matrix $K/Tr(K)$ can be prepared in time $\widetilde{O}(1)$.*

Proof. The matrix $K/Tr(K)$ can be prepared by tracing out the last d systems from the following state that can prepared making d queries to the $QRAM$,

$$|\Phi\rangle = \sum_{i\in[m],j\in[n]^d} (a_i^{\otimes d})_j \, |i, j\rangle = \sum_{i\in[m]} |i, a_i^{\otimes d}\rangle \tag{5.18}$$

As $K = \Phi\Phi^t$, this is an application of the density matrix preparation method in equation (2.9). □

Given copies of the density matrix $K/Tr(K)$, algorithm 3.2.1 can be used to perform singular value estimation and create the quantum state corresponding to the feature space representation of $\theta(\mu)$. If $b = \sum_{i\in[m]} \beta_i v_i$ in the spectral basis for K, then:

$$\Phi\theta(\mu) = K(K + m\mu)^{-1}b = \sum_{i\in[m]} \frac{\lambda_i(K)\beta_i}{\lambda_i(K) + m\mu} v_i \tag{5.19}$$

Assuming that the data is normalized so that $|a_i| = 1, \forall i \in [m]$, the trace of the polynomial kernel $Tr(K) = \sum_{i\in m}\langle a_i|a_i\rangle^d = m$, thus the state (5.19) can be prepared by performing singular value estimation for $K/Tr(K)$, applying a conditional rotation and post selecting as in step 2 of algorithm 5.2.1,

$$|b\rangle \xrightarrow{5.2.1} \sum_i \beta_i \, |v_i\rangle \, |\overline{\sigma_i}\rangle \left(\frac{\overline{\sigma_i}}{\mu + \overline{\sigma_i}} \, |0\rangle + \left(1 - \frac{\overline{\sigma_i}}{\mu + \overline{\sigma_i}}^2\right)^{1/2} |1\rangle \right) \tag{5.20}$$

The squared distance between the state $|\overline{\Phi\theta(\mu)}\rangle$ obtained by post selecting on $|0\rangle$ and the state $|\Phi\theta(\mu)\rangle$ can be bounded as follows for $\epsilon = \mu\delta$,

$$\begin{aligned} |\, |\overline{\Phi\theta(\mu)}\rangle - |\Phi\theta(\mu)\rangle \,|^2 &= \sum_i \beta_i^2 \left(\frac{\sigma_i}{\mu + \sigma_i} - \frac{\sigma_i \pm \epsilon}{\mu + \sigma_i \pm \epsilon} \right)^2 \\ &\leq \sum_i \beta_i^2 \left(\frac{\epsilon}{\mu - \epsilon} \right)^2 \leq \frac{\delta^2}{(1-\delta)^2} \end{aligned} \tag{5.21}$$

For $\delta < 1/2$ the output state is within squared distance $O(\delta^2)$ of the state $|\Phi\theta(\mu)\rangle$ corresponding to the solution. If the angle between b and $Col(K)$ is θ, then the probability of measuring $|0\rangle$ in (5.20) is at least $\frac{\cos^2(\theta)\sigma_{min}(K)^2}{(\mu+1)^2}$, the number of iterations required for amplitude amplification is $O(1/\cos(\theta)\lambda_{min}(K))$. We therefore have the following claim,

Claim 5.2.4. *Given $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ stored in the augmented QRAM and the degree d polynomial kernel $K_{ij} = \langle a_i|a_j\rangle^d$, there is a quantum algorithm that outputs $|\overline{\Phi\theta(\mu)}\rangle$ such that $||\overline{\Phi\theta(\mu)}\rangle - |\Phi\theta(\mu)\rangle|^2 = O(\delta^2)$ in time $\widetilde{O}(1/\mu^3\delta^3\sigma_{min}(K)\cos(\theta))$.*

Note that unlike the algorithm for ridge regression, we do not obtain a vector state corresponding to the solution $\theta(\mu)$. However, $\Phi\theta(\mu) \in \mathbb{R}^m$ is an approximation to b and the inner product $\langle\Phi\theta(\mu)|b\rangle$ provides a measure for the quality of fit. The inner product can be estimated using the swap test 5.1.1, the quantum algorithm can therefore be used for model selection by comparing polynomial kernels with different degrees and regularization parameters.

Classical computation of polynomial kernel models is computationally expensive as requires space $O(m^2)$ to store the dense kernel matrices and time $O(m^3)$ for matrix inversion, the quantum algorithms complement the classical algorithms by providing a method for selecting model parameters without incurring the large classical computation overheads.

Summary and discussion

The running times of the quantum algorithms for ℓ_2 regularized regression problems are summarized in the table 5.1. The classical algorithms compute exact solutions for the regression problems, the quantum algorithms produce vector states as answers and can be used to compare models and select regularization parameters. Quantum algorithms do not provide a speedup for all the problems, for example in the case of page rank computation, classical algorithms for approximate page rank computation are faster.

Problem	**Quantum Algorithm**	**Classical Algorithm**
Ridge regression	$\widetilde{O}(1/\mu^2\delta)$	$O(mn^2)$
Approximate Pagerank	$\widetilde{O}(n^2/\beta^2\delta)$	$O(m\log(1/\delta)/\beta)$
Polynomial kernels	$\widetilde{O}(1/\mu^3\delta^3\sigma_{min}(K)\cos(\theta))$	$O(m^3)$

Table 5.1. Running times for quantum and classical regression algorithms.

The quantum regression algorithms are of limited utility as they produce vector states as answers and obtaining regression coefficients from the vector states requires polynomial time. However, the quantum algorithms can complement classical algorithms for comparing models and selecting regularization parameters.

Bibliography

[1] A. Belovs and B. W. Reichardt, "Span programs and quantum algorithms for st-connectivity and claw detection," in *Algorithms–ESA 2012*. Springer, 2012, pp. 193–204.

[2] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM journal on Computing*, vol. 26, no. 5, pp. 1510–1523, 1997.

[3] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, "Efficient quantum algorithms for simulating sparse hamiltonians," *Communications in Mathematical Physics*, vol. 270, no. 2, pp. 359–371, 2007.

[4] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *arXiv: quant-ph:0005.055*, 2000.

[5] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast monte carlo algorithms for matrices i: Approximating matrix multiplication," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 132–157, 2006.

[6] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-error cur matrix decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 844–881, 2008.

[7] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, "Faster least squares approximation," *Numerische Mathematik*, vol. 117, no. 2, pp. 219–249, 2011.

[8] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.

[9] V. Giovannetti, S. Lloyd, and L. Maccone, "Architectures for a quantum random access memory," *Physical Review A*, vol. 78, no. 5, p. 052310, 2008.

[10] ——, "Quantum random access memory," *Physical review letters*, vol. 100, no. 16, p. 160501, 2008.

[11] D. Gottesman and I. Chuang, "Quantum digital signatures," *arXiv preprint quant-ph/0105032*, 2001.

[12] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

[13] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[14] M. Hardt, "Robust subspace iteration and privacy-preserving spectral analysis," *arXiv preprint arXiv:1311.2495*, 2013.

[15] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.

[16] T. Hastie, R. Tibshirani, and J. J. H. Friedman, *The elements of statistical learning*. Springer New York, 2001, vol. 1.

[17] H. Hotelling, "Relations between two sets of variates," *Biometrika*, pp. 321–377, 1936.

[18] C. Jordan, "Essai sur la géométrie à n dimensions," *Bulletin de la Société mathématique de France*, vol. 3, pp. 103–174, 1875.

[19] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A simple, combinatorial algorithm for solving sdd systems in nearly-linear time," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 911–920.

[20] A. Y. Kitaev, "Quantum measurements and the abelian stabilizer problem," *arXiv preprint quant-ph/9511026*, 1995.

[21] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 831–838, 1980.

[22] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," *Arxiv preprint:1307.0411*, 2013.

[23] ——, "Quantum self analysis," *Arxiv preprint:1307.1401*, 2013.

[24] ——, "Quantum support vector machine for big feature and big data classification," *Arxiv preprint:1307.0471*, 2013.

[25] M. W. Mahoney, P. Drineas, M. Magdon-Ismail, and D. P. Woodruff, "Fast approximation of matrix coherence and statistical leverage." in *ICML*, 2012.

[26] M. W. Mahoney and L. Orecchia, "Implementing regularization implicitly via approximate eigenvector computation," *Proceedings of the 28th International Conference on Machine Learning*, pp. 121–128, 2011.

[27] C. Marriott and J. Watrous, "Quantum arthur–merlin games," *Computational Complexity*, vol. 14, no. 2, pp. 122–152, 2005.

[28] M. McLoone and J. V. McCanny, "Efficient single-chip implementation of sha-384 and sha-512," in *Proceedings. 2002 IEEE International Conference on Field-Programmable Technology, 2002.* IEEE, 2002, pp. 311–314.

[29] M. Mihail and C. Papadimitriou, "On the eigenvalue power law," in *Randomization and approximation techniques in computer science*. Springer, 2002, pp. 254–262.

[30] B. Misra and E. C. G. Sudarshan, "The zeno paradox in quantum theory," *Journal of Mathematical Physics*, vol. 18, no. 4, pp. 756–763, 2008.

[31] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." *Stanford InfoLab*, 1999.

[32] A. Peres, *Quantum theory: concepts and methods*. Springer, 1995, vol. 57.

[33] J. Preskill, "Lecture notes for physics 229: Quantum information and computation," *California Institute of Technology*, 1998.

[34] B. W. Reichardt, F. Unger, and U. Vazirani, "A classical leash for a quantum system: Command of quantum systems via rigidity of chsh games," *arXiv preprint arXiv:1209.0448*, 2012.

[35] M. Rudelson and R. Vershynin, "Sampling from large matrices: An approach through geometric functional analysis," *Journal of the ACM (JACM)*, vol. 54, no. 4, p. 21, 2007.

[36] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM journal on computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[37] A. J. Smola, Z. L. Ovari, and R. C. Williamson, "Regularization with dot-product kernels," *Advances in Neural Information Processing Systems*, pp. 308–314, 2001.

[38] D. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *Proceedings of the 40th annual ACM symposium on Theory of computing*, pp. 563–568, 2008.

[39] M. Szegedy, "Quantum speed-up of markov chain based algorithms," in *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on.* IEEE, 2004, pp. 32–41.

[40] G. Wang, "Quantum algorithms for approximating the effective resistances of electrical networks," *arXiv preprint arXiv:1311.1851*, 2013.

[41] N. Wiebe, D. W. Berry, P. Høyer, and B. C. Sanders, "Simulating quantum dynamics on a quantum computer," *Journal of Physics A: Mathematical and Theoretical*, vol. 44, no. 44, p. 445308, 2011.

[42] N. Wiebe, A. Kapoor, and K. Svore, "Quantum nearest-neighbor algorithms for machine learning," *arXiv preprint arXiv:1401.2142*, 2014.