Quantum gradient descent for linear systems and least squares

Iordanis Kerenidis [*]   Anupam Prakash [†]

May 28, 2019

Abstract

Quantum machine learning and optimization are exciting new areas that have been brought forward by the breakthrough quantum algorithm of Harrow, Hassidim and Lloyd for solving systems of linear equations. The utility of classical linear system solvers extends beyond linear algebra as they can be leveraged to solve optimization problems using iterative methods like gradient descent. In this work, we provide the first quantum method for performing gradient descent when the gradient is an affine function. Performing τ steps of the gradient descent requires time $O(\tau C_S)$ for weighted least squares problems, where C_S is the cost of performing one step of the gradient descent quantumly, which at times can be considerably smaller than the classical cost. We illustrate our method by providing two applications: first, for solving positive semidefinite linear systems, and, second, for performing stochastic gradient descent for the weighted least squares problem with reduced quantum memory requirements. We also provide a quantum linear system solver in the QRAM data structure model that provides significant savings in cost for large families of matrices.

---

[*]CNRS, IRIF, Université Paris Diderot, Paris, France and Centre for Quantum Technologies, National University of Singapore, Singapore. Email: `jkeren@irif.fr`.

[†]CNRS, IRIF, Université Paris Diderot, Paris, France. Email: `anupamprakash1@gmail.com`.

1 Introduction

Quantum Machine Learning has seen a flurry of recent exciting developments, mainly due to the breakthrough algorithm of Harrow, Hassidim, and Lloyd [14], that takes as input a sparse and well-conditioned system of linear equations, and in time polylogarithmic in the system's dimension outputs the solution vector as a quantum state. In other words, given a matrix A and a vector b, it outputs the quantum state $|A^{-1}b\rangle$ corresponding to the solution.

The HHL algorithm has been very influential, and several works have obtained quantum algorithms for machine learning problems under similar assumptions [24, 26, 23]. However, the applications of the HHL algorithm to machine learning had several caveats as pointed out in [1]. Namely, the algorithm produces a quantum state $|A^{-1}b\rangle$ instead of a classical solution and achieves an exponential speedup only when the matrix is well conditioned and has sparsity poly-logarithmic in the dimension. In machine learning settings, the input matrices are not expected to be sparse or well structured, thus restricting the applications of the HHL algorithm.

We introduced the QRAM data structure model [18] in order to overcome some of these caveats. We presented a quantum algorithm for competitive recommendation systems in [18] that provides a good recommendation to most users in expected time $O(\frac{\sqrt{k}}{\epsilon}\text{polylog}(mn))$, which is poly-logarithmic in matrix dimensions and has a square root dependence on the rank k which is much smaller than the matrix dimensions. The best known classical recommendation systems at the time required time polynomial in the matrix dimensions for the same task.

Very recently, a classical recommendation system with running time $O(poly(k, \frac{1}{\epsilon})\text{polylog}(mn))$ was given [31] using classical instantiations of the QRAM data structures. We note that the classical recommendation algorithm [31] incurs a large polynomial overhead over the quantum algorithm and has a running time $O(\frac{k^{12}}{\epsilon^{12}}\text{polylog}(mn))$, which for inputs of interest make the new classical algorithm slower even than older classical algorithms. It remains an open question to obtain classical algorithms with a smaller polynomial overhead in k and $1/\epsilon$.

In this work, we generalize the quantum singular value estimation algorithm using QRAM data structures [18] and obtain speedups for classes of dense matrices beyond the sparse and the low-rank cases. Further, we develop a framework for quantum gradient descent with affine updates with cost polynomial in the number of steps and present applications to weighted least squares problems.

Our work is motivated by the applications of classical linear system solvers to optimization and machine learning using iterative methods. Examples of iterative methods include first-order methods like gradient descent and second order methods like the interior point method for linear and semidefinite programs. These methods allow us to solve a variety of problems that do not have a closed form solution and thus cannot be solved using linear systems alone.

The iterative methods start with some initial solution state θ_0 and update it iteratively according to a rule of the form $\theta_{t+1} = \theta_t + \alpha r_t$. In many cases, the updates are implemented using linear algebra operations like matrix multiplication and inversion. This motivates the question of whether quantum linear system solvers can be similarly used to solve more general optimization problems through iterative methods.

However, there are obvious obstacles towards realizing a quantum iterative method. Quantum routines for matrix multiplication and inversion output a quantum state and not a classical vector that can be used directly for the next step of the method. Further, they succeed only with a fairly small probability and destroy the input when they fail. Hence, if the quantum procedure for finding the updates fails at any step, one needs to restart the iterative method from the very

beginning. Algorithmically this implies that the running time of a naive quantum iterative method is exponential in the number of steps. Another problem is that the HHL based quantum linear systems solvers provide exponential speedups only for matrices that are poly-logarithmically sparse in the dimension.

In this work, we make progress on both the challenges described above. First, we provide a first order quantum iterative method for performing gradient descent with affine update rules in time polynomial in the number of steps. Secondly, we design a quantum linear systems solver in the QRAM data structure model that allows us to obtain potentially significant speedups for many classes of dense matrices. We provide as applications iterative quantum algorithms for the weighted least squares problem including one that can perform stochastic gradient descent.

The weighted least squares problem is widely used for regression or data fitting. The data matrix for least squares is often very skewed, i.e., the number of data points is vastly larger than the dimension of the points. Stochastic gradient descent is advantageous in such settings as it estimates the gradient by computations on a small sized batch of data points and avoids costly computations on large matrices. The quantum stochastic gradient descent algorithm allows us to reduce the size of the required QRAM as at any given time only a small subset of the data needs to be in quantum memory.

Quantum algorithms for gradient estimation have been studied before [15, 7, 16, 11], where a quantum algorithm is used to estimate the gradient of a given function, while a classical algorithm performs the gradient descent. A quantum gradient descent algorithm for polynomial optimization problems has also been presented [27], but the running time of the algorithm depends exponentially on the number of steps. Our algorithm achieves a polynomial scaling instead in the number of steps when the gradient is an affine function, a case that frequently arises in practice for solving least squares problems.

More recently, we presented a quantum interior point method for linear and semidefinite programs [19] which provides an example of a second order quantum iterative method and demonstrates that like classical linear system solvers, quantum linear system solvers can be also be leveraged to iteratively solve optimization problems.

1.1   The QRAM data structure model

The QRAM data structure model requires a full-scale quantum computer equipped with a quantum random access memory. The QRAM is analogous to a classical random access memory and the model can be viewed as a quantum analog of using data structures to speed up computation. Let us first define the QRAM more precisely and compare it to the more standard oracle model.

Definition 1. *A quantum random access memory (QRAM) is a device that stores data $(i, x_i) \in [n]$ and allows queries of the form $|i, 0\rangle \to |i, x_i\rangle$ to be made in superposition in time $O(\mathrm{polylog}(n))$.*

The standard oracle model also assumes that one can make the query $|i, 0\rangle \to |i, x_i\rangle$ in poly-logarithmic time. However, in the absence of a QRAM, queries $|i, 0\rangle \to |i, x_i\rangle$ can be made efficiently only if x_i is a function of i that can be computed using a quantum circuit of size poly-logarithmic in N. The QRAM implements the standard oracle for arbitrary data.

In addition, we require a notion of efficient QRAM data structures. A QRAM data structure is efficient if it avoids additional time and space overheads over storing the data sequentially in the QRAM and can be updated efficiently.

Definition 2. *A QRAM data structure storing n entries is said to be efficient it its memory requirement is $\widetilde{O}(n)$ and the cost of updating, inserting or deleting a single entry is $O(\mathrm{polylog}(n))$.*

Let us elaborate this notion a bit further. A dataset of size n can be stored in the QRAM by storing each of its entries sequentially in a simple list or array. The total memory used is $\widetilde{O}(n)$ (with logarithmic factors to account for memory management) and the update time for a single entry $O(\log(n))$. An efficient data structure allows us to store the dataset in a more elaborate form, for example in a binary tree, with the same memory and update time requirements as for when the dataset is stored sequentially.

We are now ready to define the QRAM data structure model used for our algorithms,

Definition 3. *An algorithm in the QRAM data structure model that processes a data-set of size m has two steps:*
  1. *A pre-processing step with complexity $\widetilde{O}(m)$ that constructs efficient QRAM data structures for storing the data.*
  2. *A computational step where the quantum algorithm has access to the QRAM data structures constructed in step 1.*

The complexity algorithms in this model is measured by the cost for step 2.

In order to see the relevance of this model, consider a data-processing setting where it is feasible to store a dataset of size n, but infeasible to perform a super-linear computation with cost $O(n^2)$ or $O(n^3)$. In this setting, a quantum algorithm in the QRAM data structure model requires space $\widetilde{O}(n)$ and could perform the same intensive computation in time sub-linear in n, thus making the application feasible. It is therefore an effective model for data-processing applications.

As our algorithms require a QRAM, we briefly discuss the feasibility of implementing a QRAM. A bucket brigade architecture for a QRAM has been proposed [12], where only $O(\log n)$ gates from a circuit of size $O(n)$ are activated when a query is made. An error analysis for this architecture was given [3], where it was shown that the error per gate for applications like the HHL algorithm and other quantum machine learning algorithms needs to be inverse in the number of active gates, i.e., inverse in the logarithm, and quantum error correction may not be required for such applications.

We also briefly discuss subsequent work on quantum linear system solvers in the QRAM data structure model. It was observed in [8] that the results in this paper are equivalent to the statement that QRAM data structures allow the construction of efficient unitary block encodings with parameter $\mu(A)$ (defined in section 1.2) for an arbitrary matrix A. Given efficient unitary block encodings, quantum linear system solvers with nearly optimal dependence on precision and condition number were developed [8], these results provide nearly optimal quantum linear system solvers in the QRAM data structure model.

We are now ready to present our main results and discuss the ideas used for the proofs. We present our results on linear system solvers in Section 1.2, the quantum gradient iterative method in Section 1.3 and applications of the quantum iterative method in Section 1.4.

1.2   An improved quantum linear systems solver

We first introduce some quantum and linear algebra notation that helpful for the discussion of our main results and is used throughout the paper. The set $\{1, 2, \cdots, n\}$ is denoted by $[n]$, the standard basis vectors in $\mathbb{R}^n$ are denoted by $e_i, i \in [n]$. For a vector $x \in \mathbb{R}^n$ we denote the ℓ_p-norm as $\|x\|_p = (\sum_i x_i^p)^{1/p}$. The Euclidean norm $\|x\|_2$ is denoted as $\|x\|$. The rank of a matrix is

denoted as $rk(A)$. A matrix is positive semidefinite (psd) if it is symmetric and has non-negative eigenvalues, the notation $A \succeq 0$ indicates that A is a psd matrix. The singular value decomposition of a symmetric matrix $A \in \mathbb{R}^{n \times n}$ is written as $A = \sum_i \lambda_i v_i v_i^T$ where $\lambda_i \geq 0$ are the eigenvalues and v_i are the corresponding eigenvectors.

The singular value decomposition of $A \in \mathbb{R}^{m \times n}$ is written as $A = \sum_i \sigma_i u_i v_i^T$ where σ_i are the singular values and u_i and v_i are the left and right singular vectors. The Frobenius norm is defined as $\|A\|_F^2 := \sum_{ij} A_{ij}^2 = \sum_i \sigma_i^2$ while the spectral norm $\|A\| = \sigma_{max}$, where $\sigma_{\max}$ is the largest singular value. The condition number $\kappa(A) = \sigma_{max}/\sigma_{min}$.

The i-th row of matrix $A \in \mathbb{R}^{m \times n}$ is denoted as a_i and the j-th column is denoted as a^j. The $\circ$ operator denotes the Hadamard product, that is $A = P \circ Q$ implies that $A_{ij} = P_{ij}.Q_{ij}$ for $i \in [m], j \in [n]$. For a matrix $A \in \mathbb{R}^{m \times n}$, the maximum of the p-th power of the ℓ_p norm for the row vectors is denoted $s_p(A) := \max_{i \in [m]} \|a_i\|_p^p$, the maximum of the p-th power of the ℓ_p norm of the column vectors is $s_p(A^T)$. The sparsity $s(A)$ is the maximum number of non-zero entries in a row of A. The $\widetilde{O}$ notation is used to suppress factors poly-logarithmic in vector or matrix dimensions, that is $O(f(n)\text{polylog}(mn))$ is written as $\widetilde{O}(f(n))$.

A vector $v \in \mathbb{R}^n$ is encoded into $O(\log n)$ dimensional normalized vector state $|v\rangle = \frac{1}{\|v\|} \sum_{i \in [n]} v_i |i\rangle$. However, there is one exception to this use of notation, in the gradient descent procedure we denote the unnormalized garbage state by $|G\rangle$. In all other cases, the notation $|v\rangle$ where $v \in \mathbb{R}^n$ denotes the normalized vector state.

We first present our results on linear system solvers in the QRAM data structure model. The linear system solver that we present here can offer considerable speedups over classical algorithms for the problem sampling from the solutions of linear systems for large families of dense matrices, including matrices whose rows have bounded ℓ_1 norm.

The main technical tool for our linear system solver is algorithm for singular value estimation (SVE) that generalizes the algorithm in [18]. More precisely, let $A = \sum_{i \in [k]} \sigma_i u_i v_i^T$ be the singular value decomposition for $A \in \mathbb{R}^{m \times n}$ where $k = \min(m, n)$ and σ_i are the (possibly 0) singular values and u_i and v_i are the left and right singular vectors. A given vector b can be viewed as a superposition of the singular vectors of the matrix A, i.e. $|b\rangle = \sum_i \beta_i |v_i\rangle$.

Definition 4. *Let $A = \sum_{i \in [k]} \sigma_i u_i v_i^T$ be the singular value decomposition for $A \in \mathbb{R}^{m \times n}$ for $k = \min(m, n)$ and let $\delta > 0$. The singular value estimation (SVE) problem with error δ is defined as: Given $|b\rangle = \sum_{i \in [k]} \beta_i |v_i\rangle$, to perform the mapping*

$$\sum_{i \in [k]} \beta_i |v_i\rangle |0\rangle \to \sum_{i \in [k]} \beta_i |v_i\rangle |\overline{\sigma}_i\rangle,$$

such that $|\overline{\sigma}_i - \sigma_i| \leq \delta$ for all $i \in [k]$ with probability $1 - 1/poly(n)$.

We had had provided an SVE algorithm with running time $\widetilde{O}(\|A\|_F/\delta)$ in our work on recommendation systems [18]. This algorithm relied on a particular efficient QRAM data structure for storing A. In this work, we provide efficient QRAM data structures for storing A that allow us to obtain an SVE algorithm with a running time that depends on the maximum ℓ_1-norm of the rows of A, instead of the $\|A\|_F$.

Our SVE algorithms are based on the relation between quantum walks and singular values [29, 9]. We show that for every decomposition $A/\mu = P \circ Q$ such the norms of the rows of P and the columns of Q are at most 1 and $\circ$ denotes the entrywise (Hadamard) product, if we have access

4

to unitaries that prepare normalized quantum states corresponding to the rows and columns of P, Q in time $\widetilde{O}(1)$, there is an SVE algorithm with running time $\widetilde{O}(\mu/\delta)$.

Result 1. *[Theorem 4.3] Let $A \in \mathbb{R}^{m \times n}$ be a matrix and suppose there exist $P, Q \in \mathbb{R}^{m \times n}$ and $\mu > 0$ such that $\|p_i\|_2 \leq 1\ \forall i \in [m]$, $\|q^j\|_2 \leq 1\ \forall j \in [n]$ and*

$$A/\mu = P \circ Q. \tag{1}$$

If unitaries $U : |i\rangle |0^{\lceil \log(n+1) \rceil}\rangle \to |i\rangle |\overline{p}_i\rangle$ and $V : |0^{\lceil \log(m+1) \rceil}\rangle |j\rangle \to |\overline{q}^j\rangle |j\rangle$ can be implemented in time $\widetilde{O}(\log(mn))$ then there is an algorithm that performs SVE for A in time $\widetilde{O}(\mu/\delta)$.

This result also explains the connection between a QRAM data structure and an SVE algorithm. The QRAM data structure is designed with a particular factorization $A/\mu = P \circ Q$ in mind, it provides an $\widetilde{O}(1)$ time implementation of the unitaries U and V and therefore an SVE algorithm with running time $\widetilde{O}(\mu/\delta)$ by Result 1.

The SVE algorithm in [18] falls in this framework, it utilizes an efficient QRAM data structure corresponding to a factorization of form $A/\|A\|_F = P \circ Q$. In this work, we provide efficient QRAM data structures for a more general class of factorizations. Let $s_p(A) := \max_{i \in [n]} \|a_i\|_p^p$ where $\|a_i\|_p$ denotes the ℓ_p norm of the i-th row of A, then we have the following result.

Result 2. *[Theorem 4.4] For all $p \in [0, 1]$ there are efficient QRAM data structures for storing $A \in \mathbb{R}^{m \times n}$, such that a quantum algorithm with access to such data structures can perform SVE for A with error δ in time $\widetilde{O}(\mu(A)/\delta)$ for $\mu(A) := \left(\sqrt{s_{2p}(A) s_{2(1-p)}(A^T)} \right)$.*

We next sketch how an SVE algorithm can be used to obtain a linear system solver. Let us first assume that the matrix A is positive semidefinite, solving a linear system in A reduces to performing SVE, then applying a conditional rotation by an angle proportional to the inverse of each singular value, and erasing the estimate given by the SVE algorithm.

$$\sum_i \beta_i |v_i\rangle |\overline{\sigma}_i\rangle |0\rangle \to \sum_i \beta_i |v_i\rangle |0\rangle \left(\frac{\sigma_{min}}{\overline{\sigma}_i} |0\rangle + \sqrt{1 - \frac{\sigma_{min}^2}{\overline{\sigma}_i^2}} |1\rangle \right)$$

Postselecting on the last register being $|0\rangle$, one gets a good approximation to the desired output $|A^{-1}b\rangle = \sum_i \frac{\beta_i}{\sigma_i} |v_i\rangle$. To complete the analysis of the linear system solver, we boost the success probability for the SVE (which is $O(\frac{1}{\kappa^2(A)})$) using amplitude amplification by repeating the procedure $O(\kappa(A))$ times. Further, we show in order to obtain a δ-approximate output for the linear system solver the SVE precision should be $O(\delta/\kappa(A))$. A similar analysis applies to matrix multiplication. The same steps can be used for general matrices using the procedure for recovering the sign of the eigenvalues given in [33]. If the matrix A is singular we can achieve the guarantee $\| |z\rangle - |A^+b\rangle \| \leq \delta$ for the Moore-Penrose pseudo-inverse, that is we invert only the non zero eigenvalues.

There are different QRAM data structures for storing A that can achieve $\mu(A) = \|A\|_F$ by [18] and $\mu(A) = \sqrt{s_{2p}(A) s_{2(1-p)}(A^T)}$ for every value of $p \in [0, 1]$ by Result 2. If we are allowed two passes over the stream of the matrix entries A, in the first pass we can compute the minimum of the quantities $\|A\|_F$ and $\sqrt{s_{2p}(A) s_{2(1-p)}(A^T)}$ (for an $O(1)$ sized set $\mathcal{P}$ of values for p in $[0, 1]$) and determine the optimal data structure which is constructed in the second pass. As the two-pass processing requires linear time, this algorithm is in the QRAM data structure model defined in Definition 3.

Result 3. *[Theorem 4.7] There exists an algorithm in the QRAM data structure model that given $|b\rangle$, outputs a quantum state $|z\rangle$ with $\|\,|z\rangle - |\mathcal{A}b\rangle\| \leq \delta$ for $\mathcal{A} \in \{A, A^{-1}\}$ with running time $\tilde{O}(\frac{\kappa^2(A)\mu(A)}{\delta})$ for $\mu(A) := \min_{p \in \mathcal{P}} \left(\|A\|_F, \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)} \right)$ where $\mathcal{P}$ is a set of values in $[0, 1]$ with $|\mathcal{P}| = O(1)$.*

If we are restricted to a single pass over the matrix entries, we can construct two data structures corresponding to $\mu(A) = \|A\|_F$ and $\mu(A) = s_1(A)$ for $p = 1/2$ with a constant overhead, at the end of the pass we will know which data structure achieves the smaller value for μ. These two data structures are expected to be the most useful in practice as they cover the case of low rank matrices and those with bounded ℓ_1 norm.

Let us now see how the linear system solvers in the QRAM data structure model compare to the HHL algorithm and its improvements. The HHL algorithm assumes a weaker input model where the transformation $O_A : |i, j, 0\rangle \to |i, j, a_{ij}\rangle$ can be carried out efficiently. We note that O_A can be implemented efficiently for matrix A stored in the QRAM but also without the QRAM if the matrix A is well structured. Linear system solvers in the QRAM data structure model should not be compared directly to the HHL algorithm and its variants as the input models are different, it is however illuminating to see the improvements one can obtain over HHL in the stronger QRAM data structure model.

The HHL algorithm has running time $\tilde{O}(s(A)^2 \kappa(A)^2/\delta)$ if the absolute values of the eigenvalues of A lie in the interval $[1/\kappa, 1]$ [13] and where $s(A)$ is the sparsity (the maximum number of non zero entries in a row), $\kappa(A)$ the condition number for the matrix A and δ is the approximation error. Subsequent works have improved the running time of quantum linear system solvers to $\tilde{O}(s(A)\kappa(A)\log(1/\delta))$ [2, 10]. In the case of dense matrices, these algorithms require time linear in the dimension of the matrix.

Quantum linear system solvers in the QRAM data structure model have sub-linear running time even for dense matrices. Instead of sparsity, their running time depends on the parameter $\mu(A)$ defined in Result 3.

Let us first consider dense matrices A such that absolute values of the eigenvalues lie in the interval $[1/\kappa, 1]$. The sparsity $s(A) = \Omega(n)$ while $\mu(A) \leq \|A\|_F \leq \sqrt{n}$, the QRAM data structure model is therefore obtains a worst case quadratic speedup over linear system solvers in the HHL input model as observed in [33]. As $\|A\|_F = (\sum_i \sigma_i^2)^{1/2} \leq \sqrt{rk(A)}$, this also is an exponential speedup for dense matrices whose rank is poly-logarithmic in the matrix dimensions. Let us now see an example where Result 3 using the QRAM data structure model provides an exponential speedup over previous algorithms. Let $A = I + J/n$ where J is the all ones matrix, we have $s(A) = \Omega(n)$, $\|A\|_F = \Omega(\sqrt{n})$, while $\mu(A) \leq s_1(A) = O(1)$.

Let us next consider the case of symmetric sparse matrices A such that all entries have absolute value bounded by 1. In this case, we have $s_1(A) \leq s(A)$ for all A. With this scaling, we can compare with the quantum linear system solver [10] in the HHL input model with running time depends on $s(A)$, the QRAM data structure based approach for this case has running time depending on $s_1(A) \leq s(A)$. Lastly, we note that $\mu(A) = \Omega(\sqrt{n})$ for some matrices, meaning that the QRAM data structure based linear system solvers do not provide exponential savings for all matrices.

We also note that a quantum walk that can be used for linear systems with scaling $\mu(A) = \||A|\|_2$ is known [25], where $|A|$ is obtained by taking absolute values of the entries of A. However, an efficient QRAM data structure to implement this walk is not known, such an algorithm would involve updating the singular vectors corresponding to the largest singular value for $|A|$ with cost polylog(n). Finally we note that the running time dependence for the QRAM data-structure based

linear system solver has been further improved to linear in $\kappa(A)$ and then the error to $\log(1/\delta)$ in the recent work [8].

1.3 Quantum iterative methods

Before explaining the quantum iterative method, we provide some necessary information about classical iterative methods.

Classical iterative methods for empirical risk minimization. We consider classical iterative methods in the framework of empirical risk minimization [5], where we are given m examples from a training set (x_i, y_i) with variables $x_i \in \mathbb{R}^n$ and outcome $y_i \in \mathbb{R}$. The model is parametrized by $\theta \in \mathbb{R}^n$ and is obtained by minimizing the following objective function,

$$F(\theta) = \frac{1}{m} \sum_{i \in [m]} \ell(\theta, x_i, y_i) + R(\theta).$$

The loss function $\ell(\theta, x_i, y_i)$ assigns a penalty when the model does not predict the outcome y_i well for the example (x_i, y_i) while the regularization term $R(\theta)$ penalizes models with high complexity.

The first order method for problems in this framework is called gradient descent. The algorithm starts with $\theta_0 \in \mathbb{R}^n$, and for τ steps updates θ via the following update rule:

$$\theta_{t+1} = \theta_t + \alpha \nabla F(\theta_t)$$

In the end, for a large class of loss functions $\ell(\theta, x_i, y_i)$ it outputs θ_τ which is guaranteed to be close to the solution for sufficiently large τ. The running time is τC_S, where C_S is the cost of a single step, in other words it is the cost of the update.

An important subclass of the empirical loss minimization framework is when the gradient is an affine function, as for weighted least squares and ridge regression problems. More generally, the gradient is an affine function for quadratic optimization problems of the form $\min_{x \in \mathbb{R}^n} x^T A x + b^T x + c$ for some $A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n, c \in \mathbb{R}$. In these cases, the method starts with some θ_0 and for $t \geq 0$ updates it via an update rule of the form,

$$\theta_{t+1} = \theta_t + \alpha r_t$$

where α is a scalar that denotes the step size and $r_t = A\theta_t + b := L(\theta_t)$. It is easy to see that this also implies that $r_{t+1} = S(r_t)$ for a linear operator S. Indeed,

$$r_{t+1} = L(\theta_{t+1}) = L(\theta_t + \alpha r_t) = A(\theta_t + \alpha r_t) + b = r_t + \alpha A r_t := S(r_t)$$

The final state of a linear update iterative method can hence be written as

$$\theta_\tau = \theta_0 + \alpha \sum_{t=0}^{\tau-1} r_t = \theta_0 + \alpha L(\theta_0) + \alpha \sum_{t=1}^{\tau-1} S^t(r_0). \tag{2}$$

where S^t is the operator that applies S for t time steps and S^0 is the identity operator.

We are slightly going to change notation in order to make the presentation of the quantum algorithm clearer. We rename θ_0 as r_0, which means that $L(\theta_0)$ is renamed as $L(r_0)$. This way, we have

$$\theta_\tau = r_0 + \alpha L(r_0) + \alpha \sum_{t=1}^{\tau-1} S^t(L(r_0)) = r_0 + \alpha \sum_{t=1}^{\tau} S^{t-1}(L(r_0)).$$

Without loss of generality we assume the initial point has unit norm, i.e. $\|r_0\| = 1$.

A quantum gradient descent algorithm with affine updates. Let us make things simpler for this exposition by looking at the case where we take $r_0 = 0$ and $\alpha = 1$, meaning that we want to output the unnormalized state $|\theta_\tau\rangle = \sum_t r_t$. We only make this assumption here for conveying the main ideas and not in the proofs where we address the most general case.

Imagine that there was a procedure that performs the following mapping perfectly

$$|t\rangle \,\|\theta_t\|\, |\theta_t\rangle \to |t+1\rangle \,\|\theta_{t+1}\|\, |\theta_{t+1}\rangle$$

Then, our task would be easy, since applying this unitary τ times would provide us with the desired state $|\theta_\tau\rangle$. However, this is not the case. The mapping θ_t to θ_{t+1} is not even a unitary transformation, and the norm of θ_{t+1} can be larger than the one of θ_t. Even so, imagine one could in fact perform this mapping with some "probability" (meaning mapping θ_t to some state $(\beta\|\theta_{t+1}\| |\theta_{t+1}\rangle |0\rangle + \sqrt{1-\beta^2} |G\rangle |1\rangle)$, for some garbage state G). The main problem is that one cannot amplify this amplitude, since the state $|\theta_{t+1}\rangle$ is unknown, being the intermediate step of the iterative method, and in the quantum case we only have a single copy of this state. Hence, the issue with the iterative method is that one needs to perform τ sequential steps, where each one may have some constant probability of success without the possibility of amplifying this probability. The probability of getting the desired final state is proportional to the product of the success probabilities for each step, which drops exponentially with the number of steps τ. This is also the reason previous attempts for a quantum gradient descent algorithm break down after a logarithmic number of steps [27].

Here we manage to overcome this obstacle in the following way. The first idea is to deal with the vectors r_t instead of the θ_t's, since in this case, we know that the norm of r_{t+1} is smaller than the norm of r_t. Our goal would be to find a unitary mapping that, in some sense, maps r_t to r_{t+1}. Again, there is the problem that the norms are not equal, but in this case, since the norm of r_{t+1} is smaller, we can make it into a unitary mapping by adding some garbage state. Indeed, we define the quantum step of the quantum iterative method via the following unitary

$$|t\rangle \,\|r_t\|\, |r_t\rangle \xrightarrow{V} |t+1\rangle \,(\|r_{t+1}\| |r_{t+1}\rangle |0\rangle + |G\rangle |1\rangle),$$

where the norm of the garbage state is such that the norm of the right hand side is equal to $\|r_t\|$. Note that the above vectors are not unit norm but V is still length preserving. Since we are dealing with linear updates, the above transformation is a matrix multiplication, and we use the SVE procedure to perform it with high accuracy.

The second idea is noticing that our goal now is not to obtain the final state r_τ, but the sum of all the vectors $\sum_t r_t$. Let us see how to construct this efficiently. Given a procedure for performing one step of the iterative method as above, we design another procedure U that given as input a time t and the initial state r_0 can map r_0 to r_t. We do this by applying t-times the unitary V, conditioned on the first register. In other words, we can perform the mapping

$$|t, r_0\rangle \xrightarrow{U} |t\rangle \,(\|r_t\| |r_t\rangle |0\rangle + |G\rangle |1\rangle).$$

Note that if the cost of V is C_V, then the cost of U can be at most τC_V by applying V sequentially, where the error also increases by a factor of τ. We will see that in fact for some cases we can implement C_U in time $O(C_V + \log \tau)$.

We are now ready for the last step of the algorithm that consists in starting with a superposition of time steps from 0 to τ and applying U, in order to get a superposition of the form

$$\frac{1}{\sqrt{\tau}} \sum_t |t\rangle |r_0\rangle \to \frac{1}{\sqrt{\tau}} \sum_t |t\rangle \left(\|r_t\| |r_t\rangle |0\rangle + |G\rangle |1\rangle \right).$$

Then, we can "erase" the time register by performing a Hadamard on the first register and accepting the result when the first register is 0. In other words, we are having a state of the form

$$\frac{1}{\tau} \sum_t \|r_t\| |r_t\rangle |0\rangle + |G'\rangle |1\rangle$$

Using Amplitude Amplification, we can get the desired state $\frac{1}{\|\theta_\tau\|} \sum_t \|r_t\| |r_t\rangle$, in overall time $O(\frac{\tau}{\|\theta_\tau\|})$ times the cost of applying the unitary U, and since in our applications $\|\theta_\tau\| = \Omega(1)$ we get the efficient quantum gradient descent algorithm.

Result 4 (Informal version of Theorem 3.3, Proposition 4.10). *Given a unitary V that approximately applies one step of the iterative method with affine update rules in time C_V, there exists a quantum algorithm that performs τ steps of the iterative method in equation (2) and outputs a state close to θ_τ, in time at most $O(\tau C_V)$.*

Theorem 3.3 provides a precise statement, the cost is $O(\tau C_U)$ for a unitary U with cost at most $O(\tau C_V)$. Proposition 4.10 shows that $C_U = O(C_V)$ for the iterative method in equation (2) where the matrix A is fixed. If the matrices used for the iterative method are variable then the running time is $O(\tau^2 C_V)$, this case occurs for the quantum stochastic gradient descent algorithm. For the classical iterative method in equation (2), our running time is linear in the number of steps times the cost of taking one step of the iterative method. A single step of the iterative method in the quantum case uses the SVE algorithm and its cost can be considerably smaller than that for classical algorithms.

Our algorithm does not create all the intermediate states θ_t, which we do not know how to achieve with non-negligible probability for large τ. Instead, we observe that the final state θ_τ is equal to the sum of all the update states r_t and then we try to create the sum of these states. We first go to a superposition of all time steps from 0 to τ and then conditioned on the time being t we apply coherently t updates to the initial state r_0 in order to create a sort of "history" quantum state. This is reminiscent of the "history" states in Kitaev's QMA-completeness proof for the Local-Hamiltonian problem [21]. Last, erasing the register that keeps the time can be done in time linear in the number of time steps, which is still efficient. History states have been used in the past in many different scenarios, for example, a construction using history states to compose solutions of the HHL algorithm was developed independently for quantum differential equation solvers [4]. We believe this technique will have further applications in quantum machine learning.

1.4 Applications of quantum gradient descent

Positive semidefinite linear systems. The simplest application for quantum gradient descent is an iterative algorithm for solving positive semidefinite linear systems. The iterative algorithm method for this application incurs an extra factor of $\kappa(A)$ compared to a direct method using Result 3. It is still interesting to have an alternative method for linear systems even if the asymptotic running time is slightly worse.

Classically, in machine learning settings with noisy data one uses gradient descent methods for solving linear systems instead of the direct method as it offers a tradeoff between time and accuracy. If the data is noisy, we do not require high accuracy solutions to the linear system, and gradient descent provides the solutions faster than the direct method.

For this application, we are given a positive semidefinite matrix A and a vector b, output a state close to $|A^{-1}b\rangle$. Let $A|b$ denote the matrix with row b added to A. We have the following result,

Result 5. *[Theorem 5.1] Let A be a positive semidefinite matrix and b a vector, there is an iterative QRAM-based quantum algorithm that outputs a state $|z\rangle$ such that $\||z\rangle - |A^{-1}b\rangle\| \le 2\delta$ with expected running time $\tilde{O}(\frac{\kappa(A)^3 \mu(A|b)}{\delta})$.*

We will see in the next application to least squares problems that the quantum iterative method offers additional advantages.

Stochastic gradient descent for weighted least squares. Our main application of the quantum iterative method is to the weighted least squares problem. Here, we are given a matrix X of examples and a corresponding vector y of labels, as well as a vector w of weights, and the goal is to find θ that minimizes the squared loss $\sum_i w_i(y_i - x_i^T \theta)^2$. The problem has a closed form solution given by

$$\theta = (X^T W X)^{-1} X^T W y,$$

so it can also be solved using a direct method. Quantum algorithms for unweighted least squares problems with a polynomial dependence on sparsity in the HHL input model have been described [32]. Other approaches to least squares via column sampling have also been considered [22]. We extend these works in two ways: first, using our improved SVE algorithm we can perform matrix multiplication and inversion efficiently for a larger class of matrices; second, we solve the weighted version of the problem.

More importantly, we are able to give an iterative stochastic gradient method for this problem which has many advantages in practical settings (see for example [5] for a more detailed discussion). The least squares problem is used in practice for regression or data fitting, where in many cases the data matrix is skewed in shape since the number of data points is an order of magnitude larger than the dimension of the data points. In such cases, it is prohibitive to perform classical linear algebra operations using the entire data set, and moreover, due to redundancy in the data, the gradient can be estimated efficiently over small batches.

For these reasons, the gradient is estimated over randomly sampled batches of the training set, an approach which is called stochastic gradient descent. This way, the stochastic gradient descent avoids having to perform linear algebra operations on large matrices, which would be the case if we were to solve the problem directly or using gradient descent. Our quantum iterative method can also be used to perform stochastic gradient descent for the above problems.

There are two significant advantages of the quantum stochastic gradient descent algorithm. First, the stochastic gradient descent algorithm has much reduced quantum memory requirements compared to the HHL algorithm or directly solving the linear system in the QRAM data structure model. Similar to the classical setting, the data is split randomly into batches, and for every step of the iterative method, only one batch needs to be in quantum memory. This would be very useful in a situation where the size of QRAMs is limited by the difficulty of maintaining coherent superpositions over a large number of qubits. In such a situation, it may be possible to build a 1Mb QRAM but impossible to combine 10 such devices to obtain a 10Mb QRAM. Due to

the difficulties in designing QRAMs we believe that this is a significant advantage offered by the quantum stochastic gradient descent algorithm.

In order to reduce the slowdown caused by the loading of data into the QRAM, the quantum stochastic gradient descent method can be used with multiple quantum memory devices each with limited capacity. In this setting, the algorithm would perform a computation using one of the memory devices, while data is loaded concurrently into the others. Such memory architectures can simulate a larger memory using smaller sized devices, they are also widely used in classical computing. Again, the actual quantum speedup achieved for this setting would depend on a number of hardware parameters, the advantage of the quantum stochastic gradient descent method is to provide greater flexibility and to enable quantum data processing applications for large sized datasets. Further analyses of such architectures can be interesting future work.

Secondly, the quantum stochastic gradient algorithm is particularly useful in settings where the linear system matrix is not available a priori but only revealed partially as the algorithm makes progress. Such settings are common for dynamic programming type algorithms in reinforcement learning [28]. They arise for instance for the problem of playing video games where the states of the game and the actions are revealed to the algorithm only when the algorithm moves ahead into the game and are not known in the very beginning of the game. Stochastic gradient descent can be used for optimizing the strategy in such a game. This offers an example where a direct application of the HHL algorithm would not suffice as the entire matrix is not known in advance, but the quantum stochastic gradient descent algorithm would enable such an application since at every step the gradient would be estimated via the information that has already been revealed.

The main technical difference between our algorithms for linear systems and least squares is that in the latter, one needs to perform matrix operations for a matrix that is not a priori stored in memory. More precisely, we have in memory the matrix X, the diagonal matrix W and a vector y and we need to perform matrix multiplication with the matrix $(X^TWX)^{-1}$ and also create the vector X^TWy. We do this by first using the known weight vector W to update the QRAM data structure then invoking the matrix multiplication procedure with the updated data structure.

Result 6. *[Theorem* 5.2*] Let X be an arbitrary matrix, W a diagonal matrix and y a vector, there is an iterative QRAM-based quantum algorithm that outputs $|z\rangle$ such that $\|\,|z\rangle - |(X^TWX)^{-1}X^TWy\rangle\| \leq 2\delta$ in expected time $\tilde{O}(\frac{\kappa(X^TWX)^3\mu(\sqrt{W}X|y)}{\delta})$.*

2 Quantum Preliminaries

2.1 Quantum Algorithms

We will use phase estimation and variants of amplitude amplification that we recall below. The time required to implement a unitary operator U will be denoted by $T(U)$.

Theorem 2.1. *[Consistent phase estimation, [30]] Let U be a unitary operator with eigenvectors $|v_j\rangle$ and eigenvalues $e^{\iota\theta_j}$ for $\theta_j \in [-\pi, \pi]$. There exists a quantum algorithm with running time $O(T(U)\frac{\log n}{\epsilon})$ that transforms $|\phi\rangle = \sum_{j\in[n]} \alpha_j |v_j\rangle \to \sum_{j\in[n]} \alpha_j |v_j\rangle |\overline{\theta_j}\rangle$ where $\overline{\theta_j}$ is an estimate such that $|\overline{\theta_j} - \theta_j| \leq \epsilon$ for all $j \in [n]$ with probability at least $1 - 1/\mathrm{poly}(n)$.*

The difference with phase estimation result in [20] is that the usual phase estimation does not output a fixed estimate $\overline{\theta_j}$, the estimate can in some cases assume one of two possible values, while

here we require that the estimate be fixed with high probability. The estimate however is not deterministic, it depends on the random shifts used in the consistent phase estimation algorithm. An alternate consistent eigenvalue estimation procedure is sketched in [2, 17]. We also need a version of amplitude amplification and estimation [6],

Theorem 2.2. *[Amplitude amplification and estimation [6]] If there is unitary operator U such that $U |0\rangle^l = |\phi\rangle = \sin(\theta) |x, 0\rangle + \cos(\theta) |G, 0^\perp\rangle$ where $|G\rangle$ is an arbitrary garbage state and $|0^\perp\rangle$ is any state orthogonal to $|0\rangle$ then $\sin^2(\theta)$ can be estimated within error $(1 \pm \epsilon)$ in time $O(\frac{T(U)}{\epsilon \sin(\theta)})$ and $|x\rangle$ can be generated in expected time $O(\frac{T(U)}{\sin(\theta)})$.*

Note that for this statement, if the amplitude amplification succeeds then we measure $|0\rangle$ and obtain the exact state $|x\rangle$, there is no approximation error involved. Last we provide a simple claim that shows that if two unnormalized vectors $\phi, \tilde{\phi}$ are close to each other, then their normalized versions $|\phi\rangle, |\tilde{\phi}\rangle$ are also relatively close.

Claim 2.3. *Let θ be the angle between vectors $\phi, \tilde{\phi}$ and assume that $\theta < \pi/2$. Then, $\|\phi - \tilde{\phi}\| \leq \epsilon$ implies $\||\phi\rangle - |\tilde{\phi}\rangle\| \leq \frac{\sqrt{2}\epsilon}{\|\phi\|}$.*

Proof. We bound the ℓ_2 distance $\||\phi\rangle - |\tilde{\phi}\rangle\|$ using the following argument. Let θ be the angle between $\phi, \tilde{\phi}$. For the unnormalized vectors we have $\|\phi - \tilde{\phi}\| \leq \epsilon$, and assuming that $\theta < \pi/2$ we have $\epsilon \geq \|\phi\| \sin(\theta)$. The distance between the normalized states can thus be bounded as,

$$\||\phi\rangle - |\tilde{\phi}\rangle\|^2 = (2\sin(\theta/2))^2 \leq 2\sin^2(\theta) \leq \frac{2\epsilon^2}{\|\phi\|^2} \tag{3}$$

$\square$

3 The Quantum Gradient Descent algorithm

In this section, we define a quantum step of the quantum gradient descent in the case of a linear update rule and then describe the full quantum procedure that performs the quantum iterative method.

3.1 The quantum step

We assume that the classical iterative method has an update rule of the form,

$$\theta_\tau = r_0 + \alpha \sum_{t=1}^{\tau} S^{t-1}(L(r_0)) = r_0 + \alpha \sum_{t \in [\tau]} r_t$$

for an affine operator L, a linear, contracting operator S, and an arbitrary initial vector r_0 with $\|r_0\| = 1$. This is the case, for example, for solving linear systems or least squares.

First, we define the notion of an approximate quantum step of the quantum iterative method. Let us denote by τ the number of steps of the classical iterative algorithm, and let $\tau + 1 = 2^\ell$ (if not just increase one can τ to the next power of 2). A contracting linear operator L is an operator such that $\|Lx\| \leq \|x\|$ for all x.

Definition 3.1. *An (ϵ, δ)-approximate quantum step corresponding to a classical iterative method with update rule $\theta_\tau = r_0 + \alpha \sum_{t=1}^{\tau} S^{t-1}(L(r_0))$ for an affine operator L, a linear, contracting operator S, initial vector r_0 and residual vectors $r_t = S^{t-1}(L(r_0))$ is a transformation V such that for any $1 \leq t \leq \tau - 1$,*

$$V : |0\rangle |r_0\rangle |0\rangle \rightarrow |1\rangle \left(\alpha \|\tilde{L}(r_0)\| |\tilde{L}(r_0)\rangle |0\rangle + |G_1\rangle |1\rangle \right)$$

$$: |t\rangle \|r_t\| |r_t\rangle |0\rangle \rightarrow |t+1\rangle \left(\|\tilde{S}(r_t)\| |\tilde{S}(r_t)\rangle |0\rangle + |G_{t+1}\rangle |1\rangle \right),$$

where $|G_{t+1}\rangle$, for $0 \leq t \leq \tau - 1$ is an unnormalised garbage state, $\tilde{L}$ is an approximation to $L : r_0 \rightarrow r_1$, $\tilde{S}$ is an approximation to $S : r_t \rightarrow r_{t+1}$, such that $\|L(r_0) - \tilde{L}(r_0)\| \leq \epsilon$ with probability $\geq 1 - \delta$, and for each $t \in [\tau - 1]$ we have $\|S(r_t) - \tilde{S}(r_t)\| \leq \epsilon$ with probability $\geq 1 - \delta$.

Notice that $\|\tilde{L}(r_0)\|$ might be larger than 1, but by taking α a small constant we have $\alpha \|\tilde{L}(r_0)\| \leq 1$. The way we defined V, it is norm preserving, but the vectors that we wrote do not have unit norm. We can, of course, normalize them by dividing both sides by $\|r_t\|$. We prefer this notation in order to resemble more the classical mapping of the unnormalized vectors $r_t \rightarrow r_{t+1}$.

We also note that the transformation V will be eventually implemented using a singular value estimation procedure what succeeds with probability $1 - 1/poly(n)$, thus $\delta = 1/poly(n)$ is the probability with which the estimates for $L(r_0)$ and $S(r_t)$ are correct. We can define the following procedure U similar to the ideal case.

Claim 3.2. *Given access to an (ϵ, δ)-approximate quantum step procedure V with cost C_V, there exists an (ϵ, δ)-approximate quantum multistep procedure U with cost at most $O(\tau C_V)$, such that for any $t \in [\tau]$,*

$$U : |0\rangle |0\rangle |0\rangle |r_0\rangle |0\rangle \rightarrow |0\rangle |0\rangle |0\rangle |r_0\rangle |0\rangle$$

$$: |t\rangle |0\rangle |0\rangle |r_0\rangle |0\rangle \rightarrow |t\rangle |0\rangle \left(\alpha \|\tilde{S}^{t-1}(\tilde{L}(r_0))\| |t\rangle |\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle |0\rangle + |G'_t\rangle |1\rangle \right),$$

where $|G'_t\rangle$ is an unnormalised garbage state, and with probability at least $(1 - t\delta)$ it holds that $\|S^{t-1}(L(r_0)) - \tilde{S}^{t-1}(\tilde{L}(r_0))\| \leq t\epsilon$, where S, L are as in Definition 3.1.

Note that out of the five registers used in this claim, registers 1 and 3 store the time step, 4 stores the quantum state for the iterative method while 2 and 5 are control qubits or flags. Moreover, we take into account the fact that the error in U increases by a factor of t.

Proof. We define the operator W on four registers, such that if the control register is 0, then it applies a V on the other three registers and then a CNOT to copy the last register into the control register. If the control register is 1, then it does nothing. Namely

$$W : |0\rangle |0\rangle |r_0\rangle |0\rangle \rightarrow |0\rangle |1\rangle \alpha \|\tilde{L}(r_0)\| |\tilde{L}(r_0)\rangle |0\rangle + |1\rangle |1\rangle |G_1\rangle |1\rangle$$

$$: |0\rangle |t\rangle \|r_t\| |r_t\rangle |0\rangle \rightarrow |0\rangle |t+1\rangle \|\tilde{S}(r_t)\| |\tilde{S}(r_t)\rangle |0\rangle + |1\rangle |t+1\rangle |G_{t+1}\rangle |1\rangle, \quad t \in [1, \tau - 1]$$

$$: |1\rangle |t\rangle |b\rangle |1\rangle \rightarrow |1\rangle |t\rangle |b\rangle |1\rangle \quad t \in [0, \tau - 1]$$

We define the following procedure U that acts as identity for $t = 0$ and for $t \in [\tau - 1]$ it does the following:

$$|t\rangle |0\rangle |0\rangle |r_0\rangle |0\rangle \quad \xrightarrow{W^t} \quad |t\rangle W^t |0\rangle |0\rangle |r_0\rangle |0\rangle$$

$$= \quad |t\rangle |0\rangle |t\rangle \alpha \|\tilde{S}^{t-1}(\tilde{L}(r_0))\| |\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle |0\rangle + |t\rangle |1\rangle \sum_{i=1}^{T} |i\rangle |G_i\rangle |1\rangle \quad (4)$$

$$\xrightarrow{CNOT_{5,2}} \quad |t\rangle |0\rangle \left(\alpha \|\tilde{S}^{t-1}(\tilde{L}(r_0))\| |t\rangle |\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle |0\rangle + |G'_t\rangle |1\rangle \right)$$

The equality in equation (4) follows from the definition of W. The cost of applying W is cost C_V, thus the entire procedure can be perfomed in time $O(\tau C_V)$. We prove the properties by induction on t. For $t = 1$ we use the definition of the quantum step V and the property holds. Assume it holds for $t - 1$, i.e. with probability $1 - (t-1)\delta$ we have $\|S^{t-2}(L(r_0)) - \tilde{S}^{t-2}(\tilde{L}(r_0))\| \leq (t-1)\epsilon$.

Then, we have

$$
\begin{aligned}
\|S^{t-1}(L(r_0)) - \tilde{S}^{t-1}(\tilde{L}(r_0))\| \quad &\leq \quad \|S(S^{t-2}(L(r_0))) - S(\tilde{S}^{t-2}(\tilde{L}(r_0)))\| \\
&\quad + \|S(\tilde{S}^{t-2}(\tilde{L}(r_0))) - \tilde{S}(\tilde{S}^{t-2}(\tilde{L}(r_0)))\| \\
&\leq \quad \|S^{t-2}(L(r_0)) - \tilde{S}^{t-2}(\tilde{L}(r_0))\| \\
&\quad + \|S(\tilde{S}^{t-2}(\tilde{L}(r_0))) - \tilde{S}(\tilde{S}^{t-2}(\tilde{L}(r_0)))\|
\end{aligned}
$$

where we used the fact that S is contractive. Also, by definition of the iterative step, with probability $(1 - \delta)$ we have $\|S(\tilde{S}^{t-2}(\tilde{L}(r_0))) - \tilde{S}(\tilde{S}^{t-2}(\tilde{L}(r_0)))\| \leq \epsilon$ and with probability $1 - (t-1)\delta$, by induction hypothesis, we have $\|S^{t-2}(L(r_0)) - \tilde{S}^{t-2}(\tilde{L}(r_0))\| \leq (t-1)\epsilon$. Hence overall, with probability at least $1 - t\delta$, we have

$$\|S^{t-1}(L(r_0)) - \tilde{S}^{t-1}(\tilde{L}(r_0))\| \leq t\epsilon.$$

$$\square$$

3.2 The Quantum Iterative Method: general case

The main part of the quantum iterative method is the efficient construction of a unitary Q defined below using the results from Claim 3.2. We then use amplitude amplification and estimation on Q to improve the running time of the method.

The Quantum Iterative Method Use Amplitude Amplification and Estimation with unitary Q

$$Q : |0\rangle^\ell \to \frac{1}{T} |\tilde{\theta}_\tau\rangle |0\rangle + |G\rangle |0^\perp\rangle$$

where $|G\rangle$ is a garbage state and $|0^\perp\rangle$ denotes a state that is orthogonal to $|0\rangle$, to output $|\tilde{\theta}_\tau\rangle$ and an estimate for $\|\tilde{\theta}_\tau\|$. The parameter T is taken to be $\frac{\tau + 1}{\|\tilde{\theta}_\tau\|}$. We next provide a procedure for implementing Q.

Implementation of the unitary $Q : |0\rangle^\ell \to \frac{1}{T} |\tilde{\theta}_\tau\rangle |0\rangle + |G\rangle |0^\perp\rangle$

1. Create the state $\frac{1}{\sqrt{\tau+1}}\sum_{t=0}^{\tau}|t\rangle\,|0\rangle\,|0\rangle\,|r_0\rangle\,|0\rangle$

2. Apply the unitary procedure U from Claim 3.2 and trace out the second register to get

$$\frac{1}{\sqrt{\tau+1}}\,|0\rangle\,|0\rangle\,|r_0\rangle\,|0\rangle + \frac{1}{\sqrt{\tau+1}}\sum_{t=1}^{\tau}|t\rangle\left(\alpha\|\tilde{S}^{t-1}(\tilde{L}(r_0))\|\,|t\rangle\,|\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle\,|0\rangle + |G_t'\rangle\,|1\rangle\right)$$

3. Conditioned on the last register being 0, perform a $CNOT_{1,2}$ to erase the second copy of t and then by exchanging the place of the second and third register we get

$$\frac{1}{\sqrt{\tau+1}}\,|0\rangle\,|r_0\rangle\,|0\rangle\,|0\rangle + \frac{1}{\sqrt{\tau+1}}\sum_{t=1}^{\tau}|t\rangle\left(\alpha\|\tilde{S}^{t-1}(\tilde{L}(r_0))\|\,|\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle\,|0\rangle\,|0\rangle + |G_t'\rangle\,|1\rangle\right)$$

4. Conditioned on the last register being 0 perform a Hadamard on all qubits of the first register and then by exchanging the place of the first and second register we get

$$\frac{1}{\tau+1}\sum_{y=0}^{\tau}\left(|r_0\rangle + \alpha\sum_{t=1}^{\tau}(-1)^{y\cdot t}\|\tilde{S}^{t-1}(\tilde{L}(r_0))\|\,|\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle\right)|y\rangle\,|0\rangle\,|0\rangle + |G''\rangle\,|1\rangle =$$

$$\frac{\|\tilde{\theta}_\tau\|}{\tau+1}\left(\frac{1}{\|\tilde{\theta}_\tau\|}\left(|r_0\rangle + \alpha\sum_{t=1}^{\tau}\|\tilde{S}^{t-1}(\tilde{L}(r_0))\|\,|\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle\right)\right)|0\rangle\,|0\rangle\,|0\rangle + |G\rangle\,(|0\rangle\,|0\rangle\,|0\rangle)^\perp =$$

$$\frac{1}{T}\,|\tilde{\theta}_\tau\rangle\,|0\rangle + |G\rangle\,|0^\perp\rangle$$

with $|\tilde{\theta}_\tau\rangle = \frac{1}{\|\tilde{\theta}_\tau\|}\left(|r_0\rangle + \alpha\sum_{t=1}^{\tau}\|\tilde{S}^{t-1}(\tilde{L}(r_0))\|\,|\tilde{S}^{t-1}(\tilde{L}(r_0))\rangle\right)$ and $T = \frac{\tau+1}{\|\tilde{\theta}_\tau\|}$.

3.3   Analysis

The main result for this section that proves the correctness and bounds the running time for the quantum iterative method is the following:

Theorem 3.3. *Given unitary U with cost C_U that implements an (ϵ,δ)-approximate quantum multistep procedure (as in Claim 3.2) for an iterative method with affine update rules and τ steps, there is a quantum algorithm with running time $O(T(C_U + \log\tau))$ with $T = \frac{\tau+1}{\|\tilde{\theta}_\tau\|}$ that outputs a state $|\tilde{\theta}_\tau\rangle$ such that,*

$$\||\theta_\tau\rangle - |\tilde{\theta}_\tau\rangle\| \le \frac{\sqrt{2}\alpha\tau^2\epsilon}{\|\theta_\tau\|}.$$

We prove separately the statements about the correctness and the running time for the quantum iterative method below, together with some additional comments about the applications.

3.3.1   Correctness

We calculate the ℓ_2 distance between the final state $|\tilde{\theta}_\tau\rangle$ and the correct state which is given by $|\theta_\tau\rangle = \frac{1}{\|\theta_\tau\|}\left(|r_0\rangle + \alpha\sum_{t=1}^{\tau}\|S^{t-1}(L(r_0))\|\,|S^{t-1}(L(r_0))\rangle\right)$. We first compute the non-normalised distance when a (ϵ,δ) approximate step is used for implementing Q,

$$\|\theta_\tau - \tilde{\theta}_\tau\| \le \alpha \sum_{t=1}^{\tau} \|S^{t-1}(L(r_0)) - \tilde{S}^{t-1}(\tilde{L}(r_0))\| \le \alpha \sum_{t=1}^{\tau} t\epsilon \le \alpha\tau^2\epsilon$$

Then by Claim 2.3 we can also bound the normalized distance,

$$\||\theta_\tau\rangle - |\tilde{\theta}_\tau\rangle\| \le \frac{\sqrt{2}\alpha\tau^2\epsilon}{\|\theta_\tau\|}.$$

This proves the statement about correctness for Theorem 3.3.

Further, we note that Amplitude Estimation outputs an estimate $\|\tilde{\theta}_\tau\|$ such that $\frac{\|\tilde{\theta}_\tau\|}{\|\theta_\tau\|} \in (1 \pm \xi)$ with a constant overhead (where ξ is a small constant). In the applications we will consider, $\|\theta_\tau\|$ is at least $\Omega(1)$ and at most $O(\tau)$ and $\alpha = O(1)$. Hence, again, by taking $\epsilon = O(\frac{1}{\tau^2})$ appropriately small, we can ensure that the approximation error for the ℓ_2 distance is less than a suitably small constant.

3.3.2 Running time

The expected running time is the expected running time of Amplitude Amplification (which is the same as that of Amplitude Estimation), which is T times the cost of implementing the unitary Q, which is $O(C_U + \log \tau)$. Overall, the expected running time is $O(T(C_U + \log \tau))$ as claimed in Theorem 3.3.

Let us make some additional remarks. In our applications we will have $\|\theta_\tau\| = \Omega(1)$, which also implies that $\|\tilde{\theta}_\tau\| \ge \|\theta_\tau\| - \alpha\tau^2\epsilon = \Omega(1)$ for appropriately small ϵ. Hence, the running time for the applications will be $O(\tau(C_U + \log \tau))$. In many applications including psd linear systems and weighted least squares problems, each step of the iterative method involves multiplication by a fixed matrix. In this case, we can implement U with cost $C_U = O(C_V + \log \tau)$ as shown in Proposition 4.10) and hence get an overall running time $O(\tau(C_V + \log \tau))$. However, the cost C_U can be $O(\tau C_V)$ for the case when each step of the iterative method involves multiplication by a different matrix. In this case the worst-case running time is $O(\tau^2 C_V)$, this running time occurs in our application to stochastic gradient descent for the weighted least squares problem.

For the quantum iterative method to approximate the output state to error δ', we would need the error ϵ in implementing V to be $O(\frac{\delta'}{\tau^2})$, which would make the error in U at most $O(\frac{\delta'}{\tau})$. We implement V using our linear system solver (Theorem 5.1) and therefore achieve a running time of $O(\tau(\tau^2\mu(A))/\delta)$, which is superlinear in τ.

The running time of linear system solvers in the QRAM data structure model has been improved to $O(\mu(A)\kappa(A)\text{polylog}(1/\epsilon))$ in the recent work [8]. More precisely, they presented a quantum linear system solver with running time $O(\mu(A)\kappa(A)\text{polylog}(1/\epsilon))$ given a unitary block encoding for A, and also showed that our results on QRAM data structures presented here are equivalent to the construction of unitary block encodings with parameter $\mu(A)$. If we use the improved linear system solver, the running time would be $O(\tau\mu(A)\log(\tau^2/\delta))$, that is the dependence on the number of steps is indeed linear.

4 Improved quantum algorithms for matrix multiplication and linear systems

In Section 4.1 and 4.2 we generalize the data structure for state preparation and the quantum algorithm used for singular value estimation that we had proposed in [18]. The generalized singular value estimation algorithm has a faster running time for several classes of matrices. We use the improved singular value estimation algorithm for solving quantum linear systems and quantum matrix multiplication in Section 4.3. Finally, in Section 4.4 we show how to implement a single step of the quantum iterative method.

4.1 A generalized state preparation data structure

The QRAM data structure in [18] enabled the efficient preparation of vector states $|v\rangle$ for vectors v stored in the QRAM. We provide a more general data structure that enables the efficient preparation of more general normalized states corresponding to the rows/columns of a matrix. We begin by defining the notion of a normalized state.

Definition 4.1. *The normalized vector state corresponding to vector $x \in \mathbb{R}^n$ and $M \in \mathbb{R}$ such that $\|x\|^2 \leq M$ is the quantum state $|\overline{x}\rangle = \frac{1}{\sqrt{M}} \left(\sum_{i \in [n]} x_i |i\rangle + (M - \|x\|^2)^{1/2} |n+1\rangle \right)$.*

We recall that in the QRAM data structure model, the entries of the matrix A arrive in an online manner and are stored in a data structure. The insertion and update times for the data structure are poly-logarithmic per entry, so the time required to construct the data structure is $O(w \log^2(mn))$ where w is the number of non zero entries in A. We have the following theorem that describes the efficient QRAM data structure as in Definition 2, this data structure will be used to obtain the improved singular value estimation algorithm.

Theorem 4.2. *Let $A \in \mathbb{R}^{m \times n}$ and $M = \max_{i \in [m]} \|a_i\|^2$. There is an efficient QRAM data structure for storing matrix entries (i, j, a_{ij}) such that access to this data structure allows a quantum algorithm to implement the following unitary in time $\widetilde{O}(\log(mn))$.*

$$U |i, 0^{\lceil \log(n+1) \rceil}\rangle = |i\rangle \frac{1}{\sqrt{M}} \left(\sum_{j \in [n]} a_{ij} |j\rangle + (M - \|a_i\|^2)^{1/2} |n+1\rangle \right) \tag{5}$$

Proof. The data structure maintains an array of m binary trees $B_i, i \in [m]$ one for each row of the matrix. The leaf node j of tree B_i, if present, stores $(a_{ij}^2, sign(a_{ij}))$. An internal node u stores the sum of the values of the leaf nodes in the subtree rooted at u. In addition, there is an extra node M that, at any instant of time stores the maximum row norm $M = \max_{i \in [m]} \|a_i\|^2$ for the matrix A currently stored in the data structure.

The data structure is initially empty, and the value stored in node M is 0. We next describe the update when entry (i, j, a_{ij}) is inserted or updated, note that the insertion and update must be carried out in time $O(\text{polylog}(n))$ for the data structure to be efficient. After giving the update procedure, we provide the efficient implementation for the unitary U in (5) given access to the data structure.

The update algorithm on receiving input (i, j, a_{ij}) creates leaf node j in tree B_i if not present and updates it otherwise. Deletion is a special case of update and is equivalent to the case $(0, i, j)$.

The algorithm updates the value of all nodes in the path between the leaf and the root of the tree. The update requires time $O(\log^2(mn))$ as at most $O(\log n)$ nodes on the path from node j to the root in the tree B_i are updated and each update requires time $O(\log(mn))$ to find the address and the value of the node being updated. At the end of each update the root of B_i stores the squared norm $\|a_i\|^2$ for the vector stored in B_i. The algorithm then compares M with $\|a_i\|^2$ and updates the maximum if $\|a_i\|^2 > M$, this additional step requires time $O(\log n)$. It follows that the data structure is efficient and that after the update, the value stored in the node M is $\max_{i \in [m]} \|a_i\|^2$.

In order to implement U we first perform a controlled rotation on the second register using the values M and $\|a_i\|$ which is stored at the root of B_i. We also introduce a tag qubit and tag the part of the state on the second register with value $|n+1\rangle$,

$$|i, 0^{\lceil \log(n+1) \rceil}\rangle \rightarrow |i\rangle \frac{1}{\sqrt{M}} \left(\|a_i\| \, |0^{\lceil \log(n+1) \rceil}\rangle \, |0\rangle + (M - \|a_i\|^2)^{1/2} \, |n+1\rangle \, |1\rangle \right) \tag{6}$$

We then proceed similarly to the construction in [18]. Let $B_{i,k}$ be the value of an internal node k of tree B_i at depth t. We apply a series of conditional rotations to the second register, conditioned on the first register being $|i\rangle$ and the first t-qubits of the second register being $|k\rangle$ and the tag qubit being $|0\rangle$, the rotation applied is:

$$|i\rangle \, |k\rangle \, |0\rangle \rightarrow |i\rangle \, |k\rangle \frac{1}{\sqrt{B_{i,k}}} \left(\sqrt{B_{i,2k}} \, |0\rangle + \sqrt{B_{i,2k+1}} \, |1\rangle \right)$$

We take positive square roots except for the leaf nodes where the sign of the square root is the same as $sign(a_{ij})$ of the entry stored at the leaf node. The tag qubit is uncomputed after all the conditional rotations have been performed by mapping $|n+1\rangle \, |1\rangle$ to $|n+1\rangle \, |0\rangle$.

Correctness follows since conditioned on the tag qubit being $|0\rangle$ the conditional rotations produce the state $\frac{1}{\sqrt{M}\|a_i\|} \sum_j a_{ij} \, |j\rangle$ and the amplitude for the tagged part is $\sqrt{(M - \|a_i\|^2)/M}$ matching the amplitudes in equation (5). The time for implementing U is $\widetilde{O}(\log(mn))$ as the number of quantum queries to the data structure is $\log^2(mn)$ and each query takes poly-logarithmic time. $\square$

4.2 Improved Singular Value Estimation

We first recall the definition of singular value estimation from the introduction,

Definition 5. *Let $A = \sum_{i \in [k]} \sigma_i u_i v_i^T$ be the singular value decomposition for $A \in \mathbb{R}^{m \times n}$ for $k = \min(m, n)$ and let $\delta > 0$. The singular value estimation (SVE) problem with error δ is given $|b\rangle = \sum_{i \in [k]} \beta_i \, |v_i\rangle$, to perform the mapping*

$$\sum_{i \in [k]} \beta_i \, |v_i\rangle \, |0\rangle \rightarrow \sum_{i \in [k]} \beta_i \, |v_i\rangle \, |\overline{\sigma}_i\rangle,$$

such that $|\overline{\sigma}_i - \sigma_i| \leq \delta$ for all $i \in [k]$ with probability $1 - 1/poly(n)$.

The theorem below provides a generalized singular value estimation algorithm that extends the algorithm from [18] and the quantum walk algorithms used for linear systems [10].

Theorem 4.3. *Let $A \in \mathbb{R}^{m \times n}$ be a matrix and suppose there exist $P, Q \in \mathbb{R}^{m \times n}$ and $\mu > 0$ such that $\|p_i\|_2 \leq 1 \; \forall i \in [m]$, $\|q^j\|_2 \leq 1 \; \forall j \in [n]$ and*

$$A/\mu = P \circ Q. \tag{7}$$

If unitaries $U : |i\rangle \, |0^{\lceil \log(n+1) \rceil}\rangle \to |i\rangle \, |\overline{p}_i\rangle$ and $V : |0^{\lceil \log(m+1) \rceil}\rangle \, |j\rangle \to |\overline{q}^j\rangle \, |j\rangle$ can be implemented in time $O(\log^2(mn))$ then there is a quantum algorithm for SVE with error δ in time $\widetilde{O}(\mu/\delta)$.

Proof. Let $\overline{P}, \overline{Q} \in \mathbb{R}^{(m+1) \times (n+1)}$ be matrices with rows and columns respectively equal to the normalized states $\overline{p}_i, \overline{q}^j$ for $i \in [m], j \in [n]$ and an additional row or column $\overline{p}_{m+1} = e_{m+1}, \overline{q}^{n+1} = e_{n+1}$. Let $\overline{A} = \begin{pmatrix} A & 0 \\ 0 & \mu \end{pmatrix}$ be an extension of A of size $(m+1) \times (n+1)$ so that the factorization $\overline{A}/\mu = \overline{P} \circ \overline{Q}$ holds.

As $\overline{A}$ is a block diagonal matrix, its singular value decomposition is given by $\sum_i \sigma_i \overline{u}_i \overline{v}_i^T + \mu e_{m+1}^T e_{n+1}$ where σ_i are singular values for A and $\overline{u}_i, \overline{v}_i$ are obtained by appending an additional 0 coordinate to the singular vectors u_i, v_i of A. Define the operators $\widetilde{P} \in \mathbb{R}^{(m+1)(n+1) \times (m+1)}, \widetilde{Q} \in \mathbb{R}^{(m+1)(n+1) \times (n+1)}$ as,

$$\widetilde{P} |i\rangle = |i\rangle \, |\overline{p}_i\rangle , \quad \widetilde{Q} |j\rangle = |\overline{q}^j\rangle \, |j\rangle . \tag{8}$$

The columns of $\widetilde{P}, \widetilde{Q}$ are orthogonal unit vectors so we have $\widetilde{P}^T \widetilde{P} = I_{m+1}$ and $\widetilde{Q}^T \widetilde{Q} = I_{n+1}$. Multiplication by $\widetilde{P}, \widetilde{Q}$ can be implemented efficiently using the unitaries U, V in the theorem statement. The unitary $\widetilde{U}$ on input $|i\rangle \, |0^{\lceil \log(n+1) \rceil}\rangle$ acts as U for $i \in [m]$ and maps $|m+1\rangle \, |0^{\lceil \log(n+1) \rceil}\rangle \to |e_{n+1}\rangle$, it can easily be implemented using U. We illustrate below for multiplication by $\widetilde{P}$,

$$|z\rangle \to |z, 0^{\lceil \log(n+1) \rceil}\rangle \xrightarrow{\widetilde{U}} \sum_{i \in [m+1]} z_i |i, \overline{p}_i\rangle = |\widetilde{P}z\rangle \tag{9}$$

Multiplication by $\widetilde{Q}$ can be implemented similarly using $\widetilde{V}$, thus the reflections $2\widetilde{P}\widetilde{P}^T - I$ and $2\widetilde{Q}\widetilde{Q}^T - I$ can be performed in time $\widetilde{O}(\log(mn))$.

Finally, the factorization $\widetilde{P}^T \widetilde{Q} = \overline{A}/\mu$ implies that the unitary $W = (2\widetilde{P}\widetilde{P}^T - I).(2\widetilde{Q}\widetilde{Q}^T - I)$ has eigenspaces $Span(\widetilde{P}\overline{u}_i, \widetilde{Q}\overline{v}_i)$ with eigenvalues $e^{\iota \theta_i}$ such that $\cos(\theta_i/2) = \sigma_i/\mu$. This relation between the eigenvalues of W and the singular values of A is known (for example, see [18], Lemma 5.3). Phase estimation for W on $|\widetilde{Q}\overline{v}_i\rangle$ recovers an estimate $\overline{\theta}_i$ for θ_i up to additive error δ/μ in time $\widetilde{O}(\mu/\delta)$, which provides an estimate $\cos(\overline{\theta}_i/2)$ for σ_i within additive error $O(\delta)$. The generalized singular value estimation is stated as Algorithm 4.1.

Algorithm 4.1 Generalized quantum singular value estimation.

Require: $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, efficient implementation of unitaries U, V. Input state $|x\rangle$ and precision parameter $\epsilon > 0$.

1. Let $|\overline{x}\rangle = \sum_{i \in [n]} \alpha_i |\overline{v}_i\rangle$ be the decomposition of the input in the basis of singular vectors for $\overline{A}$. Create state $|0^{\lceil \log(m+1) \rceil}, \overline{x}\rangle$ and apply unitary $\widetilde{V}$ to obtain $|\widetilde{Q}\overline{x}\rangle = \sum_i \alpha_i |\widetilde{Q}\overline{v}_i\rangle$.
2. Perform phase estimation with precision $2\epsilon > 0$ on input $|\widetilde{Q}\overline{x}\rangle$ for the unitary W in Theorem 4.3.
3. Compute $\overline{\sigma}_i = \cos(\overline{\theta}_i/2)\mu(A)$ where $\overline{\theta}_i$ is the estimate from phase estimation, and uncompute the output of the phase estimation to obtain $\sum_i \alpha_i |\widetilde{Q}\overline{v}_i, \overline{\sigma}_i\rangle$.
4. Apply the inverse of $\widetilde{V}$ to multiply the first register with the inverse of $\widetilde{Q}$ and obtain $\sum_i \alpha_i |\overline{v}_i\rangle \, |\overline{\sigma}_i\rangle$.

Note that in step 1 we work with $\overline{x} = (x, 0) \in \mathbb{R}^{n+1}$, this is the same as working with $|x\rangle$ as adding an additional 0 coordinate does not change the corresponding vector state.

$\square$

The main theorem for this section is obtained by using the data structure from Theorem 4.2 to efficiently implement the unitaries U, V required for the generalized singular value estimation procedure.

Theorem 4.4. *For (i) $\mu(A) = \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}$ for all $p \in [0,1]$ and (ii) $\mu(A) = \|A\|_F$, there are efficient QRAM data structures for storing $A \in \mathbb{R}^{m \times n}$ such that a quantum algorithm with access to such data structures can perform SVE for A with error δ in time $\widetilde{O}(\mu(A)/\delta)$.*

Proof. We first prove part (i). Theorem 4.3 holds for any choice of P, Q such that $A/\mu = P \circ Q$ provided the unitaries U and V can be implemented efficiently in time $\widetilde{O}(\log(mn))$, that is if the normalized states corresponding to the rows of P and the columns of Q can be prepared efficiently. For a given $p \in [0,1]$, the data structure in Theorem 4.2 allows us to prepare the normalized states for the following choice of P and Q,

$$p_{ij} = \frac{sgn(a_{ij})a_{ij}^p}{\max\|a_i\|_{2p}^{2p}}, \qquad q_{ij} = \frac{a_{ij}^{1-p}}{\max\|a^j\|_{2(1-p)}^{2(1-p)}} \tag{10}$$

Indeed, in order to implement the unitaries U and V corresponding to this choice of P, Q, we create two copies of the data structure in Theorem 4.2 that respectively store the rows and the columns of A. Given entry (i, j, a_{ij}) instead of a_{ij}, we store $sgn(a_{ij})a_{ij}^p$ and a_{ij}^{1-p}. The normalization factor μ for this factorization is $\mu_p(A) = \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}$, where $s_p(A) = \max_{i \in [m]}\|a_i\|_p^p$ and $s_p(A^T) = \max_{j \in [n]}\|a^j\|_p^p$.

Part (ii) follows from the data structure used for SVE algorithm given in [18], it corresponds to the factorization $A/\|A\|_F = P \circ Q$ with $p_{ij} = \frac{a_{ij}}{\|a_i\|}$ and $q_{ij} = \frac{\|a_i\|}{\|A\|_F}$. $\square$

Theorem 4.4 gives different efficient data structures that provide an SVE algorithm with running time $\widetilde{O}(\mu(A)/\delta)$ for different values of μ. We show next that in the QRAM data structure model, one can achieve the minimum $\mu(A)$ over a subset of these data structures.

Theorem 4.5. *Let $\mathcal{P}$ be a set of values in $[0,1]$ such that $|\mathcal{P}| = O(1)$. There is algorithm in the QRAM data structure model (Definition 3), that performs SVE for $A \in \mathbb{R}^{m \times n}$ in time $\widetilde{O}(\mu(A)/\delta)$ with $\mu(A) = \min_{p \in \mathcal{P}} \left(\|A\|_F, \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}\right)$.*

Proof. Let m be the number of non zero entries in A, we make two passes over the entries (i, j, a_{ij}). In the first pass we find the optimal value $\mu(A) = \min_{p \in [0,1]} \left(\|A\|_F, \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}\right)$. The cost for this is $O(|\mathcal{P}|m)$ as we need to updating the $2|\mathcal{P}|$ values $s_{2p}(A), s_{2(1-p)}(A^T)$ each time we process a matrix entry (i, j, a_{ij}). At the end of the first pass we know the data structure that corresponds to the optimal value of $\mu(A)$. In the second pass we stream over the matrix entries and contrucct the data structure corresponding to $\mu(A)$ as in Theorem 4.4.

The overall complexity is linear in m as $|\mathcal{P}|$ is a constant, the pre-processing step therefore satisfies the requirements for the QRAM data structure model in Definition 3. Given access to the optimal data structure, Theorem 4.4 implies that there is a quantum algorithm for SVE with error δ in time $\widetilde{O}(\mu(A)/\delta)$ for $\mu(A) = \min_{p \in \mathcal{P}} \left(\|A\|_F, \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}\right)$. $\square$

If we are restricted to a single pass over the matrix entries, we can construct two data structures for A for $\mu(A) = \|A\|_F$ and $\mu(A) = s_1(A)$ (corresponding to $p = 1/2$) with a constant overhead and then use the data structure that achieves the smaller value for $\mu(A)$. This construction covers the case of sparse and low rank matrices that arise often in practice and will be the one that we shall use for our iterative method.

Theorem 4.6. *There is algorithm in the QRAM data structure model (Definition 3), that makes a single pass over the entries of $A \in \mathbb{R}^{m \times n}$ in the pre-processing step and performs SVE for A in time $\widetilde{O}(\mu(A)/\delta)$ for $\mu(A) = \min\left(\|A\|_F, s_1(A)\right)$.*

For our applications, one could also use the more general Theorem 4.5 but we use Theorem 4.6 instead as it simplifies the analysis, involves only one pass over the matrix entries in the pre-processing step and covers the useful cases of bounded ℓ_1 norm and low-rank A for which there is a significant quantum speed-up.

4.3 Quantum matrix multiplication and linear systems

We provide algorithms for quantum linear systems and quantum matrix multiplication using the improved singular value estimation algorithm. We will see that once we perform singular value estimation for a matrix A, then multiplication with the matrix consists of a conditional rotation by an angle proportional to each singular value. Similarly, solving the linear system corresponding to the matrix A is multiplication with the inverse of A, in other words, a conditional rotation by an angle proportional to the inverse of each singular value of A.

The two algorithms are therefore very similar. We will also extend our matrix multiplication algorithm, i.e., the application of a linear operator, to the case of an affine operator, namely, given matrix A and vector b in memory, the algorithm maps any state $|x\rangle$ to a state close to $|Ax + b\rangle$. Last, we discuss briefly the cases for which our algorithm improves upon the running time of existing quantum linear system solvers.

If $A \in \mathbb{R}^{m \times n}$ is a rectangular matrix, then multiplication by A reduces to multiplication by the square symmetric matrix $A' = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$ as $A'(0^m, x) = (Ax, 0)$. Therefore, without loss of generality we restrict our attention to symmetric matrices for the quantum matrix multiplication problem. We state the quantum matrix multiplication Algorithm 4.2 for a positive semidefinite matrix as in applications of the iterative method. We note that linear systems for general symmetric matrices are not much harder than the case described in Algorithm 4.2.

More precisely, the SVE procedure estimates the absolute value of the eigenvalues $|\lambda_i|$ for a symmetric A, and hence we also need to recover the sign of the λ_i to perform matrix multiplication or to solve linear systems. The sign can be recovered by performing singular value estimation for the matrices $A, A + \mu I$ and comparing the estimates $|\overline{\lambda_i}|, |\overline{\lambda_i + \mu}|$. The second estimate is larger if $\lambda_i > 0$, otherwise the first estimate is larger. We refer to [33] for details. We do not use this more general linear system solver in this paper, linear system solvers for positive semidefinite matrices described next suffice for our applications.

Algorithm 4.2 Quantum matrix multiplication/linear systems.

Require: Matrix $A \in \mathbb{R}^{n \times n}$ stored in the data structure in Theorem 4.5, such that eigenvalues of
    A lie in $[1/\kappa, 1]$. Input state $|x\rangle = \sum_i \beta_i |v_i\rangle \in \mathbb{R}^n$, where v_i are right singular vectors for A.
1: Perform singular value estimation with precision ϵ_1 for A on $|x\rangle$ to obtain $\sum_i \beta_i |v_i\rangle |\overline{\lambda}_i\rangle$.
2: Perform a conditional rotation and uncompute the SVE register to obtain the state:
    (i) $\sum_i \beta_i |v_i\rangle (\overline{\lambda}_i |0\rangle + \gamma |1\rangle)$ for matrix multiplication.
    (ii) $\sum_i \beta_i |v_i\rangle (\frac{1}{\kappa \overline{\lambda}_i} |0\rangle + \gamma |1\rangle)$ for linear systems.
3: Perform Amplitude Amplification with the unitary V implementing steps 1 and 2, to obtain (i)
    $|z\rangle = \sum_i \beta_i \overline{\lambda}_i |v_i\rangle$ or (ii) $|z\rangle = \sum_i \beta_i \frac{1}{\overline{\lambda}_i} |v_i\rangle$.

We next analyze the correctness and provide the running time for the above algorithm.

Theorem 4.7. *Parts (i) and (ii) of Algorithm* 4.2 *produce as output a state* $|z\rangle$ *such that* $\||Ax\rangle - |z\rangle\| \leq \delta$ *in expected time* $\widetilde{O}(\frac{\kappa^2(A)\mu(A)}{\delta})$ *for* $\mathcal{A} = A$ *and* $\mathcal{A} = A^{-1}$ *respectively.*

Proof. We first analyze matrix multiplication. The unnormalized solution state is $Ax = \sum_i \beta_i \lambda_i v_i$, while the unnormalized output of step 1 of the algorithm which performs SVE to precision ϵ_1 is $z = \sum_i (\lambda_i \pm \widetilde{\epsilon}_i) \beta_i v_i$ such that $|\widetilde{\epsilon}_i| \leq \epsilon_1$ for all i. As the v_i are orthonormal, we have $\|Ax - z\| \leq \epsilon_1 \|x\|$ and by Claim 2.3, we have $\||Ax\rangle - |z\rangle\| \leq \frac{\sqrt{2}\epsilon_1 \|x\|}{\|Ax\|} \leq \sqrt{2}\epsilon_1 \kappa(A)$.

We next analyze linear systems. The unnormalized solution state is $A^{-1}x = \sum_i \frac{\beta_i}{\lambda_i} v_i$. The unnormalized output is $z = \sum_i \frac{\beta_i}{\lambda_i \pm \widetilde{\epsilon}_i} v_i$ for $|\widetilde{\epsilon}_i| \leq \epsilon_1$. We have the bound

$$
\begin{aligned}
\|A^{-1}x - z\|^2 &\leq \sum_i \beta_i^2 \left(\frac{1}{\lambda_i} - \frac{1}{\lambda_i \pm \widetilde{\epsilon}_i} \right)^2 \leq \epsilon_1^2 \sum_i \frac{\beta_i^2}{\lambda_i^2 (\lambda_i - \epsilon_1)^2} \leq \frac{\epsilon_1^2 \kappa^2(A) \|A^{-1}x\|^2}{(1 - \kappa(A)\epsilon_1)^2} \\
&\leq 4\epsilon_1^2 \kappa^2(A) \|A^{-1}x\|^2
\end{aligned}
$$

assuming that $\kappa(A)\epsilon_1 \leq 1/2$. Applying Claim 2.3 we obtain $\||A^{-1}x\rangle - |z\rangle\| \leq 2\sqrt{2}\kappa(A)\epsilon_1$ for $\kappa(A)\epsilon_1 \leq 1/2$.

We can therefore use the SVE algorithm with precision $\epsilon_1 = \frac{\delta}{\kappa(A)}$ for both cases to obtain a solution state $\||\mathcal{A}x\rangle - |z\rangle\| \leq \delta$. The success probability for step 2 of the algorithm is $\frac{1}{\kappa^2(A)}$ for both matrix multiplication and matrix inversion. We apply Amplitude Amplification as in Theorem 2.2 with the unitary V that represents the first two steps of the algorithm to obtain an expected running time $\widetilde{O}(\frac{\kappa^2(A)\mu(A)}{\delta})$. $\qquad \square$

Let us now see how the linear system solver in the QRAM data structure model compares to the HHL algorithm and its improvements. The HHL algorithm requires an input model where the transformation $O_A : |i, j, 0\rangle \to |i, j, a_{ij}\rangle$ can be carried out efficiently. The oracle O_A can be implemented efficiently for matrix A stored in the QRAM but also without the QRAM if the matrix A is well structured. The QRAM data structure model thus makes a stronger assumption than the HHL input model. It is however illuminating to see the improvements one can obtain over HHL in the stronger QRAM data structure model.

A natural normalization for the quantum linear system problem is to assume that $\|A\| = 1$, so that the eigenvalues being estimated have been scaled down to quantities in $[1/\kappa, 1]$. We provide an algorithm to normalize the matrix A such that $\|A\| \leq 1$ in Section 4.3.1.

The HHL algorithm with this scaling produces an δ approximation to the state $|A^{-1}b\rangle$ and an estimate for $\|A^{-1}b\|$ in time $\widetilde{O}(s^2(A)\kappa^2(A)/\delta)$ where $s(A)$ is an upper bound on the number of non-zero entries per row. Subsequent work has improved the running time to $\widetilde{O}(s(A)\kappa(A)\log(1/\delta))$ [10, 2], the recent work [8] implies similar improvements for linear system solvers in the QRAM data structure model.

Quantum linear system solvers in the QRAM data structure model have sub-linear running time even for dense matrices. Instead of sparsity, their running time depends on the parameter $\mu(A)$. On one hand, this factor is smaller than the Frobenius norm, for which we have $\|A\|_F = (\sum_i \sigma_i^2)^{1/2} \leq \sqrt{rk(A)}$. Hence, Algorithm 4.2 achieves a considerable speedup for matrices whose rank is poly-logarithmic in the matrix dimensions. Moreover, while for general dense matrices the sparsity is $\Omega(n)$, we have $\mu(A) \leq \|A\|_F \leq \sqrt{n}$. The QRAM data structure based linear system solvers therefore achieve a worst case quadratic speedup over linear system solvers in the HHL input model [33].

In addition, the factor $\mu(A)$ is smaller than the maximum ℓ_1-norm $s_1(A)$ which is smaller than the maximum sparsity $s(A)$ for the normalization $\|A\|_{max} = 1$, that is the entries of A have absolute value at most 1. This normalization is used for the linear system solver [10] in the HHL input model, the QRAM data structure based linear system solver presented here is an improvement as $s_1(A) \leq s(A)$. An example where the linear system solver of Theorem 4.7 is an exponential improvement over previous approaches is for real-valued matrices with most entries close to zero and a few entries close to 1, for example $A = I + J/n$ or a small perturbation of a permutation matrix. We have $s(A) = \Omega(n)$, $\|A\|_F = \omega(\sqrt{n})$ and $s_1(A) = O(1)$ for small enough perturbations.

Algorithm 4.2 achieves a worst case quadratic speedup over linear system solvers in the HHL input model as there are matrices with $\|A\| = 1$ for which $\mu(A) = \Omega(\sqrt{n})$. An example is a random sign matrix A with $\|A\| = 1$. In this case, one can easily show that with high probability $\mu(A) = \Omega(\sqrt{n})$.

The optimal value $\mu(A) = \min_\mu\{P \circ Q = A/\mu \mid \|p_i\|_2 \leq 1, \|q^j\|_2 \leq 1\}$ in Theorem 4.3 is the spectral norm of $|A|$ where $|A|$ is the matrix obtained by replacing entries of A by their absolute values [25]. We recall that the matrix $A \in \mathbb{R}_+^{m \times n}$ has unique positive left and right eigenvectors, these eigenvectors are called the Perron-Frobenius eigenvectors and can be computed for example by iterating $x_{t+1} = Ax_t/\|Ax_t\|$. The optimal walk can be implemented efficiently if the coordinates of the Perron-Frobenius eigenvectors for $|A|$ are stored in memory prior to constructing the data structure.

However, there are no known algorithms for computing the entries of the Perron-Frobenius eigenvector in linear time in the number of matrix entries with a poly-logarithmic update time. Hence the optimal walk cannot be implemented in the quantum data structure model. We also note that the spectral norm of $|A|$ can be much larger than the spectral norm of A, for example for a random sign matrix with ± 1 entries the spectral norm of A is $\|A\| = O(\sqrt{n})$ but the spectral norm of $|A|$ is $\||A|\| = n$. Thus there are two interesting questions about quantum linear system solvers. The first is to find the optimal quantum linear system solver in the QRAM data structure model. The second is to find if there can exist more general quantum walk algorithms with $\mu(A) = \|A\|$. The latter has also been stated as an open problem [9].

We also note that as in the analysis of the HHL algorithm [14], the parameter κ does not have to be as big as the actual condition number of A. If κ is smaller, then it means that we invert only the well-conditioned part of the matrix and this may be useful for some settings.

4.3.1 Spectral Norm Estimation

We assumed above that the matrices A are normalized such that the absolute value of the eigenvalues lie in the interval $[1/\kappa, 1]$. We provide here a quantum algorithm using the QRAM data structure given in [18] for estimating the spectral norm, which can be used to rescale matrices so that the assumption $\|A\| \leq 1$ is indeed valid. Note that $0 \leq \frac{\|A\|}{\|A\|_F} \leq 1$ and that $\|A\| = \sigma_{\max}(A)$, where $\sigma_{\max}$ is the largest singular value.

Algorithm 4.3 Spectral norm estimation.

Require: $A \in \mathbb{R}^{m \times n}$ stored in data structure given in [18]. Returns an estimate for $\eta := \|A\|/\|A\|_F$ with additive error ϵ.

1. Let $l = 0$ and $u = 1$ be upper and lower bounds for η, the estimate $\tau = (l + u)/2$ is refined using binary search in steps 2-5 over $O(\log 1/\epsilon)$ iterations.
2. Prepare $|\phi\rangle = \frac{1}{\|A\|_F} \sum_{i,j} a_{ij} |i, j\rangle = \frac{1}{\|A\|_F} \sum_{i,j} \sigma_i |u_i, v_i\rangle$ and perform SVE [18] with precision ϵ to obtain $\frac{1}{\|A\|_F} \sum_{i,j} \sigma_i |u_i, v_i, \overline{\sigma_i}\rangle$. where $|\overline{\sigma}_i - \frac{\sigma_i}{\|A\|_F}| \leq \epsilon$.
3. Append single qubit register $|R\rangle$ and set it to $|1\rangle$ if $\overline{\sigma}_i \geq \tau$ and $|0\rangle$ otherwise. Uncompute the SVE output from step 2.
4. Perform amplitude estimation on $\frac{1}{\|A\|_F} \sum_{i,j} \sigma_i |u_i, v_i, R\rangle$ conditioned on $R = 1$ to estimate $\sum_{i:\overline{\sigma}_i \geq \tau} \sigma_i^2/\|A\|_F^2$ to relative error $(1 \pm \delta)$.
5. If estimate in step 4 is 0 then $u \to \tau$ else $l \to \tau$. Set $\tau = (u + l)/2$.

The following proposition proves correctness for Algorithm 4.3.1 and bounds its running time. Algorithm 4.3.1 provides an estimate for $\|A\|/\|A\|_F$, this yields an estimate for $\|A\|$ as one can easily compute $\|A\|_F$ when A is being efficiently loaded into the QRAM.

Proposition 4.8. *Algorithm 4.3.1 estimates $\|A\|$ to error $\epsilon\|A\|_F$ in time $\widetilde{O}(\log(1/\epsilon)/\epsilon\eta)$.*

Proof. We show that the algorithm estimates η within additive error ϵ. In order to prove this, it suffices to show that if $|\tau - \eta| \geq \epsilon$ then the algorithm determines $sign(\tau - \eta)$ correctly. If $|\tau - \eta| \geq \epsilon$ then the amplitude $\sum_{i:\overline{\sigma}_i \geq \tau} \sigma_i^2/\|A\|_F^2$ being estimated in step 4 is either 0 or at least η^2. Amplitude estimation as in Theorem 2.2 yields a non-zero estimate in the interval $(1 \pm \delta)\eta$ for the latter case and thus the sign is determined correctly for constant δ.

The running time for step 2 is $\widetilde{O}(1/\epsilon)$. The amplitude estimation in step 4 requires time $\widetilde{O}(1/\epsilon\delta\eta)$ as $T(U) = \widetilde{O}(1/\epsilon)$ and the amplitude being estimated is either 0 or at least η^2. Step 4 is repeated $\log(1/\epsilon)$ times and δ is a fixed constant, so the running time is $\widetilde{O}(\log(1/\epsilon)/\epsilon\eta)$. $\qquad\square$

The above algorithm can also be used to find an estimate for the condition number κ, that can be used in the linear systems solver. Let κ' be a threshold such that $\sum_{\sigma_i \leq \kappa'} \sigma_i^2/\|A\|_F^2 \geq \eta^2$, then the algorithm (with $|R\rangle$ set to 1 if $\overline{\sigma}_i \leq \tau$ in step 3) provides an additive error ϵ estimate for κ' in time $\widetilde{O}(\log(1/\epsilon)/\epsilon\eta)$. The spectral norm estimation procedure can be viewed as an auxiliary subroutine that is used once and thus contributes an additive term to the running time for the quantum linear system solver. We do not include this additive overhead in our running time estimates.

4.4 The iterative step

We now show that the (ϵ, δ)-approximate quantum step for the iterative method in Definition 3.1 can be implemented using the quantum matrix multiplication algorithm presented above. The

matrix S for iterative methods corresponding to linear systems and least squares is of the form $S = I - \alpha A$ for a positive semidefinite matrix A. Further, we can assume that the matrix A is stored in the data structure of Theorem 4.6 and that S is positive semi-definite and contractive.

Proposition 4.9. *An $(\epsilon, 1/poly(n))$-approximate quantum step (as in Definition 3.1) for the iterative method with $S = I - \alpha A, L(x) = b - Ax, \alpha \leq 1, \|b\| = 1$ and for $A|b$ stored in the data structure of Theorem 4.6, can be implemented in time $\widetilde{O}(\mu(A|b)/\epsilon)$, where $A|b$ is the matrix A with an extra row equal to b.*

Proof. We show how to implement the unitary V that implements the $(\epsilon, 1/poly(n))$-approximate quantum step (as in Definition 3.1) for the iterative method, that is

$$V : |0\rangle |r_0\rangle |0\rangle \rightarrow |1\rangle \left(\alpha \|\tilde{L}(r_0)\| \, |\tilde{L}(r_0)\rangle |0\rangle + |G_1\rangle |1\rangle \right)$$

$$: |t\rangle \|r_t\| |r_t\rangle |0\rangle \rightarrow |t+1\rangle \left(\|\tilde{S}(r_t)\| \, |\tilde{S}(r_t)\rangle |0\rangle + |G_{t+1}\rangle |1\rangle \right),$$

where $|G_1\rangle, |G_{t+1}\rangle$ are unnormalised garbage states, such that with probability $\geq 1 - \delta$, it holds that $\|L(r_0) - \tilde{L}(r_0)\| \leq \epsilon$ and we also have that $\|S(r_t) - \tilde{S}(r_t)\| \leq \epsilon$ for all $t \in [\tau - 1]$ with probability $\geq 1 - \delta$. We first implement the linear part of V that corresponds to $1 \leq t \leq \tau - 1$ and then the affine part corresponding to $t = 0$. We denote $r_t = \sum_i \beta_i |v_i\rangle$ in the basis of right singular vectors for S.

The linear part of V is implemented by performing the singular value estimation for A and then using $\overline{\lambda_i} = (1 - \alpha \overline{\lambda_i}(A))$ as estimates for singular values of S with additive error ϵ,

$$|t\rangle \|r_t\| |r_t\rangle |0\rangle \equiv |t\rangle \sum_i \beta_i |v_i\rangle |0\rangle \rightarrow |t+1\rangle \sum_i \beta_i |v_i\rangle \left(\overline{\lambda_i} |0\rangle + \sqrt{1 - \overline{\lambda_i}^2} |1\rangle \right) \tag{11}$$

As the precision for singular value estimation is ϵ, the algorithm runs in time $\widetilde{O}(\mu(A)/\epsilon)$. For bounding the difference of the norms we observe that $\|S(r_t) - \tilde{S}(r_t)\| \leq \|\sum_i \beta_i \tilde{\epsilon}_i v_i\| \leq \alpha \epsilon \leq \epsilon$ as all the errors $\tilde{\epsilon}_i \leq \epsilon$ if the singular value estimation succeeds and $\|\beta\| \leq 1$. The procedure succeeds if the singular value estimation algorithm produces the correct estimates. The success probability for the singular value estimation is $1 - 1/poly(n)$, thus δ can be taken to be $1/poly(n)$.

The affine part of V is implemented as follows. Let $A_1 = \begin{pmatrix} -A & b \\ 0 & 0 \end{pmatrix}$ and $x_1 = (x, 1)$ so that $A_1 x_1 = (b - Ax, 0)$. Then we symmetrize A_1 by defining $A' = \begin{pmatrix} 0 & A_1 \\ A_1^T & 0 \end{pmatrix}$ and $x' = (0^{n+1}, x_1)$, and have $A'x' = (A_1 x_1, 0) = (b - Ax, 0)$. The columns of A' are stored in the memory data structure so we can perform SVE for A'. We take $x = r_0$ and use the SVE algorithm for symmetric matrices on r_0' (where we add an extra factor α in the conditional rotation) to map it to $A'r_0' = (b - Ar_0, 0)$ as the last coordinates become 0. Denote $r_0' = \sum_i \beta_i |v_i'\rangle$, where v_i' are the eigenvectors of A'.

$$|0\rangle \|r_0'\| |r_0'\rangle |0\rangle \equiv |0\rangle \sum_i \beta_i |v_i'\rangle |0\rangle \rightarrow |1\rangle \sum_i \beta_i |v_i\rangle \left(\alpha \overline{\lambda_i}(A') |0\rangle + \sqrt{1 - \alpha^2 \overline{\lambda_i}(A')^2} |1\rangle \right)$$

If the precision for singular value estimation is ϵ then the algorithm runs in time $\widetilde{O}(\mu(A')/\epsilon)$ and the correctness analysis is the same as above. We next provide an upper bound for $\mu(A')$. We have $\|A'\|_F \leq 2\|A\|_F + 2\|b\|$, while $s_1(A') \leq \max(s_1(A) + \|b\|_\infty, s_1(b))$. Let's assume for simplicity that

25

$\|b\| = 1$, which is the case in our applications, then the upper bound is $O(\mu(A|b))$ where $A|b$ is a matrix obtained by adding an extra row b to A. $\qquad\square$

Finally, let us see how to implement the (ϵ, δ)-approximate quantum multistep unitary U in Claim 3.2 with cost $O(C_V + \log \tau)$ which is asymptotically the same as the cost of V.

Proposition 4.10. *Given an implementation of the unitary C_V in Proposition 4.9, an $(\epsilon, 1/poly(n))$-approximate quantum multistep step (as in Claim 3.2) for the iterative method can be implemented with cost $O(C_V + \log \tau)$.*

Proof. It is easy to see that the complexity of U is asymptotically upper bounded by the complexity of applying the unitary V^τ on $|0\rangle \, |r_0\rangle \, |0\rangle$. For this we first apply V once to get the affine transformation (with running time $\widetilde{O}(\mu(A|b)/\epsilon)$), then we apply the SVE procedure on A to obtain estimates $\overline{\lambda_i}$ of the eigenvalues of $S = (I - \alpha A)$ and then compute $\overline{\lambda_i}^{\tau-1}$ (in time $O(\log \tau)$) as the estimates for the singular values of $S^{\tau-1}$ for the conditional rotation. This gives us the unitary U in Claim 3.2 with error $t\epsilon$ for the t-th step. The running time of the second part is $\widetilde{O}(\mu(A)/\epsilon + \log \tau)$. As μ is monotone over the addition of columns, the overall running time is $O(C_V + \log \tau)$. $\qquad\square$

5 Quantum iterative algorithms

5.1 Linear systems

Let $A \succeq 0$ be a $n \times n$ positive semidefinite matrix with eigenvalues in the interval $[1/\kappa, 1]$ and let $b \in \mathbb{R}^n$ with $\|b\| = 1$. We assume that b is stored in a QRAM data structure. The goal is to solve the linear system $A\theta = b$.

The classical iterative method starts with the observation that the quadratic form $F(\theta) = \theta^T A \theta - b\theta$ is minimized at the solution to $A\theta = b$. The algorithm starts with an arbitrary θ_0 and applies the following updates,

$$\theta_{t+1} = \theta_t + \alpha(b - A\theta_t) = \theta_t + \alpha r_t \tag{12}$$

where the step size α will be a small constant and the residuals $r_t := b - A\theta_t$ for $t \geq 0$. The residuals satisfy the recurrence $r_{t+1} = b - A(\theta_t + \alpha r_t) = (I - \alpha A)r_t$ and the initial condition $r_0 = (b - A\theta_0)$.

The convergence analysis and the choice of the step size α follow from the following classical argument. Let $\theta^* = A^{-1}b$ be the optimal solution, the error $e_t := \theta_t - \theta^*$ satisfies the recurrence $e_{t+1} = (\theta_{t+1} - \theta^*) = \theta_t - \theta^* + \alpha(b - A\theta_t) = e_t + \alpha A(\theta^* - \theta_t) = (I - \alpha A)e_t$. After t steps of the iterative method,
$$\|e_t\| = \|(I - \alpha A)^t e_0\| \leq (1 - \alpha/\kappa)^t \|e_0\|.$$

The method therefore converges to the optimal solution θ^* within error ϵ in $\tau = O(\kappa \log(\|e_0\|/\epsilon)/\alpha)$ iterations. The step size α can be fixed to be a small constant say $\alpha = 0.01$ and the starting point θ_0 chosen to be a unit vector so that $\|e_0\| \leq \kappa$. With these choices the number of iterations required for convergence within error ϵ is $O(\kappa \log(\kappa/\epsilon))$.

The classical iterative method can be viewed as gradient descent with affine updates. We obtain a quantum iterative method for this problem using Proposition 4.9.

Theorem 5.1. *Given positive semidefinite $A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n$ stored in the data structure of Theorem 4.6, there is an iterative quantum algorithm that outputs a state $|z\rangle$ such that $\||z\rangle - |A^{-1}b\rangle\| \leq \delta$ with expected running time $O(\frac{\kappa(A)^3 \log^3 \frac{\kappa(A)}{\delta} \mu(A|b)}{\delta})$.*

Proof. An $(\epsilon, 1/poly(n))$ approximate iterative step for the classical iterative method described above can be implemented in time $C_V = \widetilde{O}(\mu(A|b)/\epsilon)$ using Proposition 4.9. The cost C_U of the unitary in Theorem 3.3 is the cost of implementing the powers V^t by Proposition 4.10, in this case this cost is the same as C_V. This is because we do not have to apply V sequentially t times, but once the SVE has estimated the eigenvalues, we can directly perform the conditional rotations by an angle proportional to the t-th power of each eigenvalue.

Theorem 3.3 shows that given an $(\epsilon, 1/poly(n))$ approximate step and for constant $\alpha, \|\tilde{\theta}_\tau\|$, the quantum iterative method has error $\||\theta_\tau\rangle - |\tilde{\theta}_\tau\rangle\| = O(\tau^2 \epsilon)$ and requires time $O(\tau C_U) = O(\tau C_V) = O(\frac{\tau \mu(A|b)}{\epsilon})$. We take $\epsilon = O(\frac{\delta}{\tau^2})$ in order to have $\||\theta_\tau\rangle - |\tilde{\theta}_\tau\rangle\| \leq \delta/2$ for some $\delta > 0$. The running time bound follows as we showed that $\||\theta_\tau\rangle - |A^{-1}b\rangle\| \leq \delta/2$ for $\tau = \kappa(A) \log \frac{\kappa(A)}{\delta}$.

In order to complete the proof, it remains to show that $\|\theta_\tau\| = \Omega(1)$ for $\tau = O(\kappa \log(\kappa/\epsilon))$. The solution $A^{-1}b$ to the linear system has norm at least 1 as b is a unit vector and the eigenvalues of A^{-1} are greater than 1. After τ steps we have $\|\theta_\tau - \theta^*\| \leq \epsilon \Rightarrow \|\theta_\tau\| \geq \|\theta^*\| - \epsilon \geq 1 - \epsilon$. $\qquad\blacksquare$

5.2 Weighted Least Squares

For the weighted least squares problem, we are given a matrix $X \in \mathbb{R}^{m \times n}$ and a vector $y \in \mathbb{R}^m$, as well as a vector $w \in \mathbb{R}^m$ of weights, and the goal is to find $\theta \in \mathbb{R}^n$ that minimizes the squared loss $\sum_{i \in [m]} w_i(y_i - x_i^T \theta)^2$. The closed form solution is given by,

$$\theta = (X^T W X)^{-1} X^T W y$$

and thus the problem can also be solved using a direct method. However, the direct method is prohibitive for large datasets and iterative methods are used instead as they do not involve solving large linear systems. The iterative method for weighted least squares is a gradient descent algorithm with the update rule $\theta_{t+1} = \theta_t + \rho \sum_{i \in [m]} w_i(y_i - \theta_t^T x_i) x_i$ which in matrix form can be written as,

$$\theta_{t+1} = (I - \rho X^T W X)\theta_t + \rho X^T W y \tag{13}$$

The quantum iterative method for the weighted least squares problem is given by the following theorem.

Theorem 5.2. *Let $X \in \mathbb{R}^{m \times n}, y \in \mathbb{R}^n, w \in \mathbb{R}^m$ be stored in the data structures in Theorem 4.6. Let $W = diag(w)$, $A = X^T W X$ and $b = X^T W y$, then there is an iterative quantum algorithm that outputs a state $|z\rangle$ such that $\||z\rangle - |A^{-1}b\rangle\| \leq \delta$ with expected running time $O(\frac{\kappa(A)^3 \log^3 \frac{\kappa(A)}{\delta} \mu(\sqrt{W}X|y)}{\delta})$.*

Proof. The iterative update rule for weighted least squares in equation (13) can be written as $\theta_{t+1} = \theta_t + \rho r_t$ where $r_t = X^T W y - X^T W X \theta_t$. These updates are analogous to the linear system updates in equation (12) as $r_t = b - A\theta_t$ for $b = X^T W y$ and $A = X^T W X$. The step size ρ is analogous to the step size α for the linear system. It follows from Theorem 5.1 that given an $(\epsilon, 1/poly(n))$ approximate step for the iterative method, the quantum iterative algorithm for weighted least squares would have the stated running time.

It remains to show how to implement the iterative step for the least squares problem, which is somewhat different from the case of linear systems as instead of the matrix $X^T W X$ and the vector $X^T W y$, we have the matrix X and vector y stored in memory and weights w arrive as a stream of entries (i, w_i) for $i \in [m]$. Nevertheless, the iterative step can be implemented in this setting as we show next.

Note that $A = B^T B$ where $B = \sqrt{W} X$, thus the eigenvalues of A are the squared singular values of B. It therefore suffices to perform the generalized SVE for B. We assume that the data structures for performing generalized SVE for X have been created.

In order to perform the generalized SVE for B, we append to the data structure in Theorem 4.2 a variable M_w that is initially a copy of M. Recall that the variable M stores the maximum row norm $M = \max_{i \in [m]} \|a_i\|^2$ for the matrix A stored in the data structure. The variable M_w is updated whenever w_i arrives or $\|x_i\|$ gets updated so as to store $\max_i w_i \|x_i\|^2$, that is if $M_w \leq w_i \|x_i\|^2$ then set $M_w = w_i \|x_i\|^2$. We replace $M \to M_w$ and $\|x_i\| \to \sqrt{w_i} \|x_i\|$ in equation (6) and follow exactly the same steps as in Theorem 4.2 to implement the unitary,

$$U' \ket{i, 0^{\lceil \log(n+1) \rceil}} = \ket{i} \frac{1}{\sqrt{M_w}} \left(\sum_{j \in [n]} \sqrt{w_i} x_{ij} \ket{j} + (M_w - w_i \|x_i\|^2)^{1/2} \ket{n+1} \right)$$

Using U', V instead of U, V in Theorem 4.4 we can perform generalized SVE for $B = \sqrt{W} X$ in time $\mu(B)$. In order to multiply by $A = B^T B$ for the iterative method, we perform generalized SVE for B in equation (11) and then conditional rotation with factor $\alpha \overline{\sigma}_i^2$.

Analogous to the above procedure one can also implement matrix multiplication for $B' = X^T W$. Note that the state $\ket{b}$ is not in the memory, so we can not do the first affine update used for linear systems (where we set $r_0 = b - A\theta_0$ for a random θ_0). Instead we have y and X in memory, so we first do the affine step to create $(y - X\theta_0)$ and then multiply with the matrix $X^T W$,

$$b - A\theta_0 = X^T W y - X^T W X \theta_0 = X^T W (y - X\theta_0).$$

$\square$

The procedure for performing generalized SVE for $B = \sqrt{W} X$ in the proof of Theorem 5.2 implies that given the data structure for storing $X \in \mathbb{R}^{m \times n}$ and a weight vector $w \in \mathbb{R}^m$, it is possible to prepare the data structure for performing SVE for $B = \sqrt{W} X$ in time $O(m)$ where $W = diag(w)$. We note this fact as it may be useful for other applications.

It is straightforward to add ℓ_2 regularization to the weighted least squares problem. The loss function becomes $\sum_i w_i (y_i - \theta^T x_i)^2 + \lambda \|\theta\|^2$ and the update rule changes to $r_t = b - A\theta_t$ for $b = X^T W y$ and $A = X^T W X + \lambda I$. The algorithm performs the generalized SVE for $X^T W X + \lambda I$ instead of $X^T W X$.

5.3 Stochastic gradient descent for Weighted Least Squares

In the classical setting, it is expensive to compute the gradient $\sum_{i \in [m]} w_i (y_i - \theta_t^T x_i) x_i$ by summing over the entire dataset when the dataset size is large. Moreover, due to redundancy in the dataset, the gradient can be estimated by summing over randomly sampled batches. Stochastic gradient descent utilizes this fact in the classical setting to lower the cost of the updates. Stochastic gradient descent algorithms do not compute the gradient exactly, but estimate it over batches $\sum_{i \in S_j} w_i (y_i - \theta_t^T x_i) x_i$ obtained by randomly partitioning the dataset.

In the quantum case, applying a linear system solver or the iterative method in Theorem 5.2 for a large dataset would require a large-sized QRAM and coherent operations over a large number of qubits. Stochastic gradient descent remains relevant for quantum iterative methods since it can considerably reduce the size of the QRAM as well as the number of qubits on which we need to perform coherent operations.

The stochastic gradient updates are defined for any choice of partition $\Pi = (S_1, S_2, \cdots, S_k)$ for the dataset where $S_i \subset [m]$ for $i \in [k]$ are subsets of $[m]$ of equal size. The random partitioning can be easily implemented in the quantum setting by permuting the data before entering into the QRAM, thus storing matrices $A_j|b_j$ for batches of a fixed size. For a given partition Π let X_j be the the $|S_j| \times n$ matrix obtained by selecting the rows corresponding to S_j. Define $A_j = X_j^T W_j X_j$ where W_j is the diagonal matrix of weights restricted to S_j.

The stochastic gradient descent algorithm starts with the initial condition $r_0 = (b - A_1\theta_0)$ and iteratively applies the updates $r_t = (I - \rho A_{t'})r_{t-1}$ on the residuals with $t' = t + 1 \mod k$. Note that as the number of steps τ is larger than the number of partitions k, we cyclically iterate the updates corresponding to the matrices $A_j, j \in [k]$. It is straightforward to implement these updates efficiently using Theorem 5.2 as the matrices A_j and the weights W_j are stored in memory. The running time is linear in the parameter $\mu = \max_{i \in [k]} \mu(X_k)$.

The updates in the classical stochastic gradient descent algorithm are affine, our quantum iterative method can therefore simulate these updates. The convergence analysis in equation (2) is not applicable for stochastic gradient descent, we refer to [5] for the classical convergence analysis. The correctness follows as the quantum algorithm follows is able to simulate each step of the classical stochastic gradient algorithm with sufficient precision.

In the case of linear systems and least squares, the updates were of the form $r_{t+1} = (I - \rho A)r_t$ for a fixed matrix A, and hence we could simultaneously apply t steps of the update in one step and hence have running time $C_U = O(C_V)$ using Proposition 4.10. In the stochastic gradient descent case, we have k different matrices A_t where k is the number of partitions for the dataset, and these matrices have different eigenbases. Therefore, we can only perform the linear updates sequentially, and the cost C_U for implementing U is $O(\tau C_V)$ in this case. Claim 3.3 therefore implies that the running time for the stochastic gradient descent is $C_U = O(\tau^2 C_V)$ as opposed to $C_U = O(\tau C_V)$ for the linear systems and weighted least squares.

Similar to the running time analysis in Section 3.3.2 we can also compute the running time for quantum stochastic gradient descent algorithm. If we use our linear system solver with precision dependence $O(1/\epsilon)$, then the running time is $O(\tau^4 \mu)$. However if we use a quantum linear system solver with precision dependence $\log(1/\epsilon)$, then the running time becomes $\widetilde{O}(\tau^2 \mu)$.

References

[1] S. Aaronson, "Read the fine print," *Nature Physics*, vol. 11, no. 4, pp. 291–293, 2015.

[2] A. Ambainis, "Variable time amplitude amplification and quantum algorithms for linear algebra problems," in *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, vol. 14. LIPIcs, 2012, pp. 636–647.

[3] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, and P. V. Srinivasan, "On the robustness of bucket brigade quantum ram," *New Journal of Physics*, vol. 17, no. 12, p. 123010, 2015.

[4] D. W. Berry, A. M. Childs, A. Ostrander, and G. Wang, "Quantum algorithm for linear differential equations with exponentially improved dependence on precision," *Communications in Mathematical Physics*, vol. 356, no. 3, pp. 1057–1081, 2017.

[5] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *arXiv preprint arXiv:1606.04838*, 2016.

[6] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemporary Mathematics*, vol. 305, pp. 53–74, 2002.

[7] D. Bulger, "Quantum basin hopping with gradient-based local optimisation," *arXiv preprint quant-ph/0507193*, 2005.

[8] S. Chakraborty, A. Gilyén, and S. Jeffery, "The power of block-encoded matrix powers: improved regression techniques via faster hamiltonian simulation," *arXiv preprint arXiv:1804.01973*, 2018.

[9] A. M. Childs, "On the relationship between continuous-and discrete-time quantum walk," *Communications in Mathematical Physics*, vol. 294, no. 2, pp. 581–603, 2010.

[10] A. M. Childs, R. Kothari, and R. D. Somma, "Quantum algorithm for systems of linear equations with exponentially improved dependence on precision," *SIAM Journal on Computing*, vol. 46, no. 6, pp. 1920–1950, 2017.

[11] A. Gilyén, S. Arunachalam, and N. Wiebe, "Optimizing quantum optimization algorithms via faster quantum gradient computation," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2019, pp. 1425–1444.

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

[13] A. W. Harrow, "Review of quantum algorithms for systems of linear equations," *arXiv preprint arXiv:1501.00008*, 2014.

[14] 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.

[15] S. P. Jordan, "Fast quantum algorithm for numerical gradient estimation," *Physical review letters*, vol. 95, no. 5, p. 050501, 2005.

[16] S. P. Jordan, "Permutational quantum computing," *arXiv preprint arXiv:0906.2508*, 2009.

[17] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, "q-means: A quantum algorithm for unsupervised machine learning," *arXiv preprint arXiv:1812.03584*, 2018.

[18] I. Kerenidis and A. Prakash, "Quantum recommendation systems," in *Proceedings of the 7th conference on Innovations in Theoretical Computer Science (ITCS)*, 2017.

[19] I. Kerenidis and A. Prakash, "A quantum interior point method for LPs and SDPs," *arXiv:1808.09266*, 2018.

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

[21] A. Y. Kitaev, A. Shen, and M. N. Vyalyi, *Classical and quantum computation.* American Mathematical Society Providence, 2002, vol. 47.

[22] Y. Liu and S. Zhang, "Fast quantum algorithms for least squares regression and statistical leverage scores," *Theoretical Computer Science*, vol. 657, Part A, pp. 38–47, 2017.

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

[24] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, 2014.

[25] R. Mathias, "The spectral norm of a nonnegative matrix," *Linear Algebra and its Applications*, vol. 139, pp. 269–284, 1990.

[26] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical review letters*, vol. 113, no. 13, p. 130503, 2014.

[27] P. Rebentrost, M. Schuld, F. Petruccione, and S. Lloyd, "Quantum gradient descent and Newton's method for constrained polynomial optimization," *arXiv preprint arXiv:1612.01789*, 2016.

[28] R. S. Sutton and A. G. Barto, "Introduction to reinforcement learning," 1998.

[29] 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.

[30] A. Ta-Shma, "Inverting well conditioned matrices in quantum logspace," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing.* ACM, 2013, pp. 881–890.

[31] E. Tang, "A quantum-inspired classical algorithm for recommendation systems," in *Proceedings of the fifty-first annual ACM symposium on Theory of computing (To appear), arXiv:1807.04271*, 2019.

[32] N. Wiebe, D. Braun, and S. Lloyd, "Quantum algorithm for data fitting," *Physical review letters*, vol. 109, no. 5, p. 050505, 2012.

[33] L. Wossnig, Z. Zhao, and A. Prakash, "A quantum linear system algorithm for dense matrices." *Physical review letters*, vol. 120, no. 5, p. 050502, 2018.