Quantum Algorithms for Regularized Least Squares

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computational Natural Sciences
by Research

by

Aditya Morolia
20171177
aditya.morolia@research.iiit.ac.in

International Institute of Information Technology (IIIT)
Hyderabad - 500 032, INDIA
June 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Quantum Algorithms for Regularized Least Squares" by Aditya Morolia, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Co-Adviser: Prof. Shantanav Chakraborty

_____ _____

Date Co-Adviser: Prof. Indranil Chakrabarty

To Mom and Dad.

"Physics is mathematics in a sexy black dress."

Abstract

Linear regression is a widely used technique to fit linear models and finds widespread applications across different areas of natural sciences, as well as field such as machine learning and statistics. In most real-world scenarios, however, linear regression problems are often *ill-posed* or the underlying model suffers from *overfitting*, leading to erroneous or trivial solutions. This is often dealt with by adding extra constraints, known as regularization. For instance, suppose that we are given N data points $\{(a_i, b_i)\}_{i=1}^N$ where $\forall i : a_i \in \mathbb{R}^d, \forall i : b_i \in \mathbb{R}$. The assumption is that there is a vector x such that $b_i = x^T a_i + e_i$, where e_i is a random variable (noise) with mean 0. Suppose A is the data matrix of dimension $N \times d$, such that its i^{th} row is the vector a_i and $b \in \mathbb{R}^N$ such that $b = (b_1, \cdots, b_N)^T$. Adding an ℓ_2-regularization, the objective is to obtain x that minimizes

$$\|Ax - b\|_2^2 + \lambda \|Lx\|_2^2 \tag{0.0.1}$$

where L is an appropriately chosen penalty matrix (or regularization matrix) of dimension $N \times d$ and $\lambda > 0$ is the regularization parameter, an appropriately chosen constant. This regularization technique is known as *general ℓ_2-regularization* in the literature. It is a generalization of the well known *ridge regression* which corresponds to the case when L is the identity matrix. The closed-form solution of the general ℓ_2-regularized ordinary least squares problem is given by

$$x = \left(A^T A + \lambda L^T L\right)^{-1} A^T b. \tag{0.0.2}$$

A straightforward observation is that even when $A^T A$ is singular, regularization ensures that the condition number (ratio of the maximum and the minimum singular values) of the resulting matrix is finite and therefore $A^T A + \lambda L^T L$ is invertible. Another observation is that when the condition number of A is arbitrarily large (the minimum singular value of A is arbitrarily small), the condition number of the matrix to be inverted in Equation 0.0.2 can be tamed significantly. Since the complexity of most quantum algorithms for the quantum versions of these problems depend the condition number, the complexity reduces significantly. These models can be generalized to situations where the data points are weighted or correlated which incorporate more realistic scenarios.

Quantum linear systems solvers are possibly among the most studied quantum algorithms. In this work we develop the first quantum algorithms for ordinary, weighted and generalized least squares with generalized ℓ_2-regularization. We assume access to (approximate) block-encodings of the relevant matrices and develop robust versions of quantum singular value transformations to implement linear transformations of these matrices. Our goal is to output a quantum state that is δ-close to $|x\rangle$, the state encoding the solution to the underlying regularized least squares problem. Owing to the generality of the block-encoding framework, our algorithms are applicable to a variety of input models. In order to obtain our results, we work with approximate block-encodings of matrices and apply robust quantum singular value transformations (QSVT) to them. For most quantum linear algebra applications using QSVT, access to perfect block-encoding is assumed, and the robustness of such algorithms is left out.

We develop a space-efficient variable time amplitude amplification (VTAA) procedure, which we then use in a variable-time matrix inversion algorithm. A crucial requirement for variable time matrix inversion algorithm is the application of the inversion procedure to the portion of the input state that is spanned by singular values larger than a certain threshold. In order to achieve this, prior results have made use of *Gapped Phase Estimation* (GPE), which is a simple variant of the standard phase estimation procedure. However, GPE requires extra registers that store the estimates of the phases, which are never used in the variable-time algorithm. In this work, instead of GPE, we make use of a robust version of *quantum eigenvalue discrimination* (QEVD) using QSVT, which decides whether the eigenvalue of a matrix is above or below a certain threshold without storing the eigenvalue estimates. Given the block-encoding of a matrix A with condition number κ, this procedure for implementing A^+ reduces the number of additional qubits required for our variable time by a factor of $O(\log^2(\kappa/\delta))$ as compared to prior results, where δ is the desired accuracy. We show that in order to implement A^+, the precision required in the block-encoding of A is determined in turn by the precision in the QEVD

procedure. Consequently, this improves both variable-time amplitude amplification and quantum linear systems algorithm: we achieve optimal complexity (linear dependence in the condition number and polylogarithmic dependence on inverse-accuracy) using far fewer ancilla qubits. We then use this to develop our algorithms for ordinary, weighted and generalized least squares with general ℓ_2-regularization. The primitives we have developed, and our approach to designing the algorithms might be of independent interest to the community, specially in the domain of quantum machine learning (QML) and quantum algorithms for natural sciences, where such quantum linear algebra subroutines (QLAS) are frequently used.

Acknowledgment

This work is a consequence of the amazing time I had at IIIT Hyderabad, lead above all by the amazing people I met there. This marks a humble beginning to what I hope would become a lifetime of joy in academia and learning.

I thank my amazing advisors, Prof. Indranil Chakrabary and Prof. Shantanav Chakraborty. They have been with me every step of the way, with their unconditional support and guidance. Through Prof. Indranil I was introduced to the vastness of the field, and have met some really amazing people along the way. His support and constant guidance through the time I had just been introduced to the field had build the foundation for everything to come up. In Prof. Shantanav I have found a mentor and a friend that I hope to carry along throughout my life, and no amount of text can acknowledge the fact. They also gave me an opportunity to work as a teaching assistant for their courses, which was an amazing experience that made me learn a lot. Besides our work, they have introduced me to other aspects of life including literature, music, prose and philosophy that has had a significant impact on the person I'm becoming.

Along with them, I have been fortunate to have an amazing peer group at IIIT, including Prof. Harjinder Singh (Laltu), Prof. Subhadip Mitra, Prof. Samyadeb Bhattacharya, Prof. Siddhartha Das and Prof. Kannan Srinathan. Talking to them has always been a joy and has taught me a lot. The team gave me the freedom to pursue my passion that has made this journey incredibly fun. Prof. Laltu has been an inspiration since my first week of college, and every interaction with him is something I will cherish forever.

I thank Prof. Guruprasad Kar and his team at PAMU for graciously hosting me at ISI Kolkata for the summer school and teaching me the nuances of the quantum theory. I also thank Prof. A. K. Pati for inviting me to HRI Allahabad and working with me on quantum information theory. His dedication to research in his field is an inspiration.

I also thank Prof. Barry C. Sanders for giving me an opportunity to work with him and his team. His dedication to his work and his knowledge of the field was inspiring and I do hope to reach there someday.

I thank all my friends at IIIT for helping shape the person I am. Anurudh Peduri, Mahathi Vempati and Siddhartha Bhat, you have been an inspiration since the day I met you. Every conversation I had with you has taught me many, many things, and it was through you that I found the field of quantum computing which has lead to this thesis. Anurudh, working with you has filled with learning and joy, and I was really lucky to have had this experience. I thank Utkarsh for entertaining all of my often annoying and silly questions and helping me every way he did. I thank Rohan, Keshav, Suryansh, Animesh, Zeeshan, Alapan, Kushagra, Rutvij, Jayadev, Athreya, Tanmay, Yash and Srinidhi for making the lab fun. I hope we have more conversations like the ones we did during my time in college.

Finally, I thank my parents Manish and Sanju, the most important people in my life. You gave me the freedom and privilege to worry about nothing and focus on my passion, without which I would have amounted to nothing. I thank my sister Aditi, my free therapy on call, for her love and support. And I thank my family for always believing in me during whatever it is I choose to do. Your love and support means a lot. Prof. Siddhartha, thank you for letting me annoy you with all the silly questions I had, and the long calls you spent answering them. And know that they're not gonna stop coming anytime soon. Most importantly, I thank my grandfather for raising me into the person I am. Making you proud is the most important thing I will do in my life and I hope that one day I could become a person half as inspiring as you are.

Contents

List of Theorems/Definitions/Lemmas

List of Figures

Chapter 1

Overview and Motivation

1.1 Motivation

Physics and computation have been closely related since the conception of a computers. Computers of some form have always been the tools that we use to assist the production of knowledge [1], giving rise to new discoveries in natural sciences by simulating the physics of the universe, while on the other hand advances in physics have made engineering of even more powerful computers possible, producing as a byproduct an insanely high number of dressed up golden retrievers and arrogant cat memes.

Recent connections made in the field of black hole information paradox and holographic duality have highlighted the importance of theoretical computer science in theoretical physics [2]. In fact, what is and isn't computable depends on the underlying physics of the universe we live in [1]. Therefore at the height of the quantum revolution in physical sciences, it was natural to look at what power a programmable, quantum mechanical device would give us. The vastness of the Hilbert space and the superposition principle are obvious advantages that the quantum theory would bring to the table, and this has been provably demonstrated when we care only about global properties [i] of the underlying functions to be computed (e.g. Deutsch Jozsa algorithm, Shor's algorithm). But despite immense focus on the subject, a provable advantage to more practical problems remains to be seen.

An important class of problems that shows up frequently in natural sciences is that of finding solutions to linear systems. Given a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$, we have to find a vector $x \in \mathbb{R}^n$ such that $Ax = b$, or find approximate solutions to the system, by minimizing a loss function in Ax, b, and some additional constraints. These are the class of problems that will be focused upon in this work. There has been extensive work done in this domain, some of which will be introduced and improved upon in this work. We shall begin this chapter by informally introducing linear systems and regularization, followed by an overview of some applications. Then we shall briefly state the results derived in this manuscript.

1.2 Introduction

The problem of fitting a theoretical model to a large set of experimental data appears across various fields ranging from the natural sciences to machine learning and statistics [3]. Linear regression is one of the most widely used procedures for achieving this. By assuming that, for the underlying model, there exists a linear relationship between a dependent variable and one or more explanatory variables, linear regression constructs the best linear fit to the series of data points. Usually, it does so while minimizing the sum of squared errors - known as the least squares method.

In other words, suppose that we are given N data points $\{(a_i, b_i)\}_{i=1}^N$ where $\forall i : a_i \in \mathbb{R}^d, \forall i : b_i \in \mathbb{R}$. The assumption is that there is a vector x such that $b_i = x^T a_i + e_i$, where e_i is a random variable (noise) with mean 0. Suppose A is the data matrix of dimension $N \times d$, such that its i^{th} row is the vector a_i and $b \in \mathbb{R}^N$ such that $b = (b_1, \cdots, b_N)^T$. Then the procedure, known as ordinary least squares, obtains a vector $x \in \mathbb{R}^d$ that minimizes the objective function $||Ax - b||_2^2$. This problem has a closed-form solution given by $x = (A^T A)^{-1} A^T b = A^+ b$, where A^+ denotes the Moore-Penrose inverse of the matrix A. Thus computationally, finding the best fit by linear regression reduces to finding the pseudo-inverse of a matrix that represents the data, a task that is expensive for classical machines for large data sets. Classically two of the most widely used algorithms are

[i] Global properties are those that are true for the whole input space. For example, the property of a function being constant or balanced (as considered in the Deutsch-Jozsa problem.)

steepest descent, and conjugate gradient descent. For a symmetric, positive-definite $N \times N$ matrix M with condition number κ, these algorithms run in time $\mathcal{O}\big(\text{nnz}(M)\kappa \log\left(\frac{1}{\varepsilon}\right)\big)$ and $\mathcal{O}\big(\text{nnz}(M)\sqrt{\kappa} \log\left(\frac{1}{\varepsilon}\right)\big)$ respectively, where $\text{nnz}(M)$ is the number of non-zero entries in M, and ε is the multiplicative error, relative to the error in the choice of the initial point [4].

In practice, however, least squares regression runs into problems such as *overfitting*. For instance, the solution might fit most data points, even those corresponding to random noise. Furthermore, the linear regression problem may also be *ill-posed*, for instance, when the number of variables exceeds the number of data points rendering it impossible to fit the data. These issues come up frequently with linear regression models and result in erroneous or trivial solutions. Furthermore, another frequent occurrence is that the data matrix A has linearly dependent columns. In this scenario, the matrix $A^T A$ is not full rank and therefore is not invertible.

Regularization is a widely used technique to remedy these problems, not just for linear regression but for inverse problems in general [5]. In the context of linear regression, broadly, this involves adding a penalty term to the objective function, which constrains the solution of the regression problem. For instance, in the case of ℓ_2-regularization, the objective is to obtain x that minimizes

$$\|Ax - b\|_2^2 + \lambda \|Lx\|_2^2 \tag{1.2.1}$$

where L is an appropriately chosen penalty matrix (or regularization matrix) of dimension $N \times d$ and $\lambda > 0$ is the regularization parameter, an appropriately chosen constant. This regularization technique is known as *general ℓ_2-regularization* or *Tikhonov regularization* in the literature [6, 7, 8, 9, 10]. It is a generalization of *ridge regression* which corresponds to the case when L is the identity matrix [11, 12, 13]. The closed-form solution of the general ℓ_2-regularized ordinary least squares problem is given by

$$x = \left(A^T A + \lambda L^T L\right)^{-1} A^T b. \tag{1.2.2}$$

A straightforward observation is that even when $A^T A$ is singular, regularization ensures that the condition number (ratio of the maximum and the minimum singular values) of the resulting matrix is finite and therefore $A^T A + \lambda L^T L$ is invertible.

In this work, we develop quantum algorithms for linear regression with general ℓ_2-regularization. If the optimal solution is $x = (x_1, \cdots, x_d)^T$, then our quantum algorithm outputs a quantum state that is δ-close to $|x\rangle = \sum_{j=1}^{d} x_j |j\rangle / \|x\|$, assuming access to the matrices A, L, and the quantum state $|b\rangle$ via general quantum input models.

In several practical scenarios, depending on the underlying theoretical model, generalizations of the ordinary least squares (OLS) technique are more useful to fit the data. For instance, certain samples may be of more importance (and therefore have more weight) than the others, in which case weighted least squares (WLS) is preferred. Generalized least squares (GLS) is used when the underlying samples obtained are correlated. These techniques also suffer from the issues commonplace with OLS, warranting the need for regularization [10]. Consequently, we also design algorithms for quantum WLS with general ℓ_2-regularization and quantum GLS with general ℓ_2-regularization.

1.3 Applications of Quantum Linear Algebra

Quantum Linear Systems Solvers are one of the most important quantum algorithmic primitives that find applications for solving problems in natural sciences, statistics and machine learning. Here we briefly mention some common applications of these algorithms.

1.3.1 Simulating Quantum Systems

Simulation of quantum systems was the original motivation behind the conception of quantum computers [14]. A quantum state belonging to a d-dimensional Hilbert space requires $2^d - 1$ variables for a complete description, exponential in the dimension. Any simulation of a quantum evolution would thus require manipulating *probability amplitudes* exponential in numbers in the dimension of the Hilbert space. This would allow us to compute the complete description of the final state of the system. A quantum computer will be able to bypass this requirement by encoding the description into a quantum state itself, thus exponentially reducing the compute power required for the task since we would now need to perform quantum operations only on the d states (and thus only d possible parallel operations). While this seems like a clear exponential advantage, the caveat is that to extract the information out of the final quantum state, we would have to make a measurement. According to the measurement postulate of quantum mechanics [15], the state would thus collapse to one of the eigenstates of the system, thus possibly losing information (depending on the basis in which the measurement is made). We

can only *sample from* the spectra of an observable according to the probability distribution corresponding to the amplitudes of the final state of the system, but not get the complete description of the state. This limits the possibilities, but when the task is to find certain global properties of the system, this can still be efficient. The Deutsch-Jozsa algorithm is a good example of an algorithm that makes use of this phenomenon [16]. Despite these limitations, there has been progress made towards this goal [17, 18, 19, 20].

1.3.2 Quantum Machine Learning

Machine learning is an important tool for finding patterns in data (samples from some probability distribution). Quantum systems are known to produce samples that might be hard to sample from using classical algorithms. The development of quantum algorithms gives the opportunity to use these techniques to train machine learning models on quantum computers, which could allow for improvements in the training time (and other resources) required, as well as performance [21]. A lot of problems while training ML models reduce to solving linear systems.

Least squares fitting and data analysis is an important application of quantum linear systems algorithms (QLSA). Wiebe et al. [22] used the HHL algorithm [23] to develop an algorithm that efficiently determines the quality of least squares fit. This was followed by a plethora of work in quantum computing models such as quantum principal component analysis [24] and quantum support vector machines [25]. There also has been extensive work on quantum neural networks. The simplest model considers a *Parameterized Quantum Circuit* as a machine learning model [26], which considers a quantum circuit with parameterized rotations, the parameters to which are optimized by a classical computer based on the expectation value of certain measurements made on the output qubits. Figure 1.1 from [26] is a schematic example of such a model.

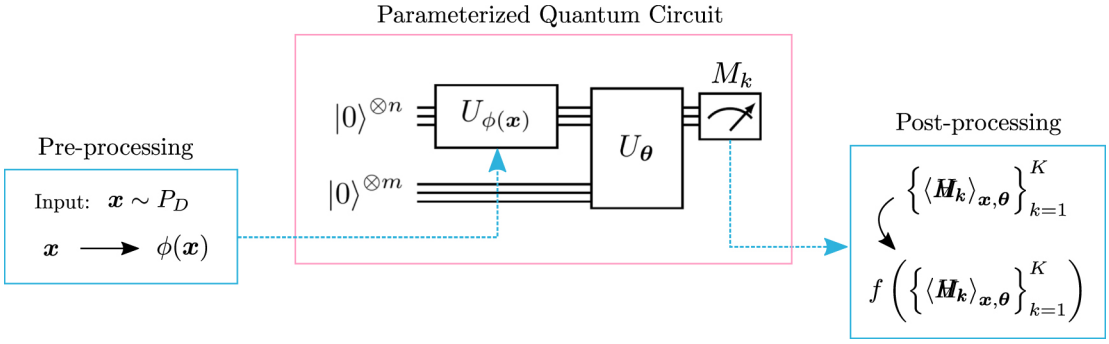


Figure 1.1: Parameterized Quantum Circuit. The pre and post processing steps are performed on a classical machine. The data vector x is fed to the quantum circuit via an encoder circuit $U_{\phi(x)}$, which is parameterized by the output of the preprocessing step $\phi(x)$. Then a variational circuit $U(\theta)$, parameterized by θ acts on the state, followed by a measurement. A classical computer then takes the measurement samples and updates the parameter θ for the next iteration over some objective which depends on the measurement outcomes.

1.3.3 Solving Linear Differential Equations

Mathematical descriptions of various physical systems are linear partial differential equations, in which the time derivative depends linearly on spatial derivatives. Examples include Maxwell's equation, the heat equation and Stokes equation. Discretization of PDEs gives sparse ordinary differential equations of very high dimensions, which can be converted to first order ODEs.

Berry [27] extended linear systems solvers to an algorithm for solving general inhomogeneous sparse linear differential equations. For an equation of the form

$$\dot{x}(t) = A(t)x(t) + b(t), \tag{1.3.1}$$

where A is an $n \times n$ sparse matrix, a classical algorithm must take time at least linear in n. On the other hand, the proposed quantum algorithm takes time $\mathcal{O}(\text{polylog}\,(n))$[ii].

Similarly, Montanaro and Pallister [29] showed how to use the QLSA to solve finite element method problems to achieve a polynomial quantum speedup, which can be used to find approximate solutions to various models in physics and mathematics such as the Black-Scholes equation in mathematical finance and for solving a PDE

[ii] As is the case with all such quantum algorithms, these prepare a quantum state with uniform amplitudes proportional to those of the desired vector, which is an important detail that should be highlighted. Refer to [28]

where the dimension grows with the number of bodies in many-body dynamics. This method discretizes the parameter space and finds an approximate solution by solving a large system of linear equations.

1.4 Prior Work

Earlier, quantum algorithms for (unregularized) linear regression was first developed by Wiebe et al. [22], wherein the authors made use of the HHL algorithm (section 4.1) for solving a linear system of equations [23]. Their algorithm assumes query access to a sparse matrix A (*sparse-access-model*) and to a procedure to prepare $|b\rangle = \sum_i b_i |i\rangle$. They first prepare a quantum state proportional to $A^T |b\rangle$, and then use the HHL algorithm to apply the operator $(A^T A)^{-1}$ to it. Overall the algorithm runs in a time scaling as κ_A^6 (the condition number of A) and inverse polynomial in the accuracy δ. Subsequent results have considered the problem of obtaining classical outputs for linear regression. For instance, in Ref. [30], A^+ is directly applied to the quantum state $|b\rangle$, followed by amplitude estimation to obtain the entries of x. On the other hand, Ref. [31] used the techniques of quantum principal component analysis in [24] to predict a new data point for the regression problem. These algorithms also work in the *sparse access model* and run in a time that scales as poly $(\kappa, 1/\delta)$. Kerenidis and Prakash [32] provided a quantum algorithm for the WLS problem wherein they used a classical data structure to store the entries of A and W. Furthermore, they assumed QRAM access to this data structure [33, 34] that would allow the preparation of quantum states proportional to the entries of A and W efficiently. They showed that in this input model (*quantum data structure model*), an iterative quantum linear systems algorithm can prepare $|x\rangle$ in time $\widetilde{O}(\mu\kappa^3/\delta)$, where κ is the condition number of the matrix $A^T\sqrt{W}$ while $\mu = \left\|\sqrt{W}A\right\|_F$. Chakraborty et al. [35] applied the framework of block-encoding along with (controlled) Hamiltonian simulation of Low and Chuang [36] to design improved quantum algorithms for solving linear systems. Quantum algorithms developed in the block-encoding framework are applicable to a wide variety of input models, including the sparse access model and the quantum data structure model of [32]. They applied their quantum linear systems solver to develop quantum algorithms for quantum weighted least squares and generalized least squares. Their quantum algorithm for WLS has a complexity that is in $\widetilde{O}\left(\alpha\kappa \text{polylog}(Nd/\delta)\right)$, where $\alpha = s$, the sparsity of the matrix $A^T\sqrt{W}$ in the sparse access model while $\alpha = \left\|\sqrt{W}A\right\|_F$, for the quantum data structure input model. For GLS, their quantum algorithm outputs $|x\rangle$ in cost $\widetilde{O}\left(\kappa_A\kappa_\Omega(\alpha_A + \alpha_\Omega\kappa_\Omega)\text{polylog}(1/\delta)\right)$, where κ_A and κ_Ω are the condition numbers of A and Ω respectively while α_A and α_Ω are parameters that depend on how the matrices A and Ω are accessed in the underlying input model.

Yu et al. [37] developed a quantum algorithm for *ridge regression* in the sparse access model using the LMR scheme [24] for Hamiltonian simulation and quantum phase estimation, which they then used to determine the optimal value of the parameter λ. Their algorithm to output $|x\rangle$ has a cubic dependence on both κ and $1/\delta$. They use this as a subroutine to determine a good value of λ. A few other works [38, 39] have considered the quantum ridge regression problem in the sparse access model, all of which can be implemented with poly$(\kappa, 1/\delta)$ cost.

Recently, Chen and de Wolf designed quantum algorithms for lasso (ℓ_1-regularization) and ridge regressions from the perspective of empirical loss minimization [40]. For both lasso and ridge, their quantum algorithms output a classical vector $\widetilde{x}$ whose loss (mean squared error) is δ-close to the minimum achievable loss. In this context, they prove a quantum lower bound of $\Omega(d/\delta)$ for ridge regression which indicates that in their setting, the dependence on d cannot be improved on a quantum computer (the classical lower bound is also linear in d and there exists a matching upper bound). Note that $\widetilde{x}$ is not necessarily close to the optimal solution x of the corresponding least squares problem, even though their respective loss values are. Moreover, their result (of outputting a classical vector $\widetilde{x}$) is incomparable to our objective of obtaining a quantum state encoding the optimal solution to the regularized regression problem.

Finally, Gilyén et al. obtained a "dequantized" classical algorithm for ridge regression assuming norm squared access to input data similar to the quantum data structure input model [41]. Furthermore, similar to the quantum setting where the output is the quantum state $|x\rangle = \sum_j x_j |j\rangle / \|x\|$ instead of x itself, their algorithm obtains samples from the distribution $x_j^2/\|x\|^2$. For the regularization parameter $\lambda = O\left(\|A\|\|A\|_F\right)$, the running time of their algorithm is in $\widetilde{O}\left(\kappa^{12} r_A^3/\delta^4\right)$, where r_A is the rank of A. Their result (and several prior results) does not have a polynomial dependence on the dimension of A and therefore rules out the possibility of generic exponential quantum speedup (except in δ) in the quantum data structure input model.

1.5 Research Focus and Contributions

In this work, we design the first quantum algorithms for OLS, WLS, and GLS with general ℓ_2-regularization. We assume that the relevant matrices are provided as input in the *block-encoding* model and use the Quantum Singular Value Transformation (QSVT) framework introduced by Gilyén et al [42]. The block-encoding model assumes that an operator (close to) A/α is encoded in the top-left block of some unitary U_A. The parameter α takes specific values depending on the underlying input model. On the other hand, QSVT allows us to implement nearly arbitrary polynomial transformations to a block of a unitary matrix using a series of parameterized, projector-controlled rotations (quantum signal processing [43]). Our results utilize both these frameworks.

More precisely, given approximate block-encodings of the data matrix A and the regularizing matrix L, and a unitary procedure to prepare the state $|b\rangle$, our quantum algorithms output a quantum state that is δ-close to $|x\rangle$, the quantum state proportional to the ℓ_2-regularized ordinary least squares (or weighted least squares or generalized least squares problem). We briefly summarize the query complexities of our results in Table 1.1.

For the OLS problem with general ℓ_2-regularization (chapter 5, Theorem 5.1.2), we design a quantum algorithm which given an $(\alpha_A, a_A, \varepsilon_A)$-block-encoding of A (implemented in cost T_A), an $(\alpha_L, a_L, \varepsilon_L)$-block-encoding of L (implemented in cost T_L), a parameter $\lambda > 0$, and a procedure to prepare $|b\rangle$ (in cost T_b), outputs a quantum state which is δ-close to $|x\rangle$. The algorithm has a cost

$$\mathcal{O}\left(\kappa \log \kappa \left(\left(\frac{\alpha_A + \sqrt{\lambda}\alpha_L}{\|A\| + \sqrt{\lambda}\|L\|}\right) \log\left(\frac{\kappa}{\delta}\right)(T_A + T_L) + T_b\right)\right)$$

where κ can be thought of as a *modified condition number*, related to the effective condition numbers of A and L. When L is a good regularizer, this is given by the expression

$$\kappa = \kappa_L \left(1 + \frac{\|A\|}{\sqrt{\lambda}\|L\|}\right),$$

Notice that κ is independent of κ_A, the condition number of the data matrix A, which underscores the advantage of regularization. The parameters α_A and α_L take specific values depending on the underlying input model. For the sparse access input model, $\alpha_A = s_A$ and $\alpha_L = s_L$, the respective scarcities of the matrices A and L. On the other hand for the quantum data structure input model, $\alpha_A = \|A\|_F$ and $\alpha_L = \|L\|_F$. Consequently, the complexity of *Quantum Ridge Regression* can be obtained by substituting $L = I$ in the above complexity as

$$\mathcal{O}\left(\log \kappa \left(\frac{\alpha_A}{\sqrt{\lambda}} \log\left(\frac{\kappa}{\delta}\right) T_A + \kappa T_b\right)\right)$$

where $\kappa = 1 + \|A\|/\sqrt{\lambda}$, by noting that the block-encoding of I is trivial while the norm and condition number of the identity matrix is one. For this problem of *quantum ridge regression*, our quantum algorithms are substantially better than prior results [38, 37, 39], exhibiting a polynomial improvement in κ and an exponential improvement in $1/\delta$.

For the ℓ_2-regularized GLS problem (chapter 5, Theorem 5.2.7), we design a quantum algorithm that along with approximate block-encodings of A and L, takes as input an $(\alpha_\Omega, a_\Omega, \varepsilon_\Omega)$-lock-encoding of the matrix Ω (implementable at a cost of T_Ω) to output a state δ-close to $|x\rangle$ at a cost of

$$\mathcal{O}\left(\kappa\sqrt{\kappa_\Omega} \log \kappa \left(\left(\frac{\alpha_A}{\|A\|}T_A + \frac{\alpha_L}{\|L\|}T_L + \frac{\alpha_\Omega \kappa_\Omega}{\|\Omega\|}T_\Omega\right) \log^3\left(\frac{\kappa\kappa_\Omega \|A\|\|L\|}{\delta\|\Omega\|}\right) + T_b\right)\right)$$

In the above complexity, when L is a good regularizer, the modified condition number κ is defined as

$$\kappa = \kappa_L \left(1 + \frac{\sqrt{\kappa_\Omega}\|A\|}{\sqrt{\lambda\|\Omega\|}\|L\|}\right)$$

The WLS problem is a particular case of GLS, wherein the matrix Ω is diagonal. However, we show that better complexities for the ℓ_2-regularized WLS problem can be obtained if we assume QRAM access to the diagonal entries of W (chapter 5, Theorem 5.2.4 and Theorem 5.2.5).

Table 1.1 summarizes the complexities of our algorithms for quantum linear regression with general ℓ_2-regularization. For better exposition, here we assume that $\|A\|, \|L\|, \|\Omega\|$ and $\lambda = \Theta(1)$. For the general expression of the complexities, we refer the readers to chapter 5.

Problem	**Unregularized**	**ℓ_2-Regularized**
Quantum OLS	$\widetilde{\mathcal{O}}\left(\alpha_A \kappa_A \log\left(1/\delta\right)\right)$	$\widetilde{\mathcal{O}}\left((\alpha_A + \alpha_L)\kappa_L \log\left(1/\delta\right)\right)$
Quantum GLS	$\widetilde{\mathcal{O}}\left(\left(\alpha_A + \alpha_\Omega \kappa_\Omega\right) \kappa_A \sqrt{\kappa_\Omega} \log^3\left(1/\delta\right)\right)$	$\widetilde{\mathcal{O}}\left(\left(\alpha_A + \alpha_L + \alpha_\Omega \kappa_\Omega\right) \kappa_L \sqrt{\kappa_\Omega} \log^3\left(1/\delta\right)\right)$

Table 1.1: Complexity of quantum linear regression algorithms with and without general ℓ_2-regularization. All of these algorithms require only $\Theta(\log \kappa)$ additional qubits.

In order to derive our results, we take advantage of the ability to efficiently perform arithmetic operations on block-encoded matrices, as outlined in section 3.2. Along with this, we use QSVT to perform linear algebraic operations on block-encoded matrices. To this end, adapt the results in Refs. [42, 44] to our setting. One of our contributions is that we work with robust versions of many of these algorithms. In prior work, QSVT is often applied to block-encoded matrices, assuming perfect block-encoding. For the quantum algorithms in this paper, we rigorously obtain the precision ε required to obtain a δ-approximation of the desired output state.

For instance, a key ingredient of our algorithm for regularized least squares is to make use of QSVT to obtain A^+, given an ε-approximate block-encoding of A. In order to obtain a (near) optimal dependence on the condition number of A by applying variable-time amplitude amplification (VTAA) [45], we recast the standard QSVT algorithm as a variable stopping-time quantum algorithm. Using QSVT instead of controlled Hamiltonian simulation ensures that the variable-time quantum procedure to prepare $A^+ |x\rangle$ has a slightly better running time (by a log factor) and considerably fewer additional qubits than Refs. [46, 35].

Furthermore, for the variable time matrix inversion algorithm, a crucial requirement is the application of the inversion procedure to the portion of the input state that is spanned by singular values larger than a certain threshold. In order to achieve this, prior results have made use of *Gapped Phase Estimation* (GPE), which is a simple variant of the standard phase estimation procedure [45, 46, 35]. However, GPE requires extra registers that store the estimates of the phases, which are never used in the variable-time algorithm. In this work, instead of GPE, we make use of a robust version of *quantum eigenvalue discrimination* (QEVD) using QSVT, which decides whether the eigenvalue of a matrix is above or below a certain threshold without storing the eigenvalue estimates [42, 44]. This further reduces the number of additional qubits required for our variable time by a factor of $O(\log^2(\kappa/\delta))$ as compared to prior results [46, 35]. We show that in order to implement A^+, the precision required in the block-encoding of A is determined in turn by the precision in the QEVD procedure. Consequently, this also implies that in our framework, quantum algorithms for (unregularized) least squares (which are special cases of our result) have better time and space complexities than those of Ref. [35].

1.6 Outline of the Thesis

In chapter 2 we discuss the prerequisites needed to build our algorithms. We begin by discussing the two major models of quantum computers, namely circuit model and adiabatic model (section 2.3). In section 2.1 we describe what ordinary, weighted and generalized least squares are, and introduce regularization of linear systems. Then in section 2.4 we describe the block-encoding input model, which is how we assume access to the inputs to our algorithms. We also describe the commonly used QROM and Sparse Access Input models. In section 2.5 we describe quantum singular value transformation, which we use to develop all our algorithms. Finally, in section 2.6 we describe the technique of variable time amplitude amplification, an amplitude amplification technique which we improve and use in our algorithms to boost success probability without a significant overhead.

In chapter 3 we build various algorithmic primitives that we will then use in our algorithms as subroutines. In chapter 4 we first describe the original HHL algorithm. Then we describe an adiabatic quantum algorithm for solving linear systems. We then build our own linear systems solver using QSVT and our improved VTAA technique to achieve optimal complexity. We use the algorithmic primitives and our linear systems solver to build quantum algorithms for regularized versions of OLS, WLS and GLS problems in chapter 5. We conclude in chapter 6 with some open problems and discussion.

1.7 Notation

In this section we lay down the notation used throughout in this thesis. For a matrix $A \in \mathbb{R}^{N\times d}$, $A_{i,:}$ denotes the i^{th} row of A, and $\|A_{i,:}\|$ denotes the vector norm of $A^T_{i,:}$. s^A_r and s^A_c denote the row and column sparsity of the matrix, which is the maximum number of non-zero entries in any row and any column respectively.

When talking about quantum input models (section 2.4), the following notation is used. $A^{(p)}$ is defined as $A^{(p)}_{i,j} = (A_{i,j})^p$. Also, $\forall q \in [0,1] : s_q(A) := \max_{i\in M} \|A_{i,:}\|^q_q$ and $\forall p \in [0,1] : \mu_p(A) := \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}$.

Singular Value Decomposition. The decomposition $A = W\Sigma V^\dagger$, where W and V are unitary and Σ is a diagonal matrix, represents the *singular value decomposition* (SVD) of A. All matrices can be decomposed in this form. The diagonal entries of Σ, usually denoted by $\sigma(A) = \{\sigma_j\}$, is the multiset of all *singular values* of A, which are real and non-negative. $\sigma_{\max}$ and $\sigma_{\min}$ denote the maximum and minimum singular values of A. $r(A) = \text{rank}(A)$ is the number of non-zero singular values of A. The columns of W, V (denoted by $\{|w_j\rangle\}$ and $\{|v_j\rangle\}$) are the left and right *singular vectors* of A. Thus $A = \sum_j^r \sigma_j |w_j\rangle \langle v_j|$. The singular vectors of A can be computed as the positive square roots of the eigenvalues of A^TA (which is Hermitian, and therefore has real eigenvalues.)

Effective Condition Number. κ_A denotes (an upper bound on) the effective condition number of A, defined as the ratio of the maximum and minimum non-zero singular values of A. Let $\sigma_{\max}(A)$ be the largest singular value of A, and $\sigma_{\min}(A)$ be the smallest singular value of A. Additionally, let $\widetilde{\sigma}_{\min}(A)$ be the smallest non-zero singular value of A. Then

$$\kappa_A \geq \frac{\sigma_{\max}(A)}{\widetilde{\sigma}_{\min}(A)} = \sqrt{\frac{\lambda_{\max}(A^\dagger A)}{\widetilde{\lambda}_{\min}(A^\dagger A)}}$$

If A is full-rank, then $\widetilde{\sigma}_{\min}(A) = \sigma_{\min}(A)$, and κ_A becomes the condition number of the matrix. In this text – unless stated otherwise – we always refer to κ_A as (an upper bound on) effective condition number of a matrix, and not the true condition number.

Norm. Unless otherwise specified, $\|A\|$ denotes the spectral norm of A, while $\|A\|_F$ denotes the Frobenius norm of A, defined as

$$\|A\| := \max_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \sigma_{\max}(A)$$

$$\|A\|_F := \sqrt{\sum_{j=1}^{r} \sigma_j^2}$$

Unless otherwise specified, when A is assumed to be normalized, it is with respect to the spectral norm.

Big-O Notation. Let f and g be real valued functions in t defined on a subset of $\mathbb{R}$ unbounded from above. Then we have the following notations to represent their asymptotic scaling with respect to each other.

$$f(t) = \mathcal{O}(g(t)) \iff \exists t_0, C \in \mathbb{R} : \forall t > t_0 : |f(t)| \leq C|g(t)|, \tag{1.7.1}$$

which means that f does not scale faster than g. Similarly,

$$f(t) = \Omega(g(t)) \iff \exists t_0, C \in \mathbb{R} : \forall t > t_0 : |f(t)| \geq C|g(t)|, \tag{1.7.2}$$

which means that f scales faster than g. And

$$f(t) = \Theta(g(t)) \iff f(t) = \mathcal{O}(g(t)) \wedge g(t) = \mathcal{O}(f(t)), \tag{1.7.3}$$

which means that they scale similarly (asymptotically.)

Soft-O Complexity. Finally, we use $f = \widetilde{\mathcal{O}}(g)$ to denote $f = \mathcal{O}(g \cdot \text{polylog}(g))$.

Controlled Unitaries. If U is a s-qubit unitary, then $C\text{-}U$ is a $(s+1)$-qubit unitary defined by

$$C\text{-}U = |0\rangle\langle 0| \otimes I_s + |1\rangle\langle 1| \otimes U$$

Throughout this text whenever we state that the time taken to implement a unitary U_A is T_A and the cost of an algorithm is $\mathcal{O}(nT_A)$, we imply that the algorithm makes n uses of the unitary U_A. Thus, if the circuit depth of U_A is T_A, the circuit depth of our algorithm is $\mathcal{O}(nT_A)$.

Chapter 2

Background

2.1 Linear Regression and Regularization

In this section we briefly describe the various kinds of linear models that we will deal with. We begin by describing ordinary least squares (OLS), the simplest model. Expanding upon it, we introduce weighted and correlated least squares, followed by an introduction to regularization of these models.

2.1.1 Ordinary Least Squares

Suppose we are given data points $\{(a_i, b_i)\}_{i=1}^N$, where $\forall i : a_i \in \mathbb{R}^d, \forall i : b_i \in \mathbb{R}$ such that $(a_i, b_i) \sim_{i.i.d} \mathcal{D}$, i.e. they are sampled i.i.d. from some unknown distribution $\mathcal{D}$, assumed to be linear. We want to find a vector $x \in \mathbb{R}^d$ such that the inner product $x^T a_j$ is a good predictor for the target b_j for some unknown a_j. This can be done by minimizing the total squared loss over the given data points,

$$\mathcal{L}_O := \sum_j (x^T a_j - b_j)^2, \tag{2.1.1}$$

leading to the ordinary least squares (OLS) optimization problem. The task then is to find $x \in \mathbb{R}^d$ that minimizes $\|Ax - b\|_2^2$, where A is the $N \times d$ data matrix such that the i^{th} row of A is a_i, and the i^{th} element of the vector b is b_i. Assuming that $A^T A$ is non-singular, one can show that Equation 2.1.1 is minimized at (proof in section A.2)

$$x = (A^T A)^{-1} A^T b = A^+ b, \tag{2.1.2}$$

which corresponds to solving a linear system of equations.

2.1.2 Weighted Least Squares

Ordinary least squares method assumed that there is a constant variance in the errors of our sample (an assumption called homoscedasticity). Suppose that the assumption is violated (heteroscedasticity). That is, suppose that out of the samples present in the data, we have higher confidence in some of them than others. In such a scenario, the i^{th} observation can be assigned a weight $w_i \in \mathbb{R}$.

This leads to a generalization of the OLS problem to *weighted least squares* (WLS). In order to obtain the best linear fit, the task is now to minimize the weighted version of the loss

$$\mathcal{L}_W := \sum_j w_j (x^T a_j - b_j)^2. \tag{2.1.3}$$

As before, assuming $A^T W A$ is non-singular, the above loss function has the following closed-form solution

$$x = (A^T W A)^{-1} A^T W b, \tag{2.1.4}$$

where W is a diagonal matrix with w_i being the i^{th} diagonal element.

For example, if the model assumes that the underlying distribution is of the form $b = Ax + e$, where e is a multivariate, normally distributed vector with mean $\mathbf{0}$ and the variance-covariance matrix

$$\begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_N^2 \end{pmatrix},$$

each observation can be weighted proportional to the inverse of the variance, $w_i \propto \frac{1}{\sigma_i^2}$.

2.1.3 Generalized Least Squares

There can arise scenarios where there exists some correlation between any two samples. For *generalized least squares* (GLS), the presumed correlations between pairs of samples are given in a symmetric, non-singular covariance matrix Ω. This method then finds the best fit by minimizing the square of Mahalanobis length[i] of the residual vector. The objective is thus to find the vector x that minimizes

$$\mathcal{L}_\Omega := \sum_{i,j} (\Omega^{-1})_{i,j} (x^T a_i - b_i)(x^T a_j - b_j). \tag{2.1.5}$$

The closed form solution for GLS is given by

$$x = (A^T \Omega^{-1} A)^{-1} A^T \Omega^{-1} b. \tag{2.1.6}$$

GLS corresponds to using OLS on a linearly transformed dataset. This can be easily observed by setting $\Omega = K^T K = KK$. Such square-root decomposition is possible since Ω is assumed to be symmetric and non-singular. Then,

$$\begin{aligned} \mathcal{L}_\Omega &= (b - Ax)^T \Omega^{-1} (b - Ax) \\ &= (b - Ax)^T K^{-1} K^{-1} (b - Ax) \\ &= \left(K^{-1}(b - Ax)\right)^T \left(K^{-1}(b - Ax)\right) \\ &= (b' - A'x)^T (b' - A'x) \\ &= \mathcal{L}'_O \end{aligned}$$

where $b' := K^{-1}b$, $A' := K^{-1}A$ are the transformed samples.

Note. In practice we seldom know what the correlations are. In that situation we use the residuals of ordinary least squares to estimate the covariance matrix, a method known as *feasible generalized least squares*. How this is approached exactly varies across fields, and typically involves some assumptions on the underlying dataset.

2.1.4 Regularization of Linear Systems

As mentioned previously, in several practical scenarios, the linear regression problem may be *ill-posed* or suffer from *overfitting*. Furthermore, the data may be such that some of the columns of the matrix A are linearly dependent. This shrinks the rank of A, and consequently of the matrix $A^T A$, rendering it singular and therefore non-invertible. Recall that the closed-form solution of OLS exists only if $A^T A$ is non-singular, which is no longer the case. Such scenarios arise even for WLS and GLS problems [10].

In such cases, one resorts to *regularization* to deal with them. Let $\mathcal{L}$ be the loss function to be minimized for the underlying least squares problem (such as OLS, WLS, or GLS). Then general ℓ_2-regularization (*Tikhonov regularization*) involves an additional penalty term so that the objective now is to find the vector $x \in \mathbb{R}^d$ that minimizes

$$\mathcal{L} + \lambda \|Lx\|_2^2. \tag{2.1.7}$$

Here λ, known as the regularization parameter, is a positive constant that controls the size of the vector x, while L is known as the penalty matrix (or regularization matrix) that defines a (semi)norm on the solution

[i] See https://en.wikipedia.org/wiki/Mahalanobis_distance for the definition.

through which the size is measured. The solution to the Tikhonov regularization problem also has a closed-form solution. For the OLS problem, when $\mathcal{L} = \mathcal{L}_O$, we have that

$$x = (A^T A + \lambda L^T L)^{-1} A^T b. \tag{2.1.8}$$

This can easily be seen as follows.

$$\begin{aligned} f(x) &:= \|Ax - b\|^2 + \lambda \|Lx\|^2 \\ &= (Ax - b)^T (Ax - b) + \lambda (Lx)^T (Lx) \\ &= x^T A^T A x - x^T A^T b - b^T A x + b^T b + \lambda x^T L^T L x \\ \implies \nabla f(x) &= 2A^T A x - 2A^T b + 2\lambda L^T L x \end{aligned}$$

Setting $\nabla f(x) = 0$ gives us

$$\begin{aligned} & 2A^T A x - 2A^T b + 2\lambda L^T L x = 0 \\ \implies & (A^T A + \lambda L^T L) x = A^T b \\ \implies & x = (A^T A + \lambda L^T L)^{-1} A^T b \end{aligned}$$

It is worth noting that when $L = I$, the ℓ_2-regularized OLS problem is known as *ridge regression*. For the unregularized OLS problem, the singular values of A, σ_j are mapped to $1/\sigma_j$. The penalty term due to ℓ_2-regularization, results in a shrinkage of the singular values. This implies that even in the scenario where A has linearly dependent columns (some $\sigma_j = 0$) and $(A^T A)^{-1}$ does not exist, the inverse $(A^T A + \lambda L^T L)^{-1}$ is well defined for $\lambda > 0$ and any positive-definite L. Throughout this article, we refer to such an L (which is positive definite) as a *good regularizer*. The penalty matrix L allows for penalizing each regression parameter differently and leads to joint shrinkage among the elements of x. It also determines the rate and direction of shrinkage. In the special case of ridge regression, as $L = I$, the penalty shrinks each element of x equally along the unit vectors e_j. Also note that by definition, I is a *good regularizer*.

When $\mathcal{L} = \mathcal{L}_w$, the optimal solution is given by

$$x = (A^T W A + \lambda L^T L)^{-1} A^T W b \tag{2.1.9}$$

This can be derived similar to the ordinary case. Similarly, when $\mathcal{L} = \mathcal{L}_\Omega$, we get

$$x = (A^T \Omega^{-1} A + \lambda L^T L) A^T \Omega^{-1} b. \tag{2.1.10}$$

We show that finding the optimal solution x in all the three cases above reduces to solving a linear system. The quantum version of these algorithms output a quantum state that is ϵ-close $|x\rangle = \sum_j x_j |j\rangle / \|x\|$.

Throughout this work, while designing our quantum algorithms, we shall assume access (via a block-encoding) to the matrices A, W, Ω, and L and knowledge of the parameter λ. Classically, however, the regularization matrix L and the optimal parameter λ are obtained via several heuristic techniques [7, 9, 10].

2.2 A Gentle Introduction to Quantum Mechanics

In this section we introduce the axioms of quantum mechanics. This is not meant to be a complete or concrete introduction to the subject, for which we refer the reader to de Wolf [47][ii]. This is meant to refresh the memory of the reader, and gently lead them from the world of classical computers to that of quantum computing. We begin by introducing how systems and its states are described in quantum mechanics, dictated by the first axiom.

Axiom 1. (Quantum Systems and States). To every quantum system there is associated a separable complex Hilbert space $(\mathcal{H}, +, \cdot, \langle \cdot | \cdot \rangle)$. The states of the system are all positive, trace-class linear maps $\rho : \mathcal{H} \mapsto \mathcal{H}$ for which $\mathrm{Tr}\rho = 1$.

The states of a system can be pure or mixed. A state $\rho : \mathcal{H} \mapsto \mathcal{H}$ is called a pure state if

$$\exists \psi \in \mathcal{H} \ : \ \forall \alpha \in \mathcal{H} \ : \ \rho(\alpha) = \frac{\langle \psi | \alpha \rangle}{\langle \psi | \psi \rangle} \psi. \tag{2.2.1}$$

[ii] Prof. Frederic Schuller's Lectures on Quantum Theory on YouTube is another great source [48].

Note that the above condition implies that to each pure state, we can associate a (non-unique) element $\phi \in \mathcal{H}$. If a state is not a pure state, it is called a mixed state.

The fundamental unit of classical information is the *bit*, which can take the values 0 or 1. Similarly, the fundamental unit of quantum information is the *qubit*, which is defined as a normalized vector $|\psi\rangle$ in a two-dimensional Hilbert space over $\mathbb{C}$. Note that $\mathcal{B} := \{|0\rangle, |1\rangle\}$ forms an orthonormal basis for this space (commonly known as the computational basis).

A single qubit can be visualized as a point on the Bloch Sphere. To see this, note that any qubit can be written as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,\ |\alpha|^2 + |\beta|^2 = 1,\ \alpha, \beta \in \mathbb{C}. \tag{2.2.2}$$

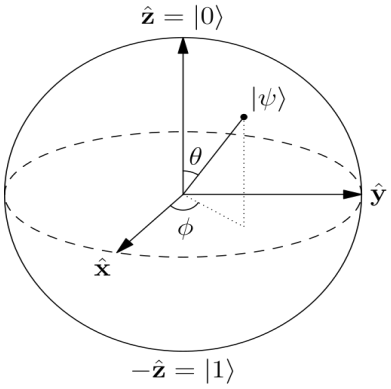


Figure 2.1: A qubit $|\psi\rangle$ on a Bloch Sphere.

Note that a qubit can also be described using the elevation and azimuthal angles θ, ϕ as

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + \exp(i\phi)\sin\frac{\theta}{2}\,|1\rangle\,. \tag{2.2.3}$$

Next, we define what the observables of the quantum theory are.

Axiom 2. (Observables). The observables of a quantum system are the self-adjoint linear maps $A : \mathcal{D}_A \mapsto \mathcal{H}$, where the subspace $\mathcal{D}_A \subseteq \mathcal{H}$ is *dense* in $\mathcal{H}$, i.e.,

$$\forall\psi \in \mathcal{H} : \forall\varepsilon > 0 : \exists\alpha \in \mathcal{D}_A : \|\alpha - \psi\| < \varepsilon. \tag{2.2.4}$$

Axiom 3. (Unitary Dynamics). In the time interval $(t_1, t_2) \subseteq \mathbb{R}$ in which no measurement occurs, the state at time t_1, denoted by $\rho(t_1)$, is related to the state at time t_2, denoted by $\rho(t_2)$ by

$$\rho(t_2) = U(t_2 - t_1)\rho(t_1)U^{-1}(t_2 - t_1) \tag{2.2.5}$$

with the unitary evolution operator defined as

$$U(t) := \exp\left(-\frac{i}{\hbar}Ht\right), \tag{2.2.6}$$

where H is the energy observable.

Measurement. Measurement operator M_j is a projection operator which projects the quantum state to an eigenvalue of M_j. The post-measurement state is given by

$$\rho_j = \frac{M_j\rho M_j^\dagger}{\mathrm{Tr}(M_j\rho M_j^\dagger)}, \tag{2.2.7}$$

where the j measurement outcome occurs with the probability $p(j) = \mathrm{Tr}(M_j\rho M_j^\dagger)$, and $\sum_j M_j^\dagger M_j = \mathcal{I}$[iii].

[iii] This definition of measurement would be sufficient for this work, but for more general and mathematically well defined notion of measurements, refer to [47].

2.3 Models of Quantum Computing

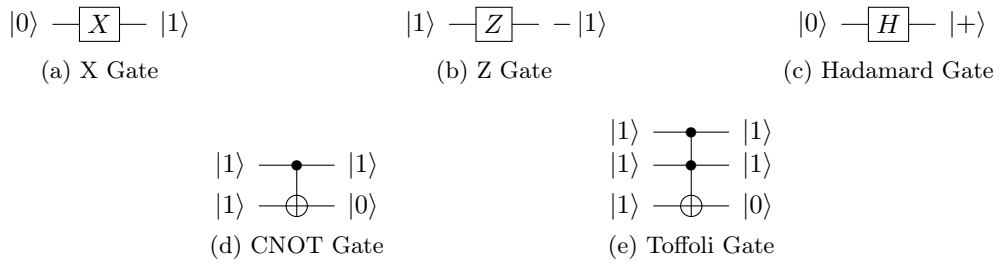
A programmable computer is a physical system based on some laws of physics, the initial state of which is programmable by the user to represent the problem. The evolution of the system encodes the solution procedure (an algorithm). Reading out the final state of the system (measurement) gives us the solution to the problem. The most familiar model of quantum computation in terms of a universal set of gates was proposed by Deutsch and Penrose [49] following the ideas of Paul Benioff and Richard Feynman for a quantum Turing machine (Benioff [50], Benioff [51], Feynman [14]), for simulating quantum mechanics of physical systems using a *programmable quantum machine*, to circumvent the problem with the exponential size of the Hilbert space. Quantum computing is based on the idea of a computational problem being encoded as a quantum system, evolving with a sequence of quantum gates (Unitaries, in accordance with the axiom of quantum mechanics), similar to that encoded as a classical system evolving with a sequence of classical logic gates.

Adiabatic quantum computing is another such framework where the initial state of the system is a Hamiltonian whose ground state is easy to prepare, which evolves according to the adiabatic theorem to a final Hamiltonian whose ground state encodes the solution to a computational problem. The adiabatic theorem guarantees that the system will remain in the ground state of the instantaneous Hamiltonian, given that the evolution takes place sufficiently slowly. The idea to encode the solution to a computational problem into the ground state of a quantum Hamiltonian appeared in 1998 by Brooke et al. [52], in trying to solve classical combinatorial optimization problem. This was called Quantum Annealing. This was introduced as a classical 'quantum inspired' algorithm, akin to Simulated Annealing (SA), and made use of simulated quantum fluctuations and tunneling (similar to thermal fluctuations simulated by SA.) This section introduces these ideas formally.

2.3.1 Circuit Model of Quantum Computation

Classical circuits. An n-bit classical *Boolean circuit* that computes a Boolean function $f : \{0,1\}^n \mapsto \{0,1\}^m$ is defined as a finite, directed acyclic graph with AND, OR and NOT gates as internal vertices, n input vertices (for the n input bits), and m output vertices. A *circuit family* is a set $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$, which contains a classical Boolean circuit for each input size in $\mathbb{N}$. If $m = 1$ and the circuit can output only 0 or 1, we say that the circuit *decides* a language $L = \{x \in \{0,1\}^* | f(x) = 1\} \subset \{0,1\}^*$.

We can generalize this notion by replacing the classical *gate set* (AND, OR and NOT) by elementary quantum gates, and allowing the inputs and outputs to be arbitrary quantum states. In line with the axioms of quantum mechanics [15], these quantum gates are unitary operators. An elementary quantum gate is a unitary transformation on 1, 2 or 3 qubits. Examples include the bit flip (X) gate, the phase flip (Z) gate, Hadamard (H) gate, and the controlled-controlled-NOT (CCNOT) gate, also called the Toffoli gate. Circuit representation of some quantum gates is depicted in Figure 2.2.



(a) X Gate (b) Z Gate (c) Hadamard Gate

(d) CNOT Gate (e) Toffoli Gate

Figure 2.2: Quantum Gates

The Toffoli gate is universal for classical reversible computation.

Universal set of quantum gates. Let $\mathcal{G}$ be a finite set of quantum gates. $\mathcal{G}$ is universal if we can use gates from $\mathcal{G}$ to approximate any unitary U on any number of qubits to any desired precision ϵ. For example, the gate set $\{H, \text{CNOT}, R_{\pi/4}\}$ is universal.

A natural question that arises is that how many gates from the universal gate set would be required to implement some unitary? The Solovay-Kitaev Theorem [15] answers this. It tells us that the number of elementary gates needed to approximate an arbitrary unitary scales poly-logarithmically with the inverse of the error.

Theorem 2.3.1 (Solovay-Kitaev Theorem)**.** *Let $\mathcal{G}$ be a finite set of elements in $SU(d)$ containing its own inverses (so $g \in \mathcal{G} \implies g^{-1} \in \mathcal{G}$). Consider some $\varepsilon > 0$. Then there is a constant c such that for any*

$U \in SU(d)$, there is a sequence S of gates from $\mathcal{G}$ of length $O(\log^c(1/\varepsilon))$ such that $\|S - U\| \leq \varepsilon$. That is, S approximates U to operator norm error.

In this model of computation the time complexity of an algorithm is proportional to the number of elementary gates that cannot be parallelised, i.e., the depth of the circuit is the critical complexity. Another important measure of complexity, with respect to an oracle, is the number of uses of an oracle that the circuit makes. This is the query complexity of algorithm with respect to the particular oracle.

2.3.2 Adiabatic Quantum Computing

A computation in this model is specified by two Hamiltonians H_i and H_f, such that the ground state of H_i is easy to prepare, and the ground state of H_f represents the solution to our computational problem. The Hamiltonians have to be *local*, i.e., they should only involve interactions between a constant number of particles. This requirement is equivalent, in the circuit model, to only allowing gates acting on a constant number of qubits at a time. The running time of the adiabatic computation is determined by the minimal spectral gap of all the Hamiltonians connecting H_i and H_f, given by

$$H(s) = (1 - s)H_i + sH_f \tag{2.3.1}$$

where $s(t) : [0, t_f] \to [0, 1]$ is called the schedule, and t_f is the annealing time.
The following definition of adiabatic quantum computing has been taken from Aharonov et al. [53].

Definition 2.3.2 (k-local Hamiltonian)**.** *A Hamiltonian H acting on n particles is called k-local H can be written as $\sum_A H_A$, where A runs over all the subsets of k out of n particles, and H_A acts trivially on all but the particles in A.*

That is, for each A, H_A is a tensor product of a Hamiltonian on A, with identity on all the other particles. Notice that for any constant k, a k-local Hamiltonian on n qubits can be described in $2^{2k}n^k = poly(n)$ space, whereas describing an arbitrary Hamiltonian requires roughly 2^{2n} space.

Definition 2.3.3 (Adiabatic Quantum Computation)**.** *A k-local adiabatic computation $AC(n, d, H_i, H_f, \epsilon)$ is specified by two k-local Hamiltonians, H_i and H_f acting on n d-dimensional particles, such that both Hamiltonians have unique ground states. The ground state of H_i is a tensor product state. The output is a state that is ϵ-close in l_2-norm to the ground state of H_f. Let t_f be the smallest time such that the final state of an adiabatic evolution according to Equation 2.3.1 for time t_f is ϵ-close in l_2-norm to the ground state of H_f. The running time of the adiabatic algorithm is defined to be*

$$cost := t_f \cdot \max_s H(s) \tag{2.3.2}$$

Notice that the definition of running time as defined in Equation 2.3.2 is invariant under the scaling of the Hamiltonians by some overall factors.

Theorem 2.3.4 (Adiabatic Theorem)**.** *Let H_i and H_f be two Hamiltonians acting on a n-qubit quantum systems, and consider the time dependent Hamiltonian as described in Equation 2.3.1. Assume that for all s, $H(s)$ has a unique ground state. Then for any fixed $\delta > 0$, if*

$$t_f \geq \Omega\left(\frac{||H_f - H_i||^{1+\delta}}{\epsilon^\delta \min_{s\in[0,1]}\{\Delta^{2+\delta}(H(s))\}}\right) \tag{2.3.3}$$

then the final state of an adiabatic evolution according to $H(s)$ for time t_f is ϵ-close in the l_2-norm to the ground state of H_f.

The matrix norm is the spectral norm. $\Delta(H(s))$ is the spectral gap, defined as the minimum eigenvalue gap between the ground state and the first excited state of the instantaneous Hamiltonian.
Aharonov et al. [53] showed that Adiabatic quantum computation with non-stoquastic Hamiltonians is polynomially equivalent to the circuit model ('Standard Quantum Computation'.)

Theorem 2.3.5 (Equivalence of Adiabatic and Circuit Model Quantum Computation)**.** *Given a quantum circuit on n qubits with L two qubit gates implementing a unitary U, and an $\epsilon > 0$, there exists a 5-local adiabatic quantum computation $AC(n + L, 2, H_i, H_f, \epsilon)$, whose running time is $poly(L, \frac{1}{\epsilon})$ and whose output after tracing out some ancilla qubits is ϵ-close in the trace distance to $U|0^{\otimes n}\rangle$. Moreover, H_i and H_f can be computed by a PTTM.*

The runtime for this algorithm is $\mathcal{O}(\epsilon^{-(5+3\delta)}L^{5+2\delta})$ for any fixed $\delta > 0$.

Note that an adiabatic quantum algorithm can always be converted to a quantum circuit that can approximate the adiabatic computation by discretizing the time dependent Hamiltonian for some finite sequence of time steps and then using a standard Hamiltonian simulation algorithm to simulate the evolution.

Quantum annealing was proposed as one of the first adiabatic quantum algorithm, to solve combinatorial optimization problems. These problems can be notoriously hard, due to the vastness of the search space. The original proposal was a 'simulated quantum annealing,' in which the evolution of a quantum system was simulated on a classical computer, such that the final state contained the solution to a computational problem. Kadowaki and Nishimori [54] and Farhi et al. [55] showed that there is an advantage in using algorithms like these. It was thought that is large, controlled, programmable quantum systems (quantum computers) could be built, then such advantages would continue, ushering the era of quantum supremacy [56]. Companies such as DWave have built physical devices, and considerable effort has gone into testing them [57, 58, 59]. Although small speedups for curated problems have been seen [57, 60], a conclusive picture does not exist.

2.4 Quantum Input Models

The complexities of quantum algorithms often depend on how the input data is accessed. For instance, in quantum algorithms for linear algebra (involving matrix operations), it is often assumed that there exists a black-box that returns the positions of the non-zero entries of the underlying matrix when queried. The algorithmic running time is expressed in terms of the number of queries made to this black-box. Such an input model, known as the *Sparse Access Model*, helps design efficient quantum algorithms whenever the underlying matrices are sparse. Various other input models exist, and quantum algorithms are typically designed and optimized for specific input models.

Kerenidis and Prakash [34] introduced a different input model, known as the *quantum data structure model*, which is more conducive for designing quantum machine learning algorithms. In this model, the input data (e.g: entries of matrices) arrive online and are stored in a classical data structure (often referred to as the KP-tree in the literature), which can be queried in superposition by using a QRAM. This facilitates efficiently preparing quantum states corresponding to the rows of the underlying matrix, that can then be used for performing several matrix operations.

Subsequently, several quantum-inspired classical algorithms have also been developed following the breakthrough result of Tang [61]. Such classical algorithms have the same underlying assumptions as the quantum algorithms designed in the data structure input model and are only polynomially slower provided the underlying matrix is low rank.

In this work, we will consider the framework of *block-encoding*, wherein it is assumed that the input matrix A (up to some sub-normalization) is stored in the left block of some unitary. The advantage of the block-encoding framework, which was introduced in a series of works [36, 43, 35, 42], is that it can be applied to a wide variety of input models. For instance, it can be shown that both the sparse access input model as well as the quantum data structure input model are specific instances of block-encoded matrices [35, 42]. Here we formally define the framework of block-encoding and also express the sparse access model as well as the quantum data structure model as block-encodings. We refer the reader to [35, 42] for proofs.

2.4.1 Unitary Block Encoding of Matrices

Definition 2.4.1 (Block Encoding, restated from [42])**.** *Suppose that A is an s-qubit operator, $\alpha, \varepsilon \in \mathbb{R}^+$ and $a \in \mathbb{N}$, then we say that the $(s+a)$-qubit unitary U_A is an (α, a, ε)-block-encoding of A, if*

$$\left\| A - \alpha(\langle 0|^{\otimes a} \otimes I) U_A (|0\rangle^{\otimes a} \otimes I) \right\| \leq \varepsilon. \tag{2.4.1}$$

Let $|\psi\rangle$ be an s-qubit quantum state. Then applying U_A to $|\psi\rangle |0\rangle^{\otimes a}$ outputs a quantum state that is $\frac{\varepsilon}{\alpha}$-close to

$$\frac{A}{\alpha} |\psi\rangle |0\rangle^{\otimes a} + |\Phi^{\perp}\rangle,$$

where $\left(I_s \otimes |0\rangle^{\otimes a} \langle 0|^{\otimes a}\right) |\Phi^{\perp}\rangle = 0$. Equivalently, suppose $\tilde{A} := \alpha \left(\langle 0|^{\otimes a} \otimes I_s\right) U_A \left(|0\rangle^{\otimes a} \otimes I_s\right)$ denotes the actual matrix that is block-encoded into U_A, then $\left\| A - \tilde{A} \right\| \leq \varepsilon$.

In the subsequent sections, we provide an outline of the quantum data structure model and the sparse access model which are particular instances of the block encoding framework.

Note that a unitary matrix is a (1, 0, 0)-block-encoding of itself.

Definition 2.4.2 (Optimal block-encoding of a matrix)**.** *Given a matrix $A \in \mathbb{C}^{2^s \times 2^s}$ and a real number $\alpha \geq \|A\|$, we can construct a $(\alpha, 1, 0)$-block-encoding of A:*

$$U_A = \begin{pmatrix} \frac{A}{\alpha} & \sqrt{I - \frac{AA^\dagger}{\alpha^2}} \\ \sqrt{I - \frac{A^\dagger A}{\alpha^2}} & \frac{A^\dagger}{\alpha} \end{pmatrix} \tag{2.4.2}$$

2.4.2 QROM Input Model

Kerenidis and Prakash introduced a quantum accessible classical data structure which has proven to be quite useful for designing several quantum algorithms for linear algebra [34]. The classical data structure stores entries of matrices or vectors and can be queried in superposition using a QRAM (quantum random access memory). We directly state the following theorem from therein.

Theorem 2.4.3 (Implementing quantum operators using an efficient data structure, [33, 34])**.** *Let $A \in \mathbb{R}^{N \times d}$, and w be the number of non-zero entries of A. Then there exists a data structure of size $\mathcal{O}\big(w \log^2(dN)\big)$ that given the matrix elements (i, j, a_{ij}), stores them at a cost of $\mathcal{O}(\log(dN))$ operations per element. Once all the non-zero entries of A have been stored in the data structure, there exist quantum algorithms that are ε-approximations to the following maps:*

$$U : |i\rangle |0\rangle \mapsto \frac{1}{\|A_{i,\cdot}\|} \sum_{j=1}^{d} a_{i,j} |i, j\rangle = |\psi_i\rangle ,$$

$$V : |0\rangle |j\rangle \mapsto \frac{1}{\|A\|_F} \sum_{i=1}^{N} \|A_{i,\cdot}\| |i, j\rangle = |\phi_j\rangle$$

where $\|A_{i,\cdot}\|$ is the norm of the i^{th} row of A and the second register of $|\psi_i\rangle$ is the quantum state corresponding to the i^{th} row of A. These operations can be applied at a cost of $\mathcal{O}(\text{polylog}(Nd/\varepsilon))$.

It was identified in Ref. [35] that if a matrix A is stored in this quantum accessible data structure, there exists an efficiently implementable block-encoding of A. We restate their result here.

Lemma 2.4.4 (Implementing block encodings from quantum data structures, [35])**.** *Let the entries of the matrix $A \in \mathbb{R}^{N \times d}$ be stored in a quantum accessible data structure, then there exist unitaries U_R, U_L that can be implemented at a cost of $\mathcal{O}(\text{polylog}(dN/\varepsilon))$ such that $U_R^\dagger U_L$ is a $(\|A\|_F, \lceil \log(d+N) \rceil, \varepsilon)$-block-encoding of A.*

Proof. The unitaries U_R and U_L can be implemented via U and V in the previous lemma. Let $U_R = U$ and $U_L = V.\texttt{SWAP}$. Then for $s = \lceil \log(d+N) \rceil$ we have

$$U_R : \ |i\rangle |0^s\rangle \to |\psi_i\rangle ,$$

and

$$U_L : \ |j\rangle |0^s\rangle \to |\phi_j\rangle ,$$

So we have that the top left block of $U_R^\dagger U_L$ if given by

$$\sum_{i=1}^{N} \sum_{j=1}^{d} \langle \psi_i | \phi_j \rangle |i, 0\rangle \langle j, 0|$$

Now

$$\begin{aligned} \langle \psi_i | \phi_j \rangle &= \sum_{k=1}^{d} \sum_{\ell=1}^{N} \frac{a_{ik}}{\|A_{i,\cdot}\|} \cdot \frac{\|A_\ell\|}{\|A\|_F} \underbrace{\langle i, k | l, j \rangle}_{:= \delta_{i,l} . \delta_{k,j}} \\ &= \frac{a_{ij}}{\|A\|_F}. \end{aligned}$$

Moreover since only ε-approximations of U and V can be implemented we have that $U_R^\dagger U_L$ is a $(\|A\|_F, \lceil \log(n+d) \rceil, \varepsilon)$ block encoding of A implementable with the same cost as U and V. □

Kerenidis and Prakash [32] argued that in certain scenarios, storing the entries of $A^{(p)}, (A^{1-p})^\dagger$ might be useful as compared to storing A, for some $p \in [0,1]$. In such cases, the quantum data structure is a $(\mu_p, \lceil \log(N+d) \rceil, \varepsilon)$ block encoding of A, where $\mu_p(A) = \sqrt{s_{2p}(A).s_{2(1-p)}(A^T)}$ such that $s_p(A) := \max_j \|A_{j,\cdot}\|_q^q$. Throughout the work, whenever our results are expressed in the quantum data structure input model, we shall state our complexity in terms of μ_A. When the entries of A are directly stored in the data structure, $\mu_A = \|A\|_F$. Although, we will not state it explicitly each time, our results also hold when fractional powers of A are stored in the database and simply substituting $\mu_A = \mu_p(A)$, yields the required complexity.

2.4.3 Sparse Access Input Model

The sparse access input model considers that the input matrix $A \in \mathbb{R}^{N \times d}$ has row sparsity s_r and column sparsity s_c. Furthermore, it assumes that the entries of A can be queried via an oracle as

$$O_A : |i\rangle |j\rangle |0\rangle^{\otimes b} \mapsto |i\rangle |j\rangle |a_{ij}\rangle \quad \forall i \in [N], j \in [d],$$

and the indices of the non-zero elements of each row and column can be queried via the following oracles:

$$O_r : |i\rangle |j\rangle \mapsto |i\rangle |r_{ij}\rangle \quad \forall i \in [N], k \in [s_r],$$
$$O_c : |i\rangle |j\rangle \mapsto |c_{ij}\rangle |j\rangle \quad \forall i \in [d], k \in [s_c]$$

where r_{ij} is the j^{th} non-zero entry of the i^{th} row of A and c_{ij} is the i^{th} non-zero entry of the j^{th} column of A. Gilyén et al. [42] showed that a block encoding of a sparse A can be efficiently prepared by using these three oracles. We restate their lemma below.

Lemma 2.4.5 (Constructing a block-encoding from sparse-access to matrices, [42])**.** *Let $A \in \mathbb{R}^{N \times d}$ be an s_r, s_c row, column sparse matrix given as a sparse access input. Then for all $\varepsilon \in (0,1)$, we can implement a $(\sqrt{s_c s_r}, \text{polylog}(Nd/\varepsilon), \varepsilon)$-block-encoding of A with $\mathcal{O}(1)$ queries to O_r, O_c, O_A and $\text{polylog}(Nd/\varepsilon)$ elementary quantum gates.*

Throughout the paper, we shall assume input matrices are accessible via approximate block-encodings. This also allows us to write down the complexities of our quantum algorithms in this general framework. Additionally, we state the complexities in both the sparse access input model as well as the quantum accessible data structure input model as particular cases.

2.5 Quantum Singular Value Transformation

In a seminal work, Gilyén et al. presented a framework to apply an arbitrary polynomial function to the singular values of a matrix, known as Quantum Singular Value Transformation (QSVT) [42]. QSVT is quite general: many quantum algorithms can be recast to this framework, and for several problems, better quantum algorithms can be obtained [42, 44]. In particular, QSVT has been extremely useful in obtaining optimal quantum algorithms for linear algebra. For instance, using QSVT, given the block-encoding of a matrix A, one could obtain A^{-c} with $c \in [0, \infty)$ with optimal complexity and by using fewer additional qubits than prior art. This section briefly describes this framework, which is a generalization of Quantum Signal Processing (QSP) [36, Section 2], [43, Theorem 2], [62]. The reader may refer to [44] for a more pedagogical overview of these techniques.

Let us begin by discussing the framework of Quantum Signal Processing. QSP is a quantum algorithm to apply a d-degree bounded polynomial transformation with parity $d \mod 2$ to an arbitrary quantum subsystem, using a quantum circuit U_Φ consisting of only controlled single qubit rotations. This is achieved by interleaving a *signal rotation operator* W (which is an x-rotation by some fixed angle θ) and a *signal processing operator* S_ϕ (which is a z-rotation by a variable angle $\phi \in [0, 2\pi]$). In this formulation, the signal rotation operator is defined as

$$W(x) := \begin{pmatrix} x & i\sqrt{1-x^2} \\ i\sqrt{1-x^2} & x \end{pmatrix}, \tag{2.5.1}$$

which is an x-rotation by angle $\theta = -2\arccos(x)$, and the signal processing operator is defined as

$$S_\phi := e^{i\phi Z}, \tag{2.5.2}$$

which is a z-rotation by an angle -2ϕ. Interestingly, sandwiching them together for some $\Phi := (\phi_0, \phi_1, \dots \phi_d) \in \mathbb{R}^{d+1}$, as shown in Equation 2.5.3, gives us a matrix whose elements are polynomial transformations of x,

$$U_\Phi := e^{i\phi_0 Z} \prod_{j=1}^{j=d} \left(W(x) e^{i\phi_j Z} \right) \tag{2.5.3}$$

$$= \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix}, \tag{2.5.4}$$

such that

1. $\deg P \leq d$; $\deg Q \leq d-1$,
2. $P(x)$ has a parity $d \mod 2$,
3. $|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1 \quad \forall x \in [-1, 1]$.

Following the application of the quantum circuit U_Φ for an appropriate Φ, one can project into the top left block of U_Φ to recover the polynomial $\langle 0| U_\Phi |0\rangle = P(x)$. Projecting to other basis allows the ability to perform more interesting polynomial transformations, which can be linear combinations of $P(x), Q(x)$, and their complex conjugates. For example, projecting to $\{|+\rangle, |-\rangle\}$ basis gives us

$$\langle +| U_\Phi |+\rangle = \Re(P(x)) + i\Re(Q(x))\sqrt{1-x^2}. \tag{2.5.5}$$

Quantum Signal Processing can be formally stated as follows.

Theorem 2.5.1 (Quantum Signal Processing, Corollary 8 from [42]). *Let $P \in \mathbb{C}[x]$ be a polynomial of degree $d \geq 2$, such that*

- *P has parity-$(d \mod 2)$,*
- *$\forall x \in [-1, 1] : |P(x)| \leq 1$,*
- *$\forall x \in (-\infty, -1] \cup [1, \infty) : |P(x)| \geq 1$,*
- *if d is even, then $\forall x \in \mathbb{R} : P(ix)P^*(ix) \geq 1$.*

Then there exists a $\Phi \in \mathbb{R}^d$ such that

$$\prod_{j=1}^{d} \left(e^{i\phi_j \sigma_z} W(x) \right) = \begin{pmatrix} P(x) & \cdot \\ \cdot & \cdot \end{pmatrix}. \tag{2.5.6}$$

Thus, QSP allows us to implement any polynomial $P(x)$ that satisfies the aforementioned requirements. Throughout this article, we refer to any such polynomial $P(x)$ as a *QSP polynomial*. Quantum Singular Value Transformation is a natural generalization of this procedure. It allows us to apply a QSP polynomial transformation to each singular value of an arbitrary block of a unitary matrix. In addition to this generalization, QSVT relies on the observation that several functions can be well-approximated by QSP polynomials. Thus, through QSVT one can transform each singular value of a block-encoded matrix by any function that can be approximated by a QSP polynomial. Since several linear algebra problems boil down to applying specific transformations to the singular values of a matrix, QSVT is particularly useful for developing fast algorithms for quantum linear algebra. Next, we introduce QSVT formally via the following theorem.

Theorem 2.5.2 (Quantum Singular Value Transformation [63], Section 3.2). *Suppose $A \in \mathbb{R}^{N \times d}$ is a matrix with singular value decomposition $A = \sum_{j=1}^{d_{\min}} \sigma_j |v_j\rangle \langle w_j|$, where $d_{\min} = \min\{N, d\}$ and $|v_j\rangle$ ($|w_j\rangle$) is the left (right) singular vector with singular value σ_j. Furthermore, let U_A be a unitary such that $A = \widetilde{\Pi} U_A \Pi$, where Π and $\widetilde{\Pi}$ are orthogonal projectors. Then, for any QSP polynomial $P(x)$ of degree n, there exists a vector $\Phi = (\phi_1, \phi_2, \cdots \phi_n) \in \mathbb{R}^n$ and a unitary*

$$U_\Phi = \begin{cases} e^{i\phi_1(2\widetilde{\Pi}-I)} U_A \left[\prod_{k=1}^{(n-1)/2} e^{i\phi_{2k}(2\widetilde{\Pi}-I)} U_A^\dagger e^{i\phi_{2k+1}(2\widetilde{\Pi}-I)} U_A \right], & n \text{ is odd} \\ \left[\prod_{k=1}^{n/2} e^{i\phi_{2k-1}(2\widetilde{\Pi}-I)} U_A^\dagger e^{i\phi_{2k}(2\widetilde{\Pi}-I)} U_A \right], & n \text{ is even,} \end{cases} \tag{2.5.7}$$

such that

$$P^{SV}(A) = \begin{cases} \widetilde{\Pi} U_\Phi \Pi, & n \text{ is odd} \\ \Pi U_\Phi \Pi, & n \text{ is even,} \end{cases} \tag{2.5.8}$$

where $P^{SV}(A)$ is the polynomial transformation of the matrix A defined as

$$P^{SV}(A) := \begin{cases} \sum_j P(\sigma_j) |v_j\rangle \langle w_j|, & P \text{ is odd} \\ \sum_j P(\sigma_j) |w_j\rangle \langle w_j|, & P \text{ is even.} \end{cases} \tag{2.5.9}$$

Theorem 2.5.2 tells us that for any QSP polynomial P of degree n, we can implement $P^{SV}(A)$ using one ancilla qubit, $\Theta(n)$ applications of U_A, $U_A^\dagger$ and controlled reflections $I - 2\Pi$ and $I - 2\widetilde{\Pi}$. Furthermore, if in some well-defined interval, some function $f(x)$ is well approximated by an n-degree QSP polynomial $P(x)$, then Theorem 2.5.2 also allows us to implement a transformation that approximates $f(A)$, where

$$f(A) := \begin{cases} \sum_j f(\sigma_j) |v_j\rangle \langle w_j|, & P \text{ is odd} \\ \sum_j f(\sigma_j) |w_j\rangle \langle w_j|, & P \text{ is even.} \end{cases} \tag{2.5.10}$$

The following theorem from Ref. [63] deals with the robustness of the QSVT procedure, i.e. how errors propagate in QSVT. In particular, for two matrices A and $\tilde{A}$, it shows how close their polynomial transformations ($P^{SV}(A)$ and $P^{SV}(\tilde{A})$, respectively) are, as a function of the distance between A and $\tilde{A}$.

Lemma 2.5.3 (Robustness of Quantum Singular Value Transformation, [63], Lemma 23)**.** *Let $P \in \mathbb{C}[x]$ be a QSP polynomial of degree n. Let $A, \tilde{A} \in \mathbb{C}^{N \times d}$ be matrices of spectral norm at most 1, such that*

$$\left\| A - \tilde{A} \right\| + \left\| \frac{A + \tilde{A}}{2} \right\|^2 \leq 1.$$

Then,

$$\left\| P^{SV}(A) - P^{SV}(\tilde{A}) \right\| \leq n \sqrt{\frac{2}{1 - \left\| \frac{A+\tilde{A}}{2} \right\|^2}} \left\| A - \tilde{A} \right\|.$$

We will apply this theorem to develop a robust version of QSVT. More precisely, in order to implement QSVT, we require access to a unitary U_A, which is a block-encoding of some matrix A. This block-encoding, in most practical scenarios, is not perfect: we only have access to a ε-approximate block-encoding of A. If we want an δ-accurate implementation of $P^{SV}(A)$, how precise should the block-encoding of A be? Such a robustness analysis has been absent from prior work involving QSVT and will allow us to develop robust versions of a number of quantum algorithms in subsequent sections. The following theorem determines the precision ε required in the block-encoding of A in terms of n, the degree of the QSP polynomial that we wish to implement and δ, the accuracy of $P^{SV}(A)$.

Theorem 2.5.4 (Robust QSVT)**.** *Let $P \in \mathbb{C}[x]$ be a QSP polynomial of degree $n \geq 2$. Let $\delta \in [0, 1]$ be the precision parameter. Let U be an (α, a, ε)-block-encoding of matrix $A \in \mathbb{C}^{N \times d}$ satisfying $\|A\| \leq \alpha/2$, implemented in cost T for some $\varepsilon \leq \alpha\delta/2n$. Then we can construct a $(1, a+1, \delta)$-block-encoding of $P(A/\alpha)$ in cost $\mathcal{O}(nT)$.*

Proof. Let $\tilde{A}$ be the encoded block of U, then $\left\| A - \tilde{A} \right\| \leq \varepsilon$. Applying QSVT on U with the polynomial P, we get a block-encoding for $P(\tilde{A}/\alpha)$, with $\mathcal{O}(n)$ uses of $U, U^\dagger$, and as many multiply-controlled NOT gates. Observe that $\left\| \frac{A}{\alpha} - \frac{\tilde{A}}{\alpha} \right\| \leq \frac{\varepsilon}{\alpha} \leq \frac{\delta}{2n} \leq \frac{1}{4}$, and,

$$\left\| \frac{\frac{A}{\alpha} + \frac{\tilde{A}}{\alpha}}{2} \right\|^2 = \left\| \frac{A}{\alpha} + \frac{\tilde{A} - A}{2\alpha} \right\|^2 \leq \left(\frac{\|A\|}{\alpha} + \frac{\left\| \tilde{A} - A \right\|}{2\alpha} \right)^2 \leq \left(\frac{1}{2} + \frac{1}{8} \right)^2 \leq \frac{1}{2}$$

Therefore the error in the final block-encoding is given by invoking Lemma 2.5.3 with matrices $A/\alpha, \tilde{A}/\alpha$:

$$\left\| P\left(\frac{A}{\alpha}\right) - P\left(\frac{\tilde{A}}{\alpha}\right) \right\| \leq n \sqrt{\frac{2}{1 - \frac{1}{2}}} \frac{\varepsilon}{\alpha} = \frac{2n\varepsilon}{\alpha} \leq \delta.$$

□

In chapter 3, we will make use of Theorem 2.5.4, to develop robust quantum algorithms for singular value discrimination, variable-time matrix inversion, and negative powers of matrices. Subsequently, in chapter 5 we shall use these primitives to design robust quantum regularized least squares algorithms.

2.6 Variable Time Amplitude Amplification

Ambainis [45] defined the notion of a *variable-stopping-time quantum algorithm* and formulated the technique of *Variable Time Amplitude Amplification* (VTAA), a tool that can be used to amplify the success probability of a variable-stopping-time quantum algorithm to a constant by taking advantage of the fact that computation on some parts of an algorithm can complete earlier than on other parts. The key idea here is to look at a quantum algorithm $\mathcal{A}$ acting on a state $|\psi\rangle$ as a combination of m quantum sub-algorithms $\mathcal{A} = \mathcal{A}_m \cdot \mathcal{A}_{m-1} \cdot \ldots \mathcal{A}_1$, each acting on $|\psi\rangle$ conditioned on some ancilla flag being set. Formally, a variable stopping time algorithm is defined as follows

Definition 2.6.1 (Variable-stopping-time Algorithm, [45]). *A quantum algorithm $\mathcal{A}$ acting on $\mathcal{H}$ that can be written as m quantum sub-algorithms, $\mathcal{A} = \mathcal{A}_m \cdot \mathcal{A}_{m-1} \cdot \ldots \mathcal{A}_1$ is called a variable stopping time algorithm if $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_\mathcal{A}$, where $\mathcal{H}_C = \otimes_{i=1}^m \mathcal{H}_{C_i}$ with $\mathcal{H}_{C_i} = \text{span}(|0\rangle, |1\rangle)$, and each unitary $\mathcal{A}_j$ acts on $\mathcal{H}_{C_j} \otimes \mathcal{H}_\mathcal{A}$ controlled on the first $j-1$ qubits $|0\rangle^{\otimes j-1} \in \otimes_{i=1}^{j-1} \mathcal{H}_{C_i}$ being in the all zero state.*

Here $\mathcal{H}_{C_i}$ is a single qubit clock register. In VTAA, $\mathcal{H}_\mathcal{A}$ has a flag space consisting of a single qubit to indicate success, $\mathcal{H}_\mathcal{A} = \mathcal{H}_F \otimes \mathcal{H}_W$. Here $\mathcal{H}_F = \text{Span}(|g\rangle, |b\rangle)$ flags the good and bad parts of the run. Furthermore, for $1 \leq i \leq m$, define the stopping times t_i such that $t_1 < t_2 < \cdots t_m = T_{\max}$, such that the algorithm $\mathcal{A}_j \mathcal{A}_{j-1} \cdots \mathcal{A}_1$ having (gate/query) complexity t_i halts with probability

$$p_j = \left\| \Pi_{C_j} \mathcal{A}_j \mathcal{A}_{j-1} \cdots \mathcal{A}_1 |0\rangle_\mathcal{H} \right\|^2,$$

where $|0\rangle_\mathcal{H} \in \mathcal{H}$ is the all zero quantum state and Π_{C_j} is the projector onto $|1\rangle$ in $\mathcal{H}_{C_j}$. From this one can define the average stopping time of the algorithm $\mathcal{A}$ defined as

$$\|T\|_2 = \sqrt{\sum_{j=1}^m p_j t_j^2}.$$

For a variable stopping time algorithm if the average stopping time $\|T\|_2$ is less than the maximum stopping time $T_{\max}$, VTAA can amplify the success probability (p_{succ}) much faster than standard amplitude amplification. In this framework, the success probability of $\mathcal{A}$ is given by

$$p_{\text{succ}} = \|\Pi_F \mathcal{A}_m \mathcal{A}_{m-1} \cdots \mathcal{A}_1 |0\rangle_\mathcal{H}\|^2$$

While standard amplitude amplification requires time scaling as $\mathcal{O}\left(T_{\max}/\sqrt{p_{\text{succ}}}\right)$, the complexity of VTAA is more involved. Following [35], we define the complexity of VTAA as follows.

Lemma 2.6.2 (Efficient variable time amplitude amplification [35]). *Let U be a state preparation unitary such that $U|0\rangle^{\otimes k} = \sqrt{p_{\text{prep}}}|0\rangle|\psi_0\rangle + \sqrt{1-p_{\text{prep}}}|1\rangle|\psi_1\rangle$ that has a query complexity T_U. And let $\mathcal{A} = \mathcal{A}_m \mathcal{A}_{m-1} \cdots \mathcal{A}_1$ be a variable stopping time quantum algorithm that we want to apply to the state $|\psi_0\rangle$, with the following known bounds: $p_{\text{prep}} \geq p'_{\text{prep}}$ and $p_{\text{succ}} \geq p'_{\text{succ}}$. Define $T'_{\max} := 2T_{\max}/t_1$ and*

$$Q := \left(T_{\max} + \frac{T_U + k}{\sqrt{p_{\text{prep}}}}\right)\sqrt{\log\left(T'_{\max}\right)} + \frac{\left(\|T\|_2 + \frac{T_U+k}{\sqrt{p_{\text{prep}}}}\right)\log\left(T'_{\max}\right)}{\sqrt{p_{\text{succ}}}}.$$

Then with success probability $\geq 1-\delta$, we can create a variable-stopping time algorithm $\mathcal{A}'$ that prepares the state $a|0\rangle\mathcal{A}'|\psi_0\rangle + \sqrt{1-a^2}|1\rangle|\psi_{garbage}\rangle$, such that $a = \Theta(1)$ is a constant and $\mathcal{A}'$ has the complexity $\mathcal{O}(Q)$.

One cannot simply replace standard amplitude amplification with VTAA to boost the success probability of a quantum algorithm. A crucial task would be to recast the underlying algorithm in the VTAA framework. We will be applying VTAA to the quantum algorithm for matrix inversion by QSVT. So, first of all, in order to apply VTAA to the algorithm must be first recast into a variable-time stopping algorithm so that VTAA can be applied.

Originally, Ambainis [45] used VTAA to improve the running time of the HHL algorithm from $\mathcal{O}(\kappa^2 \log N)$ to $\mathcal{O}(\kappa \log^3 \kappa \log N)$. Childs et al. [46] designed a quantum linear systems algorithm with a polylogarithmic

dependence on the accuracy. Additionally, they recast their algorithm into a framework where VTAA could be applied to obtain a linear dependence on κ. Later Chakraborty et al. [35] modified Ambainis' VTAA algorithm to perform variable time amplitude estimation.

In this work, to design quantum algorithms for ℓ_2-regularized linear regression, we use a quantum algorithm for matrix inversion by QSVT. We recast this algorithm in the framework of VTAA to achieve nearly linear dependence in κ (the condition number of the matrix to be inverted). QSVT instead of controlled Hamiltonian simulation improves the complexity of the overall matrix inversion algorithm (using QSVT and VTAA) by a log factor and reduces the number of additional qubits substantially. Furthermore, we replace a quantum gapped phase estimation procedure with a more efficient quantum eigenvalue discrimination algorithm using QSVT. This further reduces the number of additional qubits by $O(\log^2(\kappa/\delta))$ than in Refs. [46, 35], where κ is the condition number of the underlying matrix and δ is the desired accuracy. The details of a variable stopping time quantum algorithm for matrix inversion by QSVT is laid out in section 4.3.

Chapter 3

Algorithmic Primitives

This chapter introduces the building blocks of our quantum algorithms for quantum linear regression with general ℓ_2-regularization. As mentioned previously, we work in the block-encoding framework. We develop robust quantum algorithms for arithmetic operations, inversion, and positive and negative powers of matrices using quantum singular value transformation, assuming that we have access to approximate block-encodings of these matrices. While some of these results were previously derived assuming perfect block-encodings [42, 35], we calculate the precision required in the input block-encodings to output a block-encoding or quantum state arbitrarily close to the target.

We highlight that the primitives developed here might be of independent interest to the quantum machine learning community. In developing these primitives, we improve upon various algorithms used frequently in this domain, and in some cases highlight the robustness of the algorithms in the QSVT framework.[i]

In particular for the problem of inverting a matrix using QSVT, we reformulate QSVT-based matrix inversion as a variable stopping time algorithm such that VTAA is applicable. The use of QSVT instead of controlled Hamiltonian simulation improves the complexity of the overall matrix inversion algorithm (using QSVT and VTAA) by a log factor and reduces the number of ancilla qubits substantially. Furthermore, in order to convert the usual matrix inversion algorithm to a variable stopping time algorithm, we make use of quantum eigenvalue discrimination using QSVT [44], instead of gapped phase estimation used in [46]. This further reduces the number of additional qubits by $O(\log^2(\kappa/\delta))$, where κ is the condition number of the underlying matrix and δ is the desired accuracy.

3.1 Amplification of Block Encodings

Given a (α, a, ε)-block-encoding of a matrix A, we can efficiently amplify the sub-normalization factor from α to a constant and obtain an amplified block-encoding of A. For our quantum algorithms in Sec. 4.3, we show working with pre-amplified block-encodings often yields better complexities. We state the following lemma which was proven in Ref. [64]:

Lemma 3.1.1 (Uniform Block Amplification of Contractions, [64])**.** *Let $A \in \mathbb{R}^{N\times d}$ such that $\|A\| \leq 1$ If $\alpha \geq 1$ and U is a (α, a, ε)-block-encoding of A that can be implemented at a cost of T_U, then there is a $(\sqrt{2}, a+1, \varepsilon+\gamma)$-block-encoding of A that can be implemented at a cost of $\mathcal{O}(\alpha T_U \log(1/\gamma))$.*

Corollary 3.1.2 (Uniform Block Amplification)**.** *Let $A \in \mathbb{R}^{N\times d}$ and $\delta \in (0, 1]$. Suppose U is a (α, a, ε)-block-encoding of A, such that $\varepsilon \leq \frac{\delta}{2}$, that can be implemented at a cost of T_U. Then a $(\sqrt{2}\|A\|, a+1, \delta)$-block-encoding of A can be implemented at a cost of $\mathcal{O}\Big(\frac{\alpha T_U}{\|A\|}\log(\|A\|/\delta)\Big)$.*

Proof. We can re-interpret U as a $(\alpha/\|A\|, a, \varepsilon/\|A\|)$-block-encoding of $A/\|A\|$. Invoking Lemma 3.1.1 with $\gamma = \frac{\delta}{2\|A\|}$, we get U', a $(\sqrt{2}, a+1, \varepsilon/\|A\| + \frac{\delta}{2\|A\|})$-block-encoding of $A/\|A\|$, implemented at a cost of $\mathcal{O}\Big(\frac{\alpha}{\|A\|}T_U \log(\|A\|/\delta)\Big)$ which is a $(\sqrt{2}\|A\|, a+1, \delta)$-block-encoding of A. □

We now obtain the complexity of applying a block-encoded matrix to a quantum state, which is a generalization of a lemma proven in Ref. [35].

[i] By robustness we mean that we highlight how the error in the encoding of the input affects the encoding of the output.

Lemma 3.1.3 (Applying a Block-encoded Matrix on a Quantum State)**.** *Let A be an s-qubit operator such that its singular values lie in $[\|A\|/\kappa, \|A\|]$. Also let $\delta \in (0,1)$, and U_A be an (α, a, ε)-block-encoding of A, such that*

$$\varepsilon \leq \frac{\delta \|A\|}{2\kappa},$$

that can be implemented in time T_A. Furthermore, suppose $|b\rangle$ be an s-qubit quantum state that can be prepared in time T_b. Then we can prepare a state that is δ-close to $\frac{A|b\rangle}{\|A|b\rangle\|}$ with success probability $\Omega(1)$ at a cost of

$$\mathcal{O}\left(\frac{\alpha\kappa}{\|A\|}(T_A + T_b)\right)$$

Proof. The proof is similar to Lemma 24 of [35]. We have $\|A|b\rangle\| \geq \frac{\|A\|}{\kappa}$. By applying U_A to $|0\rangle|b\rangle$ (implementable at a cost of $T_A + T_B$), followed by $\frac{\alpha\kappa}{\|A\|}$-rounds of amplitude amplification (conditioned on having $|0\rangle$ in the first register) , we obtain a quantum state that within δ of $|0\rangle \otimes \frac{A|b\rangle}{\|A|b\rangle\|}$. □

Corollary 3.1.4 (Applying a pre-amplified Block-encoded Matrix on a Quantum State)**.** *Let A be an s-qubit operator such that its singular values lie in $[\|A\|/\kappa, \|A\|]$. Also let $\delta \in (0,1)$, and U_A be an (α, a, ε)-block-encoding of A, such that*

$$\varepsilon \leq \frac{\delta \|A\|}{4\kappa},$$

that can be implemented in time T_A. Furthermore, suppose $|b\rangle$ be an s-qubit quantum state that can be prepared in time T_b. Then we can prepare a state that is δ-close to $\frac{A|b\rangle}{\|A|b\rangle\|}$ with success probability $\Omega(1)$ at a cost of

$$\mathcal{O}\left(\frac{\alpha\kappa}{\|A\|}\log\left(\frac{\kappa}{\delta}\right)T_A + \kappa T_b\right)$$

Proof. We first pre-amplify the unitary using Corollary 3.1.2 with some $\gamma \geq 2\varepsilon$. We get a $(\sqrt{2}\|A\|, a+1, \gamma)$-block-encoding of A implemented at a cost of

$$T_{A'} := \mathcal{O}\left(\frac{\alpha T_A}{\|A\|}\log\left(\frac{\|A\|}{\gamma}\right)\right)$$

Now we invoke Lemma 3.1.3 with $\delta = \frac{2\kappa\gamma}{\|A\|}$ and the above unitary to prepare the state, which has a time complexity of

$$\mathcal{O}(\kappa(T_{A'} + T_b)) = \mathcal{O}\left(\frac{\alpha\kappa}{\|A\|}\log\left(\frac{\kappa}{\delta}\right)T_A + \kappa T_b\right)$$

□

Now, it may happen that the U_b prepares a quantum state that is only ε-close to the desired state $|b\rangle$. In such cases, we have the following lemma

Lemma 3.1.5 (Robustness of state preparation)**.** *Let A be an s-qubit operator such that its singular values lie in $[\frac{\|A\|}{\kappa}, \|A\|]$. Suppose $|b'\rangle$ is a quantum state that is $\varepsilon/2\kappa$-close to $|b\rangle$ and $|\psi\rangle$ is a quantum state that is $\varepsilon/2$-close to $A|b'\rangle/\|A|b'\rangle\|$. Then we have that $|\psi\rangle$ is ε-close to $A|b\rangle/\|A|b\rangle\|$.*

Proof. We know that

$$\||b\rangle - |b'\rangle\| \leq \frac{\varepsilon}{2\kappa}$$

and

$$\left\||\psi\rangle - \frac{A|b'\rangle}{\|A|b'\rangle\|}\right\| \leq \frac{\varepsilon}{2}$$

For small enough $\varepsilon \ll \kappa$, we can assume that $\|A|b\rangle\| \approx \|A|b'\rangle\|$.

We can derive the final error as

$$\begin{aligned}\left\||\psi\rangle - \frac{A|b\rangle}{\|A|b\rangle\|}\right\| &= \left\||\psi\rangle - \frac{A|b\rangle - A|b'\rangle + A|b'\rangle}{\|A|b\rangle\|}\right\| \\ &= \left\||\psi\rangle - \frac{A|b'\rangle}{\|A|b\rangle\|} + \frac{A|b'\rangle - A|b'\rangle}{\|A|b\rangle\|}\right\|\end{aligned}$$

$$
\begin{aligned}
&\leq \left\| |\psi\rangle - \frac{A|b'\rangle}{\|A|b'\rangle\|} \right\| + \left\| \frac{A|b'\rangle - A|b'\rangle}{\|A|b\rangle\|} \right\| \\
&\leq \frac{\varepsilon}{2} + \frac{\|A\|\||b\rangle - |b'\rangle\|}{\|A|b\rangle\|} \\
&\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} \\
&= \varepsilon
\end{aligned}
$$

□

3.2 Arithmetic with Block-Encoded Matrices

The block-encoding framework embeds a matrix on the top left block of a larger unitary U. It has been demonstrated framework allows us to obtain sums, products, linear combinations of block-encoded matrices. This is particularly useful for solving linear algebra problems in general. Here, we state some of the arithmetic operations on block-encoded matrices that we shall be using in order to design the quantum algorithms of chapter 5 and tailor existing results to our requirements.

First we prove a slightly more general form of linear combination of unitaries in the block-encoding framework, presented in [42]. To do this we assume that we are given optimal state preparation pairs, defined as follows.

Lemma 3.2.1 (Optimal State Preparation Unitary)**.** *Let $m \in \mathbb{Z}^+$, $\eta \in \mathbb{R}^m_+$, and $s = \lceil \log m \rceil$. Then there exists an s-qubit unitary P – called a η state-preparation unitary – such that $P|0\rangle$ is proportional to $\sum_j \sqrt{\eta_j}|j\rangle$*

Lemma 3.2.2 (Linear Combination of Block Encoded Matrices)**.** *For each $j \in \{0, \ldots, m-1\}$, let A_j be an s-qubit operator, and $y_j \in \mathbb{R}^+$. Let U_j be a $(\alpha_j, a_j, \varepsilon_j)$-block-encoding of A_j , implemented in time T_j. Define the matrix $A = \sum_j y_j A_j$, and the vector $\eta \in \mathbb{R}^m$ s.t. $\eta_j = y_j \alpha_j$. Let U_η be a η state-preparation unitary , implemented in time T_η. Then we can implement a*

$$\left(\sum_j y_j \alpha_j, \max_j(a_j) + s, \sum_j y_j \varepsilon_j \right)$$

block-encoding of A at a cost of $\mathcal{O}\left(\sum_j T_j + T_\eta\right)$.

Proof. The proof is similar to the one in Ref. [42], with some improvements to the bounds. Let $a = max_j(a_j) + s$ and $\alpha = \sum_j y_j \alpha_j$. For each $j \in \{0, \ldots, m-1\}$, construct the extended unitary U'_j by padding ancillas to U_j, i.e. $U'_j = I_{a-s-a_j} \otimes U_j$. Note that U'_j is a $(\alpha_j, a-s, \varepsilon_j)$-block-encoding of A_j. Let $B_j = (\langle 0|^{a_j} \otimes I_s) U_j (|0\rangle^{a_j} \otimes I_s)$ denote the top left block of U_j and U'_j, and observe that $\|A_j - \alpha_j B_j\| \leq \varepsilon_j$. We also construct P — an η state-preparation unitary s.t. $P|0\rangle = \sum_j \sqrt{y_j \alpha_j}|j\rangle$ — by invoking Lemma 3.2.1.

Consider the unitary $W = (P^\dagger \otimes I_{a-1} \otimes I_s)(\sum_j |j\rangle\langle j| \otimes U'_j)(P \otimes I_{a-1} \otimes I_s)$. This is a (α, a, ε)-block-encoding of $A = \sum_j y_j A_j$, where ε is computed as:

$$
\begin{aligned}
\|A - \alpha(\langle 0|^a \otimes I_s) W (|0\rangle^a \otimes I_s)\| &= \left\| \sum_{j=0}^{m-1} y_j A_j - \alpha(\langle 0|^a \otimes I_s) W (|0\rangle^a \otimes I_s) \right\| \\
&= \left\| \sum_j y_j A_j - \alpha(\langle 0|^a \otimes I_s)(\sum_j P^\dagger |j\rangle\langle j| P \otimes U'_j)(|0\rangle^a \otimes I_s) \right\| \\
&= \left\| \sum_j y_j A_j - \alpha \sum_j \langle 0| P^\dagger |j\rangle\langle j| P |0\rangle \otimes B_j \right\| \\
&= \left\| \sum_j \left(y_j A_j - \alpha \langle 0| P^\dagger |j\rangle\langle j| P |0\rangle B_j \right) \right\| \\
&= \left\| \sum_j \left(y_j A_j - \alpha \left(\frac{y_j \alpha_j}{\alpha} \right) B_j \right) \right\|
\end{aligned}
$$

$$\leq \sum_j y_j \|A_j - \alpha_j B_j\|$$
$$\leq \sum_j y_j \varepsilon_j = \varepsilon$$

□

We now specialize the above lemma for the case where we need a linear combination of just two unitaries. This is the case used in this work, and we obtain a better error scaling for this by giving an explicit state preparation unitary.

Corollary 3.2.3 (Linear Combination of Two Block Encoded Matrices)**.** *For $j \in \{0,1\}$, let A_j be an s-qubit operator and $y_j \in \mathbb{R}^+$. Let U_j be a $(\alpha_j, a_j, \varepsilon_j)$-block-encoding of A_j, implemented in time T_j. Then we can implement a $(y_0\alpha_0 + y_1\alpha_1, 1 + \max(a_0, a_1), y_0\varepsilon_0 + y_1\varepsilon_1)$ encoding of $y_0A_0 + y_1A_1$ in time $\mathcal{O}(T_0 + T_1)$.*

Proof. Let $\alpha = y_0\alpha_0 + y_1\alpha_1$ and $P = \frac{1}{\sqrt{\alpha}} \begin{pmatrix} \sqrt{y_0\alpha_0} & -\sqrt{y_1\alpha_1} \\ \sqrt{y_1\alpha_1} & \sqrt{y_0\alpha_0} \end{pmatrix}$. By Lemma 3.2.1, we have that P is an $\{y_0\alpha_0, y_1\alpha_1\}$ state preparation unitary. Invoking Lemma 3.2.2 with P, we get the required unitary.

□

Given block-encodings of two matrices A and B, it is easy to obtain a block-encoding of AB.

Lemma 3.2.4 (Product of Block Encodings, [42])**.** *If U_A is an (α, a, δ)-block-encoding of an s-qubit operator A implemented in time T_A, and U_B is a (β, b, ε)-block-encoding of an s-qubit operator B implemented in time T_B, then $(I^{\otimes b} \otimes U_A)(I^{\otimes a} \otimes U_B)$ is an $(\alpha\beta, a+b, \alpha\varepsilon+\beta\delta)$-block-encoding of AB implemented at a cost of $\mathcal{O}(T_A + T_B)$.*

Directly applying Lemma 3.2.4 results in a block-encoding of $\frac{AB}{\alpha\beta}$. If α and β are large, then the sub-normalization factor $\alpha\beta$ might incur an undesirable overhead to the cost of the algorithm that uses it. In many cases, the complexity of obtaining products of block-encodings can be improved if we first amplify the block-encodings (using Lemma 3.1.2) and then apply Lemma 3.2.4. We prove the following lemma:

Lemma 3.2.5 (Product of Amplified Block-Encodings)**.** *Let $\delta \in (0, 1]$. If U_A is an $(\alpha_A, a_A, \varepsilon_A)$-block-encoding of an s-qubit operator A implemented in time T_A, and U_B is a $(\alpha_B, a_B, \varepsilon_B)$-block-encoding of an s-qubit operator B implemented in time T_B, such that $\varepsilon_A \leq \frac{\delta}{4\sqrt{2}\|B\|}$ and $\varepsilon_B \leq \frac{\delta}{4\sqrt{2}\|A\|}$. Then we can implement a $(2\|A\|\|B\|, a_A + a_B + 2, \delta)$-block-encoding of AB implemented at a cost of*

$$\mathcal{O}\left(\left(\frac{\alpha_A}{\|A\|}T_A + \frac{\alpha_B}{\|B\|}T_B\right)\log\left(\frac{\|A\|\|B\|}{\delta}\right)\right).$$

Proof. Using Corollary 3.1.2 for some $\delta_A \geq 2\varepsilon_A$ we get a $(\sqrt{2}\|A\|, a_A + 1, \delta_A)$-block-encoding of A at a cost of

$$\mathcal{O}\left(\frac{\alpha_A T_A}{\|A\|}\log\left(\|A\|/\delta_A\right)\right).$$

Similarly for some $\delta_B \geq 2\varepsilon_B$ we get a $(\sqrt{2}\|B\|, a_B + 1, \delta_B)$-block-encoding of B at a cost of

$$\mathcal{O}\left(\frac{\alpha_B T_B}{\|B\|}\log\left(\|B\|/\delta_B\right)\right).$$

Now using Lemma 3.2.4 we get a $(2, a_A + a_B + 2, \sqrt{2}\left(\|A\|\delta_B + \|B\|\delta_A\right))$-block-encoding of AB. We can choose $\delta_A = \frac{\delta}{2\sqrt{2}\|B\|}$ and $\delta_B = \frac{\delta}{2\sqrt{2}\|A\|}$ which bounds the final block-encoding error by δ. □

Observe that we have assumed that A and B are s-qubit operators. For any two matrices of dimension $N \times d$ and $d \times K$, such that $N, d, K \leq 2^s$, we can always pad them with rows and columns of zero entries and convert them to s-qubit operators. Thus, in the scenario where A and B are not s-qubit operators, one can consider block encodings of padded versions of these matrices. Note that this does not affect the operations on the sub-matrix blocks encoding A and B. Thus, the above results can be used to perform block-encoded matrix products for arbitrary (compatible) matrices.

Next we show how to find the block encoding of tensor product of matrices from their block encodings. This procedure will be useful in creating the dilated matrices required for regularization.

Lemma 3.2.6 (Tensor Product of Block Encoded Matrices)**.** *Let U_1 and U_2 be $(\alpha, a, \varepsilon_1)$ and $(\beta, b, \varepsilon_2)$-block-encodings of A_1 and A_2, s and t-qubit operators, implemented in time T_1 and T_2 respectively. Define $S := \Pi_{i=1}^{s} \mathtt{SWAP}_{a+b+i}^{a+i}$. Then, $S(U_1 \otimes U_2)S^\dagger$ is an $(\alpha\beta, a+b, \alpha\varepsilon_2 + \beta\varepsilon_1 + \varepsilon_1\varepsilon_2)$ block-encoding of $A_1 \otimes A_2$, implemented at a cost of $\mathcal{O}(T_1 + T_2)$.*

Proof. From the property of Kronecker products $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$. For $j \in \{1, 2\}$ let $\tilde{A}_j = \left(\langle 0|^{\otimes a_j} \otimes I_s\right) U_j \left(|0\rangle^{\otimes a_j} \otimes I_s\right)$. Then it follows that

$$\left(\langle 0|^{\otimes a} \otimes I_s \otimes \langle 0|^{\otimes b} \otimes I_t\right)(U_1 \otimes U_2)\left(|0\rangle^{\otimes a} \otimes I_s \otimes |0\rangle^{\otimes b} \otimes I_t\right) = \tilde{A}_1 \otimes \tilde{A}_2 \tag{3.2.1}$$

Therefore $\tilde{A}_1 \otimes \tilde{A}_2$ is block-encoded in $U_1 \otimes U_2$ as a non-principal block-encoding, and we can use `SWAP` gates to move it to the principal block as follows.

$$\begin{aligned}
S\left(|0\rangle^{\otimes a} \otimes I_s |0\rangle^{\otimes b} \otimes I_t\right) &= \Pi_{i=1}^{s} \mathtt{SWAP}_{a+b+i}^{a+i}\left(|0\rangle^{\otimes a} \otimes I_s |0\rangle^{\otimes b} \otimes I_t\right) \\
&= \Pi_{i=1}^{s-1} \mathtt{SWAP}_{a+b+i}^{a+i} \mathtt{SWAP}_{a+b+s}^{a+s}\left(|0\rangle^{\otimes a} \otimes I_s |0\rangle^{\otimes b} \otimes I_t\right) \\
&= \Pi_{i=1}^{s-1} \mathtt{SWAP}_{a+b+i}^{a+i}\left(|0\rangle^{\otimes a} \otimes I_{s-1} |0\rangle^{\otimes b} \otimes I_{t+1}\right) \\
&= \dots \\
&= |0\rangle^{\otimes a+b} \otimes I_{s+t}
\end{aligned}$$

Similarly,

$$\left(\langle 0|^{\otimes a} \otimes I_s \otimes \langle 0|^{\otimes b} \otimes I_t\right) S^\dagger = \langle 0|^{\otimes a+b} \otimes I_{s+t}.$$

From Equation 3.2.1 we have

$$\begin{aligned}
\tilde{A}_1 \otimes \tilde{A}_2 &= \left(\langle 0|^{\otimes a} \otimes I_s \otimes \langle 0|^{\otimes b} \otimes I_t\right) S^\dagger S (U_1 \otimes U_2) S^\dagger S \left(|0\rangle^{\otimes a} \otimes I_s |0\rangle^{\otimes b} \otimes I_t\right) \\
&= \left(\langle 0|^{\otimes a+b} \otimes I_{s+t}\right) S (U_1 \otimes U_2) S^\dagger \left(|0\rangle^{\otimes a+b} \otimes I_{s+t}\right)
\end{aligned}$$

Next, we look at the sub-normalization and error terms.

$$\begin{aligned}
\left\| A_1 \otimes A_2 - \alpha\beta \tilde{A}_1 \otimes \tilde{A}_2 \right\|_2 &\leq \left\| (\alpha\tilde{A}_1 + \varepsilon_1 I_s) \otimes (\beta\tilde{A}_2 + \varepsilon_2 I_t) - \alpha\tilde{A}_1 \otimes \beta\tilde{A}_2 \right\|_2 \\
&= \left\| \alpha\tilde{A}_1 \otimes \varepsilon_2 I_2 + \varepsilon_1 I_s \otimes \beta\tilde{A}_2 + \varepsilon_1 I_s \otimes \varepsilon_2 I_2 \right\|_2 \\
&\leq \alpha\varepsilon_2 \left\| \tilde{A}_1 \right\|_2 + \beta\varepsilon_2 \left\| \tilde{A}_2 \right\|_2 + \varepsilon_1\varepsilon_2 \\
&= \alpha\varepsilon_2 + \beta\varepsilon_1 + \varepsilon_1\varepsilon_2
\end{aligned}$$

where we have used $\|A_1\|_2 \leq \alpha \left\| \tilde{A}_1 \right\|_2 + \varepsilon_1$ and $\left\| \tilde{A}_1 \right\|_2 \leq 1$ and similarly for A_2.

□

We will now use Lemma 3.2.6 to augment one matrix into another, given their approximate block-encodings.

Lemma 3.2.7 (Block-encoding of augmented matrix)**.** *If U_A is an $(\alpha_A, a_A, \varepsilon_A)$-block encoding of an s-qubit operator A that can be implemented in time T_A and U_B is an $(\alpha_B, a_B, \varepsilon_B)$-block encoding of an s-qubit operator B that can be implemented in time T_B, then we an implement an $(\alpha_A + \alpha_B, \max(a_A, a_B) + 2, \varepsilon_A + \varepsilon_B)$-block-encoding of*

$$A_B = \begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$$

at a cost of $\mathcal{O}(T_A + T_B)$.

Proof. Let $M_A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$. Then the SWAP gate is a $(1, 1, 0)$ block encoding of M_A. By Lemma 3.2.6, we can implement U'_A, an $(\alpha_A, a_A+1, \varepsilon_A)$-block-encoding of their tensor product $M_A \otimes A = \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix}$ at a cost of $\mathcal{O}(T_A)$. Similarly, Let $M_B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$. Then $(I \otimes X) \cdot \texttt{SWAP}$ is a $(1, 1, 0)$-block-encoding of M_B. Similarly Lemma 3.2.6, we can implement U'_B, an $(\alpha_B, a_B + 1, \varepsilon_B)$-block-encoding of $M_B \otimes B = \begin{pmatrix} 0 & 0 \\ B & 0 \end{pmatrix}$ at a cost of $\mathcal{O}(T_B)$. We add them by using Corollary 3.2.3 on U'_A and U'_B, to implement U_{A_B}, an $(\alpha_A + \alpha_B, 2 + \max(a_A, a_B), \varepsilon_A + \varepsilon_B)$-block-encoding of $A_B = \begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$. This can be implemented at a cost of $\mathcal{O}(T_A + T_B)$. □

Lemma 3.2.8 (Block-Encoding of a dilated matrix)**.** *If U_A is an (α, a, ε)-block-encoding of an s-qubit operator A implemented in time T_A, then there is a (α, a, ε)-block-encoding of an $s+1$-qubit operator $\bar{A} := \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$, that can be prepared at a cost of $\mathcal{O}(T_A)$.*

Proof. By adding an additional qubit, we can implement a controlled unitary $C\text{-}U_A$ that acts on $(a + s + 1)$ qubits such that controlled on the $(a+1)^{\text{th}}$ qubit, it implements U_A on the first a and the last s qubits. Let,

$$V = C\text{-}U_A^\dagger (X \otimes I) C\text{-}U_A = |0\rangle\langle 1| \otimes U_A + |1\rangle\langle 0| \otimes U_A^\dagger,$$

Then $\texttt{SWAP}_{a,1} V$ is an (α, a, ε)-block encoding of $\bar{A}$. Here $\texttt{SWAP}_{a,1}$ is a sequence of SWAP gates that swaps the entire a-qubit register, $|0\rangle^{\otimes a}$ with the single qubit control register, one qubit at a time. □

When using a dilated matrix $\bar{A}$, we must also extend the input state to $|\bar{b}\rangle = |0\rangle |b\rangle$. This just increases the number of input qubits by 1, but keeps the rest of the behavior identical to the non-dilated setting.

3.3 Robust Quantum Singular Value Discrimination

The problem of deciding whether the eigenvalues of a Hamiltonian lie above or below a certain threshold, known as *eigenvalue discrimination*, finds widespread applications. For instance, the problem of determining whether the ground energy of a generic local Hamiltonian is $< \lambda_a$ or $> \lambda_b$ is known to be QMA-Complete [65]. Nevertheless, quantum eigenvalue discrimination has been useful in preparing ground states of Hamiltonians. Generally, a variant of quantum phase estimation, which effectively performs a projection onto the eigenbasis of the underlying Hamiltonian, is used to perform eigenvalue discrimination [66]. Recently, it has been shown that QSVT can be used to approximate a projection onto the eigenspace of an operator by implementing a polynomial approximation of the *sign function* [67]. This was then used to design improved quantum algorithms for ground state preparation.

In our work, we design a more general primitive, known as *Quantum Singular Value Discrimination* (QSVD). Instead of eigenvalues, the algorithm distinguishes whether a singular value σ is $\leq \sigma_a$ or $\geq \sigma_b$. This is particularly useful when the block-encoded matrix is not necessarily Hermitian and hence, may not have well-defined eigenvalues. We use this procedure to develop a more space-efficient variable stopping time matrix inversion algorithm in chapter 4. Owing to the widespread use of singular values in a plethora of fields, we believe that our QSVD procedure is of independent interest.

Let us define the *sign function* $\text{sign} : \mathbb{R} \to \mathbb{R}$ as follows:

$$\text{sign}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0. \end{cases} \tag{3.3.1}$$

Given a threshold singular value c, Low and Chuang [64] showed that there exists a polynomial approximation to $\text{sign}(c - x)$ (based on its approximation of the *erf function*). We use the result of Ref. [44], where such a polynomial of even parity was considered. This is crucial, as for even polynomials, QSVT maps right (left) singular vectors to right (left) singular vectors, which enables us to use the polynomial in [44] for singular value discrimination.

Lemma 3.3.1 (Polynomial approximation to the sign function [64, 68, 44])**.** *For any $\varepsilon, \Delta, c \in (0, 1)$, there exists an efficiently computable even polynomial $P_{\varepsilon,\Delta,c}(x)$ of degree $l = \mathcal{O}\left(\frac{1}{\Delta} \log(1/\varepsilon)\right)$ such that*

1. $\forall x \in [0,1]\colon |P_{\varepsilon,\Delta,c}(x)| \leq 1$

2. $\forall x \in [0,1] \setminus \left(c - \frac{\Delta}{2}, c + \frac{\Delta}{2}\right) : |P_{\varepsilon,\Delta,c}(x) - \text{sign}(c-x)| \leq \varepsilon$

Therefore, given a matrix A with singular values between $[0,1]$, we can use QSVT to implement $P_{\varepsilon,\Delta,c}(A)$ which correctly distinguishes between singular values of A whose value is less than $c - \Delta/2$ and those whose value is greater than $c + \Delta/2$. For our purposes, we shall consider that we are given U_A, which is an (α, a, ε) block-encoding of a matrix A. Our goal would be to distinguish whether a certain singular value σ satisfies $0 \leq \sigma \leq \varphi$ or $2\varphi \leq \sigma \leq 1$. Since U_A (approximately) implements A/α, the task can be rephrased as distinguishing whether a singular value of A/α is in $[0, \varphi/\alpha]$ or in $[2\varphi/\alpha, 1]$. For this, we develop a robust version of quantum singular value discrimination ($QSVD(\phi, \delta)$), which indicates the precision ε required to commit an error that is at most δ.

Theorem 3.3.2 (Quantum Singular Value Discrimination using QSVT)**.** *Suppose $A \in \mathbb{C}^{N\times N}$ is an s-qubit operator (where $N = 2^s$) with singular value decomposition $A = \sum_{j\in[N]} \sigma_j |u_j\rangle\langle v_j|$ such that all σ_j lie in the range $[0,1]$. Let $\varphi \in \left(0, \frac{1}{2}\right)$ and $\delta \in (0,1]$ be some parameters. Suppose that for some $\alpha \geq 2$ and ε satisfying*

$$\varepsilon = o\left(\frac{\delta\varphi}{\log(1/\delta)}\right)$$

we have access to U_A, an (α, a, ε)-block-encoding of A implemented in cost T_A. Then there exists a quantum algorithm $QSVD(\varphi, \delta)$ which implements a $(1, a+1, \delta)$-block-encoding of some $(s+1)$-qubit operator $D \in \mathbb{C}^{2N\times 2N}$ satisfying the following constraints for all $j \in [N]$:

- $\sigma_j \leq \varphi \implies D|0\rangle|v_j\rangle = |0\rangle|v_j\rangle$
- $\sigma_j \geq 2\varphi \implies D|0\rangle|v_j\rangle = |1\rangle|v_j\rangle$

This algorithm has a cost of

$$\mathcal{O}\left(\frac{\alpha}{\varphi}\log\left(\frac{1}{\delta}\right)T_A\right).$$

Proof. We invoke Lemma 3.3.1 with parameters $\varepsilon' := \frac{\delta}{2}$, $c := \frac{3\varphi}{2\alpha}$ and $\Delta := \frac{\varphi}{2\alpha}$, to construct an even polynomial $P := P_{\varepsilon',\Delta,c}$ of degree $n := \mathcal{O}\left(\frac{\alpha}{\varphi}\log\left(\frac{1}{\varepsilon'}\right)\right)$, which is an ε'-approximation of $f(x) := \text{sign}\left(\frac{3\varphi}{2\alpha} - x\right)$ for $x \in \left[0, \frac{\varphi}{\alpha}\right] \cup \left[\frac{2\varphi}{\alpha}, 1\right]$. Invoking Theorem 2.5.4 with P and U_A, we get U_B – a $(1, a+1, \gamma)$-block-encoding of $B := P(A/\alpha)$, implemented in cost $\mathcal{O}(nT_A)$, where ε must satisfy $\varepsilon \leq \alpha\gamma/2n$.

Now consider the following unitary W that acts on $s + a + 2$ qubits:

$$W := \texttt{SWAP}^\dagger_{[s,s+a+1]}(H \otimes I_{s+a+1})\,(C\text{-}U_B)\,(H \otimes I_{s+a+1})\texttt{SWAP}_{[s,s+a+1]}$$

W is the required block-encoding of D, and $\texttt{SWAP}_{[l,r]}$ sequentially swaps adjacent qubits with indices in range $[l,r]$ effectively moving qubit indexed l to the right of qubit r. (where qubits are zero-indexed, with higher indices for ancillas). Let $\tilde{B}$ be the top-left block of U_B (therefore $\left\|B - \tilde{B}\right\| \leq \gamma$). Then we can extract $\tilde{D}$, the top-left block of W as follows:

$$\begin{aligned}\tilde{D} &= \left(\langle 0|^{\otimes a+1} \otimes I_{s+1}\right)\texttt{SWAP}^\dagger_{[s,s+a+1]}\left(|+\rangle\langle+| \otimes I_{s+a+1} + |-\rangle\langle-| \otimes U_B\right)\texttt{SWAP}_{[s,s+a+1]}\left(|0\rangle^{\otimes a+1} \otimes I_{s+1}\right)\\ &= |+\rangle\langle+| \otimes I_s + |-\rangle\langle-| \otimes \tilde{B}\end{aligned}$$

Let us define index sets $L, R \subseteq [N]$ where $L := \{j \in [N] \mid \sigma_j \leq \varphi\}$ and $R := \{j \in [N] \mid \sigma_j \geq 2\varphi\}$; and the corresponding subspace projections $\Pi_L := \sum_{j\in L} |v_j\rangle\langle v_j|$, $\Pi_R := \sum_{j\in R} |v_j\rangle\langle v_j|$, and $\Pi_\perp := I_s - \Pi_L - \Pi_R$. Using these we pick our required operator D as follows:

$$D := I \otimes \Pi_L + X \otimes \Pi_R + \tilde{D}(I \otimes \Pi_\perp)$$

That is, D behaves as expected on the required subspace, and acts identical to $\tilde{D}$ on the remaining space. The error in the block-encoding can be computed as

$$\left\|D - \tilde{D}\right\| = \left\|I \otimes \Pi_L + X \otimes \Pi_R + \tilde{D}(I \otimes \Pi_\perp) - \tilde{D}\right\|$$

$$
\begin{aligned}
&= \left\| I \otimes \Pi_L + X \otimes \Pi_R - \tilde{D}(I \otimes (\Pi_L + \Pi_R)) \right\| \\
&= \left\| (I \otimes I_s - \tilde{D})(I \otimes \Pi_L) + (X \otimes I_s - \tilde{D})(I \otimes \Pi_R) \right\| \\
&= \left\| \left(|-\rangle\langle -| \otimes (I_s - \tilde{B}) \right) (I \otimes \Pi_L) - \left(|-\rangle\langle -| \otimes (I_s + \tilde{B}) \right) (I \otimes \Pi_R) \right\| \\
&= \left\| (I_s - \tilde{B})\Pi_L - (I_s + \tilde{B})\Pi_R \right\| \\
&= \left\| (I_s - B)\Pi_L - (I_s + B)\Pi_R + (B - \tilde{B})(\Pi_L - \Pi_R) \right\| \\
&\leq \|(I_s - P(A/\alpha))\Pi_L - (I_s + P(A/\alpha))\Pi_R\| + \left\| B - \tilde{B} \right\| \|\Pi_L - \Pi_R\| \\
&\leq \varepsilon' + \gamma
\end{aligned}
$$

We can choose $\gamma = \delta/2$, therefore

$$\varepsilon \leq \frac{\alpha\delta}{4n} = \mathcal{O}\left(\frac{\delta\varphi}{\log\left(\frac{1}{\delta}\right)} \right)$$

□

In chapter 4, we develop a variable stopping time quantum algorithm for matrix inversion using QSVT. In order to recast the usual matrix inversion to the VTAA framework, we need to be able to apply this algorithm to specific ranges of the singular values of the matrix. This is achieved by applying a controlled QSVD algorithm, to determine whether the input singular vector corresponds to an singular value less than (or greater than) a certain threshold. Based on the outcome of controlled QSVD, the standard inversion algorithm is applied. These two steps correspond to sub-algorithms $\mathcal{A}_j$ of the VTAA framework.

In prior works such as Refs. [45, 46, 35], gapped phase estimation (GPE) was used to implement this. GPE requires an additional register of $\mathcal{O}(\log(\kappa)\log(1/\delta))$ qubits to store the estimated phases. For the whole VTAA procedure, $\log\kappa$ such registers are needed. As a result, substituting GPE with QSVD, we save $\mathcal{O}\big(\log^2(\kappa)\log(1/\delta)\big)$ qubits.

3.4 Negative Powers of Matrices using QSVT

We consider the problem: given an approximate block-encoding of a matrix A, we need to prepare a block-encoding of A^{-c}, where $c \in (0,1)$. This procedure will be used to develop algorithms for ℓ_2-regularized versions of GLS. We will directly use the results of [42].

Lemma 3.4.1 (Polynomial approximations of negative power functions, [63], Corollary 67)**.** *Let $\varepsilon, \delta \in (0, \frac{1}{2}], c > 0$ and let $f(x) := \frac{\delta^c}{2} x^{-c}$, then there exist even/odd polynomials $P_{c,\varepsilon,\delta}, P'_{c,\varepsilon,\delta} \in \mathbb{R}[x]$ such that $\|P_{c,\varepsilon,\delta} - f\|_{[\delta,1]} \leq \varepsilon$, $\|P_{c,\varepsilon,\delta}\|_{[-1,1]} \leq 1$ and $\left\|P'_{c,\varepsilon,\delta} - f\right\|_{[\delta,1]} \leq \varepsilon$, $\left\|P'_{c,\varepsilon,\delta}\right\|_{[-1,1]} \leq 1$. Moreover the degree of the polynomials are $\mathcal{O}\Big(\frac{\max(1,c)}{\delta} \log\left(\frac{1}{\varepsilon}\right)\Big)$.*

Theorem 3.4.2 (Negative fractional powers of a normalized matrix using QSVT)**.** *Let $c \in (0,1)$ be some constant and $\delta \in (0,1]$ Let A be a normalized matrix with non-zero singular values in the range $[1/\kappa, 1]$. Let U_A be a (α, a, ε)-block-encoding of a matrix A, implemented in time T_A such that $\alpha \geq 2$ and*

$$\varepsilon = o\left(\frac{\delta}{\kappa^{c+1}\log(\kappa/\delta)} \right)$$

Then we can construct a $(2\kappa^c, a+1, \delta)$-block-encoding of A^{-c} at a cost of

$$\mathcal{O}\Big(\alpha\kappa \log\left(\frac{\kappa}{\delta}\right) \; T_A \Big).$$

Proof. From Lemma 3.4.1, using $\Delta := \frac{1}{\kappa\alpha}$ and an appropriate $\varphi \in (0, \frac{1}{2}]$, we get an even QSP polynomial $P := P_{c,\varphi,\Delta}$ which is φ-close to $f(x) := \frac{1}{2\kappa^c\alpha^c x^c}$, and has degree n such that $n = \mathcal{O}\Big(\alpha\kappa \log\left(\frac{1}{\varphi}\right)\Big)$. Therefore

$$\|f(A/\alpha) - P(A/\alpha)\| \leq \varphi.$$

Using Theorem 2.5.4 we can construct U_P, a $(1, a+1, \gamma)$-block-encoding of $P(A/\alpha)$, given that $\varepsilon \leq \frac{\alpha\gamma}{2n}$. Then from triangle inequality it follows that it is a $(1, a+1,\ \varphi+\gamma)$-block-encoding of $f(A/\alpha)$. And because $f(A/\alpha) = \frac{A^{-c}}{2\kappa^c}$, U_P can be re-interpreted as a $(2\kappa^c, a+1, 2\kappa^c(\varphi+\gamma))$-block-encoding of A^{-c}. We therefore choose $\varphi = \gamma = \frac{\delta}{4\kappa^c}$, and choose ε as

$$\varepsilon = o\left(\alpha\frac{\delta}{4\kappa^c}\frac{1}{\alpha\kappa\log(4\kappa^c/\delta)}\right) = o\left(\frac{\delta}{\kappa^{c+1}\log(\kappa/\delta)}\right)$$

□

Having discussed the necessary algorithmic primitives, we are now in a position to design quantum algorithms for linear regression with general ℓ_2-regularization. We will first deal with ordinary least squares in chapter 4, followed by weighted and generalized least squares with regularization in chapter 5.

Chapter 4

Quantum Linear Systems Algorithms

Quantum linear systems (QLS) are perhaps one of the most explored topics in quantum algorithms literature. This began with the HHL algorithm (named so after the authors' initials), and has seen significant interest ever since. The vanilla problem inputs a quantum encoding of a matrix $A \in \mathbb{R}^{m\times n}$ and a quantum state proportional to a vector $b \in \mathbb{R}^m$ and requires that we output a quantum state proportional to the solution x of the equation $Ax = b$. This can be further generalized, and regularized. These problems have been explored in various quantum input models, and under various constraints.

To put things in perspective, we begin this chapter by introducing the HHL algorithm, followed by an adiabatic quantum algorithm for solving QLS by Lin and Tong [69]. Then we develop a variable-time quantum linear systems algorithm (QLSA) using the improved VTAA technique and QSVT. Our algorithm gives a linear dependence in the condition number, and unlike the previous work by Chakraborty et al. [35], uses QSVT instead of Hamiltonian simulation, and requires fewer ancillary qubits. We will then use this algorithm we develop in the next chapter to develop an algorithms for the regularized ordinary, weighted and generalized least squares problems.

4.1 Harrow-Hassidim-Lloyd (HHL) Algorithm

This algorithm deals with the quantum version of the following problem. Given a Hermitian $n \times n$ matrix A and a unit vector b, find the vector x satisfying $Ax = b$. The algorithm assumes access to b as a quantum state $|b\rangle = \sum_i b_i |i\rangle$, which can be prepared by some controlled rotations. Suppose $A = \sum_j \lambda_j |v_j\rangle \langle v_j| \, ; \forall j, \lambda_j \in [-1, \frac{1}{\kappa}] \cup [\frac{1}{\kappa}, 1]$, where κ is the condition number of A. The quantum version of the problem requires us to prepare a state $|\tilde{x}\rangle$ such that

$$\| |\widetilde{x}\rangle - |x\rangle \| \leq \varepsilon.$$

Here

$$|x\rangle = \frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|}$$

is the true solution to the linear system. The HHL algorithm uses phase estimation and controlled rotations for the task. First we apply phase estimation on A along with $|b\rangle$ using controlled Hamiltonian simulation to get

$$\sum_j c_j |v_j\rangle |\widetilde{\lambda}_j\rangle_w |0\rangle_a ,$$

where $\{\widetilde{\lambda}_j\}$ are the approximate eigenvalues of A. Next we want to transfer the contents of the register w to the phase of register a. This can be easily done by applying the arcsin function to the contents of the register and then using Theorem A.1.1. The transformation looks like

$$\sum_j c_j |v_j\rangle |\widetilde{\lambda}_j\rangle |0\rangle \longmapsto \sum_j c_j |v_j\rangle |\frac{1}{\pi} \arcsin\left(\frac{c}{\widetilde{\lambda}_j}\right)\rangle |0\rangle \tag{4.1.1}$$

$$\xrightarrow[\text{Theorem A.1.1}]{} \sum_j c_j |v_j\rangle |\frac{1}{\pi} \arcsin\left(\frac{c}{\widetilde{\lambda}_j}\right)\rangle \left[\frac{c}{\widetilde{\lambda}_j}|0\rangle + \sqrt{1-\frac{c^2}{\widetilde{\lambda}_j^2}}|1\rangle\right] \tag{4.1.2}$$

Uncomputing gives us the desired state

$$c\sum_j \frac{c_j}{\widetilde{\lambda}_j} |v_j\rangle |0\rangle |0\rangle + |\perp\rangle ,$$

where $|\perp\rangle$ has no component over $\mathcal{I} \otimes |0\rangle \langle 0|$. Therefore, post-selecting on $|0\rangle$ in the last register leaves $|\widetilde{x}\rangle$ in the first register.

4.2 Adiabatic Quantum Algorithm for Solving Linear Systems

Lin and Tong [69] gave an adiabatic quantum algorithm to solve linear systems. It allows us to prepare a target eigenstate of a given Hamiltonian if we have an initial state with a non-trivial overlap with the target eigenstate and there is a reasonable lower bound known on the spectral gap. They also discretise the algorithm for d-sparse matrices using a Hamiltonian simulation subroutine.

Assume access to a d-sparse matrix $A \in \mathbb{C}^{N\times N}$ via oracles $O_{A,1}, O_{A,2}$ as

$$O_{A,1} |j\rangle |l\rangle = |j\rangle |v(j,l)\rangle , \tag{4.2.1}$$
$$O_{A,2} |j\rangle |k\rangle |z\rangle = |j\rangle |k\rangle |A_{jk} \oplus z\rangle , \tag{4.2.2}$$

where $j, k, l, z \in [N]$ and $v(j,l)$ is the row index of the l-th non-zero element in the j-th column. Also assume access to an oracle O_B such that

$$O_B |0\rangle = |b\rangle , \tag{4.2.3}$$

where $|b\rangle \in \mathbb{C}^N$. This allows us to prepare a $(d, n+2, 0)$-block-encoding of A. The task is to prepare the quantum state

$$|x\rangle = \frac{A^{-1} |b\rangle}{||A^{-1} |b\rangle ||},$$

where $|x\rangle$ is the quantum state proportional to the solution of the linear system $Ax = b$.

They assume that the singular values of A are contained in $\left[\frac{1}{\kappa}, 1\right]$, where κ is the condition number of A. The algorithm goes as follows.

Define

$$Q_b = \mathcal{I} - |b\rangle \langle b| \tag{4.2.4}$$

The initial Hamiltonian is

$$H_0 := \begin{pmatrix} 0 & Q_b \\ Q_b & 0 \end{pmatrix} = \sigma_x \otimes Q_b. \tag{4.2.5}$$

The final Hamiltonian is

$$H_1 := \begin{pmatrix} 0 & AQ_b \\ Q_b A & 0 \end{pmatrix} = |0\rangle \langle 1| \otimes AQ_b + |1\rangle \langle 0| \otimes Q_b A. \tag{4.2.6}$$

Note that the null space of H_1 is spanned by $|0\rangle |x\rangle$ and $|1\rangle |b\rangle$. The rest of the spectrum is separated from 0 by a gap of $\frac{1}{\kappa}$. Then the adiabatic evolution is given by

$$H(f) = (1-f)H_0 + fH_1. \tag{4.2.7}$$

Using the vanilla (linear) schedule $f(s) = s$ gives a time complexity of $\mathcal{O}(\kappa^2/\epsilon)$. Using the AQC(p) schedule (for $p \in (1,2)$), given by

$$f(s) = \frac{\kappa}{\kappa - 1}\left(1 - \left(1 + s(\kappa^{p-1} - 1)\right)^{\frac{1}{1-p}}\right), \tag{4.2.8}$$

the time complexity can be reduced to $\mathcal{O}(\kappa/\epsilon)$, which is optimal in κ but not in ϵ. Using the AQC(exp) schedule, the time complexity can be further reduced to $\mathcal{O}\left(\kappa \log^2(\kappa) \log^4\left(\frac{\log \kappa}{\epsilon}\right)\right)$, which is near-optimal in both κ and ϵ, although with a high constant overhead (numerically shown in their paper.) To overcome this they developed quantum eigenstate filtering, which they use to accelerate AQC(p) and reduce the query complexity in precision to $\log\left(\frac{1}{\varepsilon}\right)$.

4.3 Variable Time Quantum Linear Systems Algorithm using QSVT

Matrix inversion by QSVT applies a polynomial approximation of $f(x) = 1/x$, satisfying the constraints laid out in Theorem 2.5.1. Here, we make use of the result of [44] to implement A^+. We adapt their result to the scenario where we have an approximate block-encoding of A as input. Finally, we convert this to a variable stopping time quantum algorithm and apply VTAA to obtain a linear dependence on the condition number of A.

Lemma 4.3.1 (Matrix Inversion polynomial (Appendix C of [44]))**.** *Given $\kappa \geq 1, \varepsilon \in \mathbb{R}^+$, there exists an odd QSP polynomial $P^{MI}_{\varepsilon,\kappa}$ of degree $\mathcal{O}(\kappa \log(\kappa/\varepsilon))$, which is an $\frac{\varepsilon}{2\kappa}$ approximation of the function $f(x) = \frac{1}{2\kappa x}$ in the range $\mathcal{D} := [-1, -\frac{1}{\kappa}] \cup [\frac{1}{\kappa}, 1]$. Also in this range $P^{MI}_{\varepsilon,\kappa}$ is bounded from above by 1, i.e. $\forall x \in \mathcal{D} : \left|P^{MI}_{\varepsilon,\kappa}(x)\right| \leq 1$.*

Theorem 4.3.2 (Inverting Normalized Matrices using QSVT)**.** *Let A be a normalized matrix with non-zero singular values in the range $[1/\kappa_A, 1]$ for some $\kappa_A \geq 1$. Let $\delta \in (0, 1]$. For some $\varepsilon = o\left(\frac{\delta}{\kappa_A^2 \log(\kappa_A/\delta)}\right)$ and $\alpha \geq 2$, let U_A be an (α, a, ε)-block-encoding of A, implemented in time T_A. Then we can implement a $(2\kappa_A, a+1, \delta)$-block-encoding of A^+ at a cost of*

$$\mathcal{O}\left(\kappa_A \alpha \log\left(\frac{\kappa_A}{\delta}\right) T_A\right).$$

Proof. We use the matrix inversion polynomial defined in Lemma 4.3.1, $P := P^{MI}_{\phi,\kappa}$ for this task, with $\kappa = \kappa_A \alpha$ and an appropriate ϕ. This has a degree of $n := \mathcal{O}(\kappa_A \alpha \log(\kappa_A \alpha/\phi))$. We invoke Theorem 2.5.4 to apply QSVT using the polynomial P above, block-encoding U_A, and an appropriate error parameter γ such that $\varepsilon \leq \alpha\gamma/2n$, to get the unitary U, a $(1, a+1, \gamma)$-block-encoding of $P(A/\alpha)$. As P is a $(\phi/2\kappa)$-approximation of $f(x) := 1/2\kappa x$, we have

$$\|f(A/\alpha) - P(A/\alpha)\| \leq \frac{\phi}{2\kappa},$$

which implies U is a $(1, a+1, \gamma + \phi/2\kappa)$-block-encoding of $f(A/\alpha)$. And because $f(A/\alpha) = \frac{\alpha A^+}{2\kappa} = A^+/2\kappa_A$, we can re-interpret U as a $(2\kappa_A, a+1, 2\kappa_A\gamma + \phi/\alpha)$-block-encoding of A^+. Choosing $2\kappa_A\gamma = \phi/\alpha = \delta/2$, the final block-encoding has an error of δ. This gives us $\phi = \alpha\delta/2$ and $\gamma = \delta/4\kappa_A$, and

$$\varepsilon \leq \frac{\alpha\gamma}{2n} = \frac{\alpha\delta}{8\kappa_A n} = \mathcal{O}\left(\frac{\delta}{\kappa_A^2 \log(\kappa_A/\delta)}\right)$$

□

Next, we design a map $W(\gamma, \delta)$ that uses QSVT to invert the singular values of a matrix if they belong to a particular domain. This helps us recast the usual matrix inversion algorithm as a variable-stopping-time algorithm and will be a key subroutine for boosting the success probability of this algorithm using VTAA. This procedure was also used in Refs. [46, 35] for the quantum linear systems algorithms.

Theorem 4.3.3 (Efficient inversion of block-encoded matrix)**.** *Let A be a normalized matrix with non-zero singular values in the range $[1/\kappa, 1]$, for some $\kappa \geq 1$. Let $\delta \in (0, 1]$; $0 < \gamma \leq 1$. Let U_A be an (α, a, ε)-block-encoding of A implemented in time T_A, such that $\alpha \geq 2$ and $\varepsilon = o\left(\frac{\delta\gamma^2}{\log(\frac{1}{\delta\gamma})}\right)$. Then for any quantum state $|b\rangle$ that is spanned by the left singular vectors of A corresponding to the singular values in the range $[\gamma, 1]$, there exists a unitary $W(\gamma, \delta)$ that implements*

$$W(\gamma, \delta) : |0\rangle_F |0\rangle_Q |b\rangle_I \mapsto \frac{1}{a_{\max}} |1\rangle_F |0\rangle_Q f(A) |b\rangle_I + |0\rangle_F |\perp\rangle_{QI} \tag{4.3.1}$$

where $a_{\max} = \mathcal{O}(\kappa_A)$ is a constant independent of γ, $|\perp\rangle_{QI}$ is an unnormalized quantum state orthogonal to $|0\rangle_Q$ and $\|f(A)|b\rangle - A^+|b\rangle\| \leq \delta$. Here F is a 1-qubit flag register, Q is an α-qubit ancilla register, and I is the $\log N$-qubit input register. This unitary has a cost

$$\mathcal{O}\left(\frac{\alpha}{\gamma} \log\left(\frac{1}{\gamma\delta}\right) T_A\right) \tag{4.3.2}$$

Proof. Since we only need to invert the singular values in a particular range, we can use the procedure in Theorem 4.3.2 with κ_A modified to the restricted range. That gives us the description of a quantum circuit $\widetilde{W}(\gamma, \delta)$ that can implement the following map

$$\widetilde{W}(\gamma, \delta) : |b\rangle_I |0\rangle_Q \mapsto \frac{\gamma}{2} f(A) |b\rangle_I |0\rangle_Q + |\perp\rangle_{QI},$$

where $|\perp\rangle$ is an unnomalized state with no component along $|0\rangle_Q$. This has the same cost as Equation 4.3.2. Here $\|f(A)|\psi\rangle - A^+|\psi\rangle\| \leq \delta$ whenever $|\psi\rangle$ is a unit vector in the span of the singular vectors of A corresponding to the singular values in $[\gamma, 1]$. This follows from the sub-multiplicativity property of the matrix-vector product.

Next, we must transform the amplitude of the good part of the state to $\Theta(\kappa)$, independent of γ. To achieve this, we will have to flag it with an ancillary qubit to use a controlled rotation to modify the amplitude. Thus we add a single qubit $|0\rangle_F$ register and flip this register controlled on register Q being in the state $|0\rangle$ (the good part). This gives us the transformation

$$\widetilde{W}'(\gamma,\delta) : |0\rangle_F |b\rangle_I |0\rangle_Q \mapsto \frac{\gamma}{2} |1\rangle_F f(A)|b\rangle_I |0\rangle_Q + |0\rangle_F |\perp\rangle_{QI}$$

Then we use a controlled rotation to replace the amplitude $\gamma/2$ with some constant $a_{\max}^{-1}$ which is independent of γ, which is achieved by introducing the relevant phase to the flag space

$$|1\rangle_F \mapsto \frac{2}{\gamma a_{\max}} |1\rangle_F + \sqrt{1 - \frac{4}{\gamma^2 a_{\max}^2}} |0\rangle_F .$$

This gives us the desired $W(\gamma,\delta)$ as in Equation 4.3.1. □

Given such a unitary $W(\gamma,\delta)$, Ref. [35] laid out a procedure for a variable time quantum algorithm $\mathcal{A}$ that takes as input the block encoding of an $N \times d$ matrix A, and a state preparation procedure $U_b : |0\rangle^{\otimes n} \mapsto |b\rangle$, and outputs a quantum state that is a bounded distance away from $A^+|b\rangle / \|A^+|b\rangle\|$. In order to determine the branches of the algorithm on which to apply VTAA at a particular iteration, [46, 35, 45] use the technique of gapped phase estimation, which given a unitary U, a threshold ϕ and one of its eigenstate $|\lambda\rangle$, decides if the corresponding eigenvalue is a bounded distance below the threshold, or a bounded distance above it. In this work, we replace gapped phase estimation with the QSVD algorithm (Theorem 3.3.2) which can be applied directly to any block-encoded (not necessarily Hermitian) matrix A, and allows for saving on $\mathcal{O}\big(\log^2(\kappa/\delta)\big)$ qubits.

The Variable time Algorithm: This algorithm will be a sequence of m sub-algorithms $\mathcal{A} = \mathcal{A}_m \cdot \mathcal{A}_{m-1} \cdot \ldots \mathcal{A}_1$, where $m = \lceil \log \kappa \rceil + 1$. The overall algorithm acts on the following registers:

- m single qubit clock registers $C_i : i \in [m]$.
- An input register I, initialized to $|0\rangle^{\otimes s}$.
- Ancillary register space Q for the block encoding of A, initialized to $|0\rangle^{\otimes a}$.
- A single qubit flag register $|0\rangle_F$ used to flag success of the algorithm.

Once we have prepared the above state space, we use the state preparation procedure to prepare the state $|b\rangle$. Now we can define how each $\mathcal{A}_j$ acts on the state space. Let $\varepsilon' = \frac{\delta}{a_{\max} m}$. The action of $\mathcal{A}_j$ can be broken down into two parts:

1. If $C_{j-1} \ldots C_1$ is in state $|0\rangle^{\otimes(j-1)}$, apply QSVD$(2^{-j}, \varepsilon')$, (Theorem 3.3.2) to the state $|b\rangle$. The output is to be written to the clock register C_j.
2. If the state of C_j is now $|1\rangle$, apply $W(2^{-j}, \varepsilon')$ to $I \otimes F \otimes Q$.

Additionally, we would need algorithms $\mathcal{A}' = \mathcal{A}'_m \cdots \mathcal{A}'_1$ which are similar to $\mathcal{A}$, except that in Step 2, it implements W' which sets the flag register to 1. That is,

$$W' |b\rangle_I |0\rangle_F |0\rangle_Q = |b\rangle_I |1\rangle_F |0\rangle_Q .$$

Now we are in a position to define the variable time quantum linear systems algorithm using QSVT.

Theorem 4.3.4 (Variable Time Quantum Linear Systems Algorithm Using QSVT)**.** *Let $\varepsilon, \delta > 0$. Let A is a normalized $N \times d$ matrix such that its non-zero singular values lie in $[1/\kappa, 1]$. Suppose that for*

$$\varepsilon = o\left(\frac{\delta}{\kappa^3 \log^2\left(\frac{\kappa}{\delta}\right)} \right),$$

we have access to U_A which is an (α, a, ε)-block-encoding of A, implemented with cost T_A. Let $|b\rangle$ be a state vector which is spanned by the left singular vectors of A. Suppose there exists a procedure to prepare the state

$|b\rangle$ in cost T_b. Then there exists a variable time quantum algorithm that outputs a state that is δ-close $\frac{A^+|b\rangle}{\|A^+|b\rangle\|}$ at a cost of

$$\mathcal{O}\left(\kappa \log \kappa \left(\alpha T_A \log\left(\frac{\kappa}{\delta}\right) + T_b\right)\right) \tag{4.3.3}$$

using $\mathcal{O}(\log(\kappa))$ additional qubits.

Proof. The correctness of the algorithm is similar to that of Refs. [46, 35], except here, we use QSVD instead of gapped phase estimation. According to Lemma 2.6.2, we need $T_{\max}$ (the maximum time any of the sub-algorithms $\mathcal{A}_j$ take), $\|T\|_2^2$ (the ℓ_2-averaged stopping time of the sub-algorithms), and $\sqrt{p_{\text{succ}}}$ (the square root of the success probability.) Now each sub-algorithm consists of two steps, implementing QEVD with precision 2^{-j} and error ε', followed by $W(2^{-j}, \varepsilon')$. From Theorem 3.3.2, the first step costs

$$\mathcal{O}\left(\alpha T_A 2^j \log\left(\frac{1}{\varepsilon'}\right)\right),$$

and the cost of implementing $W(2^{-j}, \varepsilon')$ is as described in Equation 4.3.2. Thus the overall cost of $\mathcal{A}_j$, which is the sum of these two costs, turns out to be

$$\mathcal{O}\left(\alpha T_A 2^j \log\left(\frac{2^j}{\varepsilon'}\right)\right) \tag{4.3.4}$$

Note that the time t_j required to implement $\mathcal{A}_j \ldots \mathcal{A}_1$ is also the same as Equation 4.3.4. Also,

$$\begin{aligned}
T_{\max} &= \max_j \; t_j \\
&= \max_j \; \mathcal{O}\left(\alpha T_A 2^j \log\left(\frac{2^j}{\varepsilon'}\right)\right) \\
&= \mathcal{O}\left(\alpha T_A \kappa \log\left(\frac{\kappa}{\varepsilon'}\right)\right) \\
&= \mathcal{O}\left(\alpha T_A \kappa \log\left(\frac{\kappa \log(\kappa)}{\delta}\right)\right).
\end{aligned}$$

The $\|T\|_2^2$ is dependent on the probability that $\mathcal{A}$ stops at the j^{th} step. This is given by $p_j = \left\|\Pi_{C_j} \mathcal{A}_j \ldots \mathcal{A}_1 |\psi\rangle_I |0\rangle_{CFPQ}\right\|^2$, where Π_{C_j} is the projector on $|1\rangle_{C_j}$, the j^{th} clock register. From this, $\|T\|_2^2$ can be calculated as

$$\begin{aligned}
\|T\|_2^2 &= \sum_j p_j t_j^2 \\
&= \sum_j \left\|\Pi_{C_j} \mathcal{A}_j \ldots \mathcal{A}_1 |\psi\rangle_I |0\rangle_{CFPQ}\right\|^2 t_j^2 \\
&= \sum_k |c_k|^2 \sum_j \left(\left\|\Pi_{C_j} \mathcal{A}_j \ldots \mathcal{A}_1 |v_k\rangle_I |0\rangle_{CFPQ}\right\|^2 t_j^2\right) \\
&= \mathcal{O}\left(\alpha^2 T_A^2 \sum_k \log^2\left(\frac{1}{\sigma_k \varepsilon'}\right) \frac{|c_k|^2}{\sigma_k^2}\right)
\end{aligned}$$

Therefore

$$\|T\|_2 = \mathcal{O}\left(\alpha T_A \log\left(\frac{\kappa \log \kappa}{\delta}\right) \sqrt{\sum_k \frac{|c_k|^2}{\sigma_k^2}}\right). \tag{4.3.5}$$

Next we calculate the success probability.

$$\begin{aligned}
\sqrt{p_{\text{succ}}} &= \left\|\Pi_F \frac{A^{-1}}{\alpha_{\max}} |b\rangle_I |\phi\rangle_{CFPQ}\right\| + \mathcal{O}(m\varepsilon') \\
&= \frac{1}{\alpha_{\max}} \sqrt{\sum_j \frac{|c_j|^2}{\sigma_j^2}} + \mathcal{O}\left(\frac{\delta}{\alpha_{\max}}\right)
\end{aligned}$$

$$= \Omega\left(\frac{1}{\kappa}\sqrt{\sum_j \frac{|c_j|^2}{\sigma_j^2}}\right)$$

Given these, we can use Lemma 2.6.2 to write the final complexity of matrix inversion with VTAA:

$$T_{\max} + T_b + \frac{(\|T\|_2 + T_b)\log(T'_{\max})}{\sqrt{p_{\text{succ}}}} = \mathcal{O}\Big(\kappa \log\kappa \left(\alpha T_A \log\left(\frac{\kappa}{\delta}\right) + T_b\right)\Big)$$

The upper bound on the precision required for the input block-encoding, ε, can be calculated from the bounds on the precisions for $W(\kappa, \varepsilon')$ (Theorem 4.3.3) and QSVD(κ, ε') (Theorem 3.3.2) as follows:

$$\varepsilon = o\left(\min\left(\frac{\varepsilon'}{\kappa^2 \log\left(\frac{\kappa}{\varepsilon'}\right)}, \frac{\varepsilon'}{\kappa \log\left(\frac{1}{\varepsilon'}\right)}\right)\right) = o\left(\frac{\varepsilon'}{\kappa^2 \log\left(\frac{\kappa}{\varepsilon'}\right)}\right) = o\left(\frac{\delta}{\kappa^3 \log^2\left(\frac{\kappa}{\delta}\right)}\right)$$

□

The overall complexity is better by a log factor and requires $\mathcal{O}\big(\log^2(\kappa/\delta)\big)$ fewer additional qubits as compared to the variable time algorithms in Refs. [46, 35].

Chapter 5

Regularized Quantum Regression

5.1 Quantum Ordinary Least Squares with General Tikhonov Regularization

In this section, we derive the main results of our paper, namely quantum algorithms for quantum ordinary least squares (OLS), quantum weighted least squares (WLS) and quantum generalized least squares (GLS) with ℓ_2-regularization.

5.1.1 Quantum Ordinary Least Squares

Given N data points $\{a_i, b_i\}_{i=1}^N$ such that $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$, the objective of linear regression is to find $x \in \mathbb{R}^d$ that minimizes the loss function

$$\mathcal{L}_O = \sum_{j=1}^{N} (x^T a_i - b_i)^2. \tag{5.1.1}$$

Consider the $N \times d$ matrix A (known as the data matrix) such that the i^{th} row of A is the vector a_i transposed and the column vector $b = (b_1 \cdots b_N)^T$. Then, the solution to the OLS problem is given by $x = (A^T A)^{-1} A^T b = A^+ b$.

For the ℓ_2-regularized version of the OLS problem, a penalty term is added to its objective function. This has the effect of shrinking the singular values of A which helps overcome problems such as rank deficiency and *overfitting* for the OLS problem. The loss function to be minimized is of the form

$$\|Ax - b\|_2^2 + \|Lx\|_2^2, \tag{5.1.2}$$

where L is the $N \times d$ penalty matrix and $\lambda > 0$ is the optimal regularizing parameter. The solution $x \in \mathbb{R}^d$ satisfies

$$x = (A^T A + \lambda L^T L)^{-1} A^T b. \tag{5.1.3}$$

Therefore, for quantum ordinary least squares with general ℓ_2-regularization, we assume that we have access to approximate block-encodings of the data matrix A, L and a procedure to prepare the quantum state $|b\rangle = \sum_{j=1}^N b_j |j\rangle / \|b\|$. Our algorithm outputs a quantum state that is close to

$$|x\rangle = \frac{(A^T A + \lambda L^T L)^{-1} A^T |b\rangle}{\|(A^T A + \lambda L^T L)^{-1} A^T |b\rangle\|}. \tag{5.1.4}$$

In order to implement a quantum algorithm that implements this, a straightforward approach would be the following: We first construct block-encodings of $A^T A$ and $L^T L$, given block encodings of A and L, respectively (Using Lemma 3.2.4). We could then implement a block-encoding of $A^T A + \lambda L^T L$ using these block encodings (By Lemma 3.2.2). On the other hand, we could also prepare a quantum state proportional to $A^T |b\rangle$ by using the block-encoding for A and the unitary preparing $|b\rangle$. Finally, using the block encoding of $A^T A + \lambda L^T L$, we could implement a block-encoding of $(A^T A + \lambda L^T L)^{-1}$ (using Theorem 4.3.2) and apply it to the state $A^T |b\rangle$. Although this procedure would output a quantum state close to $|x\rangle$, it is not efficient. It is easy to see that the inverse of $A^T A + \lambda L^T L$, would be implemented with a complexity that has a quadratic dependence on the condition numbers of A and L. This would be undesirable as it would perform worse than the unregularized quantum least squares algorithm, where one is able to implement A^+ directly. However, it is possible to design a quantum algorithm that performs significantly better than this.

The first observation is that it is possible to recast this problem as finding the pseudoinverse of some augmented matrix. Given the data matrix $A \in \mathbb{R}^{N\times d}$, the regularizing matrix $L \in \mathbb{R}^{N\times d}$, let us define the following augmented matrix

$$A_L := \begin{pmatrix} A & 0 \\ \sqrt{\lambda}L & 0 \end{pmatrix}. \tag{5.1.5}$$

It is easy to see that the top left block of $A_L^+ = (A^TA + \lambda L^TL)^{-1}A^T$, which is the required linear transformation to be applied to b. Consequently, our strategy would be to implement a block-encoding of A_L, given block-encodings of A and L. Following this, we use matrix inversion by QSVT to implement $A_L^+ |b\rangle |0\rangle$. The first register is left in the quantum state given in Equation 5.1.4.

From this, it is clear that the complexity of our quantum algorithm would depend on the effective condition number of the augmented matrix A_L. In this regard, we shall assume that the penalty matrix L is a *good regularizer*. That is, L is chosen such that it does not have zero singular values (positive definite). This is a fair assumption as if L has only non-zero singular values, the minimum singular value of A_L is guaranteed to be lower bounded by the minimum singular value of L. This ensures that the effective condition number of A_L depends on κ_L, even when the data matrix A has zero singular values and A^TA is not invertible. Consequently, this also guarantees that regularized least squares provide an advantage over their unregularized counterparts.

Next, we obtain bounds on the effective condition number of the augmented matrix A_L for a good regularizer L via the following lemma:

Lemma 5.1.1 (Condition number and Spectral Norm of A_L)**.** *Let the data matrix A and the positive definite penalty matrix L have spectral norms $\|A\|$ and $\|L\|$, respectively. Furthermore, suppose their effective condition numbers be upper bounded by κ_A and κ_L. Then the ratio between the maximum and minimum (non-zero) singular value of A_L is upper bounded by*

$$\kappa = \kappa_L \left(1 + \frac{\|A\|}{\sqrt{\lambda}\|L\|}\right)$$

We can also bound the spectral norm as

$$\|A_L\| = \Theta\left(\|A\| + \sqrt{\lambda}\|L\|\right)$$

Proof. To bound the spectral norm and condition number of A_L, consider the eigenvalues of the following matrix:

$$A_L^TA_L = \begin{pmatrix} A^TA + \lambda L^TL & 0 \\ 0 & 0 \end{pmatrix}$$

This implies that the non-zero eigenvalues of $A_L^TA_L$ are the same as those of $A^TA + \lambda L^TL$. Therefore, using triangle inequality, the spectral norm of A_L can be upper-bounded as follows:

$$\|A_L\| = \sqrt{\left\|A_L^TA_L\right\|} = \sqrt{\|A^TA + \lambda L^TL\|} \leq \sqrt{\|A^TA\| + \lambda\|L^TL\|} = \sqrt{\|A\|^2 + \lambda\|L\|^2} \leq \|A\| + \sqrt{\lambda}\|L\|$$

Similarly $\|A_L\| \geq \|A\|$ and $\|A_L\| \geq \sqrt{\lambda}\|L\|$, which effectively gives the tight bound for $\|A_L\|$.

As L^TL is positive definite, we have that its minimum singular value is $\sigma_{\min(L)} = \|L\|/\kappa_L$. And we also know that A^TA is positive semidefinite, so by Weyl's inequality, the minimum singular value of A_L is lower bounded by

$$\sigma_{\min}(A_L) \geq \sqrt{\sigma_{\min}(A)^2 + \lambda\sigma_{\min}(L)^2} \geq \sqrt{\lambda\frac{\|L\|^2}{\kappa_L^2}} = \sqrt{\lambda}\frac{\|L\|}{\kappa_L}$$

Thus,

$$\frac{\sigma_{\max}(A_L)}{\sigma_{\min}(A_L)} \leq \kappa = \kappa_L\left(1 + \frac{\|A\|}{\sqrt{\lambda}\|L\|}\right)$$

□

In the theorems and lemmas for regularized quantum linear regression and its variants that we develop in this section, we consider that L is a *good regularizer* in order to provide a simple expression for κ. However, this is without loss of generality. When L is not a good regularizer, the expressions for the respective complexities will remain unaltered, except that κ would now correspond to the condition number of the augmented matrix.

Now it might be possible that $|b\rangle$ does not belong to the row space of $(A^TA+\lambda L^TL)^{-1}A^T$ which is equivalent to saying $|b\rangle|0\rangle$ may not lie in $\text{row}(A_L^+)$. However, it is reasonable to expect that the initial hypothesis of the underlying model being close to linear is correct. That is, we expect $|b\rangle$ to have a good overlap with $\text{row}\left(A_L^+\right) = \text{col}\left(A_L\right)$. The quantity that quantifies how far the model is from being linear is the so called *normalized residual sum of squares*. For ℓ_2-regularized ordinary least squares, this is given by

$$\mathcal{S}_O = \frac{\left\|\left(I - \Pi_{\text{col}(A_L)}\right)|b\rangle|0\rangle\right\|^2}{\||b\rangle\|^2} = 1 - \left\|\Pi_{\text{col}(A_L)}|b\rangle|0\rangle\right\|^2. \tag{5.1.6}$$

If the underlying data can indeed be fit by a linear function, $\mathcal{S}_O$ will be low. Subsequently, we assume that $\mathcal{S}_O = 1 - \left\|\Pi_{\text{col}(A_L)}|b\rangle|0\rangle\right\|^2 \leq \gamma < 1/2$. This in turn implies that $\left\|\Pi_{\text{col}(A_L)}|b\rangle|0\rangle\right\|^2 = \Omega(1)$, implying that the data can be reasonably fit by a linear model.[i]

Now we are in a position to present our quantum algorithm for the quantum least squares problem with general ℓ_2-regularization. We also present an improved quantum algorithm for the closely related quantum ridge regression, which is a special case of the former.

Theorem 5.1.2 (Quantum Ordinary Least Squares with General ℓ_2-Regularization)**.** *Let $A, L \in \mathbb{R}^{N\times d}$ be the data and penalty matrices with effective condition numbers κ_A and κ_L respectively, and $\lambda \in \mathbb{R}^+$ be the regression parameter. Let U_A be a $(\alpha_A, a_A, \varepsilon_A)$-block-encoding of A implemented in time T_A and U_L be a $(\alpha_L, a_L, \varepsilon_L)$-block-encoding of L implemented in time T_L. Furthermore, suppose U_b be a unitary that prepares $|b\rangle$ in time T_b and define*

$$\kappa = \kappa_L\left(1 + \frac{\|A\|}{\sqrt{\lambda}\|L\|}\right)$$

Then for any $\delta \in (0,1)$ such that

$$\varepsilon_A, \sqrt{\lambda}\varepsilon_L = o\left(\frac{\delta}{\kappa^3 \log^2\left(\frac{\kappa}{\delta}\right)}\right) \tag{5.1.7}$$

we can prepare a state that is δ-close to

$$\frac{\left(A^TA + \lambda L^TL\right)^{-1}A^T|b\rangle}{\left\|\left(A^TA + \lambda L^TL\right)^{-1}A^T|b\rangle\right\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa\log\kappa\left(\left(\frac{\alpha_A + \sqrt{\lambda}\alpha_L}{\|A\| + \sqrt{\lambda}\|L\|}\right)\log\left(\frac{\kappa}{\delta}\right)(T_A + T_L) + T_b\right)\right) \tag{5.1.8}$$

using only $\mathcal{O}(\log\kappa)$ additional qubits.

Proof. We invoke Lemma 3.2.7, to obtain a unitary U, which is a $(\alpha_A + \sqrt{\lambda}\alpha_L, \max(a_A, a_L) + 2, \varepsilon_A + \sqrt{\lambda}\varepsilon_L)$-block-encoding of the matrix A_L, implemented at a cost of $\mathcal{O}(T_A + T_L)$. Note that in Lemma 3.2.7, A and L are considered to be s-qubit operators. For $N \times d$ matrices, such that $N, d \leq 2^s$, we can pad them with zero entries. Padding A and L with zeros may result in the augmented matrix A_L having some zero rows between A and L. However, this is also not an issue as we are only interested in the top left block of A_L^+ which remains unaffected.

Note that U can be reinterpreted as a $\left(\frac{\alpha_A+\sqrt{\lambda}\alpha_L}{\|A_L\|}, \max(a_A, a_L) + 2, \frac{\varepsilon_A+\sqrt{\lambda}\varepsilon_L}{\|A_L\|}\right)$-block-encoding of the normalized matrix $A_L/\|A_L\|$. Furthermore, we can prepare the quantum state $|b\rangle|0\rangle$ in time T_b. Now by using Theorem 4.3.4 with U and an appropriately chosen δ specified above, we obtain a quantum state that is δ-close to

$$\frac{(A^TA + \lambda L^TL)^{-1}A^T|b\rangle}{\|(A^TA + \lambda L^TL)^{-1}A^T|b\rangle\|}$$

in the first register. □

In the above complexity, when L is a good regularizer, κ is independent of κ_A. κ can be made arbitrarily smaller than κ_A by an appropriate choice of L. Thus the regularized version has significantly better time complexity than the unregularized case. One such example of a good regularizer is in case of *Quantum Ridge Regression*, where we use the identity matrix to regularize. The corollary below elucidates this.

[i] Our results also hold if we assume that $\mathcal{S}_O \leq \gamma$ for some $\gamma \in (0,1)$. That is, $\left\|\Pi_{\text{col}(A_L)}\right\| \geq 1 - \gamma$. In such a scenario our complexity to prepare $A_L^+|b,0\rangle / \left\|A_L^+|b,0\rangle\right\|$ is re-scaled by $1/\sqrt{1-\gamma}$.

Corollary 5.1.3 (Quantum Ridge Regression)**.** *Let A be a matrix of dimension $N \times d$ with effective condition number κ_A and $\lambda \in \mathbb{R}^+$ be the regression parameter. Let U_A be a (α, a, ε)-block-encoding of A implemented in time T_A. Let U_b be a unitary that prepares $|b\rangle$ in time T_b. If $\kappa = 1 + \|A\|/\sqrt{\lambda}$ then for any δ such that*

$$\varepsilon = o\left(\frac{\delta}{\kappa^3 \log^2\left(\frac{\kappa}{\delta}\right)}\right)$$

we can prepare a state δ-close to

$$\frac{\left(A^T A + \lambda I\right)^{-1} A^T |b\rangle}{\left\|\left(A^T A + \lambda I\right)^{-1} A^T |b\rangle\right\|}$$

at a cost of

$$\mathcal{O}\left(\log \kappa \left(\frac{\alpha_A}{\sqrt{\lambda}} \log\left(\frac{\kappa}{\delta}\right) T_A + \kappa T_b\right)\right) \tag{5.1.9}$$

with probability $\Theta(1)$ using only $\mathcal{O}(\log \kappa)$ additional qubits.

Proof. The identity matrix I is a trivial $(1, 0, 0)$-block-encoding of itself, and $\kappa_I = 1$. We invoke Theorem 5.1.2 with $L = I$ to obtain the solution. □

Being in the block-encoding framework allows us to express the complexity of our quantum algorithm in specific input models such as the *quantum data structure input model* and the *sparse access model*. We express these complexities via the following corollaries.

Corollary 5.1.4 (Quantum Ordinary Least Squares with ℓ_2-Regularization in the Quantum Data Structure Input Model)**.** *Let $A, L \in \mathbb{R}^{N \times d}$ with effective condition numbers κ_A, κ_L respectively. Let $\lambda \in \mathbb{R}^+$ and $b \in \mathbb{R}^N$. Let κ be the effective condition number of the augmented matrix A_L. Suppose that A, L and b are stored in a quantum accessible data structure. Then for any $\delta > 0$ there exists a quantum algorithm to prepare a quantum state δ-close to*

$$\frac{\left(A^T A + \lambda L^T L\right)^{-1} A^T |b\rangle}{\left\|\left(A^T A + \lambda L^T L\right)^{-1} A^T |b\rangle\right\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa \left(\frac{\mu_A + \sqrt{\lambda}\mu_L}{\|A\| + \sqrt{\lambda}\|L\|}\right) \operatorname{polylog}\left(Nd, \kappa, \frac{1}{\delta}, \lambda\right)\right). \tag{5.1.10}$$

Proof. Since b is stored in the data structure, for some $\varepsilon_b > 0$, we can prepare the state $|b'\rangle$ that is ε_b-close to $|b\rangle = \sum_i b_i |i\rangle / \|b\|$ using $T_b = \mathcal{O}(\operatorname{polylog}(N/\varepsilon_b))$ queries to the data structure (see chapter 2.) Similarly, for some parameters $\varepsilon_A, \varepsilon_L > 0$, we can construct a $(\mu_A, \lceil \log(d+N) \rceil, \varepsilon_A)$-block-encoding of A using $T_A = \mathcal{O}(\operatorname{polylog}(Nd/\varepsilon_A))$ queries to the data structure and a $(\mu_L, \lceil \log(d+N) \rceil, \varepsilon_B)$-block-encoding of L using $T_L = \mathcal{O}(\operatorname{polylog}(Nd/\varepsilon_B))$ queries.

We invoke Theorem 5.1.2 with a precision $\delta/2$ by choosing ε_A and ε_L such that equation Equation 5.1.7 is satisfied. This gives us a state that is $\delta/2$-close to

$$\frac{\left(A^T A + \lambda L^T L\right)^{-1} A^T |b'\rangle}{\left\|\left(A^T A + \lambda L^T L\right)^{-1} A^T |b'\rangle\right\|}$$

To compute the final precision as δ, we use Lemma 3.1.5 by choosing $\varepsilon_b = \frac{\delta}{2\kappa}$. The complexity can be calculated by plugging in the relevant values in Equation 5.1.8 □

In the previous corollary $\mu_A = \|A\|_F$ and $\mu_L = \|L\|_F$ when the matrix A and L are stored in the data structure. Similarly, $\mu_A = \mu_p(A)$ and $\mu_L = \mu_p(L)$ when the matrices $A^{(p)}, A^{(1-p)}$ and $L^{(p)}, L^{(1-p)}$ are stored in the data structure.

Now we discuss the complexity of quantum ordinary least squares with ℓ_2-regularization in the *sparse access input model*. We call a matrix M as (s_r, s_c) row-column sparse if it has a row sparsity s_r and column sparsity s_c.

Corollary 5.1.5 (Quantum Ordinary least squares with ℓ_2-regularization in the sparse access model)**.** *Let $A \in \mathbb{R}^{N\times d}$ be (s_r^A, s_c^A) row-column sparse, and similarly, let $L \in \mathbb{R}^{N\times d}$ be (s_r^L, s_c^L) row-column sparse, with effective condition numbers κ_A and κ_L respectively. Let $\lambda \in \mathbb{R}^+$ and $\delta > 0$. Suppose there exists a unitary that prepares $|b\rangle$ at a cost, T_b. Then there is a quantum algorithm to prepare a quantum state that is δ-close to*

$$\frac{(A^TA + \lambda L^TL)^{-1}A^T|b\rangle}{\|(A^TA + \lambda L^TL)^{-1}A^T|b\rangle\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa\left(\frac{\sqrt{s_r^As_c^A} + \sqrt{\lambda s_r^Ls_c^L}}{\|A\| + \sqrt{\lambda}\|L\|}\right)\text{polylog}\left(Nd, \kappa, \frac{1}{\delta}, \lambda\right) + \kappa\log\kappa T_b\right). \tag{5.1.11}$$

Proof. The proof is similar to Corollary 5.1.4 but with $\alpha_A = \sqrt{s_r^As_c^A}$ and $\alpha_L = \sqrt{s_r^Ls_c^L}$. □

5.2 Regularized Quantum Weighted And Generalized Least Squares

This technique of working with a augmented matrix will also hold for the other variants of ordinary least squares. In this section, we begin by briefly describing these variants before moving on to designing quantum algorithms for the corresponding problems.

Weighted Least Squares: For the WLS problem, each observation $\{a_i, b_i\}$ is assigned some weight $w_i \in \mathbb{R}^+$ and the objective function to be minimized is of the form

$$\mathcal{L}_W := \sum_j w_j(x^Ta_j - b_j)^2. \tag{5.2.1}$$

If $W \in \mathbb{R}^{N\times N}$ is the diagonal matrix with w_i being the i^{th} diagonal entry, then the optimal x satisfies

$$x = (A^TWA)^{-1}A^TWb. \tag{5.2.2}$$

The ℓ_2-regularized version of WLS satisfies

$$x = (A^TWA + \lambda L^TL)^{-1}A^TWb \tag{5.2.3}$$

Our quantum algorithm outputs a state that is close to

$$|x\rangle = \frac{(A^TWA + \lambda L^TL)^{-1}A^TW|b\rangle}{\|(A^TWA + \lambda L^TL)^{-1}A^TW|b\rangle\|} \tag{5.2.4}$$

given approximate block-encodings of A, W and L.

Much like Equation 5.1.5, finding the optimal solution reduces to finding the pseudo inverse of an augmented matrix A_L given by

$$A_L := \begin{pmatrix}\sqrt{W}A & 0\\ \sqrt{\lambda}L & 0\end{pmatrix}.$$

The top left block of $A_L^+ = (A^TWA + \lambda L^TL)^{-1}A^T\sqrt{W}$, which is the required linear transformation to be applied to the vector $y = \sqrt{W}b$. The ratio between the minimum and maximum singular values of A_L, κ, can be obtained analogously to Lemma 5.1.1.

For the ℓ_2-regularized WLS problem, *normalized residual sum of squares* is given by

$$\mathcal{S}_W = \frac{\left\|(I - \Pi_{\text{col}(A_L)})|y\rangle|0\rangle\right\|^2}{\||y\rangle\|^2} = 1 - \left\|\Pi_{\text{col}(A_L)}|y\rangle|0\rangle\right\|^2. \tag{5.2.5}$$

Subsequently, we assume that $\mathcal{S}_W = 1 - \left\|\Pi_{\text{col}(A_L)}|y\rangle|0\rangle\right\|^2 \leq \gamma < 1/2$. This in turn implies that $\left\|\Pi_{\text{col}(A_L)}|y\rangle|0\rangle\right\|^2 = \Omega(1)$, implying that the data can be reasonably fit by a linear model.

Generalized Least Squares. Similarly, we can extend this to GLS problem, where there the input data may be correlated. These correlations are given by the non-singular covariance matrix $\Omega \in \mathbb{R}^{N\times N}$. The WLS problem is a special case of the GLS problem, corresponding to when Ω is a diagonal matrix.

The objective function to be minimized is

$$\mathcal{L}_\Omega := \sum_{i,j} (\Omega^{-1})_{ij} (x^T a_i - b_i)(x^T a_j - b_j). \tag{5.2.6}$$

The optimal $x \in \mathbb{R}^d$ satisfies

$$x = (A^T \Omega^{-1} A) A^T \Omega^{-1} b \tag{5.2.7}$$

Similarly, the ℓ_2-regularized GLS solver outputs x such that

$$x = (A^T \Omega^{-1} A + \lambda L^T L) A^T \Omega^{-1} b. \tag{5.2.8}$$

So, given approximate block-encodings of A, Ω and L a quantum GLS solver outputs a quantum state close to

$$|x\rangle = \frac{(A^T \Omega^{-1} A + \lambda L^T L) A^T \Omega^{-1} |b\rangle}{\|(A^T \Omega^{-1} A + \lambda L^T L) A^T \Omega^{-1} |b\rangle\|} \tag{5.2.9}$$

The augmented matrix A_L is defined as

$$A_L := \begin{pmatrix} \Omega^{-1/2} A & 0 \\ \sqrt{\lambda} L & 0 \end{pmatrix}.$$

Then top left block of A_L^+ to the vector $y = \Omega^{-1/2} b$ yields the optimal x. Thus the quantum GLS problem with ℓ_2-regularization first prepares $\Omega^{-1/2} |b\rangle |0\rangle$ and then uses the matrix inversion algorithm by QSVT to implement $A_L^+ \Omega^{-1/2} |b\rangle |0\rangle$. Analogous to OLS and WLS, we assume that the normalized residual sum of squares $\mathcal{S}_\Omega \leq \gamma < 1/2$.

5.2.1 Regularized Quantum Weighted Least Squares

In this section, we derive the complexity of the ℓ_2-regularized WLS problem. We assume that we have a diagonal weight matrix $W \in \mathbb{R}^{N \times N}$ such that its smallest and largest diagonal entries are $w_{\min}$ and $w_{\max}$, respectively. This implies that $\|W\| = w_{\max}$ and $\kappa_W = w_{\max}/w_{\min}$. We take advantage of the fact that the matrix W is diagonal and then apply controlled rotations to directly implement a block encoding of $\sqrt{W}A$. Additionally, given a state preparation procedure for $|b\rangle$, we can easily prepare a state proportional to $\sqrt{W} |b\rangle$. We then use Theorem 5.1.2 to solve QWLS.

We first formalize this idea in Theorem 5.2.1, assuming direct access to (i) a block encoding of $B = \sqrt{W}A$, and (ii) a procedure for preparing the state $|b_w\rangle = \frac{\sqrt{W}|b\rangle}{\|\sqrt{W}|b\rangle\|}$. Subsequently, for the specific input models, we show that we can indeed efficiently obtain a block-encoding of B and prepare the state $|b_w\rangle$.

Theorem 5.2.1 (Quantum Weighted Least Squares with General ℓ_2-Regularization)**.** *Let $A, L \in \mathbb{R}^{N \times d}$, be the data and penalty matrix, with effective condition numbers κ_A and κ_L, respectively. Let $\lambda \in \mathbb{R}^+$ be the regularizing parameter. Let $W \in \mathbb{R}^{N \times N}$ be a diagonal weight matrix with the largest and smallest diagonal entries being w_{max}, w_{min}, respectively. Let U_B be a $(\alpha_B, a_B, \varepsilon_B)$ block encoding of $B := \sqrt{W}A$ implemented in time T_B and let U_L be a $(\alpha_L, a_L, \varepsilon_L)$ block encoding of L implemented in time T_L, such that $\varepsilon_B = o\left(\frac{\delta}{\kappa^3 \log^2\left(\frac{\kappa}{\delta}\right)}\right)$ and $\varepsilon_L = o\left(\frac{\delta}{\sqrt{\lambda}\kappa^3 \log^2\left(\frac{\kappa}{\delta}\right)}\right)$. Let U_{b_w} be a unitary that prepares $\frac{\sqrt{W}|b\rangle}{\|\sqrt{W}|b\rangle\|}$ in time T_{b_w}. Define*

$$\kappa := \kappa_L \left(1 + \frac{\sqrt{w_{max}}\|A\|}{\sqrt{\lambda}\|L\|}\right)$$

Then for any $\delta > 0$ we can prepare a quantum state that is δ-close to

$$\frac{(A^T W A + \lambda L^T L)^{-1} A^T W |b\rangle}{\|(A^T W A + \lambda L^T L)^{-1} A^T W |b\rangle\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa \log \kappa \left(\frac{\alpha_B + \sqrt{\lambda}\alpha_L}{\sqrt{w_{max}}\|A\| + \sqrt{\lambda}\|L\|} \log\left(\frac{\kappa}{\delta}\right)(T_B + T_L) + T_{b_w}\right)\right), \tag{5.2.10}$$

using only $\mathcal{O}(\log \kappa)$ additional qubits.

Proof. We then invoke Theorem 5.1.2 with B and L as the data and regularization matrices, respectively. This requires that $\varepsilon_B, \varepsilon_L$ such that

$$\varepsilon_B + \sqrt{\lambda}\varepsilon_L = o\left(\frac{\delta}{\kappa^3 \log^2\left(\frac{\kappa}{\delta}\right)}\right).$$

Thus, we get the upper bounds on the precision ε_B, ε_L required. This gives us a quantum state δ-close to

$$\frac{(A^T W A + \lambda L^T L)^{-1} A^T W |b\rangle}{\|(A^T W A + \lambda L^T L)^{-1} A^T W |b\rangle\|}.$$

□

Next, we construct the block encodings for $\sqrt{W}A$ and the state $\frac{\sqrt{W}|b\rangle}{\|\sqrt{W}|b\rangle\|}$ efficiently in the quantum data structure input model. This construction would also apply to the sparse access input model with slight modifications.

Lemma 5.2.2 (Efficiently preparing $\sqrt{W}A$ in the Quantum Data Structure Model)**.** *Let $W \in \mathbb{R}^{N\times N}$ such that $W = \mathrm{diag}(w_1, w_2 \dots w_N)$ and $w_{\max} := \max_i w_i$, and $A \in \mathbb{R}^{N\times d}$ be stored in a quantum-accessible data structure. Then for any $\delta > 0$ there exists a*

$$\left(\sqrt{w_{\max}}\|A\|_F, \lceil \log(N+d)\rceil, \delta\right)$$

block-encoding of $\sqrt{W}A$ that can be implemented at the cost $\mathcal{O}(\mathrm{polylog}(Nd/\delta))$.

Proof. $\forall j \in [N]$, define

$$|\psi_j\rangle := \sqrt{\frac{w_j}{w_{\max}}}|j\rangle \frac{1}{\|A_{j,\cdot}\|}\sum_{k\in[d]} A_{j,k}|k\rangle.$$

Similarly, $\forall k \in [d]$, define

$$|\phi_k\rangle := \frac{1}{\|A\|_F}\left(\sum_{j\in[N]} \|A_{j,\cdot}\||j\rangle\right)|k\rangle.$$

Observe that $\forall j \in [N], k \in [d]$,

$$\langle\psi_j|\phi_k\rangle = \sqrt{\frac{w_j}{w_{\max}}}\frac{A_{j,k}}{\|A\|_F} = \frac{\langle j|\sqrt{W}A|k\rangle}{\sqrt{w_{\max}}\|A\|_F}.$$

Given quantum data structure accesses to W and A, once can construct quantum circuits W_R and W_L similar to U_L and U_R from Lemma 2.4.4 that prepare $|\phi_k\rangle$ and $|\psi_j\rangle$ above. $|\phi_k\rangle$ can be prepared just as in Lemma 2.4.4, while $|\psi_j\rangle$ can be prepared using controlled rotations on the state $|\frac{w_j}{w_{\max}}\rangle$ (which can be constructed from the QRAM access to W) after adding an ancilla qubit and the QRAM access to A. Thus, $W_R^\dagger W_L$ is the required block encoding, which according to Theorem 2.4.3 can be implemented using $\mathrm{polylog}(Nd/\delta)$ queries. □

Lemma 5.2.3 (Efficiently preparing $\sqrt{W}|b\rangle$ in the Quantum Data Structure Model)**.** *Let $b \in \mathbb{R}^N$ and $W \in \mathbb{R}^{N\times N}$. Suppose that b and W are stored in a quantum-accessible data structure such that we have a state preparation procedure that acts as*

$$U_W : |j\rangle|0\rangle \mapsto |j\rangle|w_j\rangle,$$
$$U_b : |0\rangle \mapsto \sum_j \frac{b_j}{\|b\|}|j\rangle.$$

Then for any $\delta > 0$ we can prepare the quantum state that is δ-close to $\frac{\sqrt{W}|b\rangle}{\|\sqrt{W}|b\rangle\|}$ with constant success probability and at a cost of $\mathcal{O}\left(\sqrt{\frac{w_{\max}}{w_{\min}}}\mathrm{polylog}\left(\frac{N}{\delta}\right)\right)$.

Proof. Use U_b to prepare the state

$$|b\rangle = \frac{1}{\|b\|}\sum_j b_j|j\rangle$$

in time $\mathrm{polylog}(N)$. Then, apply the following transformation

$$|j\rangle|0\rangle|0\rangle \mapsto |j\rangle|w_j\rangle|0\rangle$$

$$\mapsto |j\rangle |w_j\rangle \left(\sqrt{\frac{w_j}{w_{\max}}} |0\rangle + \sqrt{1 - \frac{w_j}{w_{\max}}} |1\rangle\right)$$
$$\mapsto |j\rangle |0\rangle \left(\sqrt{\frac{w_j}{w_{\max}}} |0\rangle + \sqrt{1 - \frac{w_j}{w_{\max}}} |1\rangle\right)$$

which can again be applied using some controlled rotations, a square root circuit and U_W. This gives us the state (ignoring some blank registers)

$$\sum_j \left(\sqrt{\frac{w_j}{w_{\max}}} |0\rangle + \sqrt{1 - \frac{w_j}{w_{\max}}} |1\rangle\right) \frac{b_j}{\|b\|} |j\rangle . \tag{5.2.11}$$

The probability for the ancilla to be in $|0\rangle$ state is

$$\Omega\left(\frac{w_{\min}}{w_{\max}}\right).$$

Thus performing $\mathcal{O}\left(\sqrt{\frac{w_{\max}}{w_{\min}}}\right)$ rounds of amplitude amplification on $|0\rangle$ gives us a constant probability of observing $|0\rangle$, and therefore obtaining the desired state $\frac{\sqrt{W}|b\rangle}{\|\sqrt{W}|b\rangle\|}$. □

Using the above two theorems, and the quantum OLS solver (Theorem 5.1.2), we can construct an algorithm for regularized quantum WLS.

Theorem 5.2.4 (Quantum Weighted Least Squares with General ℓ_2-Regularization in the Quantum Data Structure Model)**.** *Let $A, L \in \mathbb{R}^{N\times d}$ with effective condition numbers κ_A, κ_L respectively be stored in an efficient quantum accessible data structure. Let $W \in \mathbb{R}^{N\times N}$ be a diagonal matrix with largest and smallest singular values w_{max}, w_{min} respectively, which is also stored in an efficient quantum accessible data structure. Furthermore, suppose the entries of the vector $b \in \mathbb{R}^N$ are also stored in a quantum-accessible data structure and define,*

$$\kappa := \kappa_L \left(1 + \frac{\sqrt{w_{max}}\|A\|}{\sqrt{\lambda}\|L\|}\right)$$

Then for any $\delta > 0$ we can prepare a quantum state that is δ-close to

$$\frac{(A^TWA + \lambda L^TL)^{-1}A^TW|b\rangle}{\|(A^TWA + \lambda L^TL)^{-1}A^TW|b\rangle\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa\left(\frac{\sqrt{w_{max}}\|A\|_F + \sqrt{\lambda}\|L\|_F}{\sqrt{w_{max}}\|A\| + \sqrt{\lambda}\|L\|} + \sqrt{\frac{w_{max}}{w_{min}}}\right) \text{polylog}\left(Nd, \kappa, \frac{1}{\delta}\right)\right) \tag{5.2.12}$$

Proof. Choose some precision parameter $\varepsilon > 0$ for accessing the data structure. Given accesses to W and A, we can use Lemma 5.2.2 for some $\varepsilon > 0$ to prepare a $(\sqrt{w_{\max}}\|A\|_F, \lceil\log(N+d)\rceil, \varepsilon)$-block-encoding of $\sqrt{W}A$, using $T_A := \mathcal{O}(\text{polylog}(Nd/\varepsilon))$ queries to the data structure.

Similarly, Lemma 2.4.4 allows us to build a $(\|L\|_F, \lceil\log(N+d)\rceil, \varepsilon)$-block-encoding of L using $T_L := \mathcal{O}(\text{polylog}(Nd/\varepsilon))$ queries to the data structure.

Next, using Lemma 5.2.3, for any $\varepsilon_b > 0$, we can prepare a state ε_b-close to $|b'\rangle := \frac{\sqrt{W}|b\rangle}{\|\sqrt{W}|b\rangle\|}$. This procedure requires $T_b := \mathcal{O}\left(\sqrt{\frac{w_{\max}}{w_{\min}}}\text{polylog}(N/\varepsilon_b)\right)$ queries to the data structure. Now we can invoke the OLS solver in Theorem 5.1.2 with a precision of δ_b, by considering $\sqrt{W}A$ as the data matrix and $\frac{\sqrt{W}|b\rangle}{\|\sqrt{W}|b\rangle\|}$ as the input state. In order for the input block-encoding precision to satisfy the bound in Equation 5.1.7, we choose ε such that

$$\varepsilon = o\left(\frac{\delta_b}{\kappa^3 \log^2\left(\frac{\kappa}{\delta_b}\right)}\right).$$

Finally, for the output state to be δ-close to the required state, we choose $\delta_b = \delta/2$ and $\varepsilon_b = \delta/2\kappa$ to use the robustness result from Lemma 3.1.5. This gives us

$$\log\left(\frac{1}{\varepsilon}\right) = \mathcal{O}\left(\log\left(\frac{\kappa^3 \log^2\left(\frac{\kappa}{\delta_b}\right)}{\delta_b}\right)\right)$$

$$= \mathcal{O}\left(\log\left(\frac{\kappa}{\delta}\right)\right)$$

Now we can substitute the cost of the individual components in Equation 5.1.8 to obtain the final cost as

$$\mathcal{O}\left(\kappa \log\kappa \left(\frac{\sqrt{w_{\max}}\|A\|_F + \sqrt{\lambda}\|L\|_F}{\sqrt{w_{\max}}\|A\| + \sqrt{\lambda}\|L\|} \log\left(\frac{\kappa}{\delta}\right) \text{polylog}\left(\frac{Nd}{\varepsilon}\right) + \sqrt{\frac{w_{\max}}{w_{\min}}} \text{polylog}\left(\frac{N\kappa}{\delta}\right)\right)\right)$$
$$= \mathcal{O}\left(\kappa \left(\frac{\sqrt{w_{\max}}\|A\|_F + \sqrt{\lambda}\|L\|_F}{\sqrt{w_{\max}}\|A\| + \sqrt{\lambda}\|L\|} + \sqrt{\frac{w_{\max}}{w_{\min}}}\right) \text{polylog}\left(\frac{Nd\kappa}{\delta}\right)\right)$$

□

Now, for the sparse access model, we can obtain a block encoding similar to Lemma 5.2.2 and a quantum state similar to Lemma 5.2.3, with the same query complexities. Thus we have an algorithm similar to Theorem 5.2.4 in the sparse access model as well. We directly state the complexity of this algorithm.

Theorem 5.2.5 (Quantum Weighted Least Squares with General ℓ_2-Regularization in the Sparse Access Model)**.** *Let $A \in \mathbb{R}^{N\times d}$ be (s_r^A, s_c^A) row-column sparse, and similarly, let $L \in \mathbb{R}^{N\times d}$ be (s_r^L, s_c^L) row-column sparse, with effective condition numbers κ_A and κ_L respectively. Let $\lambda \in \mathbb{R}^+$. Let $W \in \mathbb{R}^{N\times N}$ be a diagonal matrix with the largest and the smallest diagonal entries being w_{max}, w_{min}, respectively. Suppose that the diagonal entries of W are stored in a QROM such that, for any $\delta > 0$, we can compute $|j\rangle 0 \mapsto |j\rangle |w_j\rangle$ in cost $O(\text{polylog}(Nd/\delta))$ as well as w_{max}. Furthermore, suppose there exists a unitary that prepares $|b\rangle$ at a cost T_b and define,*

$$\kappa := \kappa_L\left(1 + \frac{\sqrt{w_{max}}\|A\|}{\sqrt{\lambda}\|L\|}\right)$$

Then for any $\delta > 0$ we can prepare a quantum state that is δ-close to

$$\frac{(A^TWA + \lambda L^TL)^{-1}A^TW|b\rangle}{\|(A^TWA + \lambda L^TL)^{-1}A^TW|b\rangle\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa\left(\frac{\sqrt{w_{max}}\sqrt{s_r^A s_c^A} + \sqrt{\lambda}\sqrt{s_r^L s_c^L}}{\sqrt{w_{max}}\|A\| + \sqrt{\lambda}\|L\|} + \sqrt{\frac{w_{max}}{w_{min}}}\, T_b\right)\text{polylog}\left(Nd, \kappa, \frac{1}{\delta}\right)\right) \tag{5.2.13}$$

5.2.2 Regularized Quantum Generalized Least Squares

In this section, we assume that we have block-encoded access to the correlation matrix $\Omega \in \mathbb{R}^{N\times N}$, with condition number κ_Ω. We begin by preparing a block encoding of $\Omega^{-1/2}$, given an approximate block-encoding of Ω.

Lemma 5.2.6 (Preparing $\Omega^{-1/2}$)**.** *Let $\Omega \in \mathbb{R}^{N\times N}$ be a matrix with condition number κ_Ω Let U_Ω be an $(\alpha_\Omega, a_\Omega, \varepsilon_\Omega)$-block-encoding of Ω, implemented in time T_Ω. For any δ such that*

$$\varepsilon_\Omega = o\left(\frac{\sqrt{\|\Omega\|}\delta}{\kappa^{1.5}\log\left(\frac{\kappa}{\sqrt{\|\Omega\|}\delta}\right)}\right),$$

we can prepare a $(2\sqrt{\kappa_\Omega/\|\Omega\|}, a_\Omega + 1, \delta)$-block-encoding of $\Omega^{-1/2}$ at a cost of

$$\mathcal{O}\left(\frac{\alpha_\Omega\kappa_\Omega}{\|\Omega\|}\log\left(\frac{\kappa_\Omega}{\delta\sqrt{\|\Omega\|}}\right)T_\Omega\right)$$

Moreover, the condition number of $\Omega^{-1/2}$ is bounded by $\sqrt{\kappa_\Omega}$.

Proof. U_Ω can be re-interpreted as a $(\frac{\alpha}{\|\Omega\|},\ a,\ \frac{\varepsilon}{\|\Omega\|})$-block-encoding of $\frac{\Omega}{\|\Omega\|}$. We can then prepare the required unitary by invoking Theorem 3.4.2 on U_Ω with $c = 1/2$ and some γ such that we get a $(2\sqrt{\kappa_\Omega},\ a+1,\ \gamma)$ block encoding of $\sqrt{\|\Omega\|}\Omega^{-1/2}$, which is a $(2\sqrt{\frac{\kappa}{\|\Omega\|}},\ a+1,\ \frac{\gamma}{\sqrt{\|\Omega\|}})$ block encoding of $\Omega^{-1/2}$. Fixing $\gamma = \sqrt{\|\Omega\|}\delta$ gives us the required result. □

We will now use this lemma in conjunction with Theorem 5.1.2 to develop quantum algorithms for GLS with general ℓ_2-regularization.

Theorem 5.2.7 (Quantum Generalized Least Squares with General ℓ_2-regularization)**.** *Let $A, L \in \mathbb{R}^{N\times d}$ be the data and penalty matrices with effective condition numbers κ_A, κ_L respectively. Let $\Omega \in \mathbb{R}^{N\times N}$ be the covariance matrix with condition number κ_Ω. Let $\delta > 0$ be the precision parameter. Define κ as*

$$\kappa := \kappa_L \left(1 + \frac{\sqrt{\kappa_\Omega}\|A\|}{\sqrt{\lambda\|\Omega\|}\|L\|}\right).$$

For some ε_A such that

$$\varepsilon_A = o\left(\frac{\delta\sqrt{\|\Omega\|}}{\kappa^3\sqrt{\kappa_\Omega}\log^2\frac{\kappa}{\delta}}\right)$$

we have access to U_A, an $(\alpha_A, a_A, \varepsilon_A)$-block-encoding of A implemented in time T_A. For some ε_L such that

$$\varepsilon_L = o\left(\frac{\delta}{\sqrt{\lambda}\kappa^3\log^2\frac{\kappa}{\delta}}\right)$$

we have access to U_L, an $(\alpha_L, a_L, \varepsilon_L)$-block-encoding of L implemented in time T_L. For some ε_Ω such that

$$\varepsilon_\Omega = o\left(\frac{\delta}{\|A\|\kappa^3{\kappa_\Omega}^{1.5}\log^3\frac{\kappa}{\delta}\log\left(\frac{\kappa_\Omega}{\|A\|\|\Omega\|}\right)}\right)$$

we have access to U_Ω, an $(\alpha_\Omega, a_\Omega, \varepsilon_\Omega)$-block-encoding of Ω implemented in time T_Ω. Let U_b be a unitary that prepares the state $|b\rangle$ in time T_b.

Then we can prepare the quantum state that is δ-close to

$$\frac{\left(A^T\Omega^{-1}A + \lambda L^TL\right)^{-1}A^T\Omega^{-1}|b\rangle}{\left\|\left(A^T\Omega^{-1}A + \lambda L^TL\right)^{-1}A^T\Omega^{-1}|b\rangle\right\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa\sqrt{\kappa_\Omega}\log\kappa\left(\left(\frac{\alpha_A}{\|A\|}T_A + \frac{\alpha_L}{\|L\|}T_L + \frac{\alpha_\Omega\kappa_\Omega}{\|\Omega\|}T_\Omega\right)\log^3\left(\frac{\kappa\kappa_\Omega\|A\|\|L\|}{\delta\|\Omega\|}\right) + T_b\right)\right) \tag{5.2.14}$$

using only $\mathcal{O}(\log\kappa)$ additional qubits.

Proof. Observe that by choosing $A' := \Omega^{-1/2}A, L' := L, |b'\rangle := \Omega^{-1/2}|b\rangle$ (upto normalization) in the quantum ordinary least squares, we get a state proportional to $(A'^TA' + \lambda L'^TL')^{-1}A'^T|b'\rangle = (A^T\Omega^{-1}A + \lambda L^TL)A^T\Omega^{-1}|b\rangle$, which is the desired state.

For convenience, let us define the matrix $B := \Omega^{-1/2}$ (and therefore $\kappa_B = \sqrt{\kappa_\Omega}$ and $\|B\| = \sqrt{\kappa_\Omega/\|\Omega\|}$). We now need to prepare a block-encoding of BA and the quantum state $\frac{B|b\rangle}{\|B|b\rangle\|}$, which we then use to invoke Theorem 5.1.2.

We begin by using Lemma 5.2.6 with some precision ε_B to construct a $(\alpha_B, a_B, \varepsilon_B)$-block-encoding of $B = \Omega^{-1/2}$, where $\alpha_B = 2\sqrt{\frac{\kappa_\Omega}{\|\Omega\|}} = 2\|B\|$, and $a_B = a_\Omega + 1$. This bounds ε_Ω as

$$\varepsilon_\Omega = o\left(\frac{\sqrt{\|\Omega\|}\varepsilon_B}{{\kappa_\Omega}^{1.5}\log\left(\frac{\kappa_\Omega}{\sqrt{\|\Omega\|}\varepsilon_B}\right)}\right),$$

and has a cost of

$$T_B := \mathcal{O}\left(\frac{\alpha_\Omega\kappa_\Omega}{\|\Omega\|}\log\left(\frac{\kappa_\Omega}{\varepsilon_B\sqrt{\|\Omega\|}}\right)T_\Omega\right)$$

Then using Lemma 3.2.5 with precision γ satisfying $\gamma \geq 4\sqrt{2}\max(\|B\|\varepsilon_A, \|A\|\varepsilon_B)$, we get a $(2\|A\|\|B\|, a_A + a_B + 3, \gamma)$-block-encoding of $A' := BA = \Omega^{-1/2}A$ at a cost

$$T_{A'} := \mathcal{O}\left(\left(\frac{\alpha_A}{\|A\|}T_A + \frac{\alpha_B}{\|B\|}T_B\right)\log\left(\frac{\|A\|\|B\|}{\gamma}\right)\right).$$

To prepare $\frac{B|b\rangle}{\|B|b\rangle\|}$, we use Lemma 3.1.3 with precision $\varepsilon_b \geq 2\varepsilon_B\kappa_B/\|B\|$. This prepares a state that is ε_b-close to $|b'\rangle := \frac{B|b\rangle}{\|B|b\rangle\|}$ with constant success probability at a cost of

$$T_{b'} := \mathcal{O}\left(\frac{\alpha_B\kappa_B}{\|B\|}(T_B + T_b)\right) = \mathcal{O}(\kappa_B(T_B + T_b))$$

We could invoke OLS directly using the above two, but that ends up with a product of sub-normalization factors (α terms) in the complexity. We want to avoid this, because in most common cases α-s for block-encodings are quite large. So we also pre-amplify U_L using Corollary 3.1.2: for any $\delta_L \geq 2\varepsilon_L$ we get a $(\sqrt{2}\|L\|, a_L + 1, \delta_L)$-encoding of L at a cost of

$$T_{L'} := \mathcal{O}\left(\frac{\alpha_L}{\|L\|}T_L \log\left(\frac{\|L\|}{\delta_L}\right)\right).$$

Now that we have these, we can use Theorem 5.1.2 to get a quantum state δ'-close to $|\psi\rangle := \frac{A_L^+|b'\rangle}{\|A_L^+|b'\rangle\|}$, where $A_L^+ = (A^T\Omega^{-1}A + \lambda L^T L)^{-1}A^T\Omega^{-1/2}$. This would require that $\gamma, \sqrt{\lambda}\delta_L \in o\left(\frac{\delta'}{\kappa^3 \log^2\left(\frac{\kappa}{\delta'}\right)}\right)$ and would cost

$$\mathcal{O}\left(\kappa \log \kappa \left(\left(\frac{2\|A\|\|B\| + \sqrt{2\lambda}\|L\|}{\|BA\| + \sqrt{\lambda}\|L\|}\right)\log\left(\frac{\kappa}{\delta'}\right)(T_{A'} + T_{L'}) + T_{b'}\right)\right).$$

To simplify the ratio of norms term, we can first lower-bound $\|BA\| \geq \|A\|/\left\|B^{-1}\right\| = \|A\|/\sqrt{\|\Omega\|}$. And as $\|B\| = \sqrt{\kappa_\Omega/\|\Omega\|}$, the whole term can be simplified to $\mathcal{O}\left(\sqrt{\kappa_\Omega}\right)$. This simplifies the cost expression to $\mathcal{O}\left(\kappa \log \kappa \left(\sqrt{\kappa_\Omega}\log\left(\kappa/\delta'\right)(T_{A'} + T_{L'}) + T_{b'}\right)\right)$.

We can compute the error between $|\psi\rangle$ and the expected state by using Lemma 3.1.5. For the final error to be δ, we have to choose $\varepsilon_b = \delta/2\kappa$ and $\delta' = \delta/2$. Therefore

$$\varepsilon_B \leq \frac{\varepsilon_b\|B\|}{4\kappa_B} = \Theta\left(\frac{\delta}{\kappa\sqrt{\|\Omega\|}}\right)$$

$$\gamma, \sqrt{\lambda}\delta_L \in o\left(\frac{\delta}{\kappa^3 \log^2(\kappa/\delta)}\right)$$

$$\implies \log\left(\frac{1}{\gamma}\right) = o\left(\log\left(\frac{\kappa}{\delta}\right)\right), \quad \log\left(\frac{1}{\delta_L}\right) = o\left(\log\left(\frac{\sqrt{\lambda}\kappa}{\delta}\right)\right)$$

$$\varepsilon_A = o\left(\frac{\gamma}{\|B\|}\right), \quad \varepsilon_B = o\left(\frac{\gamma}{\|A\|}\right)$$

Combining both bounds of ε_B by using sums or products, we can effectively bound

$$\varepsilon_\Omega = o\left(\frac{\delta}{\|A\|\kappa^3{\kappa_\Omega}^{1.5}\log^3\frac{\kappa}{\delta}\log\left(\frac{\kappa_\Omega}{\|A\|\|\Omega\|}\right)}\right)$$

Finally for the final costs, we calculate the respective coefficients of terms T_A, T_Ω, T_L and T_b, (excluding the common factor of $\kappa\sqrt{\kappa_\Omega}\log\kappa$ for brevity). Let us label these "coefficient extraction" functions as $\mathcal{C}$ with matching subscripts, and the total cost as T.

$$\begin{aligned}
\mathcal{C}_A(T) &= \mathcal{O}\left(\log\left(\frac{\kappa}{\delta}\right)\mathcal{C}_A(T_{A'})\right) \\
&= \mathcal{O}\left(\log\left(\frac{\kappa}{\delta}\right)\frac{\alpha_A}{\|A\|}\log\left(\frac{\|A\|\|B\|}{\gamma}\right)\right) \\
&= \mathcal{O}\left(\frac{\alpha_A}{\|A\|}\log^2\left(\frac{\kappa\kappa_\Omega\|A\|}{\delta\|\Omega\|}\right)\right) \\
\mathcal{C}_L(T) &= \mathcal{O}\left(\log\left(\frac{\kappa}{\delta}\right)\mathcal{C}_L(T_{L'})\right)
\end{aligned}$$

$$
\begin{aligned}
&= \mathcal{O}\left(\log\left(\frac{\kappa}{\delta}\right)\frac{\alpha_L}{\|L\|}\log\left(\frac{\|L\|}{\delta_L}\right)\right)\\
&= \mathcal{O}\left(\frac{\alpha_L}{\|L\|}\log^2\left(\frac{\kappa\|L\|}{\delta}\right)\right)\\
\mathcal{C}_\Omega(T) &= \mathcal{O}\left(\log\left(\frac{\kappa}{\delta}\right)\mathcal{C}_\Omega(T_{A'}) + \frac{\mathcal{C}_\Omega(T_{b'})}{\sqrt{\kappa_\Omega}}\right)\\
&= \mathcal{O}\left(\left(\log\left(\frac{\kappa}{\delta}\right)\log\left(\frac{\|A\|\|B\|}{\gamma}\right)+1\right)\mathcal{C}_\Omega(T_B)\right)\\
&= \mathcal{O}\left(\log^2\left(\frac{\kappa\kappa_\Omega\|A\|}{\delta\|\Omega\|}\right)\frac{\alpha_\Omega\kappa_\Omega}{\|\Omega\|}\log\left(\frac{\kappa_\Omega}{\varepsilon_B\sqrt{\|\Omega\|}}\right)\right)\\
&= \mathcal{O}\left(\frac{\alpha_\Omega\kappa_\Omega}{\|\Omega\|}\log^3\left(\frac{\kappa\kappa_\Omega\|A\|}{\delta\|\Omega\|}\right)\right)\\
\mathcal{C}_b(T) &= \mathcal{O}\left(\frac{\mathcal{C}_\Omega(T_{b'})}{\sqrt{\kappa_\Omega}}\right) = \mathcal{O}(1)
\end{aligned}
$$

And hence the final complexity is given by the expression

$$
\begin{aligned}
T &= \mathcal{O}(\kappa\sqrt{\kappa_\Omega}\log\kappa\,(\mathcal{C}_A(T)\cdot T_A + \mathcal{C}_L(T)\cdot T_L + \mathcal{C}_\Omega(T)\cdot T_\Omega + \mathcal{C}_b(T)\cdot T_b))\\
&= \mathcal{O}\left(\kappa\sqrt{\kappa_\Omega}\log\kappa\left(\frac{\alpha_A}{\|A\|}\log^2\left(\frac{\kappa\kappa_\Omega\|A\|}{\delta\|\Omega\|}\right)T_A + \frac{\alpha_L}{\|L\|}\log^2\left(\frac{\kappa\|L\|}{\delta}\right)T_L + \frac{\alpha_\Omega\kappa_\Omega}{\|\Omega\|}\log^3\left(\frac{\kappa\kappa_\Omega\|A\|}{\delta\|\Omega\|}\right)T_\Omega + T_b\right)\right)\\
&= \mathcal{O}\left(\kappa\sqrt{\kappa_\Omega}\log\kappa\left(\left(\frac{\alpha_A}{\|A\|}T_A + \frac{\alpha_L}{\|L\|}T_L + \frac{\alpha_\Omega\kappa_\Omega}{\|\Omega\|}T_\Omega\right)\log^3\left(\frac{\kappa\kappa_\Omega\|A\|\|L\|}{\delta\|\Omega\|}\right) + T_b\right)\right)
\end{aligned}
$$

□

One immediate observation is that for the special case of the (unregularized) quantum GLS problem (when $L = 0$ and $\lambda = 0$), our algorithm has a slightly better complexity than [35] and requires fewer additional qubits. Now, we will state the complexities of this algorithm in specific input models, namely the quantum data structure model and the sparse-access input model.

Corollary 5.2.8 (Quantum Generalized Least Squares with General ℓ_2-Regularization in the Quantum Data Structure Model)**.** *Let $A, L \in \mathbb{R}^{N\times d}$ be the data and penalty matrices with effective condition numbers κ_A, κ_L respectively. and $\Omega \in \mathbb{R}^{N\times N}$ be the covariance matrix with condition number κ_Ω. Let the matrices A, L, Ω and the vector b be stored in a quantum-accessible data structure. Define κ as*

$$
\kappa := \kappa_L\left(1 + \frac{\sqrt{\kappa_\Omega}\|A\|}{\sqrt{\lambda\|\Omega\|}\|L\|}\right)
$$

Then for any $\delta > 0$, we can prepare the quantum state that is δ-close to

$$
\frac{\left(A^T\Omega^{-1}A + \lambda L^TL\right)^{-1}A^T\Omega^{-1}|b\rangle}{\left\|\left(A^T\Omega^{-1}A + \lambda L^TL\right)^{-1}A^T\Omega^{-1}|b\rangle\right\|}
$$

with probability $\Theta(1)$, at a cost of

$$
\mathcal{O}\left(\kappa\sqrt{\kappa_\Omega}\left(\frac{\mu_A}{\|A\|} + \frac{\mu_L}{\|L\|} + \frac{\kappa_\Omega\mu_\Omega}{\|\Omega\|}\right)\mathrm{polylog}\left(Nd, \kappa, \frac{1}{\delta}, \frac{\kappa_\Omega}{\|\Omega\|}, \|A\|, \|L\|, \lambda\right)\right) \tag{5.2.15}
$$

Proof. The proof is very similar to Corollary 5.1.4 with the extra input of Ω. We can use the data structure to prepare the block-encodings for A, L, Ω and the state $|b\rangle$, with precisions $\varepsilon_A, \varepsilon_L, \varepsilon_\Omega, \varepsilon_b$ respectively. We invoke Theorem 5.2.7 with a precision of δ_b, and choose the above ε terms to be equal to their corresponding upper-bounds. And finally we use Lemma 3.1.5 with $\varepsilon_b = \delta/2\kappa$ and $\delta_b = \delta/2$ to get the final error as δ. □

Now, $\mu_A = \|A\|_F$ (similarly for μ_L and μ_Ω). As $\|A\|_F \le \sqrt{r(A)}\|A\|$, where $r(A)$ is the rank of A, we have that the complexity of Corollary 5.2.8 can be re-expressed as

$$
\mathcal{O}\left(\kappa\sqrt{\kappa_\Omega}\left(\sqrt{r(A)} + \sqrt{r(L)} + \sqrt{r(\Omega)}\kappa_\Omega\right)\mathrm{polylog}\left(\frac{Nd\kappa}{\delta}\right)\right). \tag{5.2.16}
$$

Corollary 5.2.9 (Quantum Generalized Least Squares with General ℓ_2-Regularization in the Sparse Access Model)**.** *Let $A \in \mathbb{R}^{N\times d}$ be a (s_r^A, s_c^A) row-column sparse data matrix. Let $L \in \mathbb{R}^{N\times d}$ be a (s_r^L, s_c^L) row-column sparse penalty matrix. Let $\Omega \in \mathbb{R}^{N\times N}$ be a (s_r^Ω, s_c^Ω) row-column sparse covariance matrix. Suppose we have a procedure to prepare $|b\rangle$ in cost T_b. Define κ as*

$$\kappa := \kappa_L \left(1 + \frac{\sqrt{\kappa_\Omega}\|A\|}{\sqrt{\lambda\|\Omega\|}\|L\|}\right)$$

Then for any $\delta > 0$, we can prepare the quantum state that is δ-close to

$$\frac{\left(A^T\Omega^{-1}A + \lambda L^T L\right)^{-1} A^T\Omega^{-1}|b\rangle}{\left\|\left(A^T\Omega^{-1}A + \lambda L^T L\right)^{-1} A^T\Omega^{-1}|b\rangle\right\|}$$

with probability $\Theta(1)$, at a cost of

$$\mathcal{O}\left(\kappa\sqrt{\kappa_\Omega}\left(\frac{\sqrt{s_r^A s_c^A}}{\|A\|} + \frac{\sqrt{s_r^L s_c^L}}{\|L\|} + \frac{\kappa_\Omega\sqrt{s_r^\Omega s_c^\Omega}}{\|\Omega\|} + T_b\right)\text{polylog}\left(Nd, \kappa, \frac{1}{\delta}, \frac{\kappa_\Omega}{\|\Omega\|}, \|A\|, \|L\|, \lambda\right)\right) \tag{5.2.17}$$

Proof. The algorithm is similar to Corollary 5.2.8, but with $\alpha_A = \sqrt{s_r^A s_c^A}$, $\alpha_L = \sqrt{s_r^L s_c^L}$, $\alpha_\Omega = \sqrt{s_r^\Omega s_c^\Omega}$. □

Chapter 6

Conclusion and Future Directions

In this work we have developed quantum algorithms for regularized ordinary, weighted and generalized least squares. In the process, we have improved various other quantum algorithms commonly used in the literature. QSVT is a novel approach to quantum algorithms, and to the best of our knowledge this work was the first piece of literature to highlight the robustness of all the algorithms designed using it. The primitives we have developed, and our approach to designing the algorithms might be of independent interest to the community, specially in the domain of quantum machine learning (QML), where such quantum linear algebra subroutines (QLAS) are frequently used. QSVT is an interesting framework to analyze applications and limitations of quantum computation, and is known to produce algorithms often matching the previous SoTA or beating them. The generality of the block-encoding model also implies quantum algorithms in other input models (a fact also highlighted in chapter 5). However, due to the nature of the framework involving applications of controlled unitaries and the dependence of the algorithms on the block-encoding parameter (α), this might not be a good candidate for near term implementations.

Our algorithms for quantum linear regression with general ℓ_2-regularization made use of QSVT to implement various several matrix operations. However, it is possible to use QSVT directly to obtain the solution to *quantum ridge regression*. This requires computing a polynomial approximation for the transformation $\sigma \mapsto \sigma/(\sigma^2 + \lambda)$, to be applied on the singular values of A, which lie between $[1/\kappa_A, 1]$. However, it is unclear how to extend this while considering general ℓ_2-regularization. For instance, even when the data matrix and the penalty matrix share the same right singular vectors, this approach involves obtaining polynomial approximations to directly implement transformations of the form $\sigma \mapsto \sigma/(\sigma^2 + \lambda\widetilde{\sigma}^2)$, where $\widetilde{\sigma}$ is a singular value of the penalty matrix L. A monomial is no longer sufficient to approximate this quantum singular value transformation. It would be interesting to explore whether newly developed ideas of M-QSVT [70] can be used to implement such transformations directly with improved complexity.

While developing quantum machine learning algorithms, it is important to point out the caveats, even at the risk of being repetitive [28]. Our quantum algorithms output a quantum state $|x\rangle$ whose amplitudes encode the solution of the classical (regularized) linear regression problem. While given access to the data matrix and the penalty matrix, we achieve an exponential advantage over classical algorithms, this advantage is not generic. If similar assumptions (ℓ_2-sample and query access) are provided to a classical device, Gilyén et al. developed a quantum algorithm [41] for ridge regression (building upon [71]) which has a running time in $\mathcal{O}(\text{poly}(\kappa, \text{rank}(A), 1/\delta))$, which implies that any quantum algorithm for this problem can be at most polynomially faster in κ. One might posit that similar quantum-inspired classical algorithms for general ℓ_2-regression can also be developed.

Another future direction of research would be to recast our algorithms in the framework of adiabatic quantum computing (AQC) following the works of [72, 73]. Quantum algorithms for linear systems in this framework have the advantage that a linear dependence on κ can be obtained without using complicated subroutines like variable-time amplitude amplification. The strategy is to implement these problems in the AQC model and then use time-dependent Hamiltonian simulation [74] to obtain their complexities in the circuit model. One caveat is that, so far, time-dependent Hamiltonian simulation algorithms have only been developed in the sparse-access model and therefore the advantage of the generality of the block-encoding framework is lost.

In the future, it would also be interesting to explore other quantum algorithms for machine learning such as principal component regression and linear support vector machines [75] using QSVT. Finally, following the results of [40], it would be interesting to investigate techniques for quantum machine learning that do not require the quantum linear systems algorithm as a subroutine.

Appendix A

Miscellaneous Algorithms

A.1 Using controlled rotations to modify phases

In this section we show how to shift the contents of a resister to phases of an ancillary qubit. Specifically, we show how to apply the following transformation

$$|0\rangle |\theta\rangle \mapsto [\cos(\pi\theta) |0\rangle + \sin(\pi\theta) |1\rangle] |\theta\rangle . \tag{A.1.1}$$

Theorem A.1.1. *Let $\theta = 0.\theta_{d-1}\theta_{d-2}\dots\theta_0$ be a d-bit representation of θ. Then there exists a unitary U_θ acting on $d+1$-qubits that performs the transformation in Equation A.1.1.*

Proof. For the Pauli-y matrix

$$\sigma_y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},$$

observe that

$$\exp(-i\tau\sigma_y) = \cos\tau\mathcal{I} - i\sin\tau\sigma_y R_y(\tau), \tag{A.1.2}$$

is a single qubit rotation around the y axis. Apply a series of controlled-y rotations on the ancilla qubit, controlled first on the $d-1$ qubit, then on $d-2$, and so on, by angles $\frac{\pi}{2}, \frac{\pi}{2^2}, \dots \frac{\pi}{2^d}$. This gives us the required transformation. The circuit diagram is given in Figure A.1.

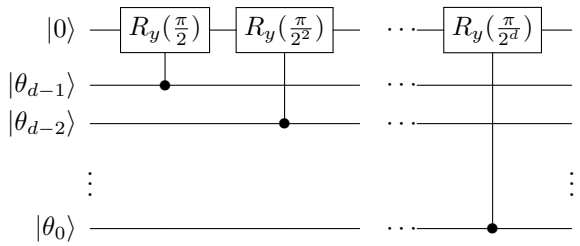


Figure A.1: Circuit diagram for Equation A.1.1

□

A.2 Closed-form Solution for OLS

We have to show that assuming (A^TA) is non-singular, $f(x) := \|Ax - b\|^2$ is minimized at $x = (A^TA)^{-1}A^Tb$.

Proof.

$$f(x) := \|Ax - b\|^2 \tag{A.2.1}$$
$$= (Ax - b)^T(Ax - b) \tag{A.2.2}$$
$$= x^TA^TAx - x^TA^Tb - b^TAx + b^Tb \tag{A.2.3}$$
$$\implies \nabla f(x) = 2A^TAx - 2A^Tb \tag{A.2.4}$$

Setting $\nabla f(x) = 0$ gives us

$$2A^TAx - 2A^Tb = 0 \tag{A.2.5}$$
$$\implies (A^TA)x = A^Tb \tag{A.2.6}$$
$$\implies x = (A^TA)^{-1}A^Tb \tag{A.2.7}$$

□

Bibliography

[1] David Deutsch. *The beginning of infinity: Explanations that transform the world*. Penguin UK, 2011.

[2] Scott Aaronson. The complexity of quantum states and transformations: From quantum money to black holes, 2016. URL https://arxiv.org/abs/1607.05256.

[3] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. URL https://mitpress.mit.edu/books/machine-learning-1.

[4] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, USA, 1994.

[5] H.W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Mathematics and Its Applications. Springer Netherlands, 1996. ISBN 9780792341574. URL https://link.springer.com/book/9780792341574.

[6] William J. Hemmerle. An explicit solution for generalized ridge regression. *Technometrics*, 17(3):309–314, 1975. ISSN 00401706. URL http://www.jstor.org/stable/1268066.

[7] Martin Hanke and Per Christian Hansen. Regularization methods for large-scale problems. *Surv. Math. Ind*, 3(4):253–315, 1993.

[8] Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1): 108–116, 1995. doi:10.1162/neco.1995.7.1.108.

[9] Gene H. Golub, Per Christian Hansen, and Dianne P. O'Leary. Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications*, 21(1):185–194, 1999. doi:10.1137/S0895479897326432. URL https://doi.org/10.1137/S0895479897326432.

[10] Wessel N. van Wieringen. Lecture notes on ridge regression, 2015. URL https://doi.org/10.48550/ARXIV.1509.09169.

[11] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000. ISSN 00401706. URL http://www.jstor.org/stable/1271436.

[12] Donald W. Marquardt. Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12(3):591–612, 1970. ISSN 00401706. URL http://www.jstor.org/stable/1267205.

[13] Hrishikesh D. Vinod. A survey of ridge regression and related techniques for improvements over ordinary least squares. *The Review of Economics and Statistics*, 60(1):121–131, 1978. ISSN 00346535, 15309142. URL http://www.jstor.org/stable/1924340.

[14] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982. ISSN 1572-9575. doi:10.1007/BF02650179. URL https://doi.org/10.1007/BF02650179.

[15] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information, 2010.

[16] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992. doi:10.1098/rspa.1992.0167. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1992.0167.

[17] null null, Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Benjamin Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Edward Farhi, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Matthew P. Harrigan, Alan Ho, Sabrina Hong, Trent Huang, William J. Huggins, Lev Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Erik Lucero, Orion Martin, John M. Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mruczkiewicz, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Hartmut Neven, Murphy Yuezhen Niu, Thomas E. O'Brien, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Doug Strain, Kevin J. Sung, Marco Szalay, Tyler Y. Takeshita, Amit Vainsencher, Theodore White, Nathan Wiebe, Z. Jamie Yao, Ping Yeh, and Adam Zalcman. Hartree-fock on a superconducting qubit quantum computer. *Science*, 369(6507):1084–1089, 2020. doi:10.1126/science.abb9811. URL https://www.science.org/doi/abs/10.1126/science.abb9811.

[18] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, sep 2017. doi:10.1038/nature23879. URL https://doi.org/10.1038%2Fnature23879.

[19] P. J. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, P. V. Coveney, P. J. Love, H. Neven, A. Aspuru-Guzik, and J. M. Martinis. Scalable quantum simulation of molecular energies. *Phys. Rev. X*, 6:031007, Jul 2016. doi:10.1103/PhysRevX.6.031007. URL https://link.aps.org/doi/10.1103/PhysRevX.6.031007.

[20] J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi. Computation of molecular spectra on a quantum processor with an error-resilient algorithm. *Phys. Rev. X*, 8:011021, Feb 2018. doi:10.1103/PhysRevX.8.011021. URL https://link.aps.org/doi/10.1103/PhysRevX.8.011021.

[21] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, Sep 2017. ISSN 1476-4687. doi:10.1038/nature23474. URL http://dx.doi.org/10.1038/nature23474.

[22] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Phys. Rev. Lett.*, 109: 050505, Aug 2012. doi:10.1103/PhysRevLett.109.050505. URL https://doi.org/10.1103/PhysRevLett.109.050505.

[23] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), Oct 2009. ISSN 1079-7114. doi:10.1103/physrevlett.103.150502. URL https://doi.org/10.1103/physrevlett.103.150502.

[24] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, Sep 2014. ISSN 1745-2481. doi:10.1038/nphys3029. URL https://doi.org/10.1038/nphys3029.

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

[26] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, Nov 2019. ISSN 2058-9565. doi:10.1088/2058-9565/ab4eb5. URL https://dx.doi.org/10.1088/2058-9565/ab4eb5.

[27] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, feb 2014. doi:10.1088/1751-8113/47/10/105301. URL https://doi.org/10.1088%2F1751-8113%2F47%2F10%2F105301.

[28] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, Apr 2015. ISSN 1745-2481. doi:10.1038/nphys3272. URL https://doi.org/10.1038/nphys3272.

[29] Ashley Montanaro and Sam Pallister. Quantum algorithms and the finite element method. *Physical Review A*, 93(3), mar 2016. doi:10.1103/physreva.93.032324. URL https://doi.org/10.1103%2Fphysreva.93.032324.

[30] Guoming Wang. Quantum algorithm for linear regression. *Phys. Rev. A*, 96:012335, Jul 2017. doi:10.1103/PhysRevA.96.012335. URL https://doi.org/10.1103/PhysRevA.96.012335.

[31] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. Prediction by linear regression on a quantum computer. *Phys. Rev. A*, 94:022342, Aug 2016. doi:10.1103/PhysRevA.94.022342. URL https://doi.org/10.1103/PhysRevA.94.022342.

[32] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Phys. Rev. A*, 101:022316, Feb 2020. doi:10.1103/PhysRevA.101.022316. URL https://doi.org/10.1103/PhysRevA.101.022316.

[33] Anupam Prakash. *Quantum Algorithms for Linear Algebra and Machine Learning*. PhD thesis, EECS Department, University of California, Berkeley, Dec 2014. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.html.

[34] Iordanis Kerenidis and Anupam Prakash. Quantum Recommendation Systems. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:21, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-029-3. doi:10.4230/LIPIcs.ITCS.2017.49.

[35] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-109-2. doi:10.4230/LIPIcs.ICALP.2019.33.

[36] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, Jul 2019. ISSN 2521-327X. doi:10.22331/q-2019-07-12-163. URL https://doi.org/10.22331/q-2019-07-12-163.

[37] Chao-Hua Yu, Fei Gao, and Qiao-Yan Wen. An improved quantum algorithm for ridge regression. *IEEE Transactions on Knowledge and Data Engineering*, 33(3):858–866, 2021. doi:10.1109/TKDE.2019.2937491.

[38] Changpeng Shao and Hua Xiang. Quantum regularized least squares solver with parameter estimate. *Quantum Information Processing*, 19(4):113, Feb 2020. ISSN 1573-1332. doi:10.1007/s11128-020-2615-9. URL https://doi.org/10.1007/s11128-020-2615-9.

[39] Menghan Chen, Chaohua Yu, Gongde Guo, and Song Lin. Faster quantum ridge regression algorithm for prediction. *International Journal of Machine Learning and Cybernetics*, Apr 2022. ISSN 1868-808X. doi:10.1007/s13042-022-01526-6. URL https://doi.org/10.1007/s13042-022-01526-6.

[40] Yanlin Chen and Ronald de Wolf. Quantum algorithms and lower bounds for linear regression with norm constraints. *arXiv preprint*, 2021. doi:10.48550/ARXIV.2110.13086.

[41] András Gilyén, Zhao Song, and Ewin Tang. An improved quantum-inspired algorithm for linear regression, 2020. URL https://doi.org/10.48550/ARXIV.2009.07268.

[42] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 193–204, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367059. doi:10.1145/3313276.3316366. URL https://doi.org/10.1145/3313276.3316366.

[43] Guang Hao Low and Isaac L. Chuang. Optimal hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, Jan 2017. doi:10.1103/PhysRevLett.118.010501. URL https://doi.org/10.1103/PhysRevLett.118.010501.

[44] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2:040203, Dec 2021. doi:10.1103/PRXQuantum.2.040203.

[45] Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 636–647, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-35-4. doi:10.4230/LIPIcs.STACS.2012.636.

[46] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, Jan 2017. ISSN 1095-7111. doi:10.1137/16M1087072. URL https://doi.org/10.1137/16M1087072.

[47] Ronald de Wolf. Quantum computing: Lecture notes, 2019. URL https://arxiv.org/abs/1907.09415.

[48] Prof. Frederic Schuller. Lectures on quantum theory. https://www.youtube.com/playlist?list=PLPH7f_7ZlzxQVx5jRjbfRGEzWY_upS5K6, 2016.

[49] David Elieser Deutsch and Roger Penrose. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989. doi:10.1098/rspa.1989.0099. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1989.0099.

[50] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563–591, May 1980. ISSN 1572-9613. doi:10.1007/BF01011339. URL https://doi.org/10.1007/BF01011339.

[51] Paul Benioff. Quantum mechanical models of turing machines that dissipate no energy. *Phys. Rev. Lett.*, 48:1581–1585, Jun 1982. doi:10.1103/PhysRevLett.48.1581. URL https://link.aps.org/doi/10.1103/PhysRevLett.48.1581.

[52] J. Brooke, D. Bitko, T. F., null Rosenbaum, and G. Aeppli. Quantum annealing of a disordered magnet. *Science*, 284(5415):779–781, 1999. doi:10.1126/science.284.5415.779. URL https://www.science.org/doi/abs/10.1126/science.284.5415.779.

[53] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Journal on Computing*, 37(1):166–194, 2007. doi:10.1137/S0097539705447323. URL https://doi.org/10.1137/S0097539705447323.

[54] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Phys. Rev. E*, 58:5355–5363, Nov 1998. doi:10.1103/PhysRevE.58.5355. URL https://link.aps.org/doi/10.1103/PhysRevE.58.5355.

[55] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001. doi:10.1126/science.1057726. URL https://www.science.org/doi/abs/10.1126/science.1057726.

[56] John Preskill. Quantum computing and the entanglement frontier. *arXiv:1203.5813*, 2012. doi:10.48550/arXiv.1203.5813.

[57] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Phys. Rev. X*, 6:031015, Aug 2016. doi:10.1103/PhysRevX.6.031015. URL https://link.aps.org/doi/10.1103/PhysRevX.6.031015.

[58] Joshua Job and Daniel Lidar. Test-driving 1000 qubits. *Quantum Science and Technology*, 3(3):030501, 2018.

[59] Sergio Boixo, Troels F Rønnow, Sergei V Isakov, Zhihui Wang, David Wecker, Daniel A Lidar, John M Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10(3):218, 2014.

[60] Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 2014. doi:10.1126/science.1252319. URL https://www.science.org/doi/abs/10.1126/science.1252319.

[61] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 217–228, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367059. doi:10.1145/3313276.3316310. URL https://doi.org/10.1145/3313276.3316310.

[62] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. Methodology of resonant equiangular composite quantum gates. *Phys. Rev. X*, 6:041067, Dec 2016. doi:10.1103/PhysRevX.6.041067.

[63] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. *arXiv preprint*, jun 2018. doi:10.48550/arXiv.1806.01838. URL https://doi.org/10.48550/arXiv.1806.01838.

[64] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by uniform spectral amplification. *arXiv:1707.05391*, 2017. doi:10.48550/ARXIV.1707.05391.

[65] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006. doi:10.1137/S0097539704445226. URL https://doi.org/10.1137/S0097539704445226.

[66] Yimin Ge, Jordi Tura, and J. Ignacio Cirac. Faster ground state preparation and high-precision ground energy estimation with fewer qubits. *Journal of Mathematical Physics*, 60(2):022202, 2019. doi:10.1063/1.5027484. URL https://doi.org/10.1063/1.5027484.

[67] Lin Lin and Yu Tong. Near-optimal ground state preparation. *Quantum*, 4:372, December 2020. ISSN 2521-327X. doi:10.22331/q-2020-12-14-372. URL https://doi.org/10.22331/q-2020-12-14-372.

[68] Guang Hao Low. *Quantum signal processing by single-qubit dynamics*. PhD thesis, Massachusetts Institute of Technology, 2017. URL http://hdl.handle.net/1721.1/115025.

[69] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, November 2020. ISSN 2521-327X. doi:10.22331/q-2020-11-11-361. URL https://doi.org/10.22331/q-2020-11-11-361.

[70] Zane M. Rossi and Isaac L. Chuang. Multivariable quantum signal processing (m-qsp): prophecies of the two-headed oracle. *arXiv preprint*, 2022. doi:10.48550/ARXIV.2205.06261.

[71] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. *Sampling-Based Sublinear Low-Rank Matrix Arithmetic Framework for Dequantizing Quantum Machine Learning*, page 387–400. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450369794. URL https://doi.org/10.1145/3357713.3384314.

[72] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, November 2020. ISSN 2521-327X. doi:10.22331/q-2020-11-11-361. URL https://doi.org/10.22331/q-2020-11-11-361.

[73] Dong An and Lin Lin. Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm. *ACM Transactions on Quantum Computing*, 3(2), mar 2022. ISSN 2643-6809. doi:10.1145/3498331. URL https://doi.org/10.1145/3498331.

[74] Guang Hao Low and Nathan Wiebe. Hamiltonian simulation in the interaction picture. *arXiv preprint*, 2018. doi:10.48550/arXiv.1805.00675.

[75] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113:130503, Sep 2014. doi:10.1103/PhysRevLett.113.130503.