University of Ottawa

Department of Mathematics

Summer 2019 Undergraduate Research Project

Quantum Computing Algorithms for Applied Linear Algebra

Author
James Dickens

Supervisor
Dr. Hadi Salmasian

August 19, 2019



Contents

Contents **1**

1 Introduction: **4**

2 Quantum Computing Concepts: **6**

2.1 Qubits: . . . 6
2.2 Hilbert Spaces: . . . 6
2.3 Operations on Qubits/Elementary Properties of Unitary Matrices: . . . 7
2.4 Tensor Products and Multiple Qubits: . . . 7
2.5 Dirac Notation and Outer Products: . . . 8
2.6 Partial Measurements: . . . 9

3 Basic Quantum Computing Algorithms: **9**

3.1 Superdense Coding: . . . 10
3.2 Quantum Teleportation Algorithm: . . . 10
3.3 Deutsch's Algorithm: . . . 12
3.3.1 Hadamard Transforms: . . . 12
3.3.2 Deutch's Algorithm: . . . 13
3.4 Deutsch-Jozsa Algorithm: . . . 13
3.5 Simon's Algorithm: . . . 15
3.6 Quantum Gates and the Bloch Sphere: . . . 17
3.6.1 Bloch Sphere: . . . 17
3.6.2 Pauli Gates: . . . 17
3.6.3 Controlled U-gates: . . . 18
3.6.4 Toffoli Gate: . . . 18
3.6.5 Rotation Operators: . . . 18
3.7 Discrete Fourier Transform/Quantum Fourier Transform: . . . 19
3.7.1 Concepts and Definitions: . . . 19
3.7.2 Implementing the Quantum Fourier Transform: . . . 20
3.8 Phase Estimation: . . . 21
3.8.1 Performance and requirements: . . . 23
3.9 Order Finding Using Phase Estimation: . . . 24
3.10 Shor's Algorithm and Factoring Discussion: . . . 26
3.11 Grover's Algorithm: . . . 27

4 Hamiltonian Simulation: **30**

4.1 Efficient Quantum Algorithms for Simulating Sparse Hamiltonians: . . . 31
4.2 Related Concepts: . . . 31

4.3 Problem 1: . . . 32
4.4 Finding Exponential Product Formulas of Higher Orders: . . . 33
4.5 Higher-Order integrators: . . . 33
4.6 Proving Lemma 1: . . . 35
4.7 Problem 2: . . . 36
4.8 Lemma 2: . . . 37
4.9 Proof of Theorem 2: . . . 41

5 Amplitude Amplification: 41
5.1 Quadratic Speedup: . . . 41
5.2 QSearch: . . . 44
5.3 Controlled Rotations: . . . 44

6 Oracle QRAM: 46
6.1 Oracle QRAM and Amplitude Amplification: . . . 47

7 HHL algorithm: 48
7.1 Algorithm Sketch: . . . 48
7.2 Algorithm Details: . . . 50
7.3 The Proposed Filter Functions are Lipschitz Continuous: . . . 54
7.4 Error Analysis: . . . 55
7.5 Matrix Inversion is BQP-Complete: . . . 55

8 Compiling Basic Linear Algebra Subroutines for Quantum Computers: 58
8.1 Assumptions of the paper: . . . 58
8.2 Embedding: . . . 59
8.3 Quantum Inner Product Estimation: . . . 60
8.4 Sub-routine 1: . . . 61
8.5 Sub-routine 2: . . . 62
8.6 Exponential of the Kronecker Sum: . . . 63
8.7 Exponentiation of the Tensor Product of Matrices: . . . 64
8.8 Exponentiation of the Hadamard Product: . . . 64

9 Appendix: 66
9.1 Elementary Complexity Theory, Turing Machines, and the class BQP: . . . 66
9.1.1 Concepts: . . . 66
9.1.2 Formal Definition of a Turing Machine: . . . 66
9.1.3 The Halting Problem and an Uncomputable Function: . . . 68
9.1.4 Basic Complexity Classes and the Class BQP: . . . 69

9.2 Discussion of Relevant Linear Algebra material: 70
9.2.1 Some Properties of the Exponential of a Matrix: 70
9.2.2 Singular Value Decomposition: 71
9.2.3 Existence: 72
9.2.4 Matrix Norms: 73
9.2.5 Element-wise norms: 73
9.2.6 Induced or Operator Norms: 73
9.2.7 The Schatten norm: 74
9.2.8 The Condition Number of a Matrix: 74
9.3 Approximation by Rational Numbers: 75
9.3.1 Finite Continued Fractions: 75
9.3.2 Infinite Continued Fractions: 76

10 References: **82**

1 Introduction:

Quantum computers are devices that utilize the principles of quantum mechanics to perform computations. Through the use of superposition and entanglement of quantum states, researchers have constructed descriptions, simulations, and real implementations of quantum circuits used to perform algorithms commonly employed in modern computer science. One of the landmark achievements of quantum computing is the development of Shor's algorithm, an integer factorization approach that operates in polylogarithmic time complexity, an exponential speed up when compared to the best currently available classical algorithms. Qubits, the quantum analogue of the bit, can be represented with two-level quantum mechanical systems such as the polarization encoding of light, the spin of an eletron, or the dot spin of a quantum dot. One of the main motivations of research into this topic by mathematicians, physicists, and computer scientists alike is the development of algorithms for quantum computing that could offer faster time complexity than classical algorithms, to achieve a so-called *quantum supremacy*.

The aim of this report is to gain familiarity with fundamental quantum computing concepts and algorithms, and to present a selection of modern quantum algorithms in the context of applied linear algebra. For general computation, efficient processing of basic linear algebra operations is paramount for a wide range of problems. Indeed solving linear equations is an essential tool in modern science and engineering, and hence is the motivation for this report's substance.

- **Section 2** presents elementary quantum computing concepts. In particular the notion of a qubit is introduced alongside with the relevant operators on qubits used in quantum circuits, along with other concepts.

- **Section 3** presents a sample of commonly studied quantum algorithms, leading notably to the *phase estimation* algorithm used in Shor's factoring algorithm, which is also presented. The complexity of some algorithms is given in detail, while others are stated without proof. Additionally a selection of important quantum gates is introduced.

- **Sections 4-6** introduces the pre-requisite concepts used in the HHL algorithm (section 7), namely the techniques of Hamiltonian Simulation, Amplitude Amplification, and Oracle QRAM (the quantum analogue of Random Access Memory)

- **Section 7** describes the HHL Algorithm, which provides an estimate to the solution of a linear equation $A\vec{x} = \vec{b}$ for A an invertible matrix, and notably a proof that

matrix inversion is **BQP**-complete.

- **Section 8** explores a section of sub-routines presented in the paper *Compiling Basic Linear Algebra Subroutines for Quantum Computers*, in particular quantum inner product estimation, and computing estimates $x^\dagger Ay$ for a complex matrix A satisfying certain condition, alongside vectors x, y. Additionally sub-routines for computing matrix exponentials of Kronecker Sum, the tensor product, and the Hadamard product of two matrices are discussed.

- **The Appendix** consists of three sections. First a discussion of elementary complexity theory is presented, followed by a compilation of relevant linear algebra needed in some of this report. Finally a section on continued fractions is presented, a technique used in the order-finding algorithm presented in section 3.8.

I would like to thank Dr. Hadi Salmasian for assisting me in the process of writing this report and understanding these new concepts. Additionally, I would like to dedicate this work to Dr. Seth Lloyd at MIT, one of the authors of the HHL algorithm, for his brilliant research and informative videos on quantum computing.

2 Quantum Computing Concepts:

Here we will discuss the basic mathematics of quantum computing as presented in the lecture notes provided by John Watrous, as well as the textbook *Quantum Computation and Quantum Information* by Nielsen and Chang [1], [2], with supplementary definitions provided by additional resources which are cited as needed.

2.1 Qubits:

A single qubit is the quantum analgoue of a bit, represented by a vector in $\mathbb{C}^2$ of norm 1, for example $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$. The vector represents what is referred to as a superposition or a state. The components of the vector are referred to as *amplitudes*, and as we are representing a two-level quantum mechanical system, upon measurement, the qubit $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ will yield one of two outcomes. The two outcomes can be said to be indexed by a set of states $\{0, 1\}$ where $|\alpha|^2$ corresponds to the probability of measuring state 0 (represented by $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$), and $|\beta|^2$ corresponds to the probability of measuring 1, (represented by $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$).

The **quantum superposition principle** states that if a quantum system may be in one of many configurations—arrangements of particles or fields—then the most general state is a combination of all of these possibilities, where the amount in each configuration is specified by a complex number.

Pure quantum states correspond to vectors in a Hilbert space, where each observable quantity (a physical quantity that can be measuread) is associated with a mathematical operator. The eigenvalues of the operator correspond to possible values of the observable. The corresponding eigenvector is referred to in physics literature as an eigenstate.

2.2 Hilbert Spaces:

A Hilbert space H [3] is a real or complex inner product space that is also a complete metric space with respect to the distance function induced by the inner product. A complex inner product space has an inner product $\langle \cdot \rangle$ that satisfies

1. $\langle x, y \rangle = \overline{\langle y, x \rangle}$
2. For all complex a, b, vectors x_1, x_2, y, that $\langle ax_1 + bx_2, y \rangle = a\langle x_1, y \rangle + b\langle x_2, y \rangle$
3. $\langle x, x \rangle = 0 \iff x = 0$.

Hilbert spaces have a norm given by $||x|| = \sqrt{\langle x, x \rangle}$, and a distance function induced by the norm defined as $d(x, y) = ||x - y||$. A complete metric space is a metric space in which every Cauchy sequence is convergent.

2.3 Operations on Qubits/Elementary Properties of Unitary Matrices:

In order to perform operations on a qubit (or qubits), the operator in question must preserve the length of the vector, and so operations on qubits are represented by a unitary matrix $U \in Mat_n(\mathbb{C})$ with the property that $UU^\dagger = I_{n \times n}$, where $U^\dagger$ is the conjugate transpose of U. Recall that in $\mathbb{C}^n$ the dot product of two n-vectors u, v is $u^\dagger v = \sum_{i=0}^{n} {u_i}^\dagger v_i$, where $u^\dagger$ is the conjugate tranpose of the vector u. Three important properties of unitary matrices of dimension n will be used in these notes, which relate to the well known spectral theorem for unitary matrices:

1) Unitary matrices preserve the length of vectors, i.e $||Uv|| = ||v||$
2) The eigenvalues of unitary matrices corresponding to distinct eigenvectors are orthogonal, and have modulus 1.
3) There exists a basis of orthonormal eigenvectors of U, say $\{u_1, u_2, \cdots, u_n\}$.

pf of 1) $(\langle Ux, Ux \rangle)^2 = (x^\dagger U^\dagger \cdot Ux)^2 = (x^\dagger \cdot (U^\dagger)(U) \cdot x)^2 = (x^\dagger \cdot x)^2 = (\langle x, x \rangle)^2$.

2.4 Tensor Products and Multiple Qubits:

Bilinear map: If V, W, X are vector spaces over the same field F, a bilinear function $f : V \times W \to X$ is a function such that for any $w \in W$, the map $F_W : v \to B(v, w)$ is a linear map from V to X, and for any $v \in V$ the map $f_V : w \to B(v, w)$ is a linear map from W to X.

Tensor Product: If U, V are finite dimension real or complex vector spaces, a tensor product of U, V is a vector space W together with a bi-linear map $f : U \times V \to W$ with the property that if g is any bilinear map from $U \times V$ to a vector space X, there exists a unique map h from W to X, referred to as the *universal property.*

If (W, ϕ) is a tensor product and $\{e_1 \cdots e_n\}$ is a basis for U and $\{f_1 \cdots f_m\}$ is a basis for V, then $\{\phi(e_j, f_k) | 1 \leq j \leq n, 1 \leq k \leq m\}$ is a basis for W, where $dim(U \otimes V) = (dimU)(dimV)$. The basis elements of W can be written as $u \otimes v$

instead of $\phi(u, v)$ where $e_j \otimes f_k$ is a basis for W. [4].

Applied to matrices, the tensor product, sometimes referred to as the Kronecker product, is defined for matrices $A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ \cdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix}$ and $B = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,l} \\ \cdots & & & \\ b_{k,1} & b_{k,2} & \cdots & b_{k,l} \end{bmatrix}$ as $A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,m}B \\ \cdots & & & \\ a_{n,1}B & a_{n,2}B & \cdots & a_{n,m}B \end{bmatrix}$.

Some elementary properties of tensor products are :
1) Tensor products are associative, i.e. $(A \otimes B) \otimes C = A \otimes (B \otimes C)$.
2) $(A \otimes B) \times (C \otimes D) = (A \times C) \otimes (B \times D)$, where $\times$ refers to standard matrix multiplication. This is sometimes referred to as the mixed product property
3) For a scalar α, it is true that $(\alpha A) \otimes B = A \otimes (\alpha B) = \alpha(A \otimes B)$.
4) $A \otimes (B + C) = A \otimes B + A \otimes C$.
5) $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$

For two qubits represented by the quantum states $X = \begin{bmatrix} a & b \end{bmatrix}^\dagger, Y = \begin{bmatrix} c & d \end{bmatrix}^\dagger$, the superposition of the pair of quantum states (X, Y) is given by $X \otimes Y = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$, which represents two qubits. The possible outcomes upon measurement are indexed by the set $\{00, 01, 10, 11\}$ corresponding to vectors $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\dagger$, $\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^\dagger$, $\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^\dagger$, $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^\dagger$ respectively. For a quantum superposition $\begin{bmatrix} a & b & c & d \end{bmatrix}^\dagger$, the probability of measuring 00 is given by $|a|^2$, and for 01 is $|b|^2$ and so on.

In general, for n, m qubits represented by quantum states x, y, the superposition of $x \otimes y$ yields $n + m$ qubits where the possible outcomes after measurement are indexed by the set $\{0, 1\}^{n+m}$ with corresponding basis vectors $e_i \in C^{2^{n+m}}$, where for a given superposition $\begin{bmatrix} a_1 & \dots & a_{2^{n+m}} \end{bmatrix}^\dagger$, the probability of measuring e_i is $|a_i|^2$.

2.5 Dirac Notation and Outer Products:

Dirac notation is used to represent vectors efficiently, and is commonplace notation in physics. Define: $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, which are referred to as *kets*. We use symbols $|\theta\rangle$, $|\phi\rangle$ to represent arbitrary vectors. A single qubit can be represented by

a linear combination of $|0\rangle$ and $|1\rangle$, ex: $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Juxtaposition of kets refers to the tensor product, $|\theta\rangle|\phi\rangle = |\theta\rangle \otimes |\phi\rangle$, where
$|00\rangle := |0\rangle|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $|01\rangle := |0\rangle|1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Generally for a vector of dimension $2^n \times 1$ with entries indexed by bit strings of length n, say $|\phi\rangle$, we can write $|\phi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$, for complex numbers a_x, where in the case of quantum states we know that $\sum_{x \in \{0,1\}^n} |a_x|^2 = 1$.

A *bra* of a ket $|\phi\rangle$, denoted by $\langle\phi|$, consists of the conjugate transpose of $|\phi\rangle$, and the inner product of two vectors $|x\rangle, |y\rangle$ is represented by $\langle x|y\rangle$.

Define an **outer product** of two kets $|x\rangle, |y\rangle$ denoted as $|x\rangle\langle y| = |x\rangle \otimes \langle y|$, where the tensor product of a column vector by a row vector results in a matrix. The identity $|\phi\rangle\langle x|y\rangle = \langle x|y\rangle|\phi\rangle$ holds since the inner product is a complex number (a scalar in the field).

2.6 Partial Measurements:

In a system consisting of two or more qubits, it is possible to measure only one of them. Demonstrating by example, consider a state $|\phi\rangle = \frac{1}{2}|00\rangle - \frac{i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle$. By measuring the first qubit (the leftmost qubit in Dirac notation), if we measure $|0\rangle$, then the probability of measuring the other qubit to be $|0\rangle$ is equal to $||\frac{1}{2}|00\rangle||^2 = \frac{1}{4}$, and the state of the two qubits after the measurement conditioned on the measurement outcome being $|0\rangle$ is the vector itself renormalised (divided by its length), i.e. $\frac{\frac{1}{2}|00\rangle}{||\frac{1}{2}|00\rangle||} = |00\rangle$. Measuring the first qubit as $|1\rangle$ is handled simiarly so that the state of the two qubits becomes $\frac{\frac{-i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle}{||\frac{-i}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle||}$.

3 Basic Quantum Computing Algorithms:

In this section, a section of commonly studied quantum computing algorithms is presented according to the notes provided by John Watrous (University of Waterloo), and supplemental material provided by the textbook *Quantum Computation and Quantum Information* by Michael A. Nielsen and Isaac L. Chuang [1], [2].

3.1 Superdense Coding:

In quantum information theory, superdense coding is a quantum communication protocol to transmit two classical bits of information (i.e. $00, 01, 10, 11$) from a sender (often called Alice) to a receiver (often called Bob), by sending only one qubit from Alice to Bob, under the assumption of Alice and Bob pre-sharing an entangled state. A superposition of qubits X, Y, is called an *entanglement* if it cannot be written as a tensor product of qubits. In the superdense coding protocol, Alice and Bob share an e-bit (A, B) (also referred to as an EPR pair) of qubits in the form: $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, where Alice takes A, and Bob takes B.

The protocol operates as follows:
Let $(ab)_2$ be the classical bits that Alice wishes to send.

1. If $a = 1$, Alice applies the unitary transformation $\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ to the qubit A

2. If $b = 1$, Alice applies the unitary transformation $\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ to the qubit A

3. Alice sends the qubit A to Bob

4. Bob applies a controlled-NOT operation to the pair (A, B), where A is the control and B is the target. Controlled operations will be discussed later, but for now this operation has the corresponding unitary matrix

$$\textbf{CNOT} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

5. Bob applies a Hadamard transform to A.

6. Bob measure both qubits A and B, and the output of the measurement will correspond to two states a, b which can be interpreted as $(a, b)_2$. See figure 1 for why this is the case.

3.2 Quantum Teleportation Algorithm:

Suppose that Alice has a qubit that she wants to send Bob, and we give Alice and Bob the additional resource of sharing an e-bit, as in the superdense coding algorithm, then

ab	state after step 1	state after step 2	state after step 4	state after step 5
00	$\frac{1}{\sqrt{2}}\|00\rangle + \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle + \frac{1}{\sqrt{2}}\|11\rangle$	$\left(\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle\right)\|0\rangle$	$\|00\rangle$
01	$\frac{1}{\sqrt{2}}\|00\rangle + \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|10\rangle + \frac{1}{\sqrt{2}}\|01\rangle$	$\left(\frac{1}{\sqrt{2}}\|1\rangle + \frac{1}{\sqrt{2}}\|0\rangle\right)\|1\rangle$	$\|01\rangle$
10	$\frac{1}{\sqrt{2}}\|00\rangle - \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|00\rangle - \frac{1}{\sqrt{2}}\|11\rangle$	$\left(\frac{1}{\sqrt{2}}\|0\rangle - \frac{1}{\sqrt{2}}\|1\rangle\right)\|0\rangle$	$\|10\rangle$
11	$\frac{1}{\sqrt{2}}\|00\rangle - \frac{1}{\sqrt{2}}\|11\rangle$	$\frac{1}{\sqrt{2}}\|10\rangle - \frac{1}{\sqrt{2}}\|01\rangle$	$\left(\frac{1}{\sqrt{2}}\|1\rangle - \frac{1}{\sqrt{2}}\|0\rangle\right)\|1\rangle$	$-\|11\rangle$

When Bob measures at the end of the protocol, it is clear that he sees ab as required.

Figure 1: Superdense Coding Protocol outcomes [1]

it becomes possible for Alice to transmit a qubit to Bob using classical communication by means of so called *quantum teleportation*. Two bits of classical information will be needed to perform this task.

Let $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ be the qubit that we wish to send. First an e-bit is generated, one qubit sent to Alice, one to Bob. The starting state is given by the superposition $(\alpha|0\rangle + \beta|1\rangle)(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle) = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|001\rangle + \beta|100\rangle + \beta|111\rangle)$.

Alice then performs a controlled-NOT operation conditioned on the 2 leftmost qubits, resulting in the state: $\frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle)$.

Next a Hadamard transform is applied to the leftmost qubit, which transforms the state to
$\frac{1}{2}(\alpha|000\rangle + \alpha|100\rangle + \alpha|011\rangle + \alpha|111\rangle + \beta|010\rangle - \beta|110\rangle + \beta|001\rangle - \beta|101\rangle)$
$= \frac{1}{2}|00\rangle(\alpha|0\rangle + \beta|1\rangle) + \frac{1}{2}|01\rangle(\alpha|1\rangle + \beta|0\rangle) + \frac{1}{2}|10\rangle(\alpha|0\rangle - \beta|1\rangle) + \frac{1}{2}|11\rangle(\alpha|1\rangle - \beta|0\rangle)$.

Define:
- a NOT operation on a single qubit as the unitary matrix
$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
- σ_z operation defined as
$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

Since the probability of measuring $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are each equal by the above, we have four cases to consider upon measurement of the two leftmost qubits, each with probability $\frac{1}{4}$ of occurring.

1) Alice measures 00, and so the state of the three qubits is $|00\rangle(\alpha|0\rangle + \beta|1\rangle)$. Alice transmits 00 to Bob who does nothing, whence his qubit is in the state $\alpha|0\rangle + \beta|1\rangle)$.
2) Alice measures 01 in which case she transmits the classical bits 01 to Bob, and so his qubit is now in the state $|01\rangle(\alpha|1\rangle + \beta|0\rangle)$, upon which he performs a NOT operation, resulting in the desired qubit $\alpha|0\rangle + \beta|1\rangle$.
3) Alice measures 10, and so the state of the three qubits becomes $|10\rangle(\alpha|0\rangle - \beta|1\rangle)$. Alice transmits the classical bits 10 to Bob, who performs a σ_z operation on his qubit, resulting in the desired state $\alpha|0\rangle + \beta|1\rangle$.
4) Alice measure 11. In this case the state of the three qubits becomes $|11\rangle(\alpha|1\rangle - \beta|0\rangle)$, and upon transmitting 11 to Bob, he performs a NOT operation follwoed by a σ_z operation, transforming it to the state $\alpha|0\rangle + \beta|1\rangle$.

3.3 Deutsch's Algorithm:

Suppose we have a function $f : \{0,1\} \to \{0,1\}$, i.e a bit-string of length 1 mapped to a bit string of length 1. We say the function is balanced if one bit maps to 0, and the other maps to 1, and otherwise the function is constant. With classical computation, you need two queries to determine if f is balanced or constant, whereas in quantum computation it can be down with one query using Deutsch's algorithm.

3.3.1 Hadamard Transforms:

The Hadamard transform H_m is a real $2^m \times 2^m$ sized matrix, defined recursively as $H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}$, and notably $H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$, where we consider $H = H_1$ in these notes. Further note that we define
$|+\rangle := H|0\rangle = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}|1\rangle$ and $|-\rangle := H|1\rangle = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}|1\rangle$.

More generally, for $a \in \{0,1\}$, it is true that $H|a\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^a|1\rangle$. If we have two qubits in state $|x\rangle$ where $x = x_1x_2 \in \{0,1\}^2$, and apply Hadamard transforms to both qubits, we obtain:
$(H \otimes I)(I \otimes H)|x\rangle = (H \otimes H)|x\rangle = (\frac{1}{\sqrt{2}} \sum_{y_1 \in \{0,1\}} (-1)^{x_1 \cdot y_1}|y_1\rangle)(\frac{1}{\sqrt{2}} \sum_{y_2 \in \{0,1\}} (-1)^{x_2 \cdot y_2}|y_2\rangle)$
$= \frac{1}{2} \sum_{y \in \{0,1\}^2} (-1)^{x_1y_1 + x_2y_2}|y\rangle$.

This pattern generalizes to any number of qubits where $H^{\otimes n} = H \otimes \cdots \otimes H$ (n times), and we have $H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x_1y_1 + \cdots + x_ny_n}|y\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y}|y\rangle$.

3.3.2 Deutch's Algorithm:

Assume we have a black box (sometimes referred to as an oracle) which is a quantum gate B_f defined as $B_f|a\rangle|b\rangle = |b \oplus f(a)\rangle$ which is linear, and defined on the elements $|0\rangle, |1\rangle$. The steps for Deutsch's algorithm are as follows:

1. Let $A = |0\rangle$, apply a Hadamard transform to A such that
$H|0\rangle = (\frac{1}{\sqrt{(2)}}|0\rangle + \frac{1}{\sqrt{(2)}}|1\rangle)$.
Let $B = |0\rangle$, apply a Hadamard transform to B such that
$H|1\rangle = (\frac{1}{\sqrt{(2)}}|0\rangle - \frac{1}{\sqrt{(2)}}|1\rangle)$.

Consider the tensor product
$H|A\rangle \otimes H|B\rangle = (\frac{1}{\sqrt{(2)}}|0\rangle + \frac{1}{\sqrt{(2)}}|1\rangle)(\frac{1}{\sqrt{(2)}}|0\rangle - \frac{1}{\sqrt{(2)}}|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle|0\rangle - \frac{1}{\sqrt{2}}|0\rangle|1\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle - \frac{1}{\sqrt{2}}|1\rangle|1\rangle$

2. Performing the B_f operation transforms this state to :
$\frac{1}{2}|0\rangle(|0 \oplus f(0)\rangle - (|1 \oplus f(0)\rangle) + \frac{1}{2}|1\rangle(|0 \oplus f(1)\rangle - (|1 \oplus f(1)\rangle)$

$= \frac{1}{2}(-1)^{f(0))}|0\rangle(|0\rangle - |1\rangle) + \frac{1}{2}(-1)^{f(1)}|1\rangle(|0\rangle - |1\rangle)$

$= (\frac{1}{\sqrt{2}}(-1)^{f(0))}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)})|1\rangle(\frac{1}{\sqrt{(2)}}|0\rangle - \frac{1}{\sqrt{(2)}}|1\rangle)$
(Using the fact that $|0 \oplus a\rangle - |1 \oplus a\rangle = (-1)^a(|0\rangle - |1\rangle)$). Also note the appearance of $(-1)^{f(0)}$ and $(-1)^{f(1)}$, referred to as the *phase kick-back* phenomenom. Since the second qubit is unchanged, we discard it. The state of the first qubit is:
$(\frac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle$, which can be written as $(-1)^{f(0)}(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(0)\oplus f(1)}|1\rangle)$.

Next, applying a Hadamard transform results in the state $(-1)^{f(0)}|f(0) \oplus f(1)\rangle$ since for any $a \in \{0, 1\}$ it is true that $H(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^a|1\rangle) = |a\rangle$. After measuring we get $(-1)^{f(0)}|f(0) \oplus f(1)\rangle$ and so if the result is 0 then f is constant, and 1 if f is balanced. This procedure can be generalized to functions from $f : \{0, 1\}^n \to \{0, 1\}$, and is referred to as the Deutch-Jozsa algorithm.

3.4 Deutsch-Jozsa Algorithm:

Assume that we are given a function $f : \{0, 1\}^n \to \{0, 1\}$, where n is some arbitrary positive integer, and f satisfies:
1) f is a constant function either yielding 0 for all inputs, or 1 for all inputs.
2) f is said to be *balanced*, meaning that half the inputs map to 0, and the other half map to 1.

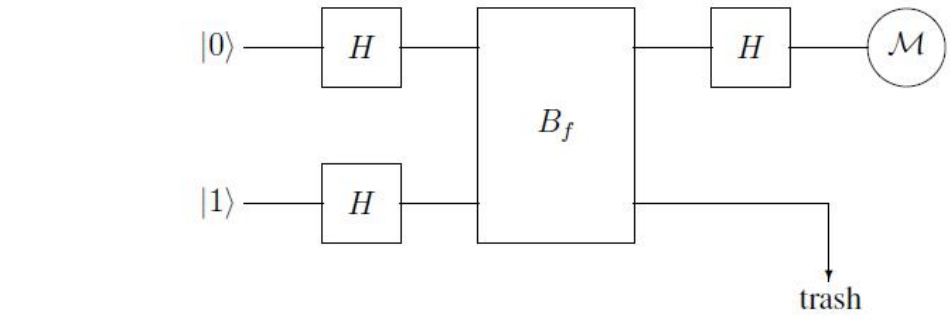


Figure 2: Diagram of Deutch's Algorithm [1]

The goal of this algorithm is to determine which of these two scenarios holds. Assume now that we have an oracle, or a black box, B_f defined as $B_f|x\rangle|b\rangle = |x\rangle|b \oplus f(x)\rangle$ for $x \in \{0,1\}^n$ and $b \in \{0,1\}$.
Classically, we can randomly choose k inputs $x_1, \cdots x_k \in \{0,1\}^n$ and evaluate f at each input. Suppose $f(x_1) = \cdots = f(x_n)$. The total number of ways to choose k distinct inputs is $\binom{2^n}{k}$, so the probability that all k inputs have the same function value for balanced f is:

$$\frac{\binom{2^{n-1}}{k} + \binom{2^{n-1}}{k}}{\binom{2^n}{k}} = \frac{2\binom{2^{n-1}}{k}}{\binom{2^n}{k}}.$$

If we choose k inputs at random with replacement, then they each individually have probability 1/2 to map to 0 and 1/2 to map to 1. So the probability that they all map to 0 or they all map to 1 is $\left(\frac{1}{2}\right)^k + \left(\frac{1}{2}\right)^k = \frac{1}{2^{k-1}}$ for balanced f. In the quantum case, 1 query to B_f will be sufficient to determine with certainty whether the function is constant or balanced. Consider Figure 3.

Keeping in mind the formulas obtained from the Hadamard transform section, the state after the first layer of Hadamard transforms is $(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle)(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$.
After performing the B_f operation on this state, the result is:
$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle(|f(x)|\rangle - |1 \oplus f(x)\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$,
where here we are again seeing the *phase kick-back* effect. Discarding the last qubit and applying n Hadamard transforms as in the diagram, we get the state:
$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}(\frac{1}{\sqrt{2^n}} \sum_{y\in 0,1^n} (-1)^{x\cdot y}|y\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (\frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{f(x)+x\cdot y}|y\rangle)$.

Consider the amplitude associated with the state $|0^n\rangle$ is $\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}$, with norm given by $|\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|^2$, where if $f(x)$ is balanced, half the terms in the sum-

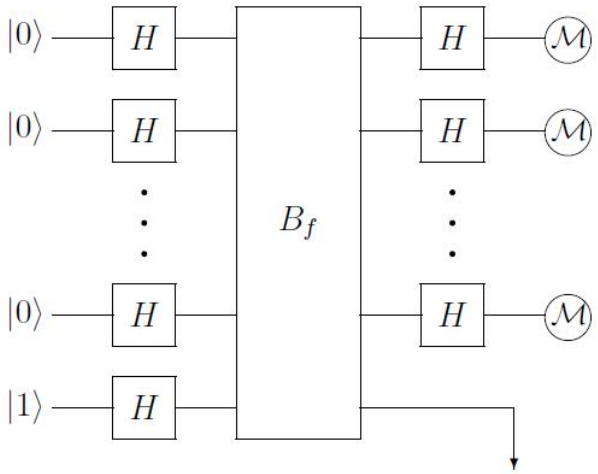


Figure 3: Deutch-Jozsa diagram [1]

mation are -1, and the other half are 1, resulting in an overall sum of 0, and if $f(x)$ is constant, we get a value of 1, thus the algorithm works as claimed with one query to B_f.

3.5 Simon's Algorithm:

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$ that is promised to obey the property that $[f(x) = f(y)] \iff [x \oplus y \in \{0^n, s\}]$. In the case $s = 0^n$, f is required to be a one-to-one function (otherwise it is a two-to-one function, that is, two inputs map to the same output). Note that $x \oplus y = 0^n$ if and only if $x = y$. Classically this is a very difficult problem even if one uses randomness and accepts a small probability of error, where it can be shown that you would need to guess $\Omega(\sqrt{2^n})$ different inputs before being likely to find a pair on which f takes the same output. Simon's algorithm for solving this task consists of iterating the following quantum circuit and doing some classical post-processing.

The circuit begins in the state $|0^n\rangle|0^n\rangle$. Hadamard transforms are performed on the first n qubits. This produces the state $\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|0^n\rangle$. Next we assume the existence of a B_f gate which acts on basis states as $B_f|x\rangle|y\rangle = |x\rangle|f(x) \oplus y\rangle$, where $\oplus$ denotes the bit-wise XOR operation. This gate differs from the previous definition of B_f because the inputs and outputs of f are n-bit strings.

The state after the B_f transformation is performed is $\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle$ since in general $a \oplus 0 = a$ for $a \in \{0,1\}^n$. After applying n Hadamard transforms to this state,

the result is the state $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle|f(x)\rangle$. If $s=0^n$, then f is a one-to-one function, and writing the state above as $\frac{1}{2^n}\sum_{y\in\{0,1\}^n}(\sum_{x\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle|f(x)\rangle)$, we see that the probability of measuring each string y is
$||\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{x\cdot y}|f(x)\rangle||^2=\frac{1}{2^n}$, (noting that $\sqrt{\sum_{i=0}^{2^n}(\frac{1}{2^n})^2}=\sqrt{\frac{1}{2^n}}$).

Now suppose that $s\neq 0^n$, therefore f is not one to one. The probability of measuring a given string is still given by :
$||\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{x\cdot y}|f(x)\rangle||^2$, but in this case there must exist 2 different strings $x_z, x'_z\in\{0,1\}^n$ such that $f(x_z)=f(x'_z)=z$, and by the definition of f, it is true that $x_z\oplus x'_z=s$. Let $A=Im(f)$. Now $(x_z\oplus x'_z)=s \implies x'_z=x_z\oplus s$, and since the XOR operation is distributive over the dot product, then $(x_z\oplus x)\cdot y=(x_z\cdot y)\oplus(s\cdot y)$, so that $||\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{x\cdot y}|f(x)\rangle||^2$
$=||\frac{1}{2^n}\sum_{z\in A}(-1)^{x_z\cdot y}+(-1)^{x'_z\cdot y}|z\rangle||^2$
$=||\frac{1}{2^n}\sum_{z\in A}(-1)^{x_z\cdot y}+(-1)^{(x_z\oplus)y}|z\rangle||^2$
$=||\frac{1}{2^n}\sum_{z\in A}(-1)^{x_z\cdot y}(1+(-1)^{s\cdot y}|z\rangle)||^2$ which equals $2^{-(n-1)}$ if $s\cdot y=0$, and 0 if $s\cdot y=1$.

Hence it follows that the measurement after the second Hadamard transforms always results in a string y that satisfies $s\cdot y=0$.

To summarize thus far, in the case where $s=0^n$, the measurement results in each string $y\in\{0,1\}^n$ with uniform probability $p_y=\frac{1}{2^n}$, and if $s\neq 0^n$, then the probability of obtaining each string y is

$$p_y=\begin{cases}2^{-(n-1)} & ,\text{if } s\cdot y=0\\ 0 & ,\text{if } s\cdot y=1\end{cases}$$

Repeating the algorithm $n-1$ times, you will get $n-1$ strings say $y_1,\ldots,y_{n-1}$ such that we obtain a matrix equation

$$As=0=\begin{cases}y_1\cdot s=0\\ \ldots\\ y_{n-1}\cdot s=0\end{cases}$$

which is a system of $n-1$ equations in n unknowns (the bits of s). The goal is to solve to obtain s. We can the procedure a sufficient number of times until each y_i is linearly independent (ocurring with probability larger than $\frac{1}{4}$). Once linear independence is achieved, we solve the system for non-zero $s'\neq 0^n$, (the rank of the matrix A is $n-1$, and so by rank-nullity its kernel is non-trivial). Now test if $f(0^n)=f(s')$, if it is true that $s=s'$, then the problem is solved. Otherwise $s=0^n$, since the unique non-zero

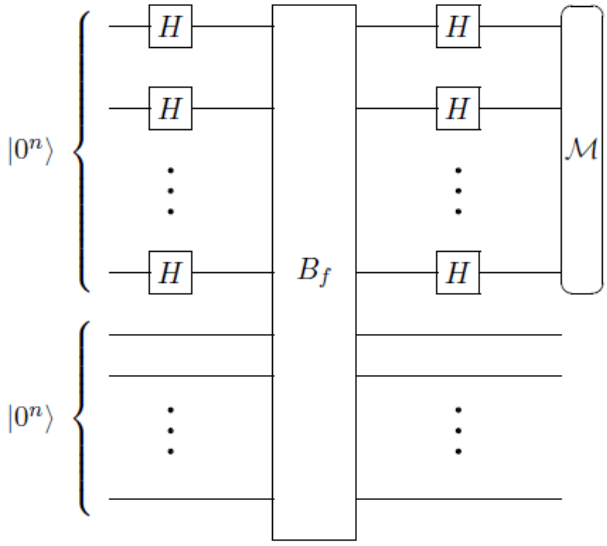


Figure 4: Simon's Algorithm [1]

solution to the linear equations would have been s.

3.6 Quantum Gates and the Bloch Sphere:

3.6.1 Bloch Sphere:

Since for any quantum state $|\psi\rangle$, we know that $\langle\psi|\psi\rangle = 1$, we can write $|\psi\rangle = cos(\theta/2)|0\rangle + e^{i\phi}sin(\theta/2)|1\rangle = cos(\theta/2)|0\rangle + (cos\phi + isin\phi)(sin\theta/2)|1\rangle$ where $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$. This last statement uses the identity $sin^2(\theta) + cos^2(\theta) = 1$, the fact $|e^{i\phi}| = 1$, and Euler's identity. The representation of $\vec{a} = (sin\theta cos\phi, sin\theta sin\phi, cos\theta)$, is said to be the geoemtrical representation on the unit sphere, referred to in this context as the *Bloch Sphere*, for the state $|\psi\rangle$.

3.6.2 Pauli Gates:

The Pauli gates act on a single qubit, they correspond to 2×2 complex matrices which are both Hermitian and unitary, named after the physicist Wolgang Pauli.

Pauli-X gate, σ_x: This is the quantum equivalent of the NOT gate for classical computers given by the matrix :

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Pauli-Y gate, σ_y: This gate equates to a rotation around the Y-axis of the Bloch

sphere by π radians.

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Pauli-Z gate, σ_z: This gate equates to a rotation around the Z-zxis of the Bloch sphere by π radians:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

The following identity holds $I^2 = \sigma_x^2 = \sigma_y^2 = \sigma_x^2 = -i\sigma_x\sigma_y\sigma_z = I$.

3.6.3 Controlled U-gates:

A controlled-U gate is a gate that operates on two qubits where the first bit is a control, and the second will be operated on by U depending on the control.

$$C(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}$$

where

$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$$

is one of Pauli matrices defined above. If the control qubit is $|0\rangle$, do nothing, otherwise if it is $|1\rangle$, apply U to the second qubit, sometimes referred to as the target qubit.

3.6.4 Toffoli Gate:

The Toffoli gate performs the transformation
$T|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|c \oplus (a \wedge) b)\rangle$. The Toffoli gate can implement all Booleans functions, alongside the use of ancilla bits.

3.6.5 Rotation Operators:

The rotation operators are defined for an angle θ in radians as:

$$R_X(\theta) = \begin{bmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix}$$

$$R_Y(\theta) = \begin{bmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix}$$

$$R_Z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

3.7 Discrete Fourier Transform/Quantum Fourier Transform:

3.7.1 Concepts and Definitions:

An **n'th root of unity** for a positive integer n is a number z satisfying the equation $z^n = 1$. They are given by $e^{\frac{2k\pi i}{n}}$, and are said to be primitive if $k = 1, 2, 3 \ldots, n-1$.

Elementary Properties of roots of unity:
Roots of unity have modulus 1.
The reciprocal of an n'th root of unity is its complex conjugate, and is also an n'th root of unity
Proof: Let z be an n'th root of unity, then $\frac{1}{z} = z^{-1} = z^{n-1} = \bar{z}$, where because $z\bar{z} = |z|^2 = 1$, when $|z| = 1$, we may conclude that $\bar{z} = 1/z$.

The **discrete Fourier transform** transforms a sequence of N complex numbers $\{s_n\} = \{x_0, \ldots, x_{N-1}\}$ into another sequence of N complex numbers $\{S_N\} = \{X_0, \ldots, X_{N-1}\}$, and is defined by $X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn}$.
The discrete Fourier transform is an invertible, linear transformation, $F : \mathbb{C}^n \to \mathbb{C}^n$.
The inverse transform is given by $x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{i2\pi kn}{N}}$

Properties of the Discrete Fourier Transform:
Another way of looking at the DFT is to express it in matrix form, as

$$F = \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} \cdots \omega_N^{0\cdot(N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} \cdots \omega_N^{1\cdot(N-1)} \\ \cdots & \cdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} \cdots \omega_N^{(N-1)\cdot(N-1)} \end{bmatrix},$$

where $\omega_N = e^{-i2\pi/N}$, is a primitive N'th root of unity. The inverse discrete Fourier transform is given by $F^{-1} = \frac{1}{N}F^\dagger$.

Quantum Fourier Transform:
The QFT on an orthonormal basis $|0\rangle, \cdots, |N-1\rangle$ is a linear operator with the action on the basis states defined as: $|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle$. The quantum Fourier transform is a unitary transformation, so that $F^{-1} = F^\dagger$.

Let $N = 2^n$ for some positive integer n with corresponding basis states $|0\rangle, \ldots, |2^n - 1\rangle$, for an n qubit quantum computer. Consider a basis state $|j\rangle$ with its binary repre-

sentation $j = (j_1 \dots j_n)_2 = j_1 2^{n-1} + \dots j_n 2^0$, where we use the notation $0.j_l \dots j_m$ to represent the binary fraction $j_l/2 + \dots + j_m/2^{m-l+1}$. Then as in the quantum Fourier transform

$|j\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle$

$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} e^{2\pi i j (\sum_{l=1}^{n} k_l 2^{-1})} |k_1 \dots k_n\rangle$ (expressing k in binary)

$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle$ (expanding as in the tensor product)

$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} [\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle]$ (using the distributive property of the tensor product)

$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} [|0\rangle + e^{2\pi i j 2^{-1}} |1\rangle]$ (evaluating the inner bracket at $k_l = 0, k_1 = 1$)

$= \frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle)}{2^{n/2}}$

(note that for any positive integer k, that $e^{2i\pi k} = 1$)

3.7.2 Implementing the Quantum Fourier Transform:

Denote $R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$. Consider the circuit in Figure 5. Applying the Hadamard gate to the first bit of $|j_1 \dots j_n\rangle$ produces the state $\frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i 0.j_1}|1\rangle)|j_2 \dots j_n\rangle$, since $e^{2\pi i 0.j_1} = -1$ when $j_1 = 1$, and is 1 otherwise. Applying the controlled-R_2 gate produces the state $\frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle)|j_2 \dots j_n\rangle$. Continuing by applying the controlled $R_3, \dots, R_n$ gates, we have the state:
$\frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle)|j_2 \dots j_n\rangle$.

Next we perform a similar procedure on the second qubit which after the Hadamard transform puts us in the state
$(\frac{1}{2^{2/2}}(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle|j_2 \dots j_n\rangle)(|0\rangle + e^{2\pi i 0.j_2}|1\rangle)|j_3 \dots j_n\rangle$
and further applying the controlled R_2 through R_n gates yields the state
$(\frac{1}{2^{2/2}}(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle|j_2 \dots j_n\rangle)(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle)|j_3 \dots j_n\rangle$.
Continuing in this fashion for each qubit, we get the final state

$\frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle)}{2^{n/2}}$, as desired. Incidentally since this circuit consists of unitary transformations, this proves that the overall quantum Fourier transform is unitary as a bonus.

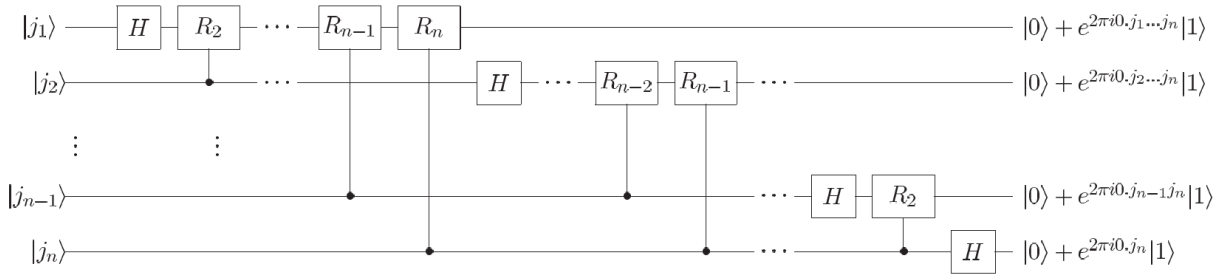


Figure 5: QFT circuit [2]

3.8 Phase Estimation:

Suppose we have a quantum circuit acting on n qubits, then the corresponding unitary matrix is of size $2^n \times 2^n$. Recall that by the spectral theorem U satisfies:
1. The eigenvalues of U have modulus/absolute value 1, and so are of the form $e^{2\pi i\theta}$ for $\theta \in [0, 1)$.
2. U has a complete, orthonormal collection of eigenvectors $|v_1\rangle, \cdots |v_N\rangle$, and any two eigenvectors corresponding to two different eigenvalues are orthogonal.
Thus for an eigenvector v, $U|v\rangle = e^{2\pi i\theta}|v\rangle$.

Phase Estimation Problem:

Input: A quantum circuit Q that performs a unitary operation U along with an eigenvector $|u\rangle$, such that $U|u\rangle = e^{2\pi i\theta}|u\rangle$.

Output: An approximation of $\theta \in [0, 1)$.

To perform the estimation, we assume that we are capable of preparing the state $|u\rangle$, an eigenvector of U, and have oracles capable of performing controlled-U^{2^j} operations, for each positive integer j. The quantum phase estimation procedure uses two registers. The first register contains m qubits initially in the state $|0\rangle^{\otimes m}$, where m is chosen according the number of digits of accuracy in our estimate of θ, as well as the probability that the overall procedure is successful. The second register contains the state $|u\rangle$. The circuit applies $H^{\otimes n}$ to the first register, followed by an application of controlled-U operations on the second register, with U raised to successive powers of 2.

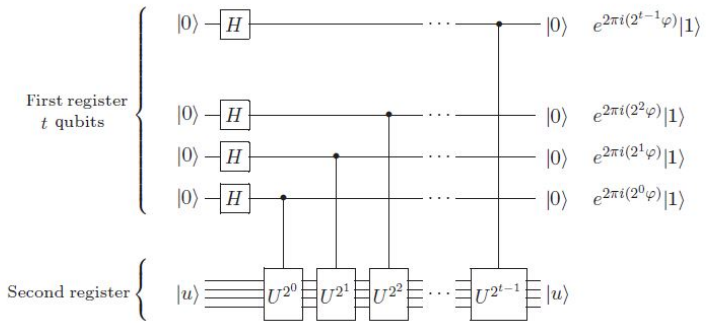


Figure 6: Phase Estimation Diagram [2]

The initial state is $|0^m\rangle|u\rangle$. After the n Hadamard transforms are performed we get the state $\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}|k\rangle|u\rangle$. Next the controlled U operations yield:
$\frac{1}{2^{m/2}}(|0\rangle+e^{2\pi i2^{m-1}\theta}|1\rangle)(|0\rangle+e^{2\pi i2^{m-2}\theta}|1\rangle)\dots(|0\rangle+e^{2\pi i2^0\theta}|1\rangle)|u\rangle = \frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}e^{2\pi ik\theta}|k\rangle|u\rangle.$

Discarding the eigenvector register, consider the case where $\theta = \frac{j}{2^m}$, for some $j \in \{0,1,2,\dots,2^m-1\}$, we want to determine j, since then we would be able to determine θ.

Then we can write $\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}e^{2\pi ik\theta}|k\rangle = \frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}w^{jk}|k\rangle$ where $w = e^{\frac{2\pi i}{2^m}}$. Defining $|\phi_j\rangle = \frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}w^{jk}|k\rangle$, we see that the $|\phi_j\rangle$ form an orthonormal set. Consider the following matrix (corresponding to the quantum Fourier transform):

$$F = \frac{1}{\sqrt{2^m}}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{2^m-1} \\ 1 & w^2 & w^4 & \cdots & 2^m-1 \\ \cdots & \cdots & \cdots & & \\ 1 & w^{2^m-1} & w^{2(2^m-1)} & \cdots & (w^{2^m-1})^{2^m-1} \end{bmatrix}$$

F is a unitary transformation that satisfies $F|j\rangle = |\phi_j\rangle$, and moreover its conjugate transpose is its inverse so $F^\dagger|\phi_j\rangle = |j\rangle$. We write $F = QFT_{2^m}$. From the previous section on the QFT and its inverse, we see that $QFT^\dagger\sum_{k=0}^{2^m-1}w^{jk}|k\rangle = |j\rangle$, and so by measuring the first register we have j, and so we know θ.

Probability in the general scenario:
After the Hadamard transforms, the controlled-U operations, and application of $QFT^\dagger$, in general we get $QFT^\dagger_{2^m}(\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}e^{2\pi ik\theta}|k\rangle) = \frac{1}{2^m}\sum_{j=0}^{2^m-1}(\sum_{k=0}^{2^m-1}e^{2\pi ik(\theta-\frac{j}{2^m})})|j\rangle$, where the

probability of measuring $|j\rangle$, say p_j, is $|\sum_{j=0}^{2^m-1} e^{2\pi ik(\theta-\frac{j}{2^m})}|^2$.
Assume that $\theta = \frac{j}{2^m} + |\epsilon|$, so that the measurement value j (divided by 2^m) becomes an estimate of θ with error ϵ. If $|\epsilon| \leq 2^{-(m+1)}$, it can be shown that $p_j > 0.4$, i.e the probability θ is accurate to m bits of precision. If ϵ satisfies $\frac{\alpha}{2^m} \leq |\epsilon| < \frac{1}{2}$ where α is an arbitrary positive number, then it can be shown that $p_j \leq \frac{1}{4\alpha^2}$. This implies that highly inaccurate results are unlikely. See the next sub-section for a more detailed analysis.
In general, if $m = k + 2$ and we run the estimation procedure several times and look for the most commonly appearing outcome, at least one outcome, which is accurate to k bits of precision, occurs with probability at least 0.4. Therefore if you take the most commonly occurring outcome and round it to k bits, the probability of correctness approaches 1 exponentially fast in the number of times the procedure is repeated.

3.8.1 Performance and requirements:

In the case that θ cannot be written exactly with an m-bit binary expansion, it is true that the procedure described will produce a pretty good approximation to θ with high probability [2].
Let b be the integer in the range 0 to 2^{t-1} such that $\frac{b}{2^t}$ is the best t-bit approximation to θ which is less than θ, or equivalently, the difference $\delta = \theta - b/2^t$ satisfies $0 \leq \delta \leq 2^{-t}$. The goal in this section is to show that the phase estimation procedure produces a result which is close to b, and thus enables us to estimate θ accurately, with high probability.

After applying the inverse quantum Fourier transform to the state $\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi ik\theta}|k\rangle$ the result can be expressed as $\frac{1}{2^t} \sum_{k,l=0}^{2^t-1} e^{\frac{-2\pi ikl}{2^t}} e^{2\pi i\theta k}|l\rangle$. Let α_l be the amplitude of $|(b+l)(mod 2^t)\rangle$ where $\alpha_l = \sum_{k=0}^{2^t-1} (e^{2\pi i(\theta-(b+l)/2^t)})^k$ (fix the index formerly considered l now for $b+l \mod (2^t)$ for each $1 \leq l \leq 2^t - 1$). This is the sum of a finite geometric series, so $\alpha_l = \frac{1}{2^t}(\frac{1-e^{2\pi i(2^t\theta-(b+l))}}{1-e^{2\pi i(\theta-(b+l)/2^t)}}) = \frac{1}{2^t}(\frac{1-e^{2\pi i(2^t\delta-l)}}{1-e^{2\pi i(\delta-l/2^t)}})$.

Suppose the outcome of the final measurement in the phase estimation procedure is m. We aim to bound the probability of obtaining a value of m where $|m-b| > e$, and e is a positive integer characterizing our desired tolerance to error ($e \leq 2^t$). The probability of observing such an m is given by the sum of the probability of observing each α_l where $|l| \pmod{2^t} > e$, and so splitting up the interval symmetrically mod 2^t we have

(*) $p(|m-b|>e) = \sum_{-2^{t-1}<l\leq -(e-1)} |\alpha_l|^2 + \sum_{e+1<l\leq 2^{t-1}} |\alpha_l|^2.$

Now noting that for any real θ, that $|1-e^{i\theta}| \leq 2$, we have that $|\alpha_l| \leq \frac{2}{2^t|1-e^{2\pi i(\delta - l/2^t)}|}$. Further by elementary caclulus, the identity $|1-e^{i\theta}| \geq 2\theta/\pi$ holds whenever $-\pi \leq \theta \leq \pi$.

For $t \geq 2$, and $|l| < 2^{t-1}$, it is true that $|\delta - l/2^t| \leq 1/2$, and so $-\pi \leq 2\pi(\delta - l/2^t) \leq \pi$, and so $|1-e^{2\pi i(\delta-l/2^t)}| \geq 4(\delta - l/2^t)$, and hence
(**) $|\alpha_l| \leq \frac{1}{2^{t+1}(\delta - l/2^t)}$ holds.

Therefore we can combine the inequalities (*) and (**) to see that
$p(|m-b|>e) = \sum_{-2^{t-1}<l\leq -(e-1)} |\alpha_l|^2 + \sum_{e+1<l\leq 2^{t-1}} |\alpha_l|^2$
$\leq \sum_{-2^{t-1}<l\leq -(e+1)} (\frac{1}{2^{t+1}(\delta - l/2^t)})^2 + \sum_{e+1<l\leq 2^{t-1}} (\frac{1}{2^{t+1}(\delta - l/2^t)})^2$
$\leq \frac{1}{4}[\sum_{-2^{t-1}<l\leq -(e+1)} \frac{1}{(l-2^t\delta)^2} + \sum_{e+1<l\leq 2^{t-1}} \frac{1}{(l-2^t\delta)^2}]$

But now $0 \leq 2^t\delta \leq 1$, so we obtain
$p(|m-b|>e) \leq \frac{1}{4}[\sum_{-2^{t-1}<l\leq -(e+1)} \frac{1}{l^2} + \sum_{e+1<l\leq 2^{t-1}} \frac{1}{(l-1)^2}]$
$\leq \frac{1}{4}[\sum_{l=-2^{t-1}+1}^{-(e+1)} \frac{1}{l^2} + \sum_{l=e+1}^{2^{t-1}} \frac{1}{(l-1)^2}]$
$\leq \frac{1}{2}\sum_{l=e+1}^{2^{t-1}} \frac{1}{(l-1)^2} = \frac{1}{2}\sum_{l=e}^{2^{t-1}-1} \frac{1}{l^2}$
$\leq \frac{1}{2}\int_{l=e}^{2^{t-1}-1} \frac{1}{l^2} dl$ (using similar reasoning to the standard analytic proof that $\sum_{i=0}^{\infty} \frac{1}{n^2}$ converges by the integral test)
$= \frac{1}{2}(\frac{1}{e-1} - \frac{2}{2-2^t}) \leq \frac{1}{2(e-1)}.$

Suppose that we wish to approximate θ to an accuracy of 2^{-n}, so that $e = 2^{t-n} - 1$. By using $t = n + p$ qubits initialized to the zero state in the phase estimation algorithm, the probability of obtaining an approximation correct to this accuracy is at least $1 - \frac{1}{2(e-1)} = 1 - \frac{1}{2(2^p-2)}$. Therefore to obtain θ accurate to n bits with probability at least $1-\epsilon$, we choose $t = n + \lceil log(2 + \frac{1}{2\epsilon})\rceil$.

3.9 Order Finding Using Phase Estimation:

Let $N > 2$ be a positive integer and let $a \in \mathbb{Z}_N^*$, i.e. an invertible element, then the order of a is the smallest positive integer k such that $a^k = 1 \mod N$. The goal of

the order-finding algorithm is to find the order of a. Let $n = \lfloor log_2(N-1) \rfloor + 1$, the number of bits needed to encode elements of $\mathbb{Z}_N$ in binary.
Define $M_a|x\rangle = |(ax) \mod N\rangle$ for $0 \leq x < N$, and to be the identity transform for $x \geq N$. M_a preserves length by mapping classical states to classical states, and is invertible and linear, and hence unitary. In order to subject this transformation to phase estimation, suppose that we want m bits of precision, then for an m-bit integer k, consider the transformation: $\Lambda_m(M_a)|k\rangle|x\rangle = |k\rangle|(a^k x) \mod N\rangle$, which for now we assume can be implemented efficiently, and is needed for the controlled-M^j operations in the phase estimation algorithm.

What are the eigenvectors and eigenvalues of M_a? Let r be the order of a.
Define $|\psi_j\rangle = \frac{1}{\sqrt{r}}(|1\rangle + w_r^{-j}|a\rangle + \cdots + w_r^{-j(r-1)}|a^{r-1}\rangle)$, where $w_r = e^{\frac{2\pi i}{r}}$, and $0 \leq j \leq r-1$ an integer, then
$M_a|\psi_j\rangle = \frac{1}{\sqrt{r}}(|a\rangle + w_r^{-j}|a^2\rangle + \cdots + w_r^{-(r-1)j}|a^r\rangle)$
$= \frac{1}{\sqrt{r}}(|1\rangle w_r^{-(r-1)j} + |a\rangle + w_r^{-j}|a^2\rangle + \cdots + w_r^{-(r-2)j}|a^{r-1}\rangle)$
$= \frac{w^j}{\sqrt{r}}(|1\rangle w_r^{-r} + w_r^{-j}|a\rangle + w_r^{-2j}|a^2\rangle + \cdots + w_r^{-(r-1)j}|a^{r-1}\rangle)$, now noting that $w_r^{-r} = 1$, we see that $M_a|\psi_j\rangle = w_r^j|\psi_j\rangle$ for any integer j, where the eigenvalue of corresponding to $|\psi_1\rangle$ is $e^{2\pi i(1/r)}$. The states $|\psi_i\rangle$ are orthonormal.

The phase estimation algorithm run on $|\psi_1\rangle$ should yield a measurement $j \in \{0, \cdots, 2^m - 1\}$ such that $\frac{j}{2^m}$ is approximately $\frac{1}{r}$, which will be sufficiently close if m is large enough. It can be shown that for $m = 2n$, that $\frac{j}{2^m} = \frac{1}{r} - \epsilon$ for $|\epsilon| \leq \frac{1}{2N^2}$. However, since we do not know the value of r, we do not know how to obtain $|\psi_1\rangle$.

Note that $\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|\psi_k\rangle = \frac{1}{r}\sum_{k=0}^{r-1}\sum_{l=0}^{r-1} w_r^{-kl}|a^l\rangle = |a^0\rangle = |1\rangle$. This is true since for a fixed $l > 0$ we have that $\sum_{k=0}^{r-1} w_r^{-kl}|a^l\rangle = \frac{w_r^{-rl}-1}{w_r^{-l}-1}|a^l\rangle = 0$ since $w_r^{-rl} = 1$.

If we were to run the phase estimation procedure on the state $|1\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|\psi_k\rangle$, the state immediately before measurement would have the form $\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|\phi_k\rangle|\psi_k\rangle$ where each $|\phi_k\rangle$ is the state of the first m qubits that you would get by running the phase procedure on $|\psi_k\rangle$ alone. Because the states $|\psi_0\rangle, \ldots, |\psi_{r-1}\rangle$ are orthonormal, the probability to measure some value j is the average over an integer $0 \leq k \leq r-1$ chosen uniformly to have measured that value starting with the eigenvector $|\psi_k\rangle$. Hence running the phase estimation procedure on $|1\rangle$ is equivalent to running the phase es-

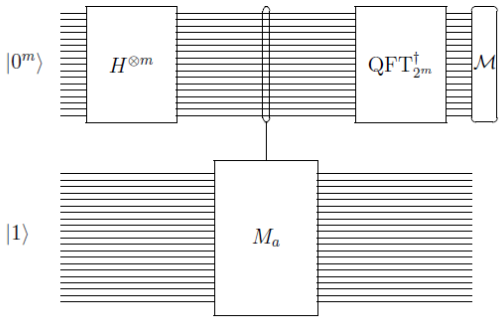


Figure 7: Order Finding Algorithm [1]

timation procedure on an eigenvector $|\psi_k\rangle$ chosen uniformly at random.

Now from the theory of convergents, it is true that given a real number $\alpha \in (0,1)$ and $N \geq 2$, there is at most one fraction $\frac{x}{y}$ with $0 \leq x, y \leq N$, $y \neq 0$, $gcd(x,y) = 1$, with $|\alpha - \frac{x}{y}| < \frac{1}{2N^2}$. See the section 10.3, *Approximation by Rational Numbers*, for a discussion of convergents. To summarize:

Order Finding Algorithm. Let $m = 2n$. Apply phase estimation to M_a with input $|0^m\rangle|1\rangle$ several times and take the most common value $j \in \{0, \cdots, 2^m - 1\}$, $\frac{j}{2^m} = \frac{k}{r}$ accurate to m bits.
Apply the continued fraction algorithm to the result in order to find $\frac{x}{y}$ with $gcd(x,y) = 1$ such that $\frac{x}{y} \approx \frac{k}{r}$. This may fail to find r if $gcd(k,r) \neq 1$, so by repeating the algorithm several times (each time possibly resulting in a different k), and taking the least common multiple of the y values, it can be shown that we will find r with high probability.

3.10 Shor's Algorithm and Factoring Discussion:

The goal of Shor's algorithm is to use quantum computers to factor integers efficiently. First consider the scenario where an integer $N \geq 2$ is a prime power, so that for some integer $k > 0$, $N = p^k$ for a prime number p. By checking every integer $1 \leq t \leq log_2(N)$, calculating $m = floor(n^{1/t})$, and checking if $N = m^t$, we will find the prime p, and power k such that $p^k = N$ in time complexity that is linear in $log_2 n$.

Now assume that N has at least 2 distinct prime factors. It is enough to find an algorithm that finds two integers $u, v \geq 2$ such that $N = uv$ and then run the algorithm recursively on u and v separately.
If N is even then we can simply choose $u = 2$, $v = N/2$, consequently suppose that N is odd and not a prime power.

Let a be a random element in Z_N^*, then it is likely that a is invertible (since if not we have found a factor in their greatest common divisor). Consider r to be the order of a, and suppose it is even, then $N|a^r - 1$ and $a^r - 1 = (a^{\frac{r}{2}} + 1)(a^{\frac{r}{2}} - 1)$, where N cannot divide $(a^{\frac{r}{2}} - 1)$ since this would contradict the definition of the order of a. If N does not divide $(a^{\frac{r}{2}} + 1)$, then the factors of N are split between $(a^{\frac{r}{2}} + 1)$ and $(a^{\frac{r}{2}} - 1)$ and so $gcd(a^{r/2} - 1, N)$ will have a non-trivial factor, giving the algorithm:

Input: an odd, composite integer N that is not a prime power.
Output: a non-trivial factor of N

Repeat
Randomly choose $a \in \{2, \ldots, N\}$
Compute $d = gcd(a, N)$
If $d \geq 2$ then
Return $u = d$ and $v = N/d$
Else
Let r be the order of a in $\mathbb{Z}_N$. (Requires the order finding algorithm)
If r is even then
Compute $x = a^{r/2} - 1(\mod N)$
Compute $d = gcd(x, N)$
If $d \geq 2$ then
Return $u = d$ and $v = N = d$. /* Answer is found. */
Repeat until the answer is found

Shor's algorithm is in the complexity class **BQP** and runs in $O((log(N)^3)$ time (polynomial in $log(N)$), and uses quantum gates of order $O((logN)^2(loglogN)(logloglogN))$ to use fast modular multiplication to compute $x = a^{r/2} - 1(modN)$. This represents an exponential speed-up over the classical factoring algorithms, such as the quadratic sieve algorithm, which typically run in sub-exponential time complexity.

3.11 Grover's Algorithm:

Suppose that we have a function $f : \{0,1\}^n \to \{0,1\}$ that is implemented by a reversible transformation B_f defined by $B_f|x\rangle|a\rangle = |x\rangle|a \oplus f(x)\rangle$ for all $x \in \{0,1\}^n$ and $a \in \{0,1\}$. The problem of *search* is simply to find a string $x \in \{0,1\}^n$ such that $f(x) = 1$, or to conclude that no such x exists if f is identically 0.

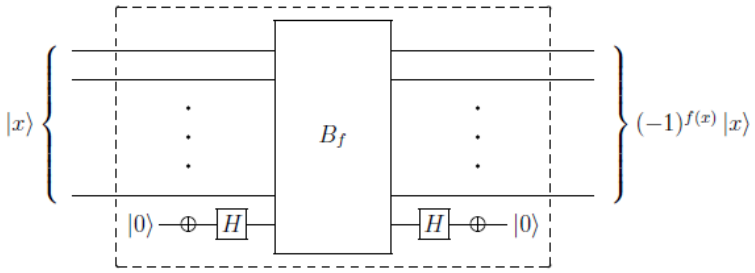


Figure 8: Z_f in Grover's Algorithm [1]

Suppose that for one value of $x \in \{0,1\}^n$, we have $f(x) = 1$. When randomly choosing k distinct elements among $\{0,1\}^n$, the probability of finding x is given by $1 - \frac{\binom{2^n-1}{k}}{\binom{2^n}{k}} = \frac{k}{2^n}$ where $\frac{\binom{2^n-1}{k}}{\binom{2^n}{k}}$ is the probability that we chose a k-subset that maps entirely to 0. Therefore if we want the probability to be larger than $1-\epsilon$ for some $0 < \epsilon < 1$, then we need $\frac{k}{2^n} > 1-\epsilon \implies k \in \Omega(2^n)$ queries to solve this problem classically. By contrast Grover's algorithm will solve the problem using $O(\sqrt{2^n})$ queries.

The algorithm uses two unitary transformations on n qubits:

$Z_f|x\rangle = (-1)^{f(x)}|x\rangle$ and $Z_0|x\rangle = \begin{cases} -|x\rangle & \text{if } x = 0^n \\ |x\rangle & \text{if } x \neq 0^n \end{cases}$.

Given the black-box transformation defined in the Deutsch-Jozsa algorithm, we can implement Z_f using a single ancillary qubit using the phase kick-back phenomenom, with just one query to B_f required to implement Z_f. The Z_0 transformation can be implemented by constructing a reversible circuit for computing the transformation $|x\rangle|a\rangle \mapsto |x\rangle|a \oplus (\neg x_1 \wedge \cdots \wedge \neg x_n)\rangle$.

Algorithm:

1. Let X be an n-qubit quantum register with starting state $|0\rangle^{\otimes n}$. Perform $H^{\otimes n}$ on X.
2. Apply to the register X the transformation $G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$.
3. Measure X and output the result.

In analyzing Grover's algorithm note that $H^{\otimes n} Z_0 H^{\otimes n}$ and $-Z_f$ may be viewed as reflections about a line, and so $G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$ is therefore a rotation by twice the angle between the two lines. This is true since by elementary geometry, two successive reflections about two lines is equivalent to a rotation of twice the angle between

the two lines.
Define sets of strings A, B as
$A = \{x \in \{0,1\}^n : f(x) = 1\}, B = \{x \in \{0,1\}^n : f(x) = 0\}$ and let $a = |A|, b = |B|$.
Define $|A\rangle = \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle$ and $|B\rangle = \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle$.

The state of register X immediately after step 1 in the algorithm is given by
$|h\rangle = H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$. Let $N = 2^n$, then $|h\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$
$= \frac{1}{\sqrt{N}}(\sqrt{a}|A\rangle + \sqrt{b}|B\rangle)$.

Note that we can express $Z_0 = I - 2|0^n\rangle\langle 0^n| = \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$.

This means that $H^{\otimes n} Z_0 H^{\otimes n} = H^{\otimes n}(I - 2|0^n\rangle\langle 0^n|)H^{\otimes n} = I - 2|h\rangle\langle h|$
using the facts i) $H^\dagger = H$, ii) $\langle 0^n|H^{\otimes n} = \langle h|$. Now consider the effect of G on $|A\rangle$ and $|B\rangle$ separately.
$G|A\rangle = -H^{\otimes n} Z_0 H^{\otimes n} Z_f |A\rangle$
$= (I - 2|h\rangle\langle h|)(-Z_f)|A\rangle$
$= (I - 2|h\rangle\langle h|)|A\rangle$ (since the elements in A satisfy $f(x) = 0$)
$= |A\rangle - 2\langle h|A\rangle|h\rangle$. Now note that $|A\rangle$ and $|B\rangle$ are orthonormal vectors, and so
$\langle h|a\rangle = (\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle)^\dagger \cdot (\frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle) = \frac{1}{\sqrt{a}\sqrt{N}} \sum_{x \in A} x \cdot x = \frac{\sqrt{a}}{\sqrt{N}}$.
Therefore $= |A\rangle - 2\langle h|A\rangle|h\rangle = |A\rangle - 2\sqrt{\frac{a}{N}}(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle)$
$= (1 - \frac{2a}{N})|A\rangle - \frac{2\sqrt{ab}}{N}|B\rangle$.

Using similar reasoning with $-Z_f|B\rangle = -|B\rangle$, the results are summarized as follows:
$G|A\rangle = (1 - \frac{2a}{N})|A\rangle - \frac{2\sqrt{ab}}{N}|B\rangle$
$G|B\rangle = \frac{2\sqrt{ab}}{N}|A\rangle - (1 - \frac{2b}{N})|B\rangle$.

The action of G on the sub-space spanned by the basis $\{|B\rangle, |A\rangle\}$ is given by the matrix $M = \begin{bmatrix} -(1 - \frac{2b}{N}) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & (1 - \frac{2a}{N}) \end{bmatrix}$, where noting that $a + b = N$, we can write $M = \begin{bmatrix} \frac{b-a}{N} & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \frac{b-a}{N} \end{bmatrix}$, where $M \cdot (1,0)^t = G|B\rangle$ and $M \cdot (0,1)^t = G|A\rangle$. Further, it is

true that $\begin{bmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{a}{N}} & \sqrt{\frac{b}{N}} \end{bmatrix}^2 = M$. Since $a \leq N, b \leq N$ and $a + b = N$, then we can choose $\theta \in (0, \pi/2)$ such that $sin\theta = \sqrt{\frac{a}{N}}$ and $cos\theta = \sqrt{\frac{b}{N}}$, and finally $R_{2\theta} = M$, and so G causes a rotation by an angle 2θ in the space spanned by $\{|B\rangle, |A\rangle\}$.

Then $\theta = sin^{-1}\sqrt{\frac{a}{N}}$, and $|h\rangle = \sqrt{\frac{b}{N}}|B\rangle + \sqrt{\frac{a}{N}}|A\rangle = cos\theta|B\rangle + sin\theta|A\rangle$ after step 1. After k iterations of G, the state will be $cos((2k+1)\theta)|B\rangle + sin((2k+1)\theta)|A\rangle$. Since ultimately we want to measure some element $x \in A$, then we would like the state of the register containing X to be as close to $|A\rangle$ as possible. Then $sin((2k+1)\theta) \approx 1 \implies (2k+1)\theta \approx \frac{\pi}{2} \implies k \approx \frac{\pi}{4\theta} - \frac{1}{2}$. But k is an integer (and in fact the number of queries to B_f), so we can only obtain approximations.

If $a = 1$, then $\theta = \sin^{-1}\sqrt{\frac{1}{N}} \approx \frac{1}{\sqrt{N}}$, so that $k = \lfloor \pi/4\sqrt{N} \rfloor$. The probability of finding the single x such that $f(x) = 1$ is $sin^2((2\lfloor \pi\sqrt{N}/4 \rfloor + 1)sin^{-1}(1/\sqrt{N}))$, which converges to 1 as N goes to infinity, and is bounded below by $\frac{1}{2}$, so that by repeating the algorithm some small constant number of times and evaluating f at the output, we will find the unique x such that $f(x) = 1$ with high probability.

If $a \geq 1$, we employ a different strategy:
1. Set $m = 1$
2. Choose $k \in \{1, \cdots, m+1\}$ uniformly and run Grover's algorithm for this choice of k. If the algorithm finds an x such that $f(x) = 1$, then output x and halt.
3. If $m > \sqrt{N}$, then output "Fail". Else, set $m = \lfloor (8/7)m \rfloor$ and go to step 2.
It can be shown that this will succeed in finding $x \in A$ with probability at least $1/4$ after $O(\sqrt{N/a})$ queries.

Finally, two cases remain. If $a = 0$, then Grover's algorithm will output a choice of $x \in \{0,1\}^n$ that is uniformly distributed and will never evaluate to 0, whereas if $b = 0$ then any iteration of Grover's algorithm will yield an x such that $f(x) = 1$.

4 Hamiltonian Simulation:

Suppose we are given a Hamiltonian operator $\hat{H}$, which is a Hermitian matrix, and the goal is to determine a quantum circuit which implements $U = e^{-i\hat{H}t}$ up to given error for time parameter t. The evolution of a quantum state $|\Psi\rangle$ under a unitary operator is given by the time-independent Schrodinger equation: $i\frac{d}{dt}|\Psi\rangle = \hat{H}|\Psi\rangle$, which has

the solution $|\Psi(t)\rangle = e^{-i\hat{H}t}|\Psi\rangle$. The challenge of simulating Hamiltonians is due to the fact that the application of matrix exponentials is computationally expensive. In particular, a quantum computer can be used to simulate the Hamiltonian operator, a task known as *Hamiltonian simulation*, which we wish to perform efficiently.

Hamiltonian Simulation: We say that a Hamiltonian $\hat{H}$ that acts on n qubits can be efficiently simulated if for any $t > 0, \epsilon > 0$, there exists a quantum circuit $U_{\hat{H}}$ consisting of $poly(n, t, 1/\epsilon)$ gates such that $||U_{\hat{H}} - e^{-i\hat{H}t}|| < \epsilon$. Since it can be shown that any quantum computation can be implemented by a sequence of Hamiltonian simulations, simulating Hamiltonians in general is **BQP**-*hard*.

The problem of simulating arbitrary Hamiltonians is not yet solved, as finding an approximate decomposition into elementary single and two qubit gates for a generic unitary is **NP**-hard, however Hamiltonians with particular structures can be simulated. If $\hat{H}$ can be efficiently simulated, then so can $c\hat{H}$ for any $c = poly(n)$. In addition since any quantum computation is reversible, $e^{-i\hat{H}t}$ is also efficiently simulatable and this must hence hold for $c < 0$. Moreover, the definition of efficiently simulatable Hamiltonians further extends to unitary matrices, since every operator $U_{\hat{H}}$ corresponds to a unitary operator, and furthermore every unitary operator can be written in the form $e^{i\hat{H}}$ for a Hermitian matrix $\hat{H}$. [6].

4.1 Efficient Quantum Algorithms for Simulating Sparse Hamiltonians:

In the paper *Efficient quantum algorithms for simulating sparse Hamiltonians*, authors Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders present an efficient quantum algorithm for simulating the evolution of a sparse Hamiltonian H for a given time t in terms of a procedure for computing the matrix entries of H. In this section we discuss problem 1 and problem 2 as outlined in the paper. [8]

4.2 Related Concepts:

The **trace norm**: of a matrix A is the sum of the singular values of A, and can be written $||A||_1 = Tr\sqrt{AA^\dagger}$, where $\sqrt{AA^\dagger}$ is some positive semidefinite matrix B such that $BB = AA^\dagger$.

The **trace distance** of two matrices is defined by half the trace norm of the difference of the matrices.
$T(A,B) := \frac{1}{2}\sqrt{Trace((A-B)^\dagger(A-B))}$.

The **iterated logarithm** of n is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1.

$$log^*n : \begin{cases} 0 & \text{, if } n \leq 1 \\ 1 + log*(logn) & \text{, if } n > 1 \end{cases}$$

A **composition** of an integer n is a way of writing n as the sum of a sequence of positive integers. A weak composition of an integer n is similar to a composition of n, but allowing terms of the sequence to be 0.

4.3 Problem 1:

The Hamiltonian is of the form $H = \sum_{j=1}^{m} H_j$. The problem is to simulate the evolution e^{-iHt} by a sequence of exponentials $e^{iH_jt'}$ such that the maximum error in the final state, as quantified by the trace distance, does not exceed some error ϵ. We wish to determine an upper bound on the number of exponentials, N_{exp}, required in this sequence.

Theorem 1. When the permissible error is bounded by ϵ, N_{exp} is bounded by $N_{exp} = 2m5^{2k}(m\tau)^{1+1/2k}/\epsilon^{1/2k}$, for $\epsilon \leq 1 \leq 2m5^{k-1}\tau$ where $\tau = ||H||t$ and k is an arbitrary positive integer.

Given the truth of theorem 1, by taking k to be sufficiently large, it is possible to obtain a scaling that is close to linear in τ. However, for a given value of τ, taking k to be too large will increase N_{exp}. Expressing N_{exp} in exponential form $2m5^{2k}(m\tau)^{1+1/2k}/\epsilon^{1/2k} = 2m^2\tau e^{2kln5+ln(m\tau/\epsilon)/2k}$, which has minimum $k = \lfloor \frac{1}{2}\sqrt{log_5(m\tau/\epsilon)+1} \rfloor$ where 1 has been added because k must take integer values. Plugging k into the original upper bound and inequality for ϵ gives :

$N_{exp} \leq 4m^2\tau e^{2\sqrt{ln5ln(m\tau/\epsilon)}}$ for $\epsilon \leq 1 \leq m\tau/25$, which for large $m\tau$ is close to linear in τ, which the authors note is effectively optimal since it is not possible to perform general simulations sublinear in τ. This leads to a discussion of finding approximations of the exponential of a sum of matrices in order to justify the bound N_{exp}.

4.4 Finding Exponential Product Formulas of Higher Orders:

One of the significant research interests of Dr. Masuo Suzuki (University of Tokyo) is his approximation technique to estimate the exponential of a sum of matrices, known as the method of higher-order integrators. In this section we discuss these higher-order integrators as outlined in the paper *Finding Exponential Product Formulas of Higher Orders*. Additionally see section 9.1 of this paper for a review of matrix exponentials. [9]

The simplest Suzuki-Trotter decomposition (the Lie Product rule is sometimes referred to as the Lie-Trotter Formula) is given by
(1) $||e^{xA}e^{xB} - e^{x(A+B)}|| \in O(x^2)$ as $x \to 0$
This can be shown comparing the taylor expansion for both expressions
$e^{x(A+B)} = I + x(A+B) + x^2/2(A^2 + AB + BA + B^2) + \cdots$, and
$e^{xA}e^{xB} = (I + xA + x^2/2A^2 + \cdots)(I + xB + x^2/2B^2 + \cdots)$
$= I + x(A+B) + x^2/2(A^2 + B^2 + 2AB) + \cdots$, where we see that
$||e^{xA}e^{xB} - e^{x(A+B)}|| \in O(x^2)$ as $x \to 0$ since $(AB \neq BA)$ and so the terms x^2 do not cancel out.

The goal of Suzuki's higher-order integrators is to obtain higher-order correction terms, i.e. to set parameters $\{p_1, \cdots, p_M\}$ such that
$e^{x(A+B)} = e^{p_1 xA}e^{p_2 xB} \cdots e^{p_{M-1}xA}e^{p_M xB} + O(x^{m+1})$. A matrix function f_m, is said to be the m'th approximant of $e^{x(A+B)}$ if $||e^{x(A+B)} - f_m(A_1, A_2)|| \in O(x^{m+1})$.

4.5 Higher-Order integrators:

The easiest improvement of formula (1) is the so-called symmetrization
(2) $S_2(x) = e^{x/2A}e^{xB}e^{x/2A} = e^{x(A+B)+x^3R_3+x^5R_5+\cdots}$. This can be shown using the Taylor Expansion of the LHS, i.e :
$e^{x/2A}e^{xB}e^{x/2A} = (I+x/2A+x^2/8A+\cdots)(I+xB+x^2/2B+\cdots)(I+x/2A+x^2/8A+\cdots)$
$= I+x(A+B)+M_2x^2+M_3x^3+\cdots = e^{x(A+B)+x^2R_2+x^3R_3+\cdots}$ for some matrices $\{M_i\}_{i=2}^{\infty}$, $\{R_i\}_{i=2}^{\infty}$, which exist by the Baker–Campbell–Hausdorff formula discussed in section 8.5. The symmetrized approximant has the property
$S_2(x)S_2(-x) = e^{x/2A}e^{xB}e^{x/2A}e^{-x/2A}e^{-xB}e^{-x/2A} = I$.

Lemma: Odd polynomials in $R[x]$ satisfying $f(-x) = -f(x)$, have even coefficients equal to 0 for any ring R.
Proof: Consider an arbitrary polynomial $\sum_{i=0}^{\infty} a_o x^k$, then if for all x,
$a_0 + a_1(-x) + a_2(-x)^2 + \cdots = -(a_0 + a_1x + a_2x^2 + \cdots) \implies \forall k \geq 0, a_{2k}x^{2k} = -a_{2k}x^{2k} \implies a_{2k} = 0$ $\square$.

Returning to our previous discussion, define the matrix logarithm $log[A] = B$ such that $e^B = A$, which exists iff A is invertible. Since $S_2(x)S_2(-x) = I$, then $S_2(x)$ is invertible and so its logarithm exists, and moreover $log[I] = 0$ since $e^0 = I$. Since $S_2(x), S_2(-x)$ commute then $0 = log[S_2(x)S_2(-x)] = log[S_2(x)] + log[S_2(-x)]$, and so $-log[S_2(x)] = log[S_2(-x)]$, and therefore $S_2(x)$ is an odd function, and so the even coefficents vanish in the Taylor expansion. Therefore we have
$||S_2(x) - e^{x(A+B)}|| \in O(x^3)$ since the matrix coefficients of powers of x less than 3 coincide.

In order to construct a symmetrized fourth-order approximant from the symmetrized second-order approximant (2), consider a product
$S(x) = S_2(sx)S_2((1-2s)x)S_2(sx) = e^{\frac{sx}{2}A}e^{sxB}e^{\frac{sx}{2}A}e^{\frac{1-2s}{2}xA}e^{(1-2s)xB}e^{\frac{1-2s}{2}xA}e^{\frac{sx}{2}A}e^{sxB}e^{\frac{sx}{2}A}$
$= e^{\frac{sx}{2}A}e^{sxB}e^{\frac{1-s}{2}xA}e^{(1-2s)xB}e^{\frac{1-s}{2}xA}e^{sxB}e^{\frac{sx}{2}A}$ which can also be written as

$S(x) = S_2(sx)S_2((1-2s)x)S_2(sx)$
$= e^{sx(A+B)+s^3x^3R_3+O(x^5)}e^{(1-2s)x(A+B)+(1-2)^3s^3x^3R_3+O(x^5)}e^{sx(A+B)+s^3x^3R_3+O(x^5)}$
$= e^{x(A+B)+[2s^3+(1-2s)^3]x^3R_3+O(x^5)}$, where we note that the first-order term in the exponent of the last line is $x(A+B)$ since $sx(A+B) + (1-2s)x(A+B) + sx(A+B) = x(A+B)$, and the even terms vanish since $S(x)S(-x) = I$. Moreover the third-order correction in the third line is the sum of the third-order corrections in the second line. Hence if s is a solution to $2s^3 + (1-2s)^3 = 0$ or $-6s^3 + 12s^2 - 6s + 1 = 0$ which has real solution $s = \dfrac{1}{2-(2)^{\frac{1}{3}}}$, then $||S(x) - e^{x(A+B)}|| \in O(x^5)$.

Following the same line of thought, we come up with another fourth-order approximant in the form $S_4(x) = S_2(s_2x)^2S_2((1-4s_2)x)S_2(s_2x)^2$, where the parameter s_2 is a solution of $4s_2^3 + (1-4s^2)^3 = 0$, or $s = \dfrac{1}{4-(4)^{1/3}}$, chosen now since $s < 1$, which the author argues is more advantageous in applications.
To construct the sixth-order approximant, let
$S_6(x) = S_4(s_4x)^2S_4((1-4s_4)x)S_4(s_4x)^2 =$
$(S_2(s_4s_2x)^2S_2(s_4(1-4s_2)x)S_2(s_4s_2x)^2)^2$
$\times S_2((1-4s_4)s_2x)^2S_2((1-4s_4)(1-4s_2)s)S_2((1-4s_4)s_2x)^2$
$\times(S_2(s_4s_2x)^2S_2(s_4(1-4s_2)x)S_2(s_4s_2x)^2)^2$
with $4s_4^5 + (1-4s_4)^5 = 0 \implies s_4 = \frac{1}{4-4^{1/5}} \approx 0.373065$

In general for $k > 1$, define
$S_{2k}(x) = S_{2k-2}(p_kx)^2S_{2k-2}((1-4p_k)x)S_{2k-2}(p_kx)^2$ where $p_k = (4 - 4^{\frac{1}{2k-1}})^{-1}$, where author Masuo Suzuki shows that $||e^{x(A+B)} - S_{2k}(x)|| \in O(x^{2k+1})$

Defining more generally for a sum of m matrices,
$S_2(x) = \prod_{j=1}^{m} e^{H_j x/2} \prod_{j'=m}^{2} e^{H'_j x/2} = e^{H_1 x/2} e^{H_2 x/2} \cdots e^{H_m x} \cdots e^{H_2 x/2} e^{H_1 x/2}$, using the same recursive formula for $S_{2k}(x)$, it is true that $||e^{(\sum_{j=1}^{m} H_j x)} - S_{2k}(x)|| \in O(x^{2k+1})$.

4.6 Proving Lemma 1:

Define $S_{2k}(\lambda) = \prod_{j=1}^{m} e^{H_j \lambda/2} \prod_{j'=m}^{1} e^{H_{j'} \lambda/2}$ for $k = 1$, and for $k > 1$,
$S_{2k}(\lambda) = [S_{2k-2}(p_k \lambda)^2][S_{2k-2}((1-p_k)\lambda)][S_{2k-2}(p_k \lambda)^2]$ where $p_k = (4 - 4^{1/(2k-1)})^{-1}$.
By the previous section we know that
(**) $||e^{\sum_{j=1}^{m} H_j \lambda} - S_{2k}(\lambda)|| \in O(|\lambda|^{2k+1})$ for $|\lambda| \to 0$, where we set $\lambda = -it$ for simulating Hamiltonians.

Lemma 1: Using integrators of order k and dividing the time into r intervals, we have the bound:
$||exp(\sum_{j=1}^{m} H_j \lambda) - S_{2k}(-it/r)^r|| \leq 5(2 \times 5^{k-1} m\tau)^{2k+1}/r^{2k}$ for
$4m5^{k-1}\tau/r \leq 1$, and $(16/3)(2 \times 5^{k-1} m\tau)^{2k+1}/r^{2k} \leq 1$.

Proof: Consider a taylor expansion of both terms in the left-hand side of (**), with $x = \lambda$. The terms containing λ to power less than $2k+1$ for a given k must cancel because the correction term is $O(|\lambda|^{2k+1})$, thus
$exp(\sum_{j=1}^{m} H_j \lambda) = S_{2k}(\lambda) + \sum_{k'=k}^{\infty} \sum_{l=1}^{L_{k'}} C_l^{k'} \prod_{q=1}^{2k'+1} H_{j_{lq}}$, where the constants C_l^k and the number of terms, denoted $L_{k'}$, depends on m and k, and $H_{j_{lq}}$ is some product of the H_i.
Because the operators H_j are in general noncommuting, expanding $(H_1 + \cdots + H_m)^{2k'+1}$ yields $m^{2k'+1}$ terms in its expansion. Therefore the Taylor expansion of $exp(\sum_{j=1}^{m} H_j \lambda)$ contains $m^{2k'+1}$ terms with $\lambda^{2k'+1}$. These terms have multiplying factors of $\frac{1}{2k'+1!}$ because that is the muliplying factor given by the Taylor expansion of the exponential.

Now we show that $S_{2k}(x)$ consists of a product of $2(m-1)5^{k-1} + 1$ exponentials for any $k \geq 1$. For $k = 1$, $S_2(x) = e^{H_1 x/2} e^{H_2 x/2} \cdots e^{H_m x} \cdots e^{H_2 x/2} e^{H_1 x/2}$ has $2m - 1 = 2(m-1)5^0 + 1$ terms. For $k > 1$, the operator S_{2k} contains $5N_{k-1} - 4$ exponentials. Note for any complex scalars t, r that matrices tA and rA commute, so that $e^{tA} e^{rA} = e^{(r+t)A}$ is counted as 1 exponential, thus at each k there are 4 exponentials that commute adjacent to one another.

Using the unfolding technique for recursive relations, we see that S_{2k} has
$N_k = 5^{k-1} N_1 - 4 \sum_{i=0}^{k-2} 5^i = 5^{k-1}(N_1 - 1) + 1$ exponentials where $N_1 = 2m - 1$ and so

$N_k = 2(m-1)5^{k-1}+1$ exponentials, as desired.

Examining each exponential individually in $S_{2k}(\lambda)$, I claim that there will be no more than $[2(m-1)5^{k-1}+1]^{2k'+1}$ terms with $\lambda^{2k'+1}$. The number of ways that an element can be chosen in each individual expansion such that the powers of λ add up to some $2k'+1$ is given by $\binom{((2k'+1)+N_k-1)!}{(N_k-1)!(2k'+1)!} < N_k^{2k'+1}$ (the weak composition combinatorial formula). Now $|p_k| < 1$ since for $k > \frac{ln(4)-ln(3)}{2ln(3)}$, $4-4^{1/(2k-1)} > 1 \implies \frac{1}{4-4^{1/(2k-1)}} > 1$, and $|1-4p_k| < 1$, so that the multiplying factors corresponding to each of the terms in S_{2k} with scalar coefficient $\lambda_{2k'+1}$ are less than one. Assume that $||H_j|| \leq ||H||$ (the trace norm, which is sub-multiplicative, yields $||H_j^k|| \leq ||H_j||^k$). Let $\Lambda = ||H||$, then

$||\sum_{k'=k}^{\infty} \lambda^{2k'+1} \sum_{l=1}^{L_{k'}} C_l^{k'} \prod_{q=1}^{2k'+1} H_{j_{lq}}|| \leq \sum_{k'=k}^{\infty} |\lambda\Lambda|^{2k'+1} L_{k'}$
$\leq \sum_{k'=k}^{\infty} |\lambda\Lambda|^{2k'+1}[m^{2k'+1} + [2(m-1)5^{k-1}+1]^{2k'+1}]$
$\leq 2\sum_{k'=k}^{\infty} |\lambda\Lambda|^{2k'+1}[2m5^{k-1}]^{2k'+1} = \frac{2|2m5^{k-1}\lambda\Lambda|^{2k+1}}{1-|2m5^{k-1}\lambda\Lambda|^2}$ where for $|2m5^{k-1}\lambda\Lambda| \leq 1/2$, the left hand side is a convergent geometric series. Therefore, we obtain the inequality :

$||e^{\lambda\sum_{j=1}^m H_j} - S_{2k}(\lambda)|| \leq (8/3)|2m5^{k-1}\lambda\Lambda|^{2k+1}$. Substituting $\lambda = -it/r$, where r is an integer, and taking the power of r gives the error bound:
$||e^{-it\sum_{j=1}^m H_j} - S_{2k}(-it/r)^r|| \leq [1+(8/3)(2m5^{k-1}\Lambda t/r)^{2k+1}]^r - 1$, for $4m5^{k-1}\lambda t/r \leq 1$.
Lemma 1 follows.

Using Lemma 1 and the fact that
$\|U_1 - U_2\| \geq \|U_1|\psi\rangle - U_2|\psi\rangle\| \geq \frac{1}{2}\,\text{Trace}\left|U_1|\psi\rangle\langle\psi|U_1^\dagger - U_2|\psi\rangle\langle\psi|U_2^\dagger\right|$

$= D(U_1|\psi\rangle\langle\psi|U_1^\dagger - U_2|\psi\rangle\langle\psi|U_2^\dagger)$, (the trace distance)
theorem 1 can be proven, where we omit the proof due to its technical detail.

4.7 Problem 2:

In order to simulate the Hamiltonian, we decompose it into the form $H = \sum_{j=1}^m H_j$ where each H_j is 1-sparse. If H_j is 1-sparse then it can be shown that it is possible to directly simulate e^{-iH_jt} with just two black-box queries to H_j. Since the value of m impacts the total cost of H, it is desirable to make m as small as possible. This motivates problem 2.

Problem 2:
The Hamiltonian H has no more than d non-zero entries in each column, and there exists a black-box function f that gives these entries. The dimension of the space which H acts upon does not exceed 2^n. If the non-zero elements in column x are

given by the indices $y_1, \ldots, y_{d'}$, where $d' \leq d$, then $f(x,i) = (y_i, H_{x,y_i})$ for $i \leq d'$, and $f(x,i) = (x,0)$ for $i > d'$. The problem is to simulate the evolution e^{-iHt} such that the maximum error in the final state, as quantified by the trace distance, does not exceed ϵ. We wish to determine the scaling of the number of call to f, denoted N_{bb} (bb for blackbox), required for the simulation.

For each x, the order of the y_i given can be arbitrary. The function f is an arbitrary black-box function, but we assume that there is a corresponding unitary U_f such that $U_f|x,i\rangle|0\rangle = |\phi_{x,i}\rangle|y_i, H_{x,y_i}\rangle$ and we may perform calls to both U_f and $U_f^\dagger$. Here $|\phi_{x,i}\rangle$ represents any additional states which are produced in the reversible calculation of f, and the notation $|a,b\rangle$ is equivalent to $|a\rangle|b\rangle$

Theorem 2:. The number of black-box calls for given k is $N_{bb} \in O((log^* n) d^2 5^{2k} (d^2\tau)^{1+1/2k}/\epsilon^{1/2k}$ with $log^* n \equiv min\{r | log_2^{(r)} n < 2\}$.

4.8 Lemma 2:

There exists a decomposition $H = \sum_{j=1}^{m} H_j$, where each H_j is 1-sparse, such that $m = 6d^2$ and each query to any H_j can be simulated by making $O(log^* n)$ queries to H.

Proof: From the black-box function defined for H in problem, we want to be able to determine corresponding black-box function for each H_j that gives the nonzero row number y, and matrix element element corresponding to each column x, if it exists. The black-box for H_j is represented by the function $g(x,j) := (y, (H_j)_{x,y})$, and if there is no non-zero element in column x, the output is $(x,0)$.
Consider the graph G_H associated with H whose vertex set is $\{0,1\}^n$ (2^n vertices corresponding to the number of columns/rows of our square matrix), where each vertex corresponds to a row or column number (depending on the order in the edge being considered), and there is an edge between vertices x, y, if the matrix element $H_{x,y}$ is non-zero. Since H is a Hermitian matrix, then $H_{i,j} = \overline{H_{i,j}}$, and therefore we know that the graph is undirected, meaning if there is an edge from x, y, there will also be an edge y, x. The edges in our graph correspond to non-zero matrix entries. Example:

$$H_1 = \begin{bmatrix} 0 & 2+i \\ 2-i & 0 \end{bmatrix}, H_1 = \begin{bmatrix} 0 & 2+i & 4-i & 1 \\ 2-i & 0 & 1 & 0 \\ 4+i & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Define an *edge-coloring* of a graph as an assignment of colors to the edges of a graph

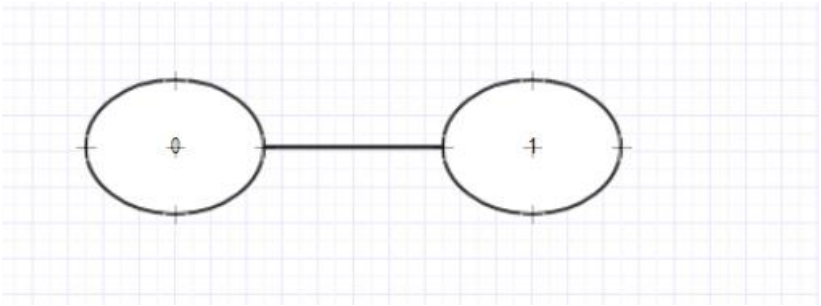


Figure 9: Graph of H_1

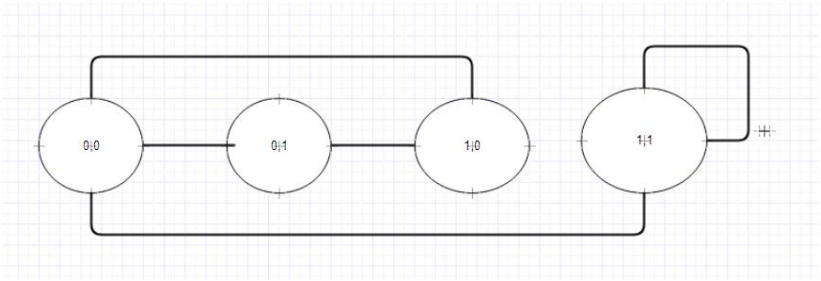


Figure 10: Graph of H_2

so that no two incident edges have the same colour. Two edges are said to be incident if they share a vertex, (ex.: (x, y), (y, z)). We wish to determine an edge-coloring of G_H, where each edge color j, corresponds to a different Hamiltonian H_j in the decomposition of H.

Index a set of labels in the set $\{1, \ldots, d\}^2$ (each vertice has at most d outgoing edges), where d is the sparsity of H. Note that for a Hermitian matrix, if no column has more than d entries, than no row has more than d entries. Denote f_y to be the y component of f, then $f_y(x, i)$ gives the i'th neighbor of vertex x in the graph. Let (x, y) be an edge of G_H such that $y = f_y(x, i)$, and $x = f_y(y, j)$, i.e. that y is the i'th neighbour of x, and x is the j'th neighbour of y. Label the edge (x, y) with the ordered pair (i, j) for $x \leq y$, or (j, i) for $x \geq y$. This labeling is not quite an edge-coloring, for if $w < x < y$ it is possible for (w, x) and (x, y) to both have the label (i, j) in the scenario where y, w are the i'th, j'th neighbours of x respectively, while x is the i'th, j'th neighbour of w, y respectively. To ensure that the labels are unique, we add the additional parameter ν, obtaining a label (i, j, ν), where next we describe how to assign values to ν. Set $x_0^{(0)} = x$, then determine a sequence of vertices $x_0^{(0)} < x_1^{(0)} < x_2^{(0)} < \cdots$ such that $x_{l+1}^{(0)} = f_y(x_l^{(0)}, i)$ and $f_y(x_{l+1}^{(0)}, j) = x_l^{(0)}$, so that the edges $(x_l^{(0)}, x_{l+1}^{(0)})$ will be labeled (i, j, ν) with the same values of i, j for each edge.

A typical chain may have only two elements, but in the case that long chains may be formed, we do not determine it any further than $x_{z_n+1}^{(0)}$ where z_n is the number of

times we must iterate $l \mapsto 2\lfloor log_2(l) \rfloor$ starting at 2^n, to obtain less than 6, this quantity is of order $log^*(n)$ (as it is an iterated logarithm), and for any realistic problem size the authors state that z_n itself will be no more than 6.

Now we construct a second sequence of values $x_l^{(1)}$ which has the same length as $x_l^{(0)}$. For each $x_l^{(0)}$ and $x_{l+1}^{(0)}$, we determine the first bit position where these two numbers differ, and record the value of this bit for $x_l^{(0)}$, followed by the binary position of this position, as $x_l^{(1)}$. The bit positions are numbered from zero (the first bit is numbered $00\ldots0$). If $x_l^{(0)}$ is at the end of the sequence, take $x_l^{(1)}$ to be the first bit of $x_l^{(0)}$ followed by the binary representation of 0. There are 2^n possible values for each of the $x_l^{(0)}$ since that is the number of vertices in our graph, and $2n$ possible values for each of the $x_l^{(1)}$ (n different positions where $x_l^{(0)}$ and $x_{l+1}^{(0)}$ could differ, and two possible values at that position).

By definition each $x_l^{(0)}$ is unique since they are strictly increasing, and moreover $x_l^{(1)}$ must differ from $x_{l+1}^{(1)}$. Consider two cases :

1) $x_{l+1}^{(0)}$ is not the last element in the first chain. Then even if the positions of the first bit where $x_l^{(0)}$ differs from $x_{l+1}^{(0)}$ and $x_{l+1}^{(0)}$ differs from $x_{l+2}^{(0)}$ are the same, the value of this bit for $x_l^{(0)}$ will be different from that of $x_{l+1}^{(0)}$. Since $x_l^{(1)}$ contains both the position and the value of the bit, even if the positions are the same, the first bit of $x_l^{(1)}$ and $x_{l+1}^{(1)}$ will differ.

2) $x_{l+1}^{(0)}$ is at the end of the first chain. In this case $x_{l+1}^{(1)}$ contains the first bit of $x_{l+1}^{(0)}$, and so if $x_l^{(0)}$ and $x_{l+1}^{(0)}$ differ at the first bit, $x_l^{(1)}$ and $x_{l+1}^{(1)}$ will differ only at the first bit, whereas if $x_l^{(0)}$ and $x_{l+1}^{(0)}$ differ at another position, then $x_l^{(1)}$ and $x_{l+1}^{(1)}$ will clearly differ.

We repeat this process until we determine the sequence of values $x_l^{(z_n)}$. At each step $x_l^{(p)}$ differs from $x_{l+1}^{(p)}$ in the same way as did $p = 1$. Note that, as we go from p to $p + 1$, the number of possible of values for $x_l^{(p)}$ is reduced as in $k \mapsto 2\lfloor log_2(k) \rfloor$ (2 possible values from $\{0, 1\} \times \lfloor log_2(k) \rfloor$ positions that could be different), and so by the definition of z_n, there six possible values for $x_0^{(z_n)}$.

Suppose that w, x are vertices such that $w < x$ with $x = f_y(w, i)$ and $w = f_y(x, j)$, then set $w_0^{(0)} = w$ and determine $w_0^{(z_n)}$ as per the above algorithm. Note that $x_1^{(0)} = w$ by our definitions. If the chain of $x_l^{(0)}$ ends sooner than $x_{z_{n+1}}^{(0)}$, then $x_l^{(p)}$ will be the same as $w_{l+1}^{(p)}$, and moreover $x_0^{(z_n)}$ will be equal to $w_1^{(z_n)}$, because the first chain of x

will simply be the corresponding chain of w, but missing the first entry w, and so continuing that pattern we see $w_0^{(z_n)} \neq x_0^{(z_n)}$.

On the other hand, if the chain of $x_l^{(0)}$ does extend all the way to $x_{z_n+1}^{(0)}$, then the chain for w will end at $w_{z_n+1}^{(0)}$ which is equal to $x_{z_n}^{(0)}$. Then $w_{z_n+1}^{(0)}$ will be calculated differently to $x_{z_n}^{(1)}$, but $w_{z_n}^{(1)}$ will be equal to $x_{z_n-1}^{(1)}$, and in general at step p, $w_{z_n-p+1}^{(p)}$ will be equal to $x_{z_n-p}^{(p)}$. In particular, at the last chain $w_1^{(z_n)}$ will be equal to $x_0^{(z_n)}$, thus as in the previous case, $w_0^{z_n} \neq x_0^{(z_n)}$.

Assign the edge (x, y) the color (i, j, ν) where $\nu = x_0^{(z_n)}$, and by construction, if the edge (w, x) has the same values of i, j as (x, y), it must have a different value of ν. Therefore, we have obtained an edge coloring of our graph.

Now we wish to describe how to calculate the black-box g using this approach. The individual Hamiltonians will follow the labelling scheme, meaning they are of the form $H_{(i,j,\nu)}$. The black box function g is now defined as $g(x, i, j, \nu) = (y, H_{(i,j,v)_{x,y}})$, where y is the row number for column x corresponding to a non-zero entry in the 1-sparse matrix $H_{(i,j,v)}$, if it exists. Define a function $T(x, i, j)$ to be equal to the ν calculated as in the above way. There are three cases where g will yield a non-zero output.

Case 1: $f_y(x, i) = x, i = j, \nu = 0$. This case corresponds to diagonal elements of the original Hamiltonian. The function will only give a non-zero result for $\nu = 0$, in order to prevent this element being repeated in different Hamiltonians $H_{(i,j,\nu)}$. In this case g will be defined to return $f(x, i)$.

Case 2: $f_y(x, i) > x, f_y(f_y(x, i), j) = x$ and $T(x, i, j) = \nu$. This case corresponds to there existing a $y > x$ such that y is the i'th neighbor of x and x is the j'th neighbor of y. In this case, we return $f(x, i)$.

Case 3: $f_y(x, i) < x, f_y(f_y(x, i), j) = x$ and $T(x, i, j) = \nu$. This case corresponds to there existing a $w < x$ such that w is the j'th neighbor of x and x is the i'th neighbor of w. In this case we return $f(x, j)$. The uniqueness of the labelling ensures that cases $2, 3$ are never true for the same values of (i, j) corresponding to the same edge in the graph.

There are d possible values for i, j, and ν may take six values, and so there are $6d^2$ colors. Let $m = 6d^2$. In determining ν, a maximum of $2(z_n + 2)$ queries are required to the black-box f since

-for each sequence element we check if $f_y(x,i) = y, f_y(f_y(x,i),j)) = x$
-The maximum length of the initial chain is the cardinality of $\{z_0, z_1, \cdots, z_n + 1\}$, which is $z_n + 2$. For a given row, our black box gives either a unique non-trivial element, or 0, and so $H_{(i,j,\nu)}$ is 1-sparse.

4.9 Proof of Theorem 2:

The number of Hamiltonians $H_{(i,j,\nu)}$ in the decomposition is $m = 6d^2$. To calculate $g(x,i,j,\nu)$, it is necessary to call the black-box $2(z_n + 2)$ times.
To simulate evolution under the Hamiltonian $H_{(i,j,\nu)}$, we require g to be implemented by a unitary operator U_g satisfying :
$U_g|x,i,j,\nu\rangle|0\rangle = |x,i,j,\nu\rangle|y, H_{(i,j,\nu)_{x,y}}\rangle$. As discussed in Problem 2, the function f may be represented by a unitary U_f, and using this unitary along with the function g defined in the lemma, it is straightforward to obtain a $\tilde{U_g}$, such that $\tilde{U_g}|x,i,j,\nu\rangle|0\rangle = |\phi_{x,i,j,\nu}\rangle|y, H_{(i,j,\nu)_{x,y}}\rangle$. We can obtain U_g in the usual way by applying $\tilde{U_g}$ copying the output, and then applying $\tilde{U_g}^\dagger$. As z_n is of order $log^*(n)$, and since by theorem 1 the bound on the number of exponentials needed to simulate e^{-iHt} for a given m by a sequence of products exponentials is bounded by $2m5^{2k}(m\tau)^{1+1/2k}/\epsilon^{1/2k}$ for a given error ϵ, then the result follows by multiplication of this bound by the number of black box calls necessary for simulating each H_j. Namely that for given integer k, taking $m = d^2$, that $N_{bb} \in O((log^*n)d^2 5^{2k}(d^2\tau)^{1+1/2k}/\epsilon^{1/2k}$.

5 Amplitude Amplification:

In this section the amplitude amplification algorithm is presented. Additionally the QSearch algorithm is given for when the probability of obtaining a *good* state is unknown in the context of a quantum algorithm that does not perform measurements. Afterwards, a section on the implementation of controlled rotations is given, which are used in several other sections of this report.

5.1 Quadratic Speedup:

Quantum amplitude amplification [10] is a generalization of Grover's search algorithm designed to boost the amplitude of being in a certain subspace of a Hilbert space. Suppose that H is a finite-dimensional Hilbert space representing the state space of a quantum system spanned by the orthonormal basis states $|x_i\rangle_{i=1}^n \in H$. Every Boolean function $\chi : \mathbb{Z} \to \{0,1\}$ induces a partition of H (mapping the indices of the basis elements) into a direct sum of two subspaces, a good subspace, and a bad subspace.

These consist of the set of basis states $|x\rangle \in H$ where $\chi(x) = 1$, and $\chi(x) = 0$, respectively. Every pure state $|\Gamma\rangle$ in H has a unique decomposition $|\Gamma\rangle = |\Gamma_0\rangle + |\Gamma_1\rangle$, where
- $|\Gamma_0\rangle$ is the projection of $|\Gamma\rangle$ onto the bad subspace, referred to as a bad state
- $|\Gamma_1\rangle$ is the projection of $|\Gamma\rangle$ onto the good subspace, referred to as a good state

Let $a_\Gamma = \langle\Gamma_1|\Gamma_1\rangle$ denote the probability that measuring Γ produces a good state, and similarly, let $b_\Gamma = \langle\Gamma_0|\Gamma_0\rangle$. Since $|\Gamma_0\rangle$ and $|\Gamma_1\rangle$ are orthogonal, we have that $a_\Gamma + b_\Gamma = 1$.

Let A be any quantum algorithm that acts on H and uses no measurements, with $|\Psi\rangle = A|0\rangle$ (let $|0\rangle = |x_0\rangle$). Define $Q(A, \chi) = -AS_0A^{-1}S_\chi$, where $S_\chi|y\rangle = -1^{\chi(y)}|y\rangle$, and S_0 which changes the sign of the amplitude iff the state is the zero state $|0\rangle$. The operator Q is well-defined since we assume that A has no measurements, and therefore has an inverse. Note that equivalently, $S_0 = I - 2|0\rangle\langle 0|$. Define $a = \langle\Psi_1|\Psi_1\rangle$, then $\langle\Psi_0|\Psi_0\rangle = 1 - a$.

Now $Q|\Psi_0\rangle = -AS_0A^{-1}S_\chi|\Psi_0\rangle = -AS_0A^{-1}|\Psi_0\rangle$, and by $A|0\rangle = |\Psi\rangle = |\Psi_0\rangle + |\Psi_1\rangle$ (as in the projection described above) , and correspondingly $\langle 0|A^{-1} = \langle\Psi_0| + \langle\Psi_1|$, therefore we have

$Q|\Psi_0\rangle = -A(I - 2|0\rangle\langle 0|)A^{-1}|\Psi_0\rangle = -(I - 2(|\Psi_0\rangle + |\Psi_1\rangle)(\langle\Psi_0| + \langle\Psi_1|))|\Psi_0\rangle$
$= -(|\Psi_0\rangle - 2(|\Psi_0\rangle + |\Psi_1\rangle)(1 - a))$
$= 2(1 - a)|\Psi_1\rangle + (1 - 2a)|\Psi_0\rangle$, expanding similarly
$Q|\Psi_1\rangle = (1 - 2a)|\Psi_1\rangle - 2a|\Psi_0\rangle$.

Suppose that $0 < a < 1$, and define H_Ψ to be the subspace spanned by the vectors $|\Psi_0\rangle$,$|\Psi_1\rangle$. The action of Q on H_Ψ is equivalent to the operator $U_\Psi U_{\Psi_0}$, where $U_{\Psi_0} = I - \frac{2}{1-a}|\Psi_0\rangle\langle\Psi_0|$, and $U_\Psi = I - 2|\Psi\rangle\langle\Psi|$, and
-U_{Ψ_0} is a reflection by the ray spanned by vector $|\Psi_0\rangle$
-U_Ψ is a reflection by the ray spanned by vector $|\Psi\rangle$

Consider the orthogonal complement $H_\Psi^\perp$ of H_Ψ, upon which AS_0A^{-1} acts as the identity. This is true since S_0 is equal to the identity unless when acting on $|0\rangle$, but $A^{-1}|y\rangle = |0\rangle$ implies $|y\rangle = |\Psi\rangle \notin H_\Psi^\perp$. Therefore Q acts as $-S_\chi$ on $H_\Psi^\perp$. But in this case $Q^2 = (-S_\chi)(-S_\chi) = I$. Take an eigenvector b of Q in $H_\Psi^\perp$, then $Q|b\rangle = \lambda|b\rangle \implies Q^2|b\rangle = \lambda^2 b = b \implies \lambda = \pm 1$. It follows that to understand the action of Q on an arbitrary initial vector $|\Gamma\rangle$ in H, it suffices to consider the action of Q on the projection of $|\Gamma\rangle$ onto H_Ψ.

Define $|\Psi_{\pm}\rangle = \frac{1}{\sqrt{2}}(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle)$. Then
$Q|\Psi_+\rangle = Q(\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{a}}|\Psi_1\rangle + \frac{i}{\sqrt{1-a}}|\Psi_0\rangle))$
$= (\frac{1}{\sqrt{2}}\frac{1}{\sqrt{a}})[(1-2a)|\Psi_1\rangle - 2a|\Psi_0\rangle] + (\frac{1}{\sqrt{2}}\frac{i}{\sqrt{1-a}})[2(1-a)|\Psi_1\rangle + (1-2a)|\Psi_0\rangle]$

Define θ_a such that $sin^2(\theta_a) = a$, then $e^{i2\theta_a} = cos(2\theta_a) + isin(2\theta_a)$
$= 1 - 2sin^2(\theta_a) + i2sin(\theta_a)cos(\theta_a) = 1 - 2a + 2i\sqrt{a}\sqrt{1-a}$, and therefore after expanding and simplifying, it is true that
$Q|\Psi_+\rangle = e^{i2\theta_a}|\Psi_+\rangle = (1 - 2a + 2i\sqrt{a}\sqrt{1-a})(\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{a}}|\Psi_1\rangle + \frac{i}{\sqrt{1-a}}|\Psi_0\rangle))$
$= (\frac{1}{\sqrt{2}}\frac{1}{\sqrt{a}})[(1-2a)|\Psi_1\rangle - 2a|\Psi_0\rangle] + (\frac{1}{\sqrt{2}}\frac{i}{\sqrt{1-a}})[2(1-a)|\Psi_1\rangle + (1-2a)|\Psi_0\rangle]$, and therefore $|\Psi_+\rangle$ is an eigenvector of Q with eigenvalue $e^{i2\theta_a}$.

Similarly $|\Psi_-\rangle := \frac{1}{\sqrt{2}}(\frac{1}{\sqrt{a}}|\Psi_1\rangle - \frac{i}{\sqrt{1-a}}|\Psi_0\rangle)$ is an eigenvector of Q with eigenvalue $e^{-i2\theta_a}$. Moreover since $\langle\Psi_0|\Psi_0\rangle = 1 - a$ and $\langle\Psi_1|\Psi_1\rangle = a$, then $\langle\Psi_+|\Psi_+\rangle = \langle\Psi_-|\Psi_-\rangle = 1$ and $\langle\Psi_+|\Psi_-\rangle = 0$, therefore $\{|\Psi_+\rangle, |\Psi_-\rangle\}$ is an orthonormal basis of H_Ψ (a subspace of H of dimension 2).

The state $|\Psi\rangle = A|0\rangle$ can be expressed in the eigenvector basis as
$|\Psi\rangle = A|0\rangle = \frac{-i}{\sqrt{2}}(e^{i\theta_a}|\Psi_+\rangle - e^{-i\theta_a}|\Psi_-\rangle)$, and after j applications of Q, we have that
$Q^j|\Psi\rangle = \frac{-i}{\sqrt{2}}(e^{i(2j+1)\theta_a}|\Psi_+\rangle - e^{-i(2j+1)\theta_a}|\Psi_-\rangle)$
$= \frac{1}{\sqrt{a}}sin((2j+1)\theta_a)|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}cos((2j+1)\theta_a)|\Psi_0\rangle$. If $0 < a < 1$, then the probability of producing a good state upon measurement is given by
$|\frac{1}{\sqrt{a}}\sin((2j+1)\theta_a)|\Psi_1\rangle|^2$
$= \frac{|\langle\Psi_1|\Psi_1\rangle|}{a}|\sin^2((2j+1)\theta_a)|^2$
$= \sin^2((2j+1)\theta_a)$ and it can be shown that if a is known and we take $m = \lfloor\frac{\pi}{4\theta_a}\rfloor$, then $sin^2((2m+1)\theta_a) \leq 1 - a$. This lends itself to the theorem :

Quadratic Speedup: Let H denote a finite dimensional Hilbert space representing the state space of a quantum system, spanned by computational basis vectors $|x_i\rangle_{i=0}^N$ for some positive integer N. Let A be any quantum algorithm acting on H that does not take measurements, and let $\chi : \mathbb{Z} \to \{0, 1\}$ be any Boolean function acting on the indices of the space. The Boolean function can be said to partition the computational basis states of the Hilbert space into the subspaces spanned by the elements $|x_i\rangle \in H$ where either $\chi(i) = 1$, the good subspace, or $\chi(i) = 0$, the bad subspace. Let a be the initial success probability of A, that is the probability of measuring a basis state $|x_i\rangle$ such that $\chi(i) = 1$ after applying $A|x_0\rangle$. Suppose that $a > 0$, and set $m = \lfloor\pi/4\theta_a\rfloor$, where θ_a is defined such that $sin^2(\theta_a) = a$ and $0 < \theta_a \leq \pi/2$. Then if we compute $Q^m A|0\rangle$ and measure the system, the outcome is in the good subspace with probability at least $max(1 - a, a)$.

The reasoning for the title given to this result, also referred to as the square-root running-time result, is that if an algorithm A has success probability $a > 0$, then after an expected number of $\frac{1}{a}$ applications of A, we will find a good solution. Applying the above theorem reduces this to an expected number of $\frac{2m+1}{max(1-a,a)} \in \Theta(\frac{1}{\sqrt{a}})$ applications of A and A^{-1}.

5.2 QSearch:

In the case that the value of a is not known, there exists a good solution without prior computation of an estimate of a known as **QSearch**.

Theorem: Quadratic Speedup without knowing a.

There exists a quantum algorithm **QSearch** with the following property. Let A be any quantum algorithm that uses non measurements, and let $\chi : \mathbb{Z} \to \{0,1\}$ be any Boolean function. Let a denote the inital success probability of A. Algorithm **QSearch** finds a good solution using an expected number of applications of A and A^{-1} which is in $\Theta(\frac{1}{\sqrt{a}})$ if $a > 0$, and otherwise runs forever. The algorithm is as follows:

Algorithm (QSearch(A, χ))

1. Set $l = 0$ and let c be any constant such that $1 < c < 2$.
2. Increase l by 1 and set $M = \lceil c^l \rceil$.
3. Apply A on the initial state $|0\rangle$, and measure the system. If the outcome $|z\rangle$ is good, that is, if $\chi(z) = 1$, then output z and stop.
4. Initialize a register of appropriate size to the state $A|0\rangle$.
5. Pick an integer j between 1 and M uniformly at random.
6. Apply Q^j to the register.
7. Measure the register. If the outcome $|z\rangle$ is good, then output z and stop. Otherwise, go to step 2.

5.3 Controlled Rotations:

Let $\theta \in \mathbb{R}$, and let $\tilde{\theta}$ be its d-bit finite precision representation. Then there is a unitary U_θ, that acts as $U_\theta : |\tilde{\theta}\rangle|0\rangle \mapsto |\tilde{\theta}\rangle(cos\tilde{\theta}|0\rangle + sin\tilde{\theta}|1\rangle)$. [6]

Proof. Define $U_\theta = \sum_{\tilde{\theta} \in \{0,1\}^d} |\tilde{\theta}\rangle\langle\tilde{\theta}| \otimes e^{-i\tilde{\theta}\sigma_y}$, where $\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$. Note that σ_y is Hermitian, and so U_θ is unitary. Observe the following. If $A = -i\theta\sigma_y$ then we have

that

$$\begin{aligned} A &= \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix} \\ A^2 &= \begin{bmatrix} -\theta^2 & 0 \\ 0 & -\theta^2 \end{bmatrix} \\ A^3 &= \begin{bmatrix} 0 & \theta^3 \\ -\theta^3 & 0 \end{bmatrix} \\ A^4 &= \begin{bmatrix} \theta^4 & 0 \\ 0 & \theta^4 \end{bmatrix} \end{aligned}$$

demonstrating that that the odd powers have nonzero off-diagonal entries, while the even powers have nonzero diagonal entries. We can therefore break up the sum for the exponential into an even part and an odd part:

$$\begin{aligned} \exp(A) &= \sum_{k=0}^{\infty} \frac{1}{k!} A^k \\ &= \sum_{k=0}^{\infty} \frac{1}{(2k)!} A^{2k} + \sum_{k=0}^{\infty} \frac{1}{(2k+1)!} A^{2k+1} \end{aligned}$$

Now notice for the even series we get:

$$\begin{aligned} \sum_{k=0}^{\infty} \frac{1}{(2k)!} \begin{bmatrix} -\theta^2 & 0 \\ 0 & -\theta^2 \end{bmatrix}^k &= \begin{bmatrix} \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} \theta^{2k} & 0 \\ 0 & \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} \theta^{2k} \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & 0 \\ 0 & \cos\theta \end{bmatrix}. \end{aligned}$$

for the odd part we get:

$$\sum_{k=0}^{\infty} \frac{1}{(2k+1)!} \begin{bmatrix} 0 & (-1)^{2k+1}\theta^{2k+1} \\ (-1)^{2k}\theta^{2k+1} & 0 \end{bmatrix} = \begin{bmatrix} 0 & -sin\theta \\ sin\theta & 0 \end{bmatrix}.$$

and so the sum of the even and odd terms is the 2 dimensional rotation matrix at θ, which is unitary. Applying U_θ to $|\tilde{\theta}\rangle|0\rangle$ yields the desired result.

The unitary operation U_θ can be implemented in $O(d)$ gates, where d is the number of bits representing $\tilde{\theta}$, using one rotation controlled on each qubit of the representation, with the angle of rotation for successive bits cut in half.

In other words, suppose $|\tilde{\theta}\rangle = |z_1\rangle \cdots |z_d\rangle$ for $|z_i\rangle \in \{|0\rangle, |1\rangle\}$ where $\theta \approx \sum_{i=1}^{d} z_i \pi / 2^i$

Define $U_{c_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\pi/2^i) & -sin(\pi/2^i) \\ 0 & 0 & sin(\pi/2^i) & cos(\pi/2^i) \end{bmatrix}$

where each U_{c_i} is unitary, and we assume there is a 2-qubit gate (or a constant number of gates) that can implement this operation. If the i'th gate acts on the i'th qubit of $|\tilde{\theta}\rangle$ and the target (the ancilla register), we will achieve

$$\prod_{i=1}^{d} R_z(\frac{z_i}{2^i}\pi) = R_z(\sum_{i=1}^{d} \frac{z_i \pi}{2^i}) = R_z(\tilde{\theta})$$

which is the desired rotation.

6 Oracle QRAM:

Author Anupam Prakash describes several models for the implementation of Quantum RAM (Random Access Memory), as part of his doctoral dissertation, *Quantum Algorithms for Linear Algebra and Machine Learning*, from which the following material is drawn [11]. Here we discuss the issue of encoding vector data into quantum states. A vector state preparation of a vector $x \in \mathbb{R}^N$ is defined to be a copy of the vector state $|x\rangle = \frac{1}{||x||} \sum_{i=1}^{N} x_i |i\rangle$, which is encoded into a quantum state with $O(logN)$ qubits. In general it is advantageous to seek encoding algorithms which run in $O(polylog(N))$ time, which is potentially achievable only if the memory model allows queries in quantum superposition. The oracle $QRAM$ is a memory device capable of answering queries in quantum superposition. It is the standard model for memory allowing queries in quantum superposition. If the $QRAM$ has N memory cells with contents x_i, $1 \leq i \leq N$, it achieves the reversible transformation:

$\sum_{i=1}^{N} \alpha_i |i\rangle \rightarrow \sum_{i=1}^{N} \alpha_i |i\rangle |x_i\rangle$

We note that for all proposed QRAM architectures, the query register is used to address memory and does not interact with the memory contents, that is a transformation of the form $|i\rangle \rightarrow |i \oplus x_i\rangle$ can not be achieved. The query time for the oracle QRAM is $\tilde{O}(logn)$ for all proposed architectures, and possible physical realizations and architectures are a subject on ongoing research.
The vector state $|x\rangle$ can be generated by the following steps

QRAM Procedure 1:

Assume that we have N memory cells with each cell encoding the i'th component of some $x \in \mathbb{R}^N$ using basis states for an appropriate number of qubits to encode the integer and fractional part.

1. Query the oracle QRAM on the uniform superposition $\sum_{i=1}^{N} \frac{1}{\sqrt{N}}|i\rangle$ yielding $\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle|x_i\rangle$.
2. Add an ancilla qubit in the state $|0\rangle$.
3. Apply a controlled rotation R_{θ_i} conditioned on the x_i for each i where $\theta_i = arccos(\frac{x_i}{||x||_\infty})$, and $|x|_\infty = max_i|x_i|$. Let $\beta_i = (1 - (\frac{x_i}{|x|_\infty}^2))^{1/2}$, then the resulting state is $\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle|x_i\rangle(\frac{x_i}{||x||_\infty}|0\rangle + \beta_i|1\rangle)$
4. Uncompute the $|x_i\rangle$ by reversing the $QRAM$ operation.
5. Post-select upon measuring the ancilla qubit to be $|0\rangle$.

If $||x|| = 1$, the probability of obtaining $|x\rangle$ is $\frac{1}{N} \sum_{i=1}^{N} \frac{x_i^2}{||x||_\infty} = \frac{1}{N||x||_\infty^2}$, where it can be shown that the worst case time complexity for preparing $|x\rangle$ using this procedure is $\tilde{O}(N)$ for a basis vector e_i. The time complexity can be improved to $\tilde{O}(\sqrt{N})$ using amplitude amplification.

6.1 Oracle QRAM and Amplitude Amplification:

Here we describe the algorithm for the creation of vector states using amplitude amplification. Suppose again that $||x|| = 1$. First note that $QRAM$ can be used to implement the unitary U defined as
$U|0\rangle^{log_N+1} \rightarrow_{QFT_N} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle \rightarrow_{QRAMprocedure1} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle(\frac{x_i}{|x|_\infty}|0\rangle + \beta_i|1\rangle) := |\psi\rangle$,
where QFT is the quantum Fourier transform, and the QRAM procedure is outlined in the previous section.

The state $|\psi\rangle$ can be decomposed as $|\psi\rangle = sin(\theta)|x\rangle|0\rangle + cos(\theta)|x'\rangle|1\rangle$ where $sin^2(\theta) = \frac{1}{N||x||_\infty^2}$ is the probability of measuring the ancilla bit as $|0\rangle$ and obtaining $|x\rangle$, and $|x'\rangle$ is the state upon measuring $|1\rangle$. The transformations U, U^{-1} can be implemented in time $\tilde{O}(logn)$ using the oracle QRAM.

As in the amplitude amplification algorithm, we define two reflections:
- $S_\psi = |\psi\rangle$ and $S_\psi|\psi^\perp\rangle = -|\psi^\perp\rangle$ where $|\psi^\perp\rangle$ is state orthogonal to $|\psi\rangle$. This can be implemented as $-US_0U^{-1}$ where S_0 flips the sign of the amplitudes of a state iff the state is $|0\rangle^{logN+1}$
- S_x is a controlled phase flip conditioned on the ancillary qubit being $|1\rangle$.

In this case the good state is $sin(\theta)|x\rangle|0\rangle$, and the bad state is $cos(\theta)|x'\rangle|1\rangle$, after $k = O(\sqrt{N}|x|_\infty)$ iterations of amplitude amplification (a.k.a $(S_\psi S_x)^k$), the probability of obtaining the state $|x\rangle$ is no less than $max(sin^2(\theta), 1 - sin^2(\theta))$.

7 HHL algorithm:

The academic paper *Quantum algorithm for linear systems of equations* written by Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd (HHL) [5] aims to solve the following problem using quantum computers. Given an invertible matrix A, and a vector $\vec{b}$, we wish to find a vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. In the non-quantum setting, when A is $s-$sparse, $N \times N$, and has condition number κ, classical algorithms can find $\vec{x}$ and estimate $\vec{x}^{\dagger}M\vec{x}$ in $O(Ns\kappa\sqrt{1/\epsilon})$ time using the method of *conjugate gradients* for error ϵ. The HHL algorithm can perform this task in $poly(logN, \kappa)$ time, delivering an exponential improvement over the best classical algorithm. Denote $f(n) \in \tilde{O}(h(n)) \implies \exists k : f(n) \in O(h(n)log^{k}((h(n)))$, where this notation is used frequently in this section/paper.

7.1 Algorithm Sketch:

Given a Hermitian $N \times N$ invertible matrix A, and a unit vector $\vec{b}$, we would like to find $\vec{x}$ satisfying $A\vec{x} = \vec{b}$. Initially, the algorithm represents $\vec{b}$ as a quantum state $|b\rangle = \sum_{i=0}^{N} b_i|i\rangle$, which we are assumed has been pre-prepared. As seen in the phase-estimation algorithm, given a unitary operator U with eigenvectors $|u_j\rangle$, and corresponding complex eigenvalues $e^{i\theta_j}$, phase-estimation allows for the following mapping to be implemented:
$|0\rangle|u_j\rangle \mapsto |\tilde{\theta_j}\rangle|u_j\rangle$, where $\tilde{\theta_j}$ is the binary representation of θ_j to a certain precision.

Since A is Hermitian, with corresponding eigenstates $|u_j\rangle$, the exponential e^{iAt} is a unitary operator, with eigenvalues $e^{i\lambda_j t}$. Thus using Hamiltonian simulation and phase estimation, we can implement a transformation :
$|0\rangle|u_j\rangle \mapsto |\tilde{\lambda_j}\rangle|u_j\rangle$, where $|\tilde{\lambda_j}\rangle$ is the binary representation of an estimate of λ_j to some precision.

The next step of the algorithm is to perform a controlled rotation conditioned on $\tilde{\lambda_j}$ for each j, where we add an ancilla register to the system in state $|0\rangle$. Performing the controlled R_{θ_j} rotation where $sin(\theta_j) = \frac{C}{\tilde{\lambda_j}}$ results in a state of the form $\sqrt{1 - \frac{C^2}{\tilde{\lambda_j}^2}}|\tilde{\lambda_j}\rangle|u_j\rangle|0\rangle + \frac{C}{\tilde{\lambda_j}}|\tilde{\lambda_j}\rangle|u_j\rangle|1\rangle$, where C is a constant of normalisation.

Enacting this procedure on the superposition $|b\rangle = \sum_{j=1}^{N} \beta_j|u_j\rangle$ (expressing b in the eigenbasis of A), we get the state $\sum_{j=1}^{N} \beta_j|\tilde{\lambda_j}\rangle|u_j\rangle(\sqrt{1 - \frac{C^2}{\tilde{\lambda_j}^2}}|0\rangle + \frac{C}{\tilde{\lambda_j}}|1\rangle)$. Uncomputing the register containing the $|\tilde{\lambda_j}\rangle$ by reversing the phase estimation procedure (which after all is unitary), we get $|0\rangle \otimes \sum_{j=1}^{N} \beta_j|u_j\rangle(\sqrt{1 - \frac{C^2}{\tilde{\lambda_j}^2}}|0\rangle + \frac{C}{\tilde{\lambda_j}}|1\rangle)$.

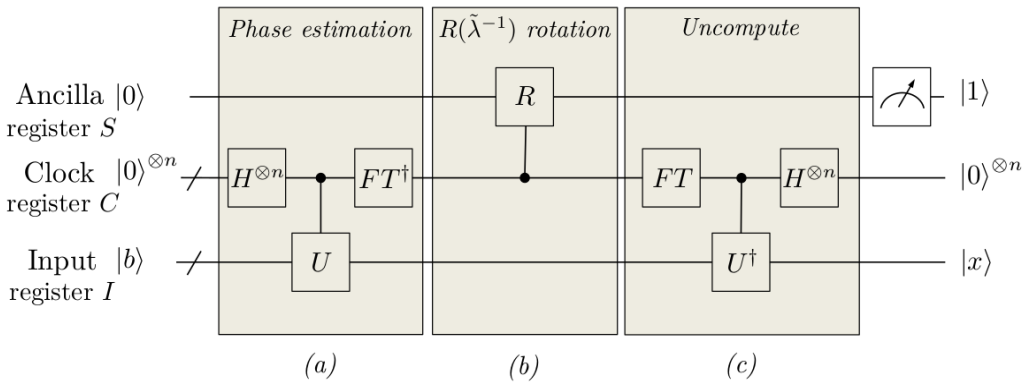


Figure 5: *HHL Algorithm Schematic:* Broadly, the algorithm proceeds in three main steps. *(a)* Phase estimation is used to estimate the eigenvalues of A, using $U = \sum_{k=0}^{T-1} |k\rangle\langle k|^C \otimes e^{iAkt_0/T}$, where $T = 2^t$, with t the number of qubits in the clock register C and $t_0 = 2\pi$. After this stage the state of the computation is $(\sum_{j=1}^{N} \beta_j |u_j\rangle^R |\tilde{\lambda}_j\rangle^I) \otimes |0\rangle^S$, where the $|u_j\rangle$ are the eigenvectors of A, the β_j are $|b\rangle$'s representation in this basis and the $|\tilde{\lambda}_j\rangle$ are the binary representations of the eigenvalues of A. *(b)* The controlled $R(\tilde{\lambda}^{-1})$ rotation extracts the eigenvalues of A^{-1} and executes a σ_y rotation, conditioned on $\tilde{\lambda}_j$, leaving the state as $\sum_{j=1}^{N} \beta_j |u_j\rangle_I |\tilde{\lambda}_j\rangle^C ((1 - C^2/\tilde{\lambda}_j^2)^{1/2} |0\rangle + C/\tilde{\lambda}_j |1\rangle)^S$, where C is a normalising constant. *(c)* The inverse phase estimation subroutine sets the register to $(|0\rangle^{\otimes n})^C$ and leaves the remaining state as $\sum_{j=1}^{N} \beta_j |u_j\rangle^I ((1 - C^2/\tilde{\lambda}_j^2)^{1/2} |0\rangle + C/\tilde{\lambda}_j |1\rangle)^S$, so postselecting on $|1\rangle^S$ gives the state $C \sum_{j=1}^{N} (\beta_j/\tilde{\lambda}_j) |u_j\rangle^I$, which is proportional to $|x\rangle^I$.

Figure 11: A simplified diagrammatic description of the HHL algorithm [6]

Now note $\lambda \neq 0$ is an eigenvalue of $A \iff \frac{1}{\lambda}$ is an eigenvalue of A^{-1}, thus a state proportional to $|x\rangle = A^{-1}|b\rangle = \sum_j \lambda_j^{-1} \beta_j |u_j\rangle$ can be constructed by postselecting on the outcome $|1\rangle$. It is possible to use amplitude amplification at this step to boost the success probability of measuring $|1\rangle$.

In analyzing the performance of the HHL algorithm, an important factor is κ, the condition number of A, which is the ratio between A's largest and smallest eigenvalues (see the section 9.2.8 for a detailed look at the condition number of a matrix). As the condition number increases, A becomes closer to a matrix which cannot be inverted, and the solutions become less reliable. Such a matrix is said to be *ill-conditioned* This algorithm assumes that the singular values of A lie between $\frac{1}{\kappa}$ and 1. In this scenario, the runtime will scale as $\kappa^2 log(N)/\epsilon$, where ϵ is the error achieved outputting the state $|x\rangle$. The greatest advantage this algorithm has over classical algorithms occurs when both κ and $\frac{1}{\epsilon}$ are $polylog(N)$ (polynomial time with input $logN$), in which case there is an exponential speedup. A simplified diagram is presented in Figure 11. The procedure yields a quantum-mechanical representation $|x\rangle$ of the desired vector $\vec{x}$. However, as the author/s note, often one is interested not just in $\vec{x}$ itself, but in some expectation value $\vec{x}^\dagger M \vec{x}$ where M is some linear operator. By mapping M to a quantum-mechanical operator, and performing the quantum measurement corre-

sponding to M, one can obtain an estimate value $\langle x|M|x\rangle$. Many features of $\vec{x}$ can be extracted including normalization, weights, moments, etc.

7.2 Algorithm Details:

First the algorithm transforms a Hermitian matrix A into a unitary operator e^{iAt} using techniques of Hamiltonian simulation. This is possible if A is s-sparse (meaning that A has at most $s < N$ non-zero entries per row). Under these assumptions, it can be shown that e^{iAt} can be simulated in $\tilde{O}(log(N)s^2t)$ time complexity.

If A is not Hermitian, define $C = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$. As C is Hermitian, we can solve the equation $C\vec{y} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix}$ to obtain $\begin{bmatrix} 0 \\ \vec{x} \end{bmatrix}$. The algorithm assumes there is an efficient procedure to prepare $|b\rangle$ (such as the oracle QRAM method discussed in section 6).

The next step is to decompose $|b\rangle$ in the eigenvector basis, using phase estimation. There are three registers in the algorithm, the ancilla register S, the register labelled C containing the state $|\Psi_0\rangle$ to be defined, and the register I containing $|b\rangle$ which we have assumed is prepared in advance.

Let $|\Psi_0\rangle := \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} sin\frac{\pi(\tau+\frac{1}{2})}{T}|\tau\rangle$ for some large positive integer T. The coefficients of $|\Psi_0\rangle$ are chosen to minimze a quadratic loss function (discussed in the calculations below).

Consider a conditional Hamiltonian evolution at time t_0 with a sum of outer products of basis elements defined as $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{iA\tau to/T}$ for $t_0 \in O(\kappa/\epsilon)$ a time parameter with error ϵ. Recall the identity $(A \otimes B) \times (|a\rangle \otimes |b\rangle) = (A \times |a\rangle) \otimes (B \times |b\rangle)$, where $\times$ denotes matrix multiplication. This is relevant since:
$(\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{iA\tau to/T}) \times |\Psi_0\rangle|b\rangle = \sum_{\tau=0}^{T-1} (|\tau\rangle\langle\tau| \times |\Psi_0\rangle) \otimes (e^{iA\tau t_0/T} \times |b\rangle)$. Now consider the effect on an eigenvector $|u_j\rangle$ in place of $|b\rangle$, the resulting state is
$|\psi_{\lambda_j t_0}\rangle = \sqrt{\frac{2}{T}} \sum_{\tau=0}^{T-1} e^{\frac{i\lambda_j t_0 \tau}{T}} sin\frac{\pi(\tau+\frac{1}{2})}{T}|\tau\rangle|u_j\rangle$.

Next, we apply a quantum Fourier transform to the C register resulting, for fixed

$|u_j\rangle$, in the state $\sum_{j=1}^{N}\beta_j\sum_{k=0}^{T-1}(\frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1}e^{\frac{i\tau}{T}(\lambda_j t_0-2\pi k)}sin\frac{\pi(\tau+\frac{1}{2})}{T})|k\rangle|u_j\rangle$, where we define $\alpha_{k|j}=\frac{\sqrt{2}}{T}\sum_{\tau=0}^{T-1}e^{\frac{i\tau}{T}(\lambda_j t_0-2\pi k)}sin(\frac{\pi(\tau+1/2)}{T})$.

Let $\delta := \lambda_j t_0 - 2\pi k$. We now claim that the following bounds for the coefficients: $|\alpha_{k|j}|^2 \leq 64\pi^2/\sigma^4$ when $|k-\lambda_j t_0/2\pi| \geq 1$. First note that
$e^{ix}-e^{-ix} = cosx + isinx - (cos(-x)+isin(-x)) = 2isinx$. Applying this identity to $\alpha_{k|j}$, we get

$$\alpha_{k|j} = \frac{1}{i\sqrt{2}T}\sum_{\tau=0}^{T-1}e^{i\frac{\tau\delta}{T}}(e^{\frac{i\pi(\tau+1/2)}{T}}-e^{-\frac{\pi(\tau+1/2)}{T}})$$
$$= \frac{1}{i\sqrt{2}T}\sum_{\tau=0}^{T-1}e^{\frac{i\pi}{2T}}e^{i\tau\frac{\delta+\pi}{T}}-e^{-\frac{i\pi}{2T}}e^{i\tau\frac{\delta-\pi}{T}}, \text{ which is a finite geometric series}$$
$$= \frac{1}{i\sqrt{2}T}(e^{\frac{i\pi}{T}}\frac{1-e^{i\pi+i\delta}}{1-e^{i\frac{\delta+\pi}{T}}}-e^{\frac{-i\pi}{T}}\frac{1-e^{i\pi+i\delta}}{1-e^{i\frac{\delta-\pi}{T}}})$$

Now note $e^{i\pi}=-1$ so that we can factor out $(1+e^{i\delta})$ and multiplying the left fraction by $e^{-\frac{i}{2T}(\delta+\pi)}$ and the right by $e^{-\frac{i}{2T}(\delta-\pi)}$, we get
$$\frac{1+e^{i\delta}}{i\sqrt{2}T}(\frac{e^{-i\delta/2T}}{e^{\frac{-i}{2T}(\delta+\pi)}-e^{\frac{i}{2T}(\delta+\pi)}}-\frac{e^{-i\delta/2T}}{e^{\frac{-i}{2T}(\delta-\pi)}-e^{\frac{i}{2T}(\delta-\pi)}})$$

Now again using the identity $e^{ix}-e^{-ix}=cosx+isinx-(cos(-x)+isin(-x))=2isinx$, and factoring the numerator we get

$$\frac{(1+e^{i\delta})(e^{1\delta/2T)}}{i\sqrt{2}T}(\frac{1}{-2isin(\frac{\delta+\pi}{2T})}-\frac{1}{-2isin(\frac{\delta-\pi}{2T})})$$

Now noting that $e^{ix/2}+e^{-ix/2}=2cos(x/2)$, we can re-write the previous equation as

$$-e^{i\frac{\delta}{2}(1-\frac{1}{T})}\frac{\sqrt{2}cos(\frac{\delta}{2})}{T}(\frac{1}{sin(\frac{\delta+\pi}{2T})}-\frac{1}{sin(\frac{\delta-\pi}{2T})}) = -e^{i\frac{\delta}{2}(1-\frac{1}{T})}\frac{\sqrt{2}cos(\frac{\delta}{2})}{T}\frac{sin(\frac{\delta-\pi}{2T})-sin(\frac{\delta+\pi}{2T})}{sin(\frac{\delta+\pi}{2T})sin(\frac{\delta-\pi}{2T})}$$
$$= -e^{i\frac{\delta}{2}(1-\frac{1}{T})}\frac{\sqrt{2}cos(\frac{\delta}{2})}{T}\frac{2cos(\frac{\delta}{2T})sin(\frac{\pi}{2T})}{sin(\frac{\delta+\pi}{2T})sin(\frac{\delta-\pi}{2T})} \text{ (using sine angle addition formulas).}$$

Assume that $2\pi \leq \delta \leq T/10$. Further using the identity $\alpha-\alpha^3/6 \leq sin\alpha$ and ignoring the phases it is true that:

$\alpha_{k|j} \leq \frac{4\pi\sqrt{2}}{(\delta^2-\pi^2)(1-\frac{\delta^2+\pi^2}{3T^2})} \leq \frac{4\pi\sqrt{2}}{(\delta^2-(\delta/2)^2)(1-\frac{\delta^2+(\delta/2)^2}{3(\delta/10)^2})} \leq \frac{8\pi}{\delta^2}$, which implies that $|\alpha_{k|j}|^2 \leq \frac{64\pi^2}{\delta^4}$ whenever $\delta \geq 2\pi$, or equivalently when $|k-\lambda_j t_0/2\pi| \leq 1$. All this to say that $|a_{k|j}|$ is large $\implies \lambda_j \approx \frac{2\pi k}{t_0}$, therefore if the phase estimation were successful, we would have $\alpha_{k|j}=1$ if $k \approx \lambda_j t_0/2\pi$ and 0 otherwise.

Hence the Fourier basis state $|k\rangle$ is an estimate of λ_j and we can relabel our Fourier basis states $|k\rangle$ as $|\tilde{\lambda_k}\rangle$, so that now for fixed u_j we can say we are in the state $\sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j}|\tilde{\lambda_k}\rangle|u_j\rangle$.

The controlled rotations, described in the previous section, on the eigenvalues of A are used to apply A^{-1} to the input state, but it is important to consider the stability of the algorithm. Suppose that for $\mu \in \mathbb{R}$ that is close to zero, that we wish to compute $\frac{1}{\mu}$. A small change in μ results in a large change in $\frac{1}{\mu}$, and so in the context of the *HHL* algorithm, we would wish to invert the well conditioned part of the matrix, which the authors define as the eigenvalues $\lambda \geq \frac{1}{\kappa}$. If not, suppose for an eigenvalue $\lambda = e_\kappa/\kappa$ for some $0 < \epsilon_\kappa < 1$, that we invert this eigenvalue as per the algorithm. A small relative error in $\frac{1}{\lambda}$ will give a result deviating by the true value by a large factor of κ, which is the *characteristic* scale of the matrix at hand, and this error would dominate all other terms in the product $A^{-1}|b\rangle$ resulting in an overly large error.

To mitigate this, the authors introduce the notion of *filter functions* $f(\lambda), g(\lambda)$ that act to invert A only on the its well-conditioned subspace, the eigenvectors corresponding to eigenvalues $\lambda \geq \frac{1}{\kappa}$. The filter functions must be Lipschitz continuous, as per the Error Analysis section of the paper.

Given two metric spaces $(X, d_X), (Y, d_Y)$, a function $f : X \to Y$ is called **Lipschitz continous** if there exists a real constant $K \geq 0$ such that for all $x_1, x_2 \in X$, $d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$. Moreover if f is a real-valued continous and differentiable function, then f is Lipschitz $\iff$ the derivative of f is bounded.

Proof: Suppose that f is a real-valued continuous, differentiable function that is Lipschitz for a constant $M \geq 0$. Then for any $x \in \mathbb{R}$, it is true that $f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h} \leq \lim_{h\to 0} \frac{M|x+h-x|}{h} = M$, so that the derivative of f is bounded by M. Conversely if the derivative of f is bounded by some constant $K \geq 0$, then by the mean-value theorem, for any $x, y \in \mathbb{R}$, it is true that for some c, that $f(x) - f(y) = f'(c)(x - y) \leq K(x - y)$, and so f is Lipschitz.

The filter functions satisfy
$f(\lambda) = 1/C\kappa\lambda$ for $\lambda \leq 1/\kappa$
$g(\lambda) = 1/C$ for $\lambda \leq 1/\kappa'$ where $\kappa' = 2\kappa$
$f^2(\lambda) + g^2(\lambda) \leq 1$ for all λ.

After the Fourier transform is applied to register C, where now we are in the state $\sum_{j=1}^{N} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} |\tilde{\lambda_k}\rangle |u_j\rangle$, the next step in the algorithm is to adjoin a three-dimensional register
$|h(\tilde{\lambda_k})\rangle := \sqrt{1 - f(\tilde{\lambda_k})^2 - g(\tilde{\lambda_k})^2}|$ nothing $\rangle + f(\tilde{\lambda_k})|\text{well}\rangle + g(\tilde{\lambda_k})|\text{ill}\rangle$, where
-*nothing* corresponds to no inversion taking place
-*well* correspondins to a successful inversion
-*ill* indicates that part of $|b\rangle$ is in the ill-conditioned subspace of A.

The three states $|nothing\rangle, |well\rangle, |ill\rangle$ are given by a superposition of three mutually orthogonal quantum states (sometimes referred to as a qutrit), and could be represented as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \phi|2\rangle$ where $|\alpha|^2 + |\beta|^2 + |\phi|^2 = 1$. One possible choice for these functions is

$$f(\lambda) = \begin{cases} \frac{1}{2\kappa\lambda} \text{ when } \lambda \geq \frac{1}{\kappa} \\ \frac{1}{2} sin(\frac{\pi}{2} \cdot \frac{\lambda - \frac{1}{\kappa'}}{\frac{1}{\kappa} - \frac{1}{\kappa'}}) \text{ when } \frac{1}{\kappa} > \lambda \geq \frac{1}{\kappa'} \\ 0 \text{ when } \frac{1}{\kappa'} > \lambda \end{cases} \quad \text{and}$$

$$g(\lambda) = \begin{cases} 0 \text{ when } \lambda \geq \frac{1}{\kappa} \\ \frac{1}{2} cos(\frac{\pi}{2} \cdot \frac{\lambda - \frac{1}{\kappa'}}{\frac{1}{\kappa} - \frac{1}{\kappa'}}) \text{ when } \frac{1}{\kappa} > \lambda \geq \frac{1}{\kappa'} \\ \frac{1}{2} \text{ when } \frac{1}{\kappa'} > \lambda \end{cases}$$

where $\kappa' = 2\kappa$. In the next section we will show that the map $\lambda \mapsto |h(\lambda)\rangle$ is $O(\kappa)$-Lipschitz.

Finally after applying the filter functions, we uncompute the C register containing the eigenvalue estimates where now we have a state proportional to
$\sum_{j,\lambda_j \geq 1/\kappa} \lambda_j^{-1} \beta_j |u_j\rangle |well\rangle + \sum_{j,\lambda_j < 1/\kappa} \beta_j |u_j\rangle |ill\rangle$.

To summarize, define U_{invert}, where the algorithm to prepare $|b\rangle$ is denoted $B|initial\rangle$, as the following algorithm:
U_{invert}
1. Prepare $|\Psi_0\rangle$ from $|0\rangle$ up to error ϵ_Ψ.
2. Apply the conditional Hamiltonian evolution $\sum_{\tau=0}^{T-1} |\tau\rangle\langle\tau| \otimes e^{iA\tau to/T}$ up to error ϵ_H
3. Apply the quantum Fourier transform to the register C. Denote the resulting basis states with $|k\rangle$ for $k = 0, \ldots, T-1$. Define $\tilde{\lambda_j} := 2\pi k/t_0$
4. Adjoin a three-dimensional register S (referred to as the flag register) in the state
$|h(\tilde{\lambda_k})\rangle := \sqrt{1 - f(\tilde{\lambda_k})^2 - g(\tilde{\lambda_k})^2}|$ nothing $\rangle + f(\tilde{\lambda_k})|\text{well}\rangle + g(\tilde{\lambda_k})|ill\rangle$
5. Reverse the steps 1-3, which are unitary, uncomputing any garbage produced along

the way.

Finally using techniques of amplitude amplification described in section 5, define the two operators :
$R_{succ} = I - 2|well\rangle\langle well|$ acting only on the S register, and
$R_{init} = I - 2|initial\rangle\langle initial|$.

Therefore, as in the amplification procedure, we start with $U_{invert}B|initial\rangle$, and repeatedly apply $U_{invert}BR_{init}B^{\dagger}U_{invert}^{\dagger}R_{succ}$, and then measure S and stop when we obtain the result $|well\rangle$ (the good state), assuming the eigenvectors are all well-conditioned. If $\tilde{p}$ denotes the initial success of measuring $|well\rangle$, the authors show that the number of repetitions required would ideally be $\pi/4\sqrt{\tilde{p}} \in O(\kappa)$. While $\tilde{p}$ is initially unknown, using the QSearch procedure described in section 5, it can be shown that a constant probability of success is obtained using $\leq 4\kappa$ repetitions.

7.3 The Proposed Filter Functions are Lipschitz Continuous:

The map $\lambda \mapsto |h(\lambda)\rangle$ is $O(\kappa)$-Lipschitz, meaning that for any $\lambda_1 \neq \lambda_2$,
$|| \, |h(\lambda_1)\rangle - |h(\lambda_2\rangle|| \leq c\kappa|\lambda_1 - \lambda_2|$ for some $c \in O(1)$.

Proof: Since the map $\lambda \mapsto |h(\lambda)\rangle$ is continous everywhere, and differentiable in λ everywhere except at $1/\kappa$ and $1/\kappa'$, it suffices to bound the norm of the derivative of $|h(\lambda)\rangle$. Consider the three pieces of the function

When $\lambda > 1/\kappa$, then
$\frac{d}{d\lambda}|h(\lambda)\rangle = \frac{1}{2\kappa^2\lambda^3\sqrt{1-\frac{1}{2\kappa^2\lambda^2}}}|nothing\rangle - \frac{1}{2\kappa\lambda^2}|well\rangle$, where the square of this norm is equal to $\frac{1}{2\kappa^2\lambda^4(2\kappa^2\lambda^2-1)} + \frac{1}{4\kappa^2\lambda^4} \leq \kappa^2$ implying the norm of the derivative is less than or equal to κ.

When $\frac{1}{k'} < \lambda < \frac{1}{\kappa}$, the norm of the derivative is $\frac{1}{2}\frac{\pi}{2}\frac{1}{\frac{1}{\kappa}-\frac{1}{\kappa'}} = \frac{\pi}{2}\kappa$. (recall that $\kappa' = 2\kappa$)

Finally $\frac{d}{d\lambda}|h(\lambda)\rangle = 0$ when $\lambda < \frac{1}{\kappa'}$, therefore if we let $c = \frac{\pi}{2}$, the largest bound where $c \in O(\kappa)$, then the desired result follows.

7.4 Error Analysis:

To produce the input state $|b\rangle$, we assume that there exists an efficiently-implementable unitary B, and neglect the possibility that B errs in prodcuing $|b\rangle$. Let T_B be the number of gates required to implement B. Next the state $|\Psi_0\rangle$ has been shown to be producible with error ϵ_Ψ in time $polylog(T/\epsilon_\Psi)$. Another required subroutine is that of Hamiltonian simulation. Assume that A is Hermitian, $s-sparse$, and let $t \leq t_0$, where $t_0 \in O(\kappa/\epsilon)$ is the time parameter of the conditional Hamiltonian applied for some error ϵ. To simulate e^{iAt} requires time $T_H = \tilde{O}(logNs^2t_0)$ according to the authors' choice of Hamiltonian simulation method.

The dominant source of error is the (modified) phase estimation (steps $2, 3, 5$). This step errs by $O(1/t_0)$ in estimating λ, which translates into a relative error of $O(1/\lambda t_0)$ in λ^{-1}. As per theorem 1 (stated below), If $\lambda_j \geq 1/\kappa$ for each j, then the result will be tensored with the $|well\rangle$ register, indicating that a succesful inversion of A has been applied, and it is shown that taking $t_0 \in O(\kappa/\epsilon)$ introduces an error of ϵ. Since the amplitude amplification process uses $O(\kappa)$ repetitions, altogether, the overall run time $\tilde{O}(\kappa T_B + \kappa^2 s^2 log(N)/\epsilon)$ where the $\tilde{O}$ suppreses the more slowly-growing term ϵ_Ψ. We state theorem 1 for completeness.

Theorem 1:
Let U be the ideal version of U_{invert} in which there is no error in any step. Let $\tilde{U}$ denote a version of U_{invert} in which everthing except the phase estimation is exact, and let $t_0 \in O(\kappa/\epsilon)$.

1. In the case when no post-selection is performed, the error is bounded as
$||\tilde{U} - U|| \leq O(\kappa/t_0)$.

2. If we post-select on the flag register being in the space spanned by $\{|well\rangle, |ill\rangle\}$ and define the normalized ideal state to be $|x\rangle$, and our actual state to be $|\tilde{x}\rangle$, then $|| \, |\tilde{x}\rangle - |x\rangle|| \leq O(\kappa/t_0)$.

3. If $|b\rangle$ is entirely within the well-conditioned subspace of A and we post-select on the flag register being $|well\rangle$, then $|| \, |x\rangle - |x\rangle|| \leq O(\kappa/t_0)$.

7.5 Matrix Inversion is BQP-Complete:

Here we present a reduction from simulating a quantum circuit to matrix inversion in order to show that matrix inversion is **BQP**-complete for a quantum circuit using N

qubits, acting on T gates which can be simulated by inverting an $O(1)$-sparse matrix A where the condition number of A is $O(T)$.

Let C be a quantum circuit acting on $n = log_2 N$ qubits which applies T two-qubit gates $U_1, \ldots, U_T$. The initial state will be given by $|0\rangle^{\otimes n}$. Adjoin an ancilla register of dimension $3T$. Define the unitary operator U as

$$U := \sum_{t=1}^{T} |t+1\rangle\langle t| \otimes U_t + |t+T+1\rangle\langle t+T| \otimes I + |t+2T+1 \mod 3T\rangle\langle t+2T| \otimes U_{T+1-t}^{\dagger}$$

For example, in the case where $T = 2$, with basis states $|1\rangle, |2\rangle, \ldots |6\rangle$, the corresponding matrix would be:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & U_1^{\dagger} \\ U_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & U_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & U_2^{\dagger} & 0 \end{bmatrix}$$

The vector $|1\rangle|\psi\rangle$ is transformed as:
$|1\rangle|\psi\rangle \to |2\rangle U_1|\psi\rangle \to |T+1\rangle U_T \ldots U_1|\psi\rangle \to |2T+1\rangle U_T \ldots U_1|\psi\rangle \to |2T+2\rangle U_{T-1} \ldots U_1|\psi\rangle \to |3T\rangle U_1|\psi\rangle \to |1\rangle|\psi\rangle$.
Indeed for $T \leq t \leq 2T$, we see that $U^t(|1\rangle|\psi_t\rangle) = |t+1\rangle \otimes (U_T \cdots U_1|\psi\rangle)$, where notably the first register is the T gates being applied to the state $|\psi\rangle$. Further $U^{3T}|1\rangle|\psi\rangle = |1\rangle|\psi\rangle$.

Lemma: Let U be a unitary matrix, $T > 0$, and $A = I - Ue^{-1/T}$. Then the condition number of A is $O(T)$ [6].

Proof. Note that since unitary matrices preseve length,
$$\frac{\lambda_{max}(A)}{\lambda_{min}(A)} \leq \frac{max_{||x||_2=1} x^{\dagger}(I - Ue^{-1/T})x}{min_{||y||_2=1} y^{\dagger}(I - Ue^{-1/T})y} = \frac{1 - min_{||x||_2=1} x^t Ux \cdot e^{-1/t}}{1 - max_{||y||_2=1} y^t Uy \cdot e^{-1/t}} = \frac{1 + e^{-1/T}}{1 - e^{-1/T}}$$

since the eigenvalues of the identity matrix are 1, and the eigenvalues of a unitary matrix are have absolute value 1. Now let $f(x) = 2x - \frac{1+e^{-1/T}}{1-e^{-1/T}}$ for $x > 0$. Using the Laurent expansion of the function approaching infinity we get
$\lim_{x\to\infty} f(x) = \frac{-1}{6x} + \frac{1}{360x^3} - \frac{1}{15120x^5} + O((1/x)^6) \in \Theta(1/x)$, and so $\lim_{x\to\infty} f(x) = 0$, and therefore the upper bound of $\kappa(A)$ tends asymptotically to $2T$ and so is $O(T)$ $\square$

.

Further observe that $(I - Ue^{-1/T}) \sum_{k=0}^{\infty} U^k e^{-k/T}$ is a telescoping series where the $j'th$

partial sum is $I - U^j e^{-j/T} \to I$ as $j \to \infty$, therefore $A^{-1} = \sum_{k=0}^{\infty} U^k e^{-k/T}$.

Applying $A^{-1}|1\rangle|0\rangle^{\otimes n}$ will yield a state $|t\rangle U_T \dots U_1|0\rangle^{\otimes n}$ for $T+1 \leq t \leq 2T$ with approximate probability (for sufficiently large T) equal to
$\frac{\sum_{t=T+1}^{2T}(e^{-t/T})^2}{\sum_{t=1}^{3T}(e^{-t/T})^2} = \frac{e^2(e^2-1)}{e^6-1} \approx 0.11731.$

Therefore measuring the second register and sampling will yield $|x\rangle = A^{-1}|0\rangle$ with probability $\geq 1/10$. By applying $A^{-1}|1\rangle|0\rangle^{\otimes n}$ an iterative number k times for sufficiently large k until a successful measurement of $T+1 \leq t \leq 2T$ in the second register, corresponding to an application of the T gates to $|0\rangle^{\otimes n}$, we obtain a geometrically distributed random variable. Hence applying A^{-1} a sufficient number of times to $|0\rangle^{\otimes n}$ will yield a simulation of the T gates of our quantum circuit with high probability, and so we have constructed a reduction from any quantum circuit to matrix inversion, and so matrix inversion is **BQP**-hard. Further, since the application of matrix inversion to a quantum state of appropriate dimension can be computed on a quantum computer, the computation is in **BQP**, and so matrix inversion is **BQP**-complete, as desired.

This implies that a classical $poly(logN, \kappa, \frac{1}{\epsilon})$ algorithm would be able to simulate a $poly(n)$-gate quantum algorithm in $poly(n)$ time, which the authors note is strongly conjectured to be false. Consider the following weaker definition of matrix inversion. We say that an algorithm solves matrix inversion if its input and output are

1. Input: An $O(1)-$sparse matrix A specified either via an oracle or via a $poly(log(N))$-time algorithm that returns the nonzero elements in a row.

2. Output: A bit that equals one with probability $\langle x|Mx\rangle \pm \epsilon$ where $M = |0\rangle\langle 0| \otimes I_{N/2}$ corresponds to measuring the first qubit where $|x\rangle$ is a normalized state proportional to $A^{-1}|b\rangle$ for $|b\rangle = 0$.
Additionally, A is a Hermitian matrix with singular values between $1/\kappa$ and 1.

The authors then show, as per theorem 4,
1. If a quantum algorithm exists for matrix inversion running in time $k^{1-\delta}polylog(N)$ for some $\delta > 0$, then **BQP=PSPACE**.

The authors argue that if one can solve matrix inversion in time complexity $k^{1-\delta}polylog(N)$, then a computation with $T \leq 2^{2n}/18$ can be simulated with a polynomial number of qubits and gates. The TBQF - totally quantified boolean formula satisfiability problem is a decision problem that asks whether a quantified propositional sentence (with

possibly both the existential and universal quantifiers) over a set of Boolean variables is true or false. This problem is the canonical **PSPACE**-*complete problem*. It can be solved exhaustively by enumerating all possible assignments to the variables in time complexity $T \leq 2^{2n}/18$, therefore a quantum polynomial time for matrix inversion running in time complexity $k^{1-\delta}polylog(N)$ could efficiently solve TBQF, implying that **PSPACE = BQP**, which the authors note is unlikely.

8 Compiling Basic Linear Algebra Subroutines for Quantum Computers:

Keeping with the theme of linear algebra subroutines, here we discuss sections from the paper paper *Compiling basic linear algebra subroutines for quantum computers* written by authors Liming Zhao, Zhikuan Zhao, Patrick Rebentrost, Joseph Fitzsimons [7]. In particular subroutines for simulating elementary linear algebraic operations on a quantum computer, given the availability of unitaries generated from these matrices, are presented. The operations considered here are the exponentiation of:
-matrix addition and multiplication
-the Kronecker sum of two matrices
-the tensor product of two matrices
-the Hadamard product of two matrices
Further we discuss quantum inner product estimation as well as computing $x^\dagger Ay$ for a matrix $A \in \mathbb{C}^{N\times N}$ and vectors $x, y \in \mathbb{C}^N$. Additionally, for a set of matrices $\{A_j\}$, a method to embed the matrices into a set of Hermitian matrices is presented and we assume as a given that a set of unitary operators generated by the embedded matrices is available.

8.1 Assumptions of the paper:

Assumption 1: Given matrices A_j for $j = 1, \dots J$, let $||X_3(A_j)||_{max}\tau = O(1)$ (the maximum element norm is used) where τ is a time parameter, where $X_3(A)$ is defined in the embedding section. Assume access to the unitaries $e^{iX_3(A_j)\tau}$ as well as given arbitrary ancilla qubits, assume access to the controlled unitaries $e^{i|1\rangle\langle 1|\otimes X_3(A_j)\tau}$. If the matrices A_j and hence $X_3(A_j)$, are sparse (have mostly 0 entries), and are stored in a sparse matrix data structure, such unitaries are provided using techniques of *Hamiltonian Simulation* (see section 5).

Assumption 2: Assume a routine that prepares a quantum state for the classical vectors $v = x, y$, consisting of $|V_i(v)\rangle = V_i(v)/|v|$ for $i = 1, 2$ defined below.

8.2 Embedding:

Here we wish to discuss embedding matrices so that the result is Hermitian, along with a method of computing matrix-vector multiplication with the resulting embedding. Let $A \in C^{N\times N}$. Define the embedding matrices $X_1(A), X_2(A), X_3(A)$ as

$X_1(A) = (R_1 \otimes A) + (R_1^\dagger \otimes A^\dagger)$
$X_2(A) = (R_2 \otimes A) + (R_2^\dagger \otimes A^\dagger)$
$X_3(A) = (R_3 \otimes A) + (R_3^\dagger \otimes A^\dagger)$
where,

$$R_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} R_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Additionally let

$$r_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, r_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Note that $(A+B)^\dagger = A^\dagger + B^\dagger$. Further $X_1(A)$ is Hermitian since
$X_1(A)^\dagger = ((R_1 \otimes A) + (R_1^\dagger \otimes A^\dagger))^\dagger$
$= (R_1 \otimes A)^\dagger + (R_1^\dagger \otimes A^\dagger)^\dagger$
$= (R_1^\dagger \otimes A^\dagger))^\dagger + (R_1 \otimes A)$. (As the conjugate transpose of a tensor product is the product of the conjugate transposes). Similarly $X_2(A), X_3(A)$ are also Hermitian.

Define $V_1(x) = r_1 \otimes x$, $V_2(y) = r_2 \otimes y$, for $x, y \in \mathbb{C}^n$, i.e.

$$V_1(x) = \begin{bmatrix} x \\ 0_n \\ 0_n \end{bmatrix}, V_2(x) = \begin{bmatrix} 0_n \\ 0_n \\ y \end{bmatrix}$$

And so

$$V_1(x)^\dagger X_3(A) V_2(y) = \begin{bmatrix} x^\dagger & 0_n & 0_n \end{bmatrix} \begin{bmatrix} 0_n & 0_n & A \\ 0_n & 0_n & 0_n \\ A^\dagger & 0_n & 0_n \end{bmatrix} \begin{bmatrix} 0_n \\ 0_n \\ y_n \end{bmatrix} = x^\dagger A y$$

Hence the value of $x^\dagger A y$ can be obtained by calculating the inner product of the corresponding embedded matrix and vectors, and the i, j'th entry of A can be computed by taking $e_i^\dagger A e_j$. Further if we define

$$P_1 = \begin{bmatrix} 0_n & 0_n & I_n \\ I_n & 0_n & 0_n \\ 0_n & I_n & 0_n \end{bmatrix} P_2 = \begin{bmatrix} 0_n & I_n & 0_n \\ I_n & 0_n & 0_n \\ 0_n & 0_n & I_n \end{bmatrix} P_3 = \begin{bmatrix} I_n & 0_n & 0_n \\ 0_n & 0_n & I_n \\ 0_n & I_n & 0_n \end{bmatrix}$$

Then $P_i X_i P_i^\dagger = X_{(i+1) \mod 3}(A)$, which will be used in the next sections.

8.3 Quantum Inner Product Estimation:

Here we summarize a quantum algorithm to estimate the inner product of two complex vectors. Given $x, y \in \mathbb{C}^n$, suppose that we are given a state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|x\rangle + |1\rangle|y\rangle)$ where $|x\rangle, |y\rangle$ have been prepared as in assumption 2.

Applying the Hadamard operator on the first qubit yields :
$H|\psi\rangle = \frac{1}{2}(|0\rangle(|x\rangle + |y\rangle) + (1\rangle(|x\rangle - |y\rangle))$. The probability of measuring $|0\rangle$ is given by $||\frac{1}{2}(x+y)||^2$, where $\langle(x+y)(x+y)\rangle = \langle x,x\rangle + \langle x,y\rangle + \langle y,x\rangle + \langle y,y\rangle = 2 + 2(Re(\langle x,y\rangle))$, and so it is equal to $\frac{1}{2}(1 + (Re(\langle x,y\rangle))$.

Phase Shift gates. A phase shift gate is of the form:

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

By applying a phase shift gate to $|\psi\rangle$ where $\theta = 3\pi/2$, we get the superposition $\frac{1}{\sqrt{2}}(|0\rangle|x\rangle - i|1\rangle|y\rangle)$. By applying a Hadamard transform to the first qubit we get:
$\frac{1}{\sqrt{2}}(H|0\rangle|x\rangle - iH|1\rangle|y\rangle) = \frac{1}{\sqrt{2}}((\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|x\rangle) - i(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|y\rangle)) =$
$\frac{1}{2}(|0\rangle(|x\rangle - i|y\rangle) + |1\rangle(|x\rangle + i|y\rangle))$. The probability of measuring $|0\rangle$ is now given by :
$\frac{1}{4}\langle x - iy|x - iy\rangle = \frac{1}{4}(||x||^2 - i\langle x|y\rangle + i\langle y|x\rangle + ||y||^2) = \frac{1}{2}(1 + Im\langle x|y\rangle)$.

This procedure is repeated a constant number of times to get an estimate of the real and imaginary parts of the inner product in question by tracking the ratio of measured $|0\rangle$ outcomes to the total number of trials. This can be modelled by a binomial distribution specified by probability p, with number of trials m with variance $mp(1-p)$, and variance of the estimate for p, say $\tilde{p}$ equal to $p(1-p)/m$.

The probability of obtaining $|0\rangle$ in the computational basis is given by
$p = \frac{1}{2}(1 + (Re(\langle x|y\rangle))$, hence $var(p) = var(\frac{1}{2}(1 + (Re(\langle x|y\rangle)) = 1/4 var(Re(\langle x|y\rangle)$, i.e. the variance of the estimate of the real part, denoted $\widetilde{Re(\langle x|y\rangle)}$, is 4 times $\tilde{p}$. The error of the real part of the inner product is given by the standard deviation

$\sigma_{\widetilde{Re(x|y)}} = \sqrt{var(\widetilde{Re(\langle x|y\rangle)})} = \sqrt{4var(\tilde{p})} = (4p(1-p)/m)^{\frac{1}{2}}$
$= (4(\frac{1}{2} + Re(\langle x|y\rangle)/2)(1 - (\frac{1}{2} + Re(\langle x|y\rangle)/2)))^{\frac{1}{2}} = (\frac{1}{m}(1 - Re(\langle x|y\rangle)^2)^{\frac{1}{2}}$, which decreases as m decreases. A similar analysis follows for the estimate of the imaginary part.

This procedure can be extended to compute $\langle V_1(x)|X_3(A)|V_2(y)\rangle$ for a matrix A satisfying the conditions of input assumption 1, using a modified version of the *HHL* algorithm presented in section 7, where here we compute $X_3(A)||V_2(y)\rangle$ without inverting the eigenvalues in A. Here we present a sketch of such an algorithm.

Given access to the controlled unitary $e^{i|1\rangle\langle 1|\otimes X_3(A)t}$ as in assumption 1, we can perform quantum phase estimation using $|V_2(y)\rangle$ as the input state, similar to applying the controlled Hamiltonian evolution in the HHL algorithm. This phase estimation results in the state $\sum_{k=0}^{N-1} \beta|u_k\rangle|\tilde{\lambda_k}\rangle$ where
- β_j correspond to the coefficents of $|V_2(y)$ in the eigenbasis of A
- $|u_j\rangle$ are the eigenvectors of A
- $|\tilde{\lambda_k}\rangle$ is an estimate of the eigenvalue associated with eigenvector $|u_j\rangle$ as in the phase estimation algorithm.

Next we perform a controlled rotation of an ancilla register initialized in $|0\rangle$ conditioned on the eigenvalue register, for an angle θ satisfying $sin(\theta) = c\lambda_k$ (note the different choice of θ than section 7!) for a constant c chosen such that $c\lambda_k \leq 1$ for all k. The controlled rotation acts on $|\lambda_k\rangle|0\rangle$ as
$R_\theta|\lambda_k\rangle|0\rangle = |\lambda_k\rangle(\sqrt{1 - c^2\lambda_k^2}|0\rangle + c\lambda_k|1\rangle)$.

We then uncompute the eigevnalue register and measure the ancilla qubit, where post-selcting on the outcome $|1\rangle$ leaves us with a desired output proportional to $\sum_{k=0}^{N-1} \beta_k\lambda_k|u_k\rangle = X_3(A)|V_2(y)\rangle$, where we can now estimate $\langle V_1(x)|X_3(A)|V_2(y)\rangle$ using the quantum inner product estimation algorithm.

8.4 Sub-routine 1:

Suppose you start with a set of matrices A_i, A_2, and by input assumption 1 we have access to the unitary operators $e^{iX_3(A_1)t/n}$, $e^{iX_3(A_2)t/n}$ where t is the desired simulation time, and n is some integer, and that we wish to compute an estimate of $e^{iX_3(A_1+A_2)t}$.

The procedure of sub-routine 1 is to sequentially apply $e^{iX_3(A_1)t/n}$, and $e^{iX_3(A_2)t/n}$ for a total of n times, a.k.a compute $(e^{iX_3(A_1)t/n}e^{iX_3(A_2)t/n})^n$, where the number of appli-

cations of the unitaries is proportional to $n = O(\frac{t^2}{\epsilon})$, for some error parameter ϵ. This method works due to the *Lie Product formula*, which states that for arbitrary $n \times n$ real or complex matrices A, B, that $e^{A+B} = lim_{N\to\infty}(e^{\frac{A}{N}}e^{\frac{B}{N}})^N$ (which is important if A, B do not necessarily commute).

8.5 Sub-routine 2:

Let $M = A_1, A_2$ for two complex matrices A_1, A_2 of the same dimension. In this case we want to be able to compute an estimate of $e^{iX_3(A_1A_2)t}$. Define a commutator:
$[X_1(A_1), X_2(A_2)] = X_1(A_1)X_2(A_2) - X_2(A_2)X_1(A_1) =$

$$\begin{bmatrix} 0 & 0 & M \\ 0 & 0 & 0 \\ -M^\dagger & 0 & 0 \end{bmatrix}$$

Note the matrix above is not Hermitian, but a Hermitian matrix can be constructed by multiplying the commutator by an imaginary factor, namely $i[X_1(A_1), X_2(A_2)] = X_3(iM)$.
Then $e^{iX_3(iM)} = e^{-[X_1(A_1),X_2(A_2)]} = e^{[iX_1(A_1),iX_2(A_2)]}$.
In general for an invertible matrix U, we have that
$Ue^MU^{-1} = U(I + M + \frac{1}{2!}M^2 + \frac{1}{3!}M^3 + \cdots)U^{-1}$
$= I + UMU^{-1} + \frac{1}{2!}UM^2U^{-1} + \frac{1}{3!}UM^3U^{-1} + \cdot)$
$= I + UMU^{-1} + \frac{1}{2!}(UMU^{-1})^2 + \frac{1}{3!}(UMU^{-1})^3 + \cdots$
$= e^{UMU^{-1}}$

Define $U_1 = \begin{bmatrix} \sqrt{-i}I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & \sqrt{i}I \end{bmatrix}$, then U is unitary and so
since $U_1 iX_3(iM)U_1^\dagger = iX_3(iM)$, then it follows that $U_1 iX_3(iA)U_1\dagger = iX_3(A)$, and finally $e^{iX_3(M)} = U_1 e^{iX_3(iA)}U_1^\dagger$.

All this to say, that if we can estimate the exponential of the commutator of matrices $iX_1(A_1), iX_2(A_2)$, then we can estimate the exponential of $iX_3(M)$.

Given access to $e^{H_1t/n}$ and $e^{H_2t/n}$, for matrices $H_1, H_2 \in \mathbb{C}^{N\times N}$, for small t we can approximate $e^{[H_1,H_2]t}$ with bounded error by using the Baker-Campbell-Hausdorff formula.

The **Baker–Campbell–Hausdorff formula** is the solution for Z to the equation $e^X e^Y = e^Z$ for possibly non-commutative X, Y in the Lie algebra of a Lie-group. In the case of the matrices $e^{H_i t/n}$ defined above for $i = 1, 2$, the first order formula yields the equations:
$e^{H_1 t/n} e^{H_2 t/n} = e^{(H_1+H_2)(t/n)+\frac{1}{2}[H_1,H_2](t/n)^2+O((t/n)^3)}$ and
$e^{-H_1 t/n} e^{-H_2 t/n} = e^{-(H_1+H_2)(t/n)+\frac{1}{2}[H_1,H_2](t/n)^2+O((t/n)^3)}$

Define $l(t) = (e^{H_1 t/n} e^{H_2 t/n} e^{-H_1 t/n} e^{-H_2 t/n}) = e^{[H_1,H_2](t/n)^2+O((t/n)^3)}$.
Then set $l(\tilde{t}) = l(t)l(-t) = l(H_1 t/n, H_2/t/n) =$
$e^{H_1 t/n} e^{H_2 t/n} e^{-H_1 t/n} e^{-H_2 t/n} e^{-H_1 t/n} e^{-H_2 t/n} e^{H_1 t/n} e^{H_2 t/n}$
$= e^{2[H_1,H_2]t^2/n^2+O((t/n)^4)}$, and now if $n' = int(n^2/2t)$ is an integer, then
$l(H_1 t/n, H_2 t/n)^{n^2/2t} \approx e^{[H_1,H_2]t+O(\frac{t^3}{n^2})}$. On the whole we see that $e^{[H_1,H_2]t}$ can be approximated to a constant ϵ error by using $n' = O(\frac{t^2}{\epsilon})$ copies of $e^{H_1 t/n}, e^{H_2 t/n}$. Consequently we can estimate $e^{iX_3(A_1,A_2)}$ for two matrices A_1, A_2, using the previous discussion. Summarizing we have:

Sub-routine 2 :
Input: A set of unitary operators according to input assumption 1 for A_1, A_2, a desired error parameter ϵ, simulation time t, and the unitary operator U_1 defined above.

Output: An operator U_{mult} which satisfies
$||u_{mult}(t) - e^{iX_3(A_1A_2)t}|| \leq \epsilon$ according to the procedure :

Construct $e^{\pm iX_1(A_t)t/n}, e^{\pm X_2(A_2)t/n}$ via permutation methods discussed previously, and apply them and U_1 as $U_{mult}(t) = U_1[\tilde{l}(iX_1(A_1)t/n, iX_2(A_2)t/n)]^{n'} U_1^\dagger$ where $n' = int(n^2/2t)$ and n is chosen such that $n' \in O(t^2/\epsilon)$.

8.6 Exponential of the Kronecker Sum:

The **Kronecker Sum:** of two matrices A, B of dimension $m \times n$, is defined as:
$A \oplus B = A \otimes I + I \otimes B$. Note that the matrices $A \otimes I$, and $I \otimes B$ commute, so that $e^{A\otimes I} e^{I\otimes B} = e^{A\otimes I + I\otimes B} = e^{A\oplus B}$.
Moreover it is true that $(A \otimes B)^n = A^n \otimes B^n$.
Proof: For $n = 1$ it is clear. Suppose now that for $n - 1 \geq 1$, that $(A \otimes B)^{n-1} = A^{n-1} \otimes B^{n-1}$, then $(A \otimes B)^n = (A \otimes B)^{n-1}(A \otimes B) = (A^{n-1} \otimes B^{n-1})(A \otimes B) = (A^{n-1}A) \otimes (B^{n-1}B) = A^n \otimes B^n$.

So that $e^{A\otimes I} = \sum_{n=0}^{\infty} \frac{1}{n!}(A \otimes I)^n = \sum_{n=0}^{\infty} \frac{1}{n!}(A^n \otimes I) = (\sum_{n=0}^{\infty} \frac{1}{n!} A^n) \otimes I = e^A \otimes I$, similarly

$e^{I\otimes B} = I \otimes e^B$.

Therefore $e^{A\oplus B} = e^{A\otimes I} e^{I\otimes B} = (e^A \otimes I)(I \otimes e^B)$ and the exponential embedded $A_1 \oplus A_2$ is then given by $e^{iX_3(A_1\oplus A_2)t} = e^{i(X_3(A_1\otimes I)+X_3(I\otimes A_2))t}$, and can be estimated using sub-routine 1.

8.7 Exponentiation of the Tensor Product of Matrices:

Consider the exponential of the tensor product of matrices A_1, A_2. Note the identity $A_1 \otimes A_2 = (A_1 \otimes I)(I \otimes A_2)$, where the Hamiltonian simulation of $A_1 \otimes A_2$ can be performed with the embedding $e^{iX_3(A_1\otimes A_2)}$ using sub-routine 2 and corresponding inputs $e^{\pm iX_3(A_1\otimes I)\tau}, e^{\pm iX_3(I\otimes A_2)\tau}$ and noting that $X_1(A), X_2(A)$ can be obtained using the permutation matrices defined earlier.

8.8 Exponentiation of the Hadamard Product:

The **Hadamard product** of two matrices A, B denoted as $A_1 \circ A_2$ is defined by $(A \circ B)_{ij} = (A)_{ij}(B)_{ij}$. For example, in the case of A, B being square matrices of dimension 3,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{bmatrix}.$$

Defining the non-Hermitian matrix
$S = \sum_{i=1}^{N-1} |i\rangle\langle i| \otimes |0^N\rangle\langle i| = \sum_{i=1}^{N-1} (e_i) \otimes (e_i)^t \otimes (e_1) \otimes (e_i)^t$, then $S^\dagger = \sum_i e_i \otimes e_i^t \otimes e_i \otimes e_1^t$. Since the matrix S is sparse, the authors note there exists an efficient quantum algorithm to simulate the embedded sparse matrix $X_3(S)$.

Therefore $(A_1 \otimes A_2)S^\dagger = \sum_i A_1 e_i \otimes e_i^t \otimes A_2 e_i \otimes e_1^t$ and

$$\begin{aligned}
S(A_1 \otimes A_2)S^\dagger &= \left(\sum_j e_j \otimes e_j^t \otimes e_1 \otimes e_j^t\right)\left(\sum_i A_1 e_i \otimes e_i^t \otimes A_2 e_i \otimes e_1^t\right) \\
&= \sum_{i,j} \left(e_j \otimes e_j^t \otimes e_1 \otimes e_j^t\right)\left(A_1 e_i \otimes e_i^t \otimes A_2 e_i \otimes e_1^t\right) \\
&= \sum_{i,j} \left((e_j \otimes e_j^t)(A_1 e_i \otimes e_i^t) \otimes (e_1 \otimes e_j^t)(A_2 e_i \otimes e_1^t)\right) \\
&= \sum_{i,j} \left((e_j^t A_1 e_i)(e_j \otimes e_i^t) \otimes (e_j^t A_2 e_i)(e_1 \otimes e_1^t)\right) \\
&= \sum_{i,j} (e_j^t A_1 e_i)(e_j^t A_2 e_i)\left((e_j \otimes e_i^t) \otimes (e_1 \otimes e_1^t)\right) \\
&= \left(\sum_{i,j} (e_j^t (A_1 \circ A_2) e_i)(e_j \otimes e_i^t)\right) \otimes (e_1 \otimes e_1^t) \\
&= (A_1 \circ A_2) \otimes (e_1 \otimes e_1^t)
\end{aligned}$$

which in turn is equal to $(A_1 \circ A_2) \otimes |0^n\rangle\langle 0^n|$, in Dirac notation, (observe that $e_j \otimes e_j^t = e_j^t \otimes e_j$). For any matrix B, it is true that

$e^{iB \otimes e_1 \otimes e_1^t} = I_n \otimes I_n + \sum_{k=1}^{\infty} \frac{1}{k!}(iB)^k \otimes (e_1 \otimes e_1^t)^k$

$= I_n \otimes I_n + (\sum_{k=1}^{\infty} \frac{1}{k!}(iB)^k) \otimes (e_1 \otimes e_1^t)$

$= I_n \otimes (I_n - e_1 \otimes e_1^t) + I_n \otimes e_1 \otimes e_1^t + (\sum_{k=1}^{\infty} \frac{1}{k!}(iB)^k) \otimes (e_1 \otimes e_1^t)$

$= I_n \otimes (I_n - e_1 \otimes e_1^t) + (\sum_{k=0}^{\infty} \frac{1}{k!}(iB)^k) \otimes (e_1 \otimes e_1^t) = I_n \otimes (I_n - e_1 \otimes e_1^t) + e^{iB} \otimes (e_1 \otimes e_1^t)$

Therefore $e^{iX_3(S(A_1 \otimes A_2)S^\dagger)t} = e^{iX_3(A_1 \circ A_2) \otimes |0^n\rangle\langle 0^n|t}$

$= e^{iX_3(A_1 \circ A_2)t} \otimes |0^n\rangle\langle 0^n| + I \otimes (I - |0^n\rangle\langle 0^n|)$, and so we can estimate $e^{iX_3(A_1 \circ A_2)t}$ by using $e^{iX_3(S(A_1 \otimes A_2)S^\dagger)t}$ with an ancilla register in the state $|0^n\rangle$ and combining methods for matrix tensor product and multiplication sub-routine 2, with assumed access to unitary operators $e^{\pm iX_3(S)\tau}, e^{\pm iX_3(S^\dagger)\tau}, e^{\pm iX_3(A_1)\tau}, e^{\pm iX_3(A_2)\tau}$.

9 Appendix:

9.1 Elementary Complexity Theory, Turing Machines, and the class BQP:

This section contains elementary concepts from complexity theory selected from *Computational Complexity: A Modern Approach* written by authors Sanjeev Arora and Boaz Barak [12], as well as known elementary concepts from this domain.

9.1.1 Concepts:

Boolean functions are functions from $f : \{0,1\}^n \mapsto \{0,1\}$.
The set L_f $\{x : f(x) = 1\}$ is said to be the *language* for f. The problem of determining the language of f is referred to as a *decision problem.*
A **formal language** L over an alphabet $\sum$ is a subset of $\sum^*$, that is, a set of words over that alphabet. Sometimes the sets of words are grouped into expressions, whereas rules and constraints may be formulated for the creation of 'well-formed expressions'.

9.1.2 Formal Definition of a Turing Machine:

Formally, a Turing machine M is described by a tuple (Γ, Q, δ), where
-Γ represents the alphabet which contains a start symbol $|\rangle$, a blank symbol and a finite number of other symbols.
-Q is the set of possible states that M's register can be in, with the property that Q has a designated start state and finish state.
-δ is a function that given the current state and the alphabet symbols read, instructs the tape head to move left, right, or stay and sets a new state in the register.

Turing machines have a set of tapes (or in some definitions a single tape), which extend infinitely to the right and are divided into cells. In the most general defintion there is an input tape, a set of work tapes, and an output tape. All tapes except for the input are initialized in their first location to the start symbol, and all other locations to the blank symbol. The input tape contains the start symbol, a finite number of non-blank symbols, and then an infinite amount of blank symbols. The halt state has the property that once it is transitioned to, no more state changes or new data is written to any of the read-write tapes, and the output of the work-tape is then considered the output of an algorithm executed on the Turing machine.

Running Time Let $f : \{0,1\}^* \to \{0,1\}^*$, and $T : \mathbb{N} \to \mathbb{N}$, then a turing machine M computes f in $T(n)$ time if for every start configuration on input x, after at most $T(|x|)$ steps it halts with $f(x)$ written on the output tape. We say that M computes f it it computes f in $T(n)$ times for some function $T : \mathbb{N} \to \mathbb{N}$.

Time Constructible: A function T is time constructible if $T(n) \geq n$ and there eixsts a Turing machine M that computes the function $x \mapsto binary(T(|x|))$ for $x \in \{0,1\}^*$, i.e. computes the binary representation of $T(|x|)$ where $|x|$ is the length in number of bits of the binary representation of x.

The following are true statements concerning Turing machines
-For every $f : \{0,1\}^* \to \{0,1\}$ and time-constructible $T : \mathbb{N} \to \mathbb{N}$, if f is computable in time $T(n)$ by a TM M using alphabet Γ, then it is computable in $4log|\Gamma|T(n)$ by a TM M^* using the alphabet $\{0, 1, blank, start\}$. The general idea is to encode the alphabet characters in binary, and for every cell of $M's$ tape, we have a corresponding $log_2\Gamma$ cells of M^*'s tape.

-For every $f : \{0,1\}^* \to \{0,1\}$ time constructible T, if f is computable in time $T(n)$ by a TM M using k tapes, then it is computable in time $5kT(n)^2$ by a TM M^* using only a single work tape. The idea being to encode the first tape at locations $1, k+1, 2k+1$, second tape at $2, k+2, 2k+2$..., and so on, and have the tape head sweep back and forth.

-A bi-directional TM is a TM whose tapes extend infinitely to the right and to the left. For every $f : \{0,1\}^* \to \{0,1\}^*$, time constructible T, if f is computable in time $T(n)$ by a bidirectional TM M , then it is computable in time $4T(n)$ by a unidirectional TM M^*. The idea is to treat M^*'s alphabet as consisting of pairs of $M's$ alphabet, and to treat each tape cell in M^* as consisting of two symbols from M by folding $M's$ tape and displaying the top and the lower half directly above each other as in Figure 12:

The importance of Turing machines is that they formalize the notion of an algorithm being *effectively computable*. This relates to David Hilbert's *Entscheidungsproblem*(German for decision problem) which asks for the existence of an algorithm that takes as input a statement of a first-order logic (where predicates cannot take inputs of other predicates), and answers "Yes" or "No" according to whether the statement is universally valid, i.e deduced from a set of axioms (including possible a finite number of additional axioms beyond usual first-order logic) using rules of logic. Alan Turing's notion of *effectively calculable* is captured by the functions computable by a Turing machine, and answers conclusively *no*.

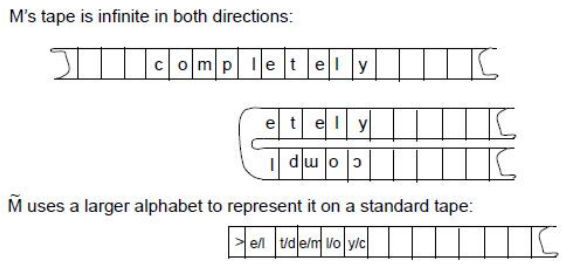


Figure 12: Converting a Bi-directional tape to a single tape [12]

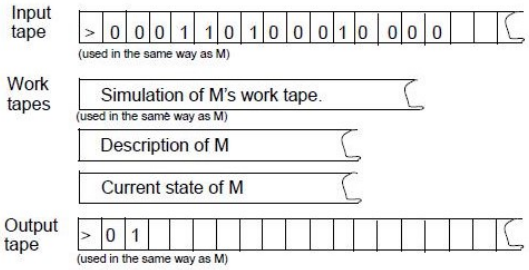


Figure 13: An outline of the Universal Turing Machine [12]

Turing machines can be encoded as a binary string (in fact infinitely many), and every binary string represents a Turing machine.

Universal Turing-Machine: There exists a TM U such that for every $x, \alpha \in \{0,1\}^*$, $U(x, \alpha) = M_\alpha(x)$ where M_α denotes the TM represented by α. Moreover, if M_α halts on input x, within T steps then $U(x, \alpha)$ halts within $CTlogT$ steps where C depends only on properties of M_α. See figure 13.

9.1.3 The Halting Problem and an Uncomputable Function:

There exists a function $UC : \{0,1\}^* \to \{0,1\}$ that is not computable by any TM.

Proof. The function UC is defined as follows, for every $\alpha \in \{0,1\}$, let M be the TM represented by α. If on input α , where yes, we are taking M on its own encoding, M halts within a finite number of steps and outputs 1, then $UC(\alpha) = 0$, otherwise $UC(\alpha) = 1$. Now assume to the contrary that there exists a TM M such that $M(\alpha) = UC(\alpha)$ for every $\alpha \in \{0,1\}^*$ (bit strings of arbitrary length). Then if β is the binary encoding of M, then $M(\beta) = UC(\beta)$. But then if $UC(\beta) = 1$, then M does not halt or does not output 1, and if $UC(\beta) = 0$, then M halts and outputs 1. In either case

we have a contradiction, and hence no Turing machine can compute UC $\square$

Define the function HALT that takes as input a pair α, x, and outputs 1 iff the TM M_α represented by α halts on input x within a finite number of steps, then it can be shown that HALT is not computable by any TM. In general the **halting problem** is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running (i.e., halt) or continue to run forever.

9.1.4 Basic Complexity Classes and the Class BQP:

P: A language L is in **P** iff there exists a deterministic Turing machine M such that M runs in polynomial time in the size of the input, and
-for all $x \in L$, M outputs 1,
-for all $x \notin L$, M outputs 0.

NP: A language L is in **NP** if and only if there exist polynomials p, q, and a deterministic Turing machine M, such that:
-for all x, y, the machine M runs in time $p(|x|)$ on input (x, y)
-for all $x \in L$, there exists a string y of length $q(|x|)$ such that $M(x, y) = 1$.
-for all $x \notin L$ and all strings y of length $q(|x|)$, $M(x, y) = 0$.

A **probabilistic Turing machine** is a non-deterministic Turing machine which chooses between the available transitions at each point according to some probability distribution.

A language L is in **BPP** if and only if there exists a probabilistic Turing machine M, such that M runs for polynomial time on all inputs and
For all $x \in L$, M outputs 1 with probability greater than or equal to $2/3$
For all $x \notin L$, M outputs 1 with probability less than or equal to $1/3$

A **quantum Turing machine** is a variation on a traditional Turing machine where the set of states is replaced by a Hilbert space Q, and the transition function is understood to be a collection of unitary matrices that are automorphisms of the Hilbert Space Q.

BQP: A language L is in **BQP** if and only if there exists a probabilistic Turing machine M, such that M runs for polynomial time on all inputs and :
For all x in L, M outputs 1 with probability greater than or equal to $2/3$.
For all $x \notin L$, M outputs 1 with probability less than or equal to $1/3$.

PSPACE: A language L is in the class **PSPACE** if it can be solved by a deterministic Turing machine using $O(t(n))$ space for some function t of the input size n.

A **reduction** is an algorithm for transforming one problem into another problem. A sufficiently efficient reduction from one problem to another may be used to show that the second problem is at least as difficult as the first.

A decision problem A is said to be **hard** if for all other decision problems B in a given complexity class, there exists a reduction from B to A. A decision problem is said to be **complete** for a complexity class if it is hard, and it is an element of the class.

9.2 Discussion of Relevant Linear Algebra material:

In this section a collection of linear algebra material is presented that seemed noteworthy for study. The material is taken from Gilbert Strang's (MIT) *Introduction to Linear Algebra* [13], and Carl Meyer's *Matrix Analysis and Applied Linear Algebra* [14].

9.2.1 Some Properties of the Exponential of a Matrix:

If X is a square real or complex matrix, then the exponential of a matrix is defined as $e^X = \sum_{k=0}^{\infty} \frac{1}{k!} X^k$ where $X^0 = I$.

Facts about exponentials of a matrix A
a) $e^0 = I$
b) $e^{(A^\dagger)} = (e^A)^\dagger$
c) If Y is an invertible matrix, then $e^{YXY^{-1}} = Ye^XY^{-1}$
d) If X, Y are matrices that commute, then $e^Xe^Y = e^{X+Y}$

Let A be an Hermitian matrix, then e^{iA} is unitary.

Proof. Suppose that A is a Hermitian matrix, then define $U = e^{iA}$, and note that $U^\dagger = (e^{iA})^\dagger = (e^{-iA^\dagger})$, and further it is true that if A, B commute, then $e^Ae^B = e^{A+B}$. Hence $UU^* = e^{iA}e^{-iA^\dagger} = e^{iA}e^{-iA} = e^0 = I$, and therefore U is unitary since $iA = iA^\dagger$, and so iA and $-iA^\dagger$ commute and sum to the 0 matrix. □

The **Lie product formula/Lie-trotter product formula** is a result named for Sophus Lie which states that for arbitrary $n \times n$ real or complex matrices A and B,

that $e^{A+B} = lim_{n\to\infty}(e^{A/N}e^{B/N})^N$.

The **Baker-Campbell-Hausdorff** is the solution for Z to the equation $e^X e^Y = e^Z$ for possibly non-commutative X, Y in the Lie algebra of a Lie-group. In the case of the matrices A, B, the first few terms of the series are :
$e^A e^B = A + B + \frac{1}{2}[X,Y] + \frac{1}{12}[X,[X,Y]] + \dots$ where $[X,Y]$ is the commutator of two matrices.

9.2.2 Singular Value Decomposition:

Any complex m by n matrix A can be factored into
$A = U\Sigma V^\dagger$ where the columns of U (m by m) are eigenvectors of $AA^\dagger$, and the columns of V (n by n) are eigenvectors of $A^\dagger A$. The $r = rank(A^\dagger A)$ singular values on the diagonal of Σ (m by n) are the square roots of the non-zero eigenvalues of both $AA^\dagger$ and $A^\dagger A$. This is referred to as the Singular-Value decomposition of A. In the special case where A is a real $m \times m$ square matrix with positive determinant, then $U, V^\dagger$ and Σ are real $m \times m$ matrices as well. It is then the case $U, V^\dagger$ are orthogonal, and hence are rotation matrices, while Σ can be viewed as a non-uniform scaling matrix, a matrix that scales a vector $p = (p_1, \cdots, p_m)$ by a scaling vector $(v_1, \cdots, v_m)$ by multiplying it by the matrix Σ consisting of diagonal entries $[\Sigma]_{ii} = v_i$, and 0 elsewhere.

Recall that the spectral theorem for a Hermitian matrix A on a vector space V states that there exists an orthonormal basis of V consisting of eigenvectors of A, and that each eigenvalue is real.
Note that for any complex M of dimension $m \times n$, it is true that $MM^\dagger$ is Hermitian since $(MM^\dagger)^\dagger = ((M^\dagger)^\dagger M^\dagger) = (MM^\dagger)$. Further $MM^\dagger$ is positive semi-definite since for any vector $x \in \mathbb{C}^m$, $x^\dagger MM^\dagger x = (x^\dagger M)(M^\dagger x)$ which is the complex dot product of a vector with itself, and so is non-negative.

The diagonalization theorem says that an $n \times n$ matrix A is diagonalizable iff it has n linearly dependent eigenvectors $\{v_1, \cdots, v_n\}$. Suppose that A has n linearly independent eigenvectors $\{v_1, \cdots, v_n\}$, define C as the matrix consisting of columns vectors which are the eigenvectors of A. Then C is invertible, and let $D = C^{-1}AC$. Now $De_i = C^{-1}ACe_i = C^{-1}Av_i = C^{-1}\lambda_i v_i = \lambda_i e_i$ where $Ce_i = v_i$ implies that $e_i = C^{-1}v_i$. Conversely suppose that $A = CDC^{-1}$ where C has columns $\{v_1, \cdots, v_n\}$ and D is diagonal with diagonal entries $\lambda_1, \cdots, \lambda_n$. Since C is invertible, its columns are linearly independent. Now we must show that v_i is an eigenvector of A with eigenvalue λ_i. But $Av_i = CDC^{-1}v_i = CDe_i = C\lambda_i e_i = \lambda_i Ce_i = \lambda_i v_i$, as desired.

9.2.3 Existence:

Consider a complex matrix A of dimension $m \times n$. Since $AA^\dagger$ is Hermitian, it has m linearly independent eigenvectors, and so by the spectral theorem and the diagonalization theorem, there exists an eigendecomposition:
$A^\dagger A = VDV^\dagger = \sum_{i=1}^{n} \lambda_i v_i v_i^\dagger = \sum_{i=1}^{n} (\sigma_i)^2 v_i v_i^\dagger$, where V is a unitary matrix whose columns are the orthonormal eigenvectors of $AA^\dagger$, and the square roots of the σ_i^2 are referred to as the singular values of A. We can write the eigenvalues of $AA^\dagger$ as squares (the σ_i^2) since the eigenvalues of $AA^\dagger$ are non-negative. In general, if λ is an eigenvalue of a positive semi-definite complex matrix R of dimension n, then for any n-vector v, we see that $v^\dagger Rv = \lambda v^\dagger v \geq 0$, and since $v^\dagger v \geq 0$, and it follows that $\lambda \geq 0$.

Let $r = rank(A) = rank(A^\dagger A)$. To see that $rank(A) = rank(A^\dagger A)$ it is sufficient to show that $Ax = 0 \iff A^\dagger Ax = 0$, since if the nullspace of two matrices are equal, they have the same column space. If $Ax = 0$ then $A^\dagger Ax = A^\dagger(Ax) = A^\dagger 0 = 0$. On the other hand, if $A^\dagger Ax = 0$, then $x^\dagger A^\dagger Ax = 0 \rightarrow (Ax)^\dagger (Ax) = 0$, implies that $Ax = 0$ since for any inner-product $x^\dagger x = 0 \iff x = 0$. Further the rank of any square matrix equals the number of nonzero eigenvalues (with multiplicity), so that the number of nonzero singular values of A equals the rank of $A^\dagger A$, i.e. after a re-ordering of the indices, $\sigma_1^2, \cdots \sigma_r^2 > 0$.

Define for $1 \leq i \leq r$, $u_i := \dfrac{Av_i}{\sigma_i}$. Note that $AA^\dagger v_i = (\sigma_i)^2 v_i$, and that $AA^\dagger u_i = AA^\dagger(\frac{Av_i}{\sigma_i}) = AA^\dagger Av_i \frac{1}{\sigma_i} = (\sigma)^2 u_i$, so that the u_i are eigenvectors $AA^\dagger$. Moreover the u_i are orthonormal since $u_i^\dagger u_j = (\dfrac{Av_i}{\sigma_i})^\dagger \dfrac{Av_j}{\sigma_j} = (\dfrac{v_i^\dagger A^\dagger}{\sigma_i})\dfrac{Av_j}{\sigma_j} = \dfrac{v_i^\dagger (\sigma_i)^2 v_j}{\sigma_i^2} = \begin{cases} 1 & \text{, if } i = j \\ 0 & \text{, if } i \neq j \end{cases}$
(the v_i are orthonormal).

Let V be an $n \times n$ matrix, where the i'th column is v_i. Since $\{u_1, \cdots u_r\}$ are r orthonormal vectors in the column space of A, if $r < m$, we can complete the set with orthonormal vectors $u_{r+1}, \cdots, u_m$ to form a basis of $\mathbb{C}^m$. Let U be an $m \times m$ matrix whose columns are the u_i so defined. Let Σ be a diagonal matrix whose i'th element is σ_i, augmented by $n - r$ columns of zeros and $m - r$ rows of zeros. Then $U = AV\Sigma^{-1}$ and so $U\Sigma = AV$, and then $A = U\Sigma V^\dagger$, where we use the fact that $VV^\dagger = I$, which is the desired singular value decomposition of A.

9.2.4 Matrix Norms:

Let K be either the field of real or complex numbers. For matrices $A, B \in L(K^m, K^n)$ a matrix norm $||\cdot||$ is a function $L(K^m, K^n) \to \mathbb{R}$ such that for all scalars $\alpha \in K$
1. $||\alpha A|| = |\alpha|\,||A||$
2. $||A + B|| \leq ||A|| + ||B||$
3. $||A|| \geq 0$
4. $||A|| = 0 \iff A = 0_{m,n}$
Additionally some norms satisfy the property of *sub-multiplicativity* where
$||AB|| \leq ||A|| \cdot ||B||$

There are different types of matrix norms, and we will consider some of the most common types.

9.2.5 Element-wise norms:

For a real or complex matrix A of dimension $m \times n$ the p-norm of A is
$||A||_p = (\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^p)^{1/p}$. Notable special cases are :
$||A||_1$, the absolute sum of all elements of A
$||A||_\infty$ is the maximum norm, the maximum absolute value among all the entries of A
$||A||_2$ is the frobenius norm given by $||A||_2 = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$. Equivalently, consider $[AA^\dagger]_{ij} = \sum_{k=1}^{n} a_{ik}\bar{a}_{jk}$, then $[AA^\dagger]_{ii} = \sum_{k=1}^{n} a_{ik}\bar{a}_{ik} = \sum_{k=1}^{n} |a_{ik}|^2$, and therefore $||A||_2 = \sqrt{trace(AA^\dagger)}$.
Moreover since the trace of a matrix is equal to the sum of its eigenvalues, then $||A||_2 = \sqrt{trace(AA^\dagger)} = \sqrt{\sum_{i=1}^{R} \lambda_i} = \sqrt{\sum_{i=1}^{R} \sigma_i^2}$, where $R \leq min(M, N)$ is the number of non-zero eigenvalues of $AA^\dagger$.

9.2.6 Induced or Operator Norms:

For a norm induced by a vector, $||A||$ is based on any vector norm $||x||$ where
$||A|| = sup_{||x||=1}||Ax|| = sup_{x \neq 0}||Ax||/||x||$. The following properties hold:
-$||A|| > 0$ if $A \neq 0$, since if the ij'th entry of A is non-zero, then $||Ae_j|| > 0$.
-$||\alpha A|| = |\alpha|||A||$, since $||\alpha A|| = sup_{||x||=1}\{||\alpha Ax||\} = |a|sup_{||x||=1}\{||Ax||\} = |a|||A||$
-$||A + B|| \leq ||A|| + ||B||$ (using the definition of this matrix norm and elementary properties of the supremum)
-$||Ax|| \leq ||A||||x||$. If $x = 0$, this is trivial, and if not $||A||||x|| = ||x||sup_{y \neq 0}\{||Ay||/||y||\} \geq ||x||||Ay||/||y||$, since y is arbitrary let $y = x$, then cancelling gives $||A||\,||x|| \geq ||Ax||$
-$||AB|| \leq ||A||\,||B||$, since $||AB|| = sup_{||x||=1}\{||ABx||\} \leq sup_{||x||=1}\{||A||\,||Bx||\}$
$= ||A||sup_{||x||=1}\{||Bx||\} = ||A||\,||B||$

In particular, if the p-norm for vectors is used for both spaces K^n, K^m, then the corresponding induced operator norm is given by $||A||_p = sup_{x\neq 0}\frac{||Ax||_p}{||x||_p}$. The special cases are

$p = 1$, which can be shown to be equivalent to the maximum absolute column sum,

$p = 2$, the largest singular value of the matrix:

Proof. $sup_{x\neq 0}\dfrac{||Ax||_2}{||x||_2} = \dfrac{||U\Sigma V'||_2}{||x||_2}$, as in the SVD of A.
Now noting that unitary matrices preserve length, then it is true that $||U\Sigma V'x||_2 = ||\Sigma x||_2$, therefore
$sup_{\{x\neq 0\}}\dfrac{||U\Sigma V'x||_2}{||x||_2} = sup_{\{x\neq 0\}}\dfrac{||\Sigma x||_2}{||x||_2} = sup_{\{x\neq 0\}}\dfrac{(\sum_{i=1}^r \sigma_i^2|x_i|^2)^{1/2}}{(\sum_{i=1}^r |x_i|^2)^{1/2}} \leq \sigma_{max}(A)$, where for $y = \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}^T$, $||\Sigma y||_2 = \sigma_1 = \sigma_{max}(A)$, is attained (after a possible re-ordering of the singular values). □

$p = \infty$, the maximum absolute row sum of the matrix.

9.2.7 The Schatten norm:

The Schatten p-norm for a real or complex matrix A of dimension $m \times n$, and $r < min(m, n)$ singular values σ_i, is defined by $||A||_p = (\sum_{i=1}^r \sigma_i^p)^{1/p}$. All schatten norms are sub-multiplicative, and unitarily invariant, meaning that $||A|| = ||UAV||$ for all matrices A, and unitary matrices U, V.

9.2.8 The Condition Number of a Matrix:

Suppose we are given a matrix equation $Ax = b$ for a complex invertible matrix A of dimension $m \times m$, and vectors x, b of dimension $m \times 1$. Beginning with a change in the right-hand side from b to $b + \delta b$, and supposing that δb is small, we know obtain an *error equation* $A(x + \delta x) = b + \delta b$, and so by subtraction $A(\delta x) = \delta b$. An error δb leads to $\delta x = A^{-1}\delta b$. The change in x is especially large when δb points in the direction that is amplified most by A^{-1}.
Suppose that A is symmetric and its eigenvalues are positive and given by $0 < \lambda_1 \leq \cdots \leq \lambda_n$. Any vector δb is a combination of the corresponding unit eigenvectors $x_1, \cdots, x_n$. The worst error δx coming from A^{-1}, is in the direction of the first eigenvector, i.e. the *worst error*:
if $\delta b = \epsilon x_1$, then $\delta x \approx \frac{\delta b}{\lambda_1}$, or in other words the error $||\delta b||$ is amplified by $\frac{1}{\lambda_1}$, the largest eigenvalue of A^{-1}, this amplification is greatest when λ_i is near zero, and A is said to be *nearly singular*. The solution $x = A^{-1}b$ and the error $\delta x = A^{-1}\delta b$ always satisfy

1)$||x|| \geq \frac{||b||}{\lambda_{max}}$
2)$||\delta x|| \leq \frac{||\delta b||}{\lambda_{min}}$
For 1), $||x|| = ||A^{-1}b||$, and decomposing $b = \alpha_1 x_1 + \cdots + \alpha_n x_n$,
$A^{-1}b = A^{-1}(\alpha_1 x_1 + \cdots + \alpha_n x_n) = (\frac{1}{\lambda_1}\alpha_1 x_1 + \cdots + \frac{1}{\lambda_n}\alpha_n x_n)$, and so
$||x|| = ||\frac{1}{\lambda_1}\alpha_1 x_1 + \cdots + \frac{1}{\lambda_n}\alpha_n x_n|| \geq ||\frac{1}{\lambda_{max}}\alpha_1 x_1 + \cdots + \frac{1}{\lambda_{max}}\alpha_n x_n|| = \frac{||b||}{\lambda_{max}}$. Equation 2 follows similarly.

Further it follows that $\frac{||\delta x||}{||x||} \leq \frac{\lambda_{max}}{\lambda_{min}}\frac{||\delta b||}{||b||}$. The *relative change* is given by $\frac{||\delta b||}{||b||}$, the *relative error* is given by $\frac{||\delta x||}{||x||}$, and the *condition number* is given by the ratio $c(A) = \frac{\lambda_{max}}{\lambda_{min}}$. The worst case in terms of error is when $||\delta x||$ is large, with δb in the direction of the eigenvector x_1, and additionally $||x||$ is small. The solution x should be as small as possible compared to b, which implies that the original problem $Ax = b$ is at the other extreme. If $b = x_n$, then $x = A^{-1}b = b/\lambda_n$. It is this combination $b = x_n$, and $\delta b = \epsilon x_1$, that makes the relative error as large as possible.

9.3 Approximation by Rational Numbers:

The following material is obtained from the course notes by Bruce Ikenaga (University of Missouri), for the course *Introduction to Number Theory* [15]. A continued fraction is an expression of the form (*)

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ddots}}}$$

for $a_i \in \mathbb{R}$, and almost always $a_i \in \mathbb{Z}_+$.

9.3.1 Finite Continued Fractions:

Continued fractions where the numerators are all 1 (as in the above example) are called *simple continued fractions*. The notation for a finite simple continued fraction is given by $[a_0; a_1, \ldots, a_n]$, which is always equal to $[a_0; a_1, \ldots, a_n - 1, 1]$. Every finite continued fraction represents a rational number. The k'th convergent of a finite continued fraction is defined as $c_k = [a_0; a_1, \cdots, a_k]$ for $0 \leq k \leq n$.

Lemma: Let $a_0, a_1, \ldots a_n$ be real numbers, and $[a_0; a_1, \ldots a_n]$ the associated finite simple continued fraction. Let

$p_0 = a_0$, $q_0 = 1$,
$p_1 = a_1a_0 + 1, q_1 = a_1$
$p_k = a_kp_{k-1} + p_{k-2}, q_k = a_kq_{k-1} + q_{k-2}$ for $k \geq 2$. Then $c_k = \frac{p_k}{q_k}$.

Proof. By induction on k:
$c_0 = \frac{p_0}{q_0} = \frac{a_0}{1} = a_0$, and $c_1 = \frac{p_1}{q_1} = \frac{a_1a_0+1}{a_1} = a_0 + \frac{1}{a_1}$. Suppose $k \geq 2$, then assume the result holds for the k'th convergent, and note that
$c_{k+1} = [a_0; a_1, \cdots, a_k, a_{k+1}] = [a_0; a_1, \cdots, a_k + \frac{1}{a_k+1}]$ (where we are allowing the last entry to be a rational number).
Then $c_{k+1} = [a_0; a_1, \cdots, a_k, a_{k+1}] = [a_0; a_1, \cdots, a_k + \dfrac{1}{a_k + 1}]$, where by induction it is true that $[a_0; a_1, \cdots, a_k + \frac{1}{a_k+1}] = \frac{p_k}{q_k} = \frac{a_kp_{k-1}+p_{k-2}}{a_kq_{k-1}+q_{k-2}}$. Further, $p_{k-1}, p_{k-2}, q_{k-1}, q_{k-2}$ are the same for both $[a_0; a_1, \cdots, a_{k+1}]$ and $[a_0; a_1, \cdots, a_k + \frac{1}{a_k+1}]$, therefore $c_{k+1} = \dfrac{(a_k + \frac{1}{a_k+1})p_{k-1} + p_{k-2}}{(a_k + \frac{1}{a_k+1})q_{k-1} + q_{k-2}} = \dfrac{a_{k+1}(a_kp_{k-1} + p_{k-2}) + p_{k-1}}{a_{k+1}(a_kq_{k-1} + q_{k-2}) + q_{k-2}} = \dfrac{a_{k+1}p_k + p_{k-1}}{a_{k+1}q_k + q_{k-1}} = \dfrac{p_{k+1}}{q_{k+1}}$. □

9.3.2 Infinite Continued Fractions:

An infinite continued fraction is a fraction of the form (*) and is denoted by $[a_0; a_1, \ldots]$ where the $a_i \geq 0$ for every i.

Lemma: $p_kq_{k-1} - p_{k-1}q_k = (-1)^{k-1}$

Proof. By induction on k, if $k = 1$, then $p_1q_0 - p_0q_1 = (a_1a_0 + 1)(1) - (a_0)(a_1) = 1 = 1^{1-1}$. Assume the result holds for $k \geq 1$ then
$p_{k+1}q_k - p_kq_{k+1} = (a_{k+1}p_k + p_{k-1})q_k - p_k(a_{k+1}q_k + q_{k-1}) = p_{k-1}q_k - p_kq_{k-1}$
$= -(p_kq_{k-1} - p_{k-1}q_k) = -(-1)^{k-1} = (-1)^k$ □

Facts about convergents:
a) $c_k - c_{k-1} = \frac{(-1)^{k-1}}{q_{k-1}q_k}$ since $c_k - c_{k-1} = \frac{p_k}{q_k} - \frac{p_{k-1}}{q_{k-1}} = \frac{p_kq_{k-1}-p_{k-1}q_k}{q_{k-1}q_k} = \frac{(-1)^{k-1}}{q_{k-1}q_k}$
b) $c_k - c_{k-2} = \frac{a_k(-1)^k}{q_{k-2}q_k}$

Lemma: The odd convergents form a strictly decreasing sequence, while the even convergents form a strictly increasing sequence. Any odd convergent is larger than any even convergent. The odd and even convergents converge to the same value.

Proof. if k is even, then $c_k - c_{k-2} = \frac{a_k(-1)^k}{q_{k-2}q_k} > 0$ since the a_i are all larger than 0, therefore the even terms get larger.
If k is odd, then $c_k - c_{k-2} = \frac{a_k(-1)^k}{q_{k-2}q_k} < 0$, so that the odd terms get smaller. Now note that $c_{2n+1} - c_{2n} = \frac{-1(^{(2n+1)-1}}{q_{k-1}q_k} > 0 \implies c_{2n+1} > c_{2n}$, and for any odd term c_{2n+1} and

any even term c_m, we have that $c_{2n+1} > c_{2n+2m+1} > c_{2n+2m} > c_{2m}$. Therefore the odd terms are bounded below by c_0 and are decreasing, and so converge, whereas the even terms are bounded above by c_1 and are increasing, and so converge. To show that they converge to the same value is the next task □

.

Note that $q_k \geq k$ for all $k \geq 1$.

Proof. By induction on k,
$q_1 = a_1 \geq 1$. For $k \geq 2$, $q_{k+1} = a_{k+1}q_k + q_{k-1} \geq a_{k+1}k + (k-1) \geq k + (k-1) = 2k - 1 \geq k + 1$.
Further note that the difference between consecutive terms converges to 0 since
$c_k - c_{k-1} = \frac{(-1)^{k-1}}{q_{k-1}q_k} \leq \frac{1}{(k-2)(k)} \to 0$ as $k \to \infty$, therefore the odd convergents converge to the same value as the even convergents □

.

Lemma: Let $x = [a_0; a_1, a_2, \ldots]$ be an infinite continued fraction where $a_i \geq 1$ for each i. Then x is an irrational number.

Proof. Suppose to the contrary that $x = \frac{p}{q}$, where p, q are integers. Odd convergents converge by decreasing to x, and even convergents converge by increasing to x. For each k, we have $c_{2k+1} > x > c_{2k}$, and so it follows that :
$c_{2k+1} - c_{2k} > x - c_{2k} > 0$, and by fact a) we have that

$$c_{2k+1} - c_{2k} = \frac{(-1)^{2k}}{q_{2k}q_{2k+1}}$$
$$\implies \frac{1}{q_{2k}q_{2k+1}} > x - c_{2k} > 0$$
$$\implies \frac{1}{q_{2k}q_{2k+1}} > x - \frac{p_{2k}}{q_{2k}} > 0$$
$$\implies \frac{1}{q_{2k+1}} > xq_{2k} - p_{2k} > 0$$
$$\implies \frac{1}{q_{2k+1}} > \frac{pq_{2k}}{q} - p_{2k} > 0, \text{ and finally}$$
$$\frac{q}{q_{2k+1}} > pq_{2k} - p_{2k}q > 0.$$

Since $q_{2k+1} \geq 2k + 1$ can be made arbitrarily large, the middle term is a positive integer smaller than a fraction less than 1, which is impossible, and therefore x must be irrational □

.

Lemma: Let $a_0, a_1, \ldots$ be a sequence of integers, where $a_k > 0$ for all $k \geq 1$. Define
$p_0 = a_0$, $q_0 = 1$,
$p_1 = a_1a_0 + 1$, $q_1 = a_1$
$p_k = a_kp_{k-1} + p_{k-2}, q_k = a_kq_{k-1} + q_{k-2}$ for $k \geq 2$. Then $q_{k+1} > q_k$ for $k > 0$.

Proof. Note that since $q_0 > 0$ and $q_1 > 0$, then $q_2 > 0$ and so inductively q_k is larger than 0 for any $k > 0$. Since by assumption $a_k > 0$ for any $k > 0$, then $q_{k+1} = a_{k+1}q_k + q_{k-1} > a_{k+1}q_k \geq 1q_k = q_k$ □

.

Theorem (Continued Fraction Algorithm): Let $x \in \mathbb{R}$ be irrational. Let $x_0 = x$. If $a_k = [x_k]$ (the integer part), and $x_{k+1} = \frac{1}{x_k - a_k}$, then $x = [a_0; a_1, a_2, \ldots]$.

Step 1: x_k is irrational for $k \geq 0$. Since x is irrational and $x_0 = x$, the result is true for $k = 0$. Assume that $k > 0$ and that the result is true for $k - 1$. Suppose that $x_k = \frac{s}{t}$, then $\frac{s}{t} = \frac{1}{x_{k-1} - a_{k-1}}$ so $x_{k-1} = a_{k-1} + \frac{t}{s}$. The right hand side is the sum of two rational numbers, which is a contradiction to the induction hypothesis that x_{k-1} is rational. Hence x_k is irrational.

Step 2: The a_k's are positive integers for $k \geq 1$. Note that the a_k's are integers, and by definition : $a_k \leq x_k \leq a_k + 1$ and since the x_k are irrational it follows that they can't have a finite decimal expansion and so a strict inequality follows of the form $a_k < x_k < a_k + 1$, and so $x_{k+1} = \frac{1}{x_k - a_k} > 1$, and $a_{k+1} = [x_{k+1}] \geq 1$

Step 3: $x = [a_0; a_1, a_2, \ldots, x_k]$. For $k = 0$, the claim is that $x = x_0$ which is true by definition. Assume the result holds for $k \geq 0$. Then $x_{k+1} = \dfrac{1}{x_k - a_k} \implies x_k - a_k = \dfrac{1}{x_{k+1}} \implies x_k = a_k + \dfrac{1}{x_{k+1}}$. Then $x = [a_0; a_1, \ldots x_k] = [a_0; a_1, \ldots, a_k + \frac{1}{x_{k+1}}] = [a_0; a_1, \ldots, a_k, x_{k+1}]$, and so this proves the overall result for $k + 1$, and therefore the result is true by induction.

Step 4: $lim_{n\to\infty}[a_0, \ldots, a_n] = x$. Consider the continued fractions $[a_0; a_1, \ldots, a_k, x_{k+1}]$ and $[a_0; a_1, a_2, \ldots, a_k, a_{k+1}, \ldots]$. Using the preceding step and the recursive formula for convergents, it is true that $x = x_0 = [a_0; a_1, a_2, \ldots, a_k, x_{k+1}] = \dfrac{x_{k+1}p_k + p_{k-1}}{x_{k+1}q_k + q_{k-1}}$, therefore

$$x - \frac{p_k}{q_k} = \frac{x_{k+1}p_k + p_{k-1}}{x_{k+1}q_k + q_{k-1}} - \frac{p_k}{q_k} = \frac{x_{k+1}p_kq_k + p_{k-1}q_k - x_{k+1}p_kq_k - p_kq_{k-1}}{(x_{k+1}q_k + q_{k-1})q_k} = \frac{p_{k-1}q_k - p_kq_{k-1}}{(x_{k+1}q_k + q_{k-1})q_k} = \frac{(-1)^k}{(x_{k+1}q_k + q_{k-1})q_k}.$$

Now taking absolute values gives $|x - \dfrac{p_k}{q_k}| = \dfrac{1}{(x_{k+1}q_k + q_{k-1})q_k}$. Next $x_{k+1} > [x_{k+1}] = a_{k+1}$ so $x_{k+1}q_k + q_{k-1} > a_{k+1}q_k + q_{k-1} = q_{k+1}$. Therefore $\dfrac{1}{x_{k+1}q_k + q_{k-1}} < \dfrac{1}{q_{k+1}} \implies \dfrac{1}{(x_{k+1}q_k + q_{k-1})q_k} < \dfrac{1}{q_{k+1}(q_k)}$. But we know that $q_k \geq k$ for all $k \geq 1$, so $|x - \dfrac{p_k}{q_k}| \leq \dfrac{1}{q_{k+1}q_k} \leq \dfrac{1}{k(k+1)}$.

Hence by the squeeze theorem $lim_{k\to\infty}|x-\frac{p_k}{q_k}|=0$, which implies that $lim_{k\to\infty}\frac{p_k}{q_k}=x$.

Theorem: Let x be irrational, and let $c_k=\frac{p_k}{q_k}$ be the k-th convergent in the continued fraction expansion of x. Suppose $p,q\in\mathbb{Z}, q>0$, and $|qx-p|<|q_kx-p_k|$, then $q\geq q_{k+1}$.

Consider the equations $p_ku+p_{k+1}v=p$ and $q_ku+q_{k+1}v=q$. Recall that the inverse of a 2 by 2 matrix $A=\begin{bmatrix}a & b\\ c & d\end{bmatrix}$ is equal to $det(A)^{-1}\begin{bmatrix}d & -b\\ -c & a\end{bmatrix}$.

We wish to solve the equation $\begin{bmatrix}p_k & p_{k+1}\\ q_k & q_{k+1}\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=\begin{bmatrix}p\\ q\end{bmatrix}$.

In this case $det(A)=p_kq_{k+1}-p_{k+1}q_k=\pm1$, and the solution can be achieved by left multiplying the right side of the equation by $\pm\begin{bmatrix}q_{k+1} & -p_{k+1}\\ -q_k & p_k\end{bmatrix}$ resulting in the vector $\begin{bmatrix}u\\ v\end{bmatrix}=\pm\begin{bmatrix}q_{k+1}p-p_{k+1}q\\ -q_kp+p_kq\end{bmatrix}$. This shows that u,v are integers.

Suppose by contradiction that $q<q_{k+1}$.

Step 1: $u,v\neq0$. Suppose that $u=0$, Then $p_ku+p_{k+1}v=p$ implies that $p_{k+1}v=p$, so $p_{k+1}q_{k+1}v=pq_{k+1}$, and similarly $q_ku+q_{k+1}v=q$ gives $q_{k+1}v=q$, and so $p_{k+1}q_{k+1}v=qp_{k+1}$.
Hence $pq_{k+1}=qp_{k+1}$, and q_{k+1} divides qp_{k+1} and since $gcd(p_{k+1},q_{k+1})=1$ (the denominator and numerator of convergents are relatively prime), then $q_{k+1}|q$, contrary to assumption that $q<q_{k+1}$, therefore $u\neq0$.
Suppose that $v=0$, then $p_ku+p_{k+1}v=p$, yields $p_ku=p$ and $q_ku=q$, hence $|qx-p|<|q_kx-p_k|$
$\implies |q_kux-p_ku|<|q_kx-p_k|$
$\implies |u||q_kx-p_k|<|q_kx-p_k|$
$\implies |u|<1$, but u is an integer, so $u=0$, a contradiction, therefore $v\neq0$.

Step 2: u,v have opposite signs
Since $v\neq0$, so that either $v>0, v<0$. Suppose $v>0$, so that $v\geq1$, since v is an integer. Then $q_{k+1}v\geq q_{k+1}>q$, and since $q_ku+q_{k+1}v=q$, then $q_ku=q-q_{k+1}v<0$, but $q_k>0$ for all $k>0$, so that u,v have opposite signs.
If $v<0$, then $q_{k+1}v<0$, and so $-q_{k+1}v>0$, therefore $q_ku=q-q_{k+1}v>0$, so that $u>0$, and u,v have opposite signs.

Step 3: $q_k x - p_k$ and $q_{k+1}x - p_{k+1}$ have opposite signs. As discussed earlier, the limit of the convergents lies above all even convergents, and below all odd convergents. So that for any $k \in \mathbb{N}$ it is true that $\frac{p_k}{q_k} < x < \frac{p_{k+1}}{q_{k+1}}$ or $\frac{p_{k+1}}{q_{k+1}} < x < \frac{p_k}{q_k}$. In the first case, the inequality gives $p_k < q_k x$ and $q_k x - p_k > 0$, while the right hand side of the inequality gives $q_{k+1}x - p_{k+1} < 0$, hence the conclusion holds. Similarly if $\frac{p_{k+1}}{q_{k+1}} < x < \frac{p_k}{q_k}$ the conclusion holds.

Step 4: Observe $u(q_k x - p_k)$ and $v(q_{k+1}x - p_{k+1})$ have the same sign.

Step 5: Final Contradiction. Since $u(q_k x - p_k)$ and $v(q_{k+1}x - p_{k+1})$ have the same sign then consider: $|qx - p| = |u(q_k x - p_k) + v(q_{k+1}x - p_{k+1})| = |u(q_k x - p_k)| + |v(q_{k+1}x - p_{k+1})|$, then $|qx - p| = |u(q_k x - p_k) + v(q_{k+1}x - p_{k+1})|$
$= |u(q_k x - p_k)| + |v(q_{k+1}x - p_{k+1})|$
$= |u||q_k x - p_k| + |v||q_{k+1}x - p_{k+1}| \geq |u||q_k x - p_k| \geq |q_k x - p_k|$, which contradicts the original hypothesis, therefore $q \geq q_{k+1}$.

Corollary. Let x be irrational, and let $c_k = \frac{p_k}{q_k}$ be the k'th convergent in the continued fraction of representation of x. Suppose that $p, q \in \mathbb{Z}, q > 0$, and $|x - \frac{p}{q}| < |x - \frac{p_k}{q_k}|$, then $q > q_k$.

Proof. By contradiction, suppose that $q \leq q_k$, then multiplying this inequality by q gives $|qx - p| < |q_k x - p_k|$, the previous theorem yields that $q \geq q_{k+1}$, and so $q_k \geq q \geq q_{k+1}$, but this contradicts the fact the q_k's increase □

.

Theorem: Let x be irrational, and let $\frac{p}{q}$ be a rational number in lowest terms with $q > 0$. Suppose that $|x - \frac{p}{q}| < \frac{1}{2q^2}$, then $\frac{p}{q}$ is a convergent in the continued fraction expansion for x.

Proof. Since as proved earlier, $q_k \geq k$ for all $k \geq 0$, there must exist some k such that $q_k \leq q < q_{k+1}$, and by the contrapositve of a theorem proven earlier, $|q_k x - p_k| \leq |qx - p| = q|x - \frac{p}{q}| < q\frac{1}{2q^2} = \frac{1}{2q}$, and so $|x - \frac{p_k}{q_k}| < \frac{1}{2qq_k}$.

Suppose by contradiction that $\frac{p}{q}$ is not a convergent in the continued fraction expansion for x. In other words, for any k, $\frac{p}{q} \neq \frac{p_k}{q_k}$, so $qp_k \neq pq_k$, we see that $|qp_k - pq_k|$ is a positive integer. Since $|qp_k - pq_k| \geq 1$, it is true that
$\frac{1}{qq_k} \leq \frac{|qp_k - pq_k|}{qq_k} = |\frac{p_k}{q_k} - \frac{p}{q}| = |\frac{p_k}{q_k} - x + x - \frac{p}{q}| \leq |\frac{p_k}{q_k} - x| + |x - \frac{p}{q}| \leq \frac{1}{2qq_k} + \frac{1}{2q^2}$, and so $q < q_k$, which is a contradiction to the assumption that $q_k \leq q$, and so the desired conclusion holds □

.

10 References:

[1] John Watrous, *CPSC 519/619 Quantum Computation notes*. University of Calgary, January 10, 2006.

[2] Michael A. Nielsen, Isaac L. Chuang. *Quantum Computation and Quantum Information*, Cambridge University Press, New York, 10th Edition, 2010.

[3] Dieudonné, Jean, *Foundations of Modern Analysis*, Academic Press.

[4] Brian Hall, *Lie Groups, Lie Algebras, and Representations*, Springer International Publishing, 2015.

[5] A. W. Harrow, A. Hassidim, and S. Lloyd. *Quantum algorithm for linear systems of equations*. Physical review letters 103.15 (2009), v2.

[6] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, and L. Wossnig, *Quantum linear systems algorithms: a primer*, arXiv preprint arXiv:1802.08227, 2018.

[7] L. Zhao, Z. Zhao, P. Rebentrost, J. Fitzsimons. *Compiling basic linear algebra subroutines for quantum computers*, arXiv:1902.10394, 2019.

[8] Berry, D. W., Ahokas, G., Cleve, R., Sanders, B.C. *Efficient quantum algorithms for simulating sparse hamiltonians*. Communications in Mathematical Physics, 270(2), 359-371, 2007.

[9] Suzuki, M., Hatano, N. *Finding exponential product formulas of higher order*, Quantum Annealing and Other Optimization Methods, Eds. A. Das and B.K. Chakrabarti (Springer, Berlin, 2005) pp. 37-68.

[10] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. *Quantum Amplitude Amplification and Estimation*, volume 305 of Contemporary Mathematics Series Millenium Volume. AMS, New York, 2002. arXiv:quant-ph/0005055

[11] A. Prakash, *Quantum algorithms for linear algebra and machine learning*, PhD Thesis, University of California, Berkeley, 2014.

[12] S. Arora, B. Boaz. *Computational Complexity: A modern approach (draft edition)*. Princeton University, 2007.

[13] Strang, Gilbert. *Introduction to Linear Algebra*. Fourth Wellesley, MA: Wellesley-Cambridge Press, 2009.

[14] Carl D. Meyer, *Matrix Analysis and Applied Linear Algebra*, chapter 5.2, p.281, Society for Industrial and Applied Mathematics, June 2000.

[15] Ikenaga B. *Introduction to Number Theory: Course Notes*, accessed August 2019. http://sites.millersville.edu/bikenaga/number-theory/approximation-by-rationals/approximation-by-rationals.html .