Quantum Computing

Nihar Shah

July 30, 2024

Quantum Computing: Theory and Practice

Nihar Shah

Guide: Professor Phani Motammari

Department of Computational and Data Science
Indian Institute of Science

Abstract

The following content consists of the work done by me as a part of my Master's Thesis at Indian Institute of Science, Bangalore. The work is done under the guidance of Prof. Phani Motammari. This work are my notes made during the 1 year work from June 2024 to August 2025 and are based on the lectures, papers, books and other resources that I have referred to during the course of my work. To a reader who is not at all familiar with Quantum Computing, this work will serve as a good starting point. I suggest him to start reading from the Appendinx to get a good grasph on the maths used in the entitre text. The work is divided into 3 parts: The first part is the introduction to Quantum Computing, the second part is the Advanced Quantum Computing and the third part is the work done by me as a part of my thesis and the Appendix. Along with the references, I have also included the code snippets that I have written during the course of my work. The references are included at the end of the document. I hope that this text helps the reader to get a good grasp on Quantum Computing. Enjoy reading!

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Phani Motammari, for the continuous support of my study and related, for his patience, and immense knowledge. His guidance helped me through the time of research and writing of this thesis.

Besides my advisor I am also grateful to the faculty members and staff at the Department of Computational and Data Science of Indian Institute of Science for their assistance and support. Special thanks to MATRIX lab for providing a stimulating and collaborative research environment.

Last but not the least, I would like to thank my family: my parents, for supporting me spiritually throughout writing this thesis and my life in general.

Contents

List of Figures

List of Tables

Notations

$\mathbb{R}$ The set of all real numbers

$\mathbb{R}^{m\times n}$ The set of all $m \times n$ real matrices

$\mathbb{Z}$ The set of all integers

$\mathbb{N}$ The set of all natural numbers

$\mathbb{Q}$ The set of all rational numbers

$\mathbb{C}$ The set of all complex numbers

$\mathbb{C}^{m\times n}$ The set of all $m \times n$ complex matrices

$\mathbb{F}$ Field

$\mathbb{F}^n$ The set of all n-dimensional vectors over the field $\mathbb{F}$

$|v\rangle$ A vector in a vector space

Chapter 1

Introduction

1.1 Axioms of Quantum Mechanics

1.1.1 Postulate 1: The State Space Postulate

The state of an isolated quantum mechanical system is completely specified by a state vector $|\psi\rangle$ which is a unit vector that belongs to a complex vector space with inner product (a Hilbert Space) $\mathbb{C}^n$, where n is the dimension of the state space of the system.

Consider some physical system that can be in N different, mutually exclusive classical states (it means a state in which the system can be found if we observe it) $|0\rangle, |1\rangle, \ldots, |N-1\rangle$. Then, a *pure quantum state* (or *state*) $|\psi\rangle$ is a superposition of classical states, written as:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \ldots + \alpha_{N-1} |N-1\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$$

where α_i are complex numbers called amplitude of $|i\rangle$ in $|\psi\rangle$. A system in quantum state $|\psi\rangle$ is in a superposition of the classical states $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ with amplitudes $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}$ respectively. Mathematically, the states $|0\rangle$, $|1\rangle$, ..., $|N-1\rangle$ forms an N-dimensional Hilbert space (i.e. N dimensional complex vector space with inner product defined). We write the quantum state $|\psi\rangle$ as a N-dimensional column

vector of its amplitudes in the Hilbert Space.

$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{pmatrix}$$

Consider a Hilbert space $\mathcal{H}_\mathcal{A}$ with orthonormal basis $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ and another Hilbert space $\mathcal{H}_\mathcal{B}$ with an orthonormal basis $|0'\rangle, |1'\rangle, \ldots, |M-1\rangle$. Then we can combine the two Hilbert spaces using tensor products to form a new Hilbert space $\mathcal{H} = \mathcal{H}_\mathcal{A} \otimes \mathcal{H}_\mathcal{B}$ which is a $N \times M$ dimensional Hilbert space with basis $|i\rangle \otimes |j\rangle$ where $|i\rangle$ is a basis vector of $\mathcal{H}_\mathcal{A}$ ($|i\rangle \in \{0, \ldots, N-1\}$) and $|j\rangle$ is a basis vector of $\mathcal{H}_\mathcal{B}$ ($|j\rangle \in \{0', \ldots, M-1\}$). The basis vectors of $\mathcal{H}$ are $|i\rangle \otimes |j\rangle$ where $i \in \{0, \ldots, N-1\}$ and $j \in \{0', \ldots, M-1\}$. Thus, an arbitrary state of $\mathcal{H}$ can be written as:

$$|\psi\rangle = \sum_{i=0}^{N-1} \sum_{j=0'}^{M-1} \alpha_{ij} |i\rangle \otimes |j\rangle$$

where α_{ij} are complex numbers called amplitudes of $|i\rangle \otimes |j\rangle$ in $|\psi\rangle$. Sucg as state is called *bipartite*. Similarly, this can be extended to tripartite states which is a tensor product of three Hilbert spaces and so on.

The simplest system is qubit - a quantum mechanical system with two-dimensional state space. Suppose $|0\rangle$ and $|1\rangle$ are two orthonormal basis vectors also generally called as **computational basis** of the state space of the qubit. Then the arbitrary state of the qubit can be represented as

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$ (since its a unit vector). The condition $|\alpha|^2 + |\beta|^2 = 1$ is called the *normalization condition*. The complex numbers α and β are called probability amplitudes for the states $|0\rangle$ and $|1\rangle$ respectively.

Intuitively, one can think of the state $|0\rangle$ and $|1\rangle$ as the analogous to the classical bit states 0 and 1. The qubit differs from the classical bit in the sense that the qubit can exist in a superposition of the states $|0\rangle$ and $|1\rangle$, i.e. of the form $\alpha |0\rangle + \beta |1\rangle$ where α and β are complex numbers, which is neither in state $|0\rangle$ or in $|1\rangle$ (or in both the sates simultaneously) which does not happen for a classical bit. This is the essence of quantum superposition which can be taken advantage of in quantum computing.

Example 1.1.1. A qubit can exist in a state such as

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

which is a superposition of the states $|0\rangle$ and $|1\rangle$ with amplitude $1/\sqrt{2}$ for being in state $|0\rangle$ and amplitude $-1/\sqrt{2}$ for being in the state $|1\rangle$.

Example 1.1.2. Say a state of a qubit is $|\psi\rangle = 7\,|0\rangle + (3 + 5\iota)\,|1\rangle$. In the matrix form, it can be written as:

$$|\psi\rangle = \begin{pmatrix} 7 \\ 3 + 5\iota \end{pmatrix}$$

In general we say that any linear combination $\sum_i \alpha_i\,|\psi_i\rangle$ is a superposition of the states $|\psi_i\rangle$ with amplitudes α_i for the state $|\psi_i\rangle$.

There are two things we can do with a quantum state: Measure it or let it evolve (without measuring it).

1.1.2 Postulate 2: Evolution Postulate

The time evolution of a closed Quantum System is described by the Schrodinger equation as:

$$i\hbar \frac{d}{dt}\,|\psi(t)\rangle = H\,|\psi(t)\rangle$$

where $|\psi(t)\rangle$ is the state of the system at time t, H is the Hamiltonian operator of the system and $\hbar$ is the reduced Planck's constant. The Hamiltonian operator is an operator that corresponds to the total energy of the system. The Schrodinger equation is a linear differential equation and is the quantum analog of Newton's second law of motion. The solution to the Schrodinger equation gives the state of the system at any time t given the initial state of the system. If we know the Hamiltonian of a system, we can predict the future state of the system. For the purpose of Quantum computing, we are interested in the time evolution of a quantum system. Thus, we will always assume that the Hamiltonian of a quantum system is known. Thus, simplifying the Schrodinger equation, we can write it as:

$$\frac{d}{dt}\,|\psi(t)\rangle = -\frac{i}{\hbar} H\,|\psi(t)\rangle$$

Rearranging the above equation, we get:

$$\int_{\psi_{t_1}}^{\psi_{t_2}} \frac{d\,|\psi\rangle}{|\psi\rangle} = -\int_{t_1}^{t_2} \frac{i}{\hbar} H dt$$

Integrating, the above equation we get,

$$\ln\left(\frac{|\psi_{t_2}\rangle}{|\psi_{t_1}\rangle}\right) = -\frac{i}{\hbar}H(t_2 - t_1)$$

$$|\psi_{t_2}\rangle = e^{-\frac{i}{\hbar}H(t_2-t_1)}\,|\psi_{t_1}\rangle$$

Now we know that the Hamiltonian operator H is Heritian and as proved in the appending we know that e^A is unitary for any Heritian operator A. Thus, the time evolution operator $U(t_2, t_1) = e^{-\frac{i}{\hbar}H(t_2-t_1)}$ is unitary. Hence, we can write the time evolution of a quantum state as:

$$|\psi(t_2)\rangle = U(t_1, t_2)\,|\psi(t_1)\rangle$$

where U is a Unitary time evolution Operator. Thus, we have described a discrete time desciption of the dynamics using unitary operators. For the purpose of Quantum Computing, we will deal with discrete time evolution of the quantum system described by the exponenetial of the Hamiltonian (Hermitian operator) which is a unitary operator as shown and not the continuous time evolution of the quantum system described by the hamitlonian. Thus, we restate the postulate for Quantum Computing as: **The evolution of a closed quantum system is described by a Unitary transformation. That is, the two states at times t_1 and time t_2 are related by a unitary operator $U(t_2, t_1)$ such that $|\psi(t_2)\rangle = U(t_2, t_1)\,|\psi(t_1)\rangle$.**

Note that in quantum computing we oftern apply a unitary operator to a closed quantum state. This is contradictory statement. The act of applying a unitary operator is in itself an external iteraction with the quantum system, thus it no longer remains closed. It turns out, that it is possible to write down a time-varying Hamiltonian for a Quantum system, in which the hamiltonian for the system is not constant, but varies according to some external control parameters which are under experimentalist's control, and could be changed during the course of an experiment. The system is therefore not closed, but it does evolve according to Schrdinger's equation with a time varying Hamiltonian, to some good approximation. Thus we will describe the evolutions of a quantum system even those system's which aren't closed using unitary operators.

1.1.3 Postulate 3: Measurement Postulate

Measurement in the computational basis

Consider a Quantum System in a state $|\psi\rangle = \sum_{j=0}^{N-1} \alpha_j\,|j\rangle$ which is some unknown

superposition of the classical states $|j\rangle$. We cannot know exactly what the superposition of the states $|\psi\rangle$ is since we can see only the classical states. Thus if we measure the state $|\psi\rangle$ it will collapse to one of the classical states (say $|j\rangle$). To which classical state will it collapse to (which will we see)? This is not known in advance, we can only predict the probabiltiy of the state collapsing to a particular classical state. The probability of the state $|\psi\rangle$ collapsing to the classical state $|j\rangle$ is given by **Born's Rule** as:

$$P(j) = |\langle j|\psi\rangle|^2 = |\alpha_j|^2$$

where α_j is the amplitude of the state $|j\rangle$ in the state $|\psi\rangle$. Thus, the probability of the state $|\psi\rangle$ collapsing to the classical state $|j\rangle$ is the square of the amplitude of the state $|j\rangle$ in the state $|\psi\rangle$. Now, since its a probability distribution this implies that the sum of the probabilities of the state $|\psi\rangle$ collapsing to any of the classical states $|j\rangle$ is 1. Thus, we have:

$$\sum_j P(j) = \sum_j |\alpha_j|^2 = 1$$

thus, α_j are called probability amplitudes. (**Note that since operations on a quantum state are unitary in nature and a Unitary operator as shown in the appendix preserves norm, thus once the amplitudes are normalized to satisfy the Born's rule we can be sure that the probabilities will remain normalized no matter the number of times we perform any unitary operation on the quantum state**). Once we measure the state $|\psi\rangle$ and it collapses to the classical state $|j\rangle$, the state of the system is now $|j\rangle$ and all the information that might have been contained in the amplitudes of α_i is gone.

Note that the probabilities of various measurement outcomes are exactly the same when we measure $|\psi\rangle$ or when we measure the state $e^{\iota\theta}|\psi\rangle$. The term $e^{\iota\theta}$ is called the **global phase** and thus has no physical significance.

Note that when the exprimentalist and their experimental apparatus observes the system,it is an external physical system in other words, and the interaction makes the quantum system no longer closed and thus not necessarily subject to the unitary evolution. Thus, more formally we can state the postulate as:
Quantum measurements are descibed by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space $|\psi\rangle$ being meaured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is given by:

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle$$

and the state of the system after the measurement is:

$$\frac{M_m |\psi\rangle}{\sqrt{p(m)}}$$

Now since the sum of the probabilities of all the measurement outcomes is 1, we have:

$$1 = \sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle$$

Thus, the measurement operators satisfy the completeness equation:

$$\sum_m M_m^\dagger M_m = I$$

Example 1.1.3. Measurement of a qubit in the computational basis. Measurement of a qubit with two outcomes defined by the two measruremnt operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$ (note that both the measurement operators are Hermitian and $M_0^2 = M_0, M_1^2 = M_1$, thus follow Completeness relation). Suppose the qubit is in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. The probability of the qubit collapsing to the state $|0\rangle$ is:

$$p(0) = |\langle 0|\psi\rangle|^2 = |\alpha|^2$$

and the probability of the qubit collapsing to the state $|1\rangle$ is:

$$p(1) = |\langle 1|\psi\rangle|^2 = |\beta|^2$$

Thus, the state of the qubit after the measurement is:

$$\frac{M_0 |\psi\rangle}{\sqrt{p(0)}} = \frac{\alpha|0\rangle}{|\alpha|}$$

$$\frac{M_1 |\psi\rangle}{\sqrt{p(1)}} = \frac{\beta|1\rangle}{|\beta|}$$

Important Note

In general, there are three important basis that we take about doing measurements in Quantum Computing:

1. **Computational Basis/Standard Basis:** The basis in which the states are represented as $|0\rangle, |1\rangle, \ldots, |N-1\rangle$.
2. **Hadamard Basis:** The basis in which the states are represented as $|+\rangle, |-\rangle$ where $|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$.
3. **Y Basis:** The basis in which the states are represented as $|+i\rangle, |-i\rangle$ where $|+i\rangle = \frac{|0\rangle + \iota|1\rangle}{\sqrt{2}}$ and $|-i\rangle = \frac{|0\rangle - \iota|1\rangle}{\sqrt{2}}$.

Example 1.1.4. Consider a state $|\psi\rangle = \left(\frac{1}{\sqrt{6}} - \frac{1}{\sqrt{3}}\right)|+\rangle + \left(\frac{1}{\sqrt{6}} + \frac{1}{\sqrt{3}}\right)|-\rangle$ (it can be given in any basis, say here it is given in hadamard basis). Find the probabilities of measurement in Hadamard basis and Standard basis.
Probability of measuring the state $|\psi\rangle$ in Hadamard basis to be $|+\rangle$ is:

$$p(+) = |\langle +|\psi\rangle|^2 = |\left(\frac{1}{\sqrt{6}} - \frac{1}{\sqrt{3}}\right)|^2 = \frac{1}{6} + \frac{1}{3} - \frac{2}{\sqrt{18}} = \frac{3 - 2\sqrt{2}}{6}$$

Probability of measuring the state $|\psi\rangle$ in Hadamard basis to be $|-\rangle$ is:

$$p(-) = |\langle -|\psi\rangle|^2 = |\left(\frac{1}{\sqrt{6}} + \frac{1}{\sqrt{3}}\right)|^2 = \frac{1}{6} + \frac{1}{3} + \frac{2}{\sqrt{18}} = \frac{3 + 2\sqrt{2}}{6}$$

Probability of measuring the state $|\psi\rangle$ in Standard basis to be $|0\rangle$ is:

$$p(0) = |\langle 0|\psi\rangle|^2 = |\frac{1}{\sqrt{2}}\left(\frac{2}{\sqrt{6}}\right)|^2 = |\frac{1}{\sqrt{3}}|^2 = \frac{1}{3}$$

Probability of measuring the state $|\psi\rangle$ in Standard basis to be $|1\rangle$ is:

$$p(1) = |\langle 1|\psi\rangle|^2 = |\frac{1}{\sqrt{2}}\left(\frac{2}{\sqrt{3}}\right)|^2 = |\frac{\sqrt{2}}{\sqrt{3}}|^2 = \frac{1}{3}$$

Important Note

Distinguishing Non-Orthogonal States

Theorem 1.1.1. *Non-orthogonal states can't be reliably distinguished.*

Consider two people Alice and Bob. Alice prepares a quantum state $|\psi_i\rangle$ $(1 \leq i \leq n)$ from some fixed set of states known to both parties and sends it to Bob . Bob has to determine which state Alice sent him.
Suppose the states $|\psi_i\rangle$ are orthonormal. Then Bob can measure the state $|\psi_i\rangle$ using the following procedure. Define Measurement operators, $M_i = |\psi_i\rangle \langle\psi_i|$, one for each possible index i, and an additional measurement operator $M_0 = I - \sum_{i=1}^{n} |\psi_i\rangle \langle\psi_i|$. The measurement operators satisfy the completeness relation $\sum_{i=0}^{n} M_i^\dagger M_i = I$. Thus, for a state $|\psi_i\rangle$ then $p(i) = \langle\psi_i| M_i |\psi_i\rangle = 1$, so the result i occurs with certianinty. Thus, it is possible to reliably distinguish the orthonormal states $|\psi_i\rangle$.
Suppose the states $|\psi_i\rangle$ are not orthonormal. then we can prove that there is no quantum measurement capable of ditinguishing the quantum states. The idea is that Bob will do a measurement described by measurement operators M_j with outcome j. Depending on the outcome of the measurment Bob tries to guess what the index i was using some rule, $i = f(j)$ (where $f(\cdot)$ is the rule to make the guess). Bob can't distinguish between the states $|\psi_1\rangle$ and $|\psi_2\rangle$ is because $|\psi_2\rangle$ can be decomposed intoa (non-zero) component parallel to $|\psi_1\rangle$, and a component orthogonal to $|\psi_1\rangle$.

Projective Measurements

A special case of the general measurement postulate. Consider a Quantum System with quantum state $|\psi\rangle$. A projective measurement on some space, with m possible outcomes, is a collection of projectors. $P_1, \ldots, P_m$ that all act on tha same space and sum to identity $\sum_{j=1}^{m} P_j = I$. These projectors are *pairwise orthogonal*, $P_i P_j = \delta_{ij} P_i$ where δ_{ij} is the Kronecker delta. The Projector P_j projects onto some subspace V_j of the total Hilbert Space V. Now every state $|\psi\rangle \in V$ can be decomposed in a unique way as $|\psi\rangle = \sum_{j=1}^{m} P_j |\psi\rangle$. Note that since the projectors are orthogonal, the subspaces V_j are orthogonal and the projection of the states onto V_j. When we apply this measurement onto the pure state $|\psi\rangle$, then we will get outcome j with probability $p(j) = \langle\psi| P_j |\psi\rangle$. If the outcome j occurs, then the state of the systema immediately after the measurement is $\dfrac{P_j |\psi\rangle}{\sqrt{p(j)}}$. Note that the probabilities sum to 1.

We cannot chose which P_j will be applied to the state but can only give a probability distribution. However, if the state $|\psi\rangle$ fully lies within one of the subspaces V_j then the measurement of the outcome will be j with certaining Note that the m projectors together form one measurement, we don't use the word "measurement" for individual P_j.

Example 1.1.5. A measurement in the computational basis in a N-dimensional state space is a specific projective measurement with m=N and $P_j = |j\rangle\langle j|$. P_j projects onto the computation basis state $|j\rangle$ and the corresponding subspace $V_j \subset V$ is the 1-dimensional subspace spanned by $|j\rangle$. Consider the state $|\psi\rangle = \sum_{j=0}^{N-1} \alpha_j |j\rangle$. $P_j |\psi\rangle = \alpha_j |j\rangle$, so applying our measurement will give outcome j with probability $|\alpha_j|^2$ and the state of the system after the measurement will be $|j\rangle$.

Example 1.1.6. Note that we can choose any orthonormal basis for the projective measurement. For example, consider a quantum state $|\psi\rangle$ expressed as superposition of standard orthonormal basis states $|0\rangle, |1\rangle, \ldots, |N-1\rangle$. We can choose any other orthonormal basis B of states say $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{N-1}\rangle$ and consider the projective measurement defined by the projectors $P_j = |\phi_j\rangle\langle\phi_j|$. This is called measureing in B basis. Applying this measurement to the state $|\psi\rangle$ will give outcome j with probability $|\langle\phi_j|\psi\rangle|^2$ and the state of the system after the measurement will be $|\phi_j\rangle$. Note that if $|\psi\rangle$ equals to one the basis vectors $|\phi_j\rangle$ then the measurement gives outcome j with probability 1.

Example 1.1.7. Note that it is not at all necessary for projectors to be of rank 1. For example, we can consider only two projectors on a N (even) dimensional space such that $P_1 = \sum_{j<N/2} |j\rangle\langle j|$ and $P_2 = \sum_{j\geq N/2} |j\rangle\langle j|$. say the state of a quantum system is $|\psi\rangle = \frac{1}{\sqrt{3}}|1\rangle + \sqrt{\frac{2}{3}}|N\rangle$. Then, it gives outcome 1 with probability $p(1) = \langle\psi| P_1 |\psi\rangle = \frac{1}{3}$ and outcome 2 with probability $p(2) = \langle\psi| P_2 |\psi\rangle = \frac{2}{3}$.

Observables: We can write projective measurements with operators $P_1, \ldots, P_m$ and associated distinct outcomes $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{R}$, can be written as one Matrix $M = \sum_{i=1}^{m} \lambda_i P_i$ which is called an observable. It has an advantage that the expected (average) value of the outcome can be easily calculated as $\langle\psi| M |\psi\rangle$. NOte that M is Hermitian. Every observable is a Hermitian.

Definition 1.1.1. Projective Measurement: A projective measurement is a measurement described by an observable M, a Hermitian operator on the state space of the system being observed. Since, its a Hermitian operator hence has a spectral decomposition as:

$$M = \sum_m m P_m$$

where P_m is the projection onto the eigen space of M with eigen value m. **The possible outcomes of the measurement correspond to the eigenvalues, m, of the observable**. Upon, measuring the state $|\psi\rangle$, the probability of getting result m is given by:

$$p(m) = \langle\psi| P_m |\psi\rangle$$

Given that the outcome m occured, the state of the quantum system immediately after the measurement is:

$$\frac{P_m |\psi\rangle}{\sqrt{p(m)}}$$

Projective measurement as a special case of Postulate 3: Suppose measurement operators in Postulate 3 in addition to satisfying the completeness relation $\sum_m M_m^\dagger M_m = I$ also satisfy the relation that M_m are orthogonal projectors, that is $M_m^\dagger = M_m$ hence are Hermitian. Then, the Postulate 3 reduces to a Projective measurement as defined. Calculation of Average values for Projective measurements. By defintion, the average value of the measurement is:

$$\begin{aligned}
\mathbf{E}(M) &= \sum_m m p(m) \\
&= \sum_m m \langle\psi| P_m |\psi\rangle \\
&= \langle\psi| \left(\sum_m m P_m \right) |\psi\rangle \\
&= \langle\psi| M |\psi\rangle \\
\langle M\rangle &= \langle\psi| M |\psi\rangle
\end{aligned}$$

Postive Operator Valued Measure (POVM) measurements

For some applications, the post-measurement of the state is of little interest. We only care about the final probability distribution on the m-outcomes, then we can use the most general type of measrurement called *positive-operator-valued-measrue (POVM)*. For example, when the system is measured only once, at the end. In such cases, the POVM measurement is more general than the projective measurement.

We specify m positive semi-definite (Psd) matrices $E_1, \ldots, E_m$ such that $\sum_m E_m = I$. When measuring a state $|\psi\rangle$, the probability of the outcome i is given by $Tr(E_i |\psi\rangle \langle\psi|) = \langle\psi| E_i |\psi\rangle$. A projective measurement is a special case of POVm where the measurement elements E_i are projectors. (That is $E_i^2 = E_i$ and $E_i^\dagger = E_i$).

The POVM elements E_i are not necessarily orthogonal. *One can show that every POVM can be simulated by a projective measusrement on a slightly larger space that yields the exact probability distribution over measurement outcomes (Neumark's theoerem)*

Example 1.1.8. Suppose in a 2-dimensional state space, we know that it is either in state $|0\rangle$ or $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Clearly, this two states are not orthogonal, so there is no measurement that distinguishes them perfectly. However, there is a POVm measurement that never makes a mistake, but sometimes gives another outcome 2, meaning "I don't know". That is, you would like to do a meaurement with three possible outcomes: 0,1 and 2, such that:

- If the state is $|0\rangle$, then you get a correct outcome 0 with proabability 1/4 and outcome 2 with probability 3/4 but never get inncorrect outcome 1.
- If the state is $|+\rangle$, then you get correct outcome 1 with probability 1/4 and outcome 2 with probabiltiy 3/4 but never get the inncorrect outcome 0.

This cannot be achieved with projective measurement on the qubit, but can be achieved with follwoing 3-outcome POVM:

- $E_0 = \frac{1}{2}|-\rangle\langle -|$, (where $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ which is orthogonal to the $|+\rangle$ state.)
- $E_1 = \frac{1}{2}|1\rangle\langle 1|$ (note that this is orthogonal to the sate $|0\rangle$ state)
- $E_I - E_0 - E_1$

Here E_0, E_1, E_2 are all psd and add up to identity, so they forma valid POVM. None of the three matrices are projectors. (note the success probability can be improved further).

Definition 1.1.2. Positive Operator Valued Measure (POVM): Suppose a measurement is described by measurement operator M_m is performed upon a quantum state $|\psi\rangle$. Then the probability of the outcome m is given by $p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle$. Suppose we define $E_m = M_m^\dagger M_m$, then clearly E_m is a positive operator. A POVM measurement is a measurement described by a set of measurement operators $\{E_m\}$ that satisfy the completeness relation $\sum_m E_m = I$ and $p(m) = \langle\psi| E_m |\psi\rangle$. Thus set of operators E_m are sufficient to determine the probabilities of the differnt measurement outcomes. The operators E_m are known as the POVM elements associated with the measurement. The complete set $\{E_m\}$ is known as POVM.

1.1.4 Postulate 4: Composite Systems

Consider a composite system amde up of two or more distinct quantum physical systems. What are the states of the composite system? How the state space of a composite system is built up from the state spaces of the component systems.

Definition 1.1.3. The state space of a composite physical system is the tensor product of the state spaces of the component physial systems. Say, if the systems are numbered through 1 to n and the system number i is prepared in the state $|\psi_i\rangle$ then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle$. This is called the tensor product state of the component states.

Example 1.1.9. Consider the orthonormal basis vectors of a single qubit system $|0\rangle$ and $|1\rangle$. Then the orthonormal basis vectors of a two qubit system are:

$$|0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\begin{pmatrix}1\\0\end{pmatrix}\\0\begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$|0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1\begin{pmatrix}0\\1\end{pmatrix}\\0\begin{pmatrix}0\\1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|1\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\begin{pmatrix}1\\0\end{pmatrix}\\1\begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

$$|1\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\begin{pmatrix}0\\1\end{pmatrix}\\1\begin{pmatrix}0\\1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Thus, $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are the orthonormal basis vectors of the two qubit Quantum System. Thus, the state of a particle for a two qubit quantum mechanical system can be written as $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ where $\alpha, \beta, \gamma, \delta$ are complex numbers such that $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$ and in the matrix form it can be written in matrix form as:

$$|\psi\rangle = \begin{pmatrix}\alpha\\\beta\\\gamma\\\delta\end{pmatrix}$$

Thus, say if $|\psi\rangle = \begin{pmatrix} 7 \\ 0 \\ \iota + 3 \\ 0 \end{pmatrix}$ then the state of the two qubit system is $|\psi\rangle = 7\,|00\rangle + (\iota + 3)\,|10\rangle$.

This, same concept of representing two-quibt quantum mechanical systems can also be extended to n-qubit quantum mechanical systems. It will be a vector in a 2^n-dimensional complex vector space. For example, in three dimensional vector space there will be $2^3 = 8$ basis vectors $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$. $|101\rangle$ will be represented in matrix form as:

$$|101\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Full and Partial Measurements of a multi-particle systems

A full measurement is a measurement that is performed on all the particles of a multi-particle system. A partial measurement is a measurement that is performed on some of the particles of a multi-particle system. Recal that when we measure a particle in the computational basis it collapses to one of the classical states.

Example 1.1.10. Consider a two qubit system in the state $|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$. A full measurement on the two qubit system will give the outcome 00 with probability $|\alpha|^2$, outcome 01 with probability $|\beta|^2$, outcome 10 with probability $|\gamma|^2$ and outcome 11 with probability $|\delta|^2$. Thus, also note that the sum of the probabilities of all the outcomes is 1, it is assumed the the state is initially normalized ($|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$).

A partial measurement on the two qubit system is a measurement performed on one of the qubits. Say, we perform a measurement on the first qubit, then the state of the second qubit will be the state of the system after the measurement on the first qubit.

Suppose on measurement of the first qubit the probability that it collapsed to state 1 is given by $|\gamma|^2 + |\delta|^2$ which is probabiltiy that the state collapses to $|10\rangle$

i.e. $|\gamma|^2$ plus the probability that the state collapses to the state $|11\rangle$ i.e. $|\delta|^2$, thus probabiltiy of the first qubit collapsing to the state 1 is given as: $|\gamma|^2 + |\delta|^2$. Then the state of the second qubit after the the measurement of the first qubit collapsed to 1 will be:

$$\frac{\gamma\,|10\rangle + \delta\,|11\rangle}{\sqrt{|\gamma|^2 + |\delta|^2}}$$

here we are normalizing the state in order to satisfy the normalized condition. Similarly, the probabilty of the first qubit collapsing to the state 0 is $|\alpha|^2 + |\beta|^2$ and the state of the second qubit after the measurement of the first qubit collapsed to 0 will be:

$$\frac{\alpha\,|00\rangle + \beta\,|01\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}}$$

Example 1.1.11. Consider a two qubit system $|\psi\rangle = \frac{1}{2}\,|00\rangle - \frac{\iota}{2}\,|10\rangle + \frac{1}{\sqrt{2}}\,|11\rangle$. Then the probability of the first qubit collapsing to the state 0 is $|\frac{1}{2}|^2 = \frac{1}{4}$ and the probability of the first qubit collapsing to the state 1 is $|-\frac{\iota}{2}|^2 + |\frac{1}{\sqrt{2}}|^2 = \frac{3}{4}$. Thus, the state of the second qubit after the measurement of the first qubit collapsed to 0 will be:

$$\frac{\frac{1}{2}\,|00\rangle}{\sqrt{|\frac{1}{2}|^2}} = |00\rangle$$

and the state of the second qubit after the measurement of the first qubit collapsed to 1 will be:

$$\frac{-\frac{\iota}{2}\,|10\rangle + \frac{1}{\sqrt{2}}\,|11\rangle}{\sqrt{|\frac{\iota}{2}|^2 + |\frac{1}{\sqrt{2}}|^2}} = \frac{-\iota}{\sqrt{3}}\,|10\rangle + \sqrt{\frac{2}{3}}\,|11\rangle$$

We can also, ask the question in reverse, that is if we measure the second qubit, then what is the state of the first qubit. The probability of the second qubit collapsing to the state 0 is $|\frac{1}{2}|^2 + |-\frac{\iota}{2}|^2 = \frac{1}{2}$ and the probability of the second qubit collapsing to the state 1 is $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$. Thus, the state of the first qubit after the measurement of the second qubit collapsed to 0 will be:

$$\frac{\frac{1}{2}\,|00\rangle - \frac{\iota}{2}\,|10\rangle}{\sqrt{|\frac{1}{2}|^2 + |-\frac{\iota}{2}|^2}} = \frac{1}{\sqrt{2}}\,|00\rangle - \frac{\iota}{\sqrt{2}}\,|10\rangle$$

and the state of the first qubit after the measurement of the second qubit collapsed to 1 will be:

$$\frac{\frac{1}{\sqrt{2}}\,|11\rangle}{\sqrt{|\frac{1}{\sqrt{2}}|^2}} = |11\rangle$$

Example 1.1.12. Consider a state $|\psi\rangle = \frac{1}{\sqrt{5}}|0000\rangle - \sqrt{\frac{2}{5}}|0100\rangle + \sqrt{\frac{1}{5}}|1111\rangle + \sqrt{\frac{1}{5}}|0110\rangle$. We wish to find the probability and resultant state is the 1st and 4th qubits are 0. The probability of the 1st qubit and 4th qubit both collapsing to 0 is $|\frac{1}{\sqrt{5}}|^2 + |-\sqrt{\frac{2}{5}}|^2 + |\sqrt{\frac{1}{5}}|^2 = \frac{4}{5}$. Thus, the resultant state of the 2nd and 3rd qubits after the measurement of the 1st and 4th qubits collapsed to 0 will be:

$$\frac{\frac{1}{\sqrt{5}}|0000\rangle - \sqrt{\frac{2}{5}}|0100\rangle + \sqrt{\frac{1}{5}}|0110\rangle}{\sqrt{\frac{4}{5}}} = \frac{1}{2}|0000\rangle - \frac{1}{\sqrt{2}}|0100\rangle + \frac{1}{2}|0110\rangle$$

Important Note

No-Cloning Theorem

Theorem 1.1.2.

Chapter 2

Qubits

A qubit is a two-dimensional quantum system. The state of a qubit can be represented as a vector in a two-dimensional complex vector space.

2.1 Bloch-Poincare sphere representation of a Qubit

Consider a Qubit in an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|0\rangle$ and $|1\rangle$ are orthonormal basis states, α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. HOw many real parameters are required to specify the state of a qubit?. Now since α and β are complex numbers, we can write them as $\alpha = |\alpha|e^{\iota\phi_\alpha}$ and $\beta = |\beta|e^{\iota\phi_\beta}$. Here, $|\alpha|$ and $|\beta|$ are real numbers and ϕ_α and ϕ_β are real numbers. Thus, we need 4 real numbers to specify the state of a qubit. NOw putting the condition that $|\alpha|^2 + |\beta|^2 = 1$, we can write $|\beta|^2 = 1 - |\alpha|^2 \implies |\beta| = \sqrt{1 - |\alpha|^2}$. Thus, we can rewrite the state of a qubit as:

$$|\psi\rangle = |\alpha|e^{\iota\phi_\alpha}|0\rangle + \sqrt{1 - |\alpha|^2}e^{\iota\phi_\beta}|1\rangle$$

Now the unknowns are $|\alpha|, \phi_\alpha, \phi_\beta$. Taking ϕ_α common we get:

$$|\psi\rangle = e^{\iota\phi_\alpha}\left(|\alpha||0\rangle + \sqrt{1 - |\alpha|^2}e^{\iota(\phi_\beta - \phi_\alpha)}|1\rangle\right)$$

Now, we know that the state of a qubit is defined upto a global phase. Thus, ignoring the term $e^{\iota\phi_\alpha}$ we can write the state of a qubit as:

$$|\psi\rangle = |\alpha||0\rangle + \sqrt{1 - |\alpha|^2}e^{\iota\phi}|1\rangle$$

where $\phi = \phi_\beta - \phi_\alpha$ is the relative phase between the states $|0\rangle$ and $|1\rangle$. Now, on substituting $|\alpha| = \cos(\theta/2)$, $\sqrt{1-|\alpha|^2} = \sin(\theta/2)$, we get:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{\iota\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

This cannot be further simplified to reduce the number of variables. Thus, we can clearly see that we need two real numbers - θ and ϕ to specify the state of a qubit. The state of a qubit can be represented as a point on the surface of a unit sphere in a three-dimensional space. This sphere is called the Bloch Sphere. A visual representation of the Bloch Sphere (or Bloch-Poincare Sphere) is shown in the figure 2.1.

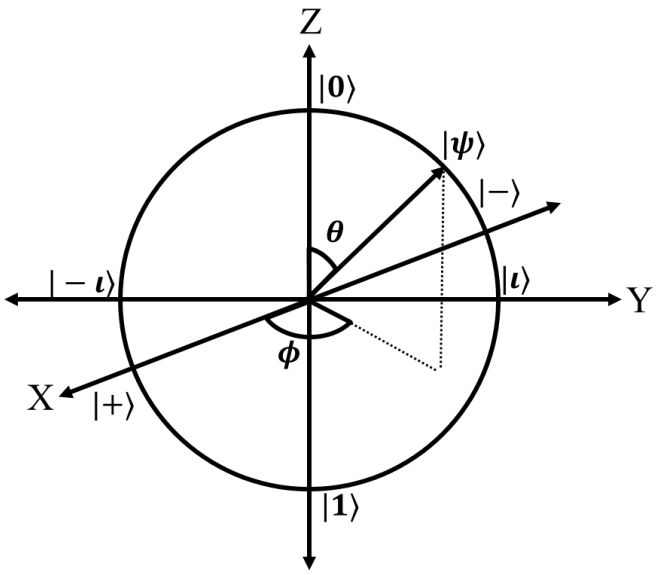


Figure 2.1: Bloch-Poincare Sphere

Thus for different values of θ and ϕ we get different points on the bloch sphere which represents the different states of a qubit. For a unique mapping from the θ and ϕ to the points on the Bloch Sphere, we restrict the value of $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. Thus, the state of a qubit can be represented as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{\iota\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

where $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$ are the polar (represents magnitude or argument) and azimuthal (represents relative phase) angles of the Bloch Sphere respectively.

The following code using Qiskit can be used to plot the Bloch Sphere. For running the code you need to install the qiskit library. Click here for guide on installing qiskit

```
import matplotlib.pyplot as plt
from qiskit.visualization import plot_bloch_vector # Import the function to plot the Bloch vector
from qiskit.visualization import plot_bloch_multivector # Import the function to plot the Bloch vector

'''
Function to plot the Bloch Sphere

plot_bloch_vector(bloch, title='', ax=None, figsize=None, coord_type='spherical', font_size=None)

Plots a Bloch Sphere

Parameters
- bloch (list[double]) - array of three elements where [<x>, <y>, <z>] (Cartesian) or
[<r>, <theta>, <phi>] (spherical in radians) <theta> is inclination angle from +z direction
<phi> is azimuth from +x direction
- title (str) - a string that represents the title of the plot
- ax (matplotlib.axes.Axes) - an axes of the current figure to plot the Bloch sphere into
- figsize (tuple) - a tuple (width, height) in inches that represents the size of the figure
- coord_type (str) - the coordinate system to use. 'spherical' or 'cartesian', default is cartesian
- font_size (int) - the font size of the tex

Returns
- matplotlib.figure.Figure - a matplotlib figure object

Raises
- ImportError - if matplotlib is not installed
'''

plot_bloch_vector([0,1,0], title='Bloch Sphere')
```

The output of the above code is shown in the figure 2.2.

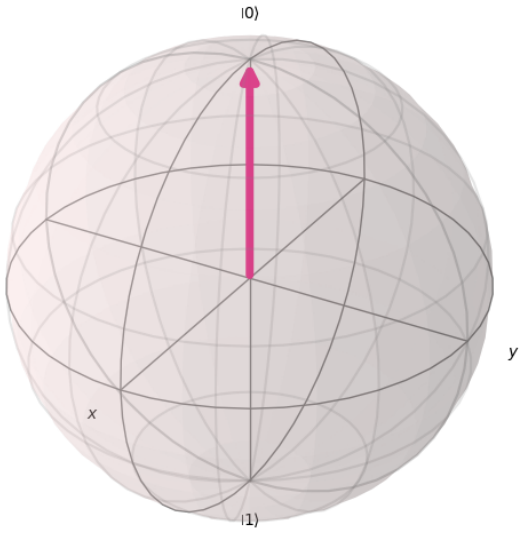


Figure 2.2: Bloch Sphere

Note that the Bloch Sphere is a geometric representation of the pure states of a qubit. The mixed states of a qubit are represented by the interior of the Bloch Sphere.

Example 2.1.1. The state of a qubit for different values of θ and ϕ is shown in the table 2.1.

θ	ϕ	State of Qubit	At Axis
0	0	$\|0\rangle$	+ve Z
π	0	$\|1\rangle$	-ve Z
$\frac{\pi}{2}$	0	$\|+\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	+ve X
$\frac{\pi}{2}$	π	$\|-\rangle = \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	-ve X
$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\|+\iota\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + i\|1\rangle)$	+ve Y
$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\|-\iota\rangle = \frac{1}{\sqrt{2}}(\|0\rangle - i\|1\rangle)$	-ve Y

Table 2.1: State of a Qubit for different values of θ and ϕ

Quantum Gates acting on Qubits can be thought of as rotations on the Bloch Sphere. This will be explained further in the later sections.

Note that orthonormal states reside at the ends of the diameter of the Bloch Sphere. Let a state $|\psi\rangle$ be represented by a point on the Bloch Sphere. The state

$|\psi\rangle$ can be written as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{\iota\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

Then the state (say $|\psi'\rangle$) exactly opposite on the Bloch sphere will have the state:

$$|\psi'\rangle = \cos\left(\frac{\pi+\theta}{2}\right)|0\rangle + e^{\iota\phi}\sin\left(\frac{\pi+\theta}{2}\right)|1\rangle$$

Thus, upon simplifying we get:

$$|\psi'\rangle = \cos\left(\frac{\pi+\theta}{2}\right)|0\rangle + e^{\iota\phi}\sin\left(\frac{\pi+\theta}{2}\right)|1\rangle$$

$$|\psi'\rangle = -\sin\left(\frac{\theta}{2}\right)|0\rangle + e^{\iota\phi}\cos\left(\frac{\theta}{2}\right)|1\rangle$$

Now taking the inner product of the states $|\psi\rangle$ and $|\psi'\rangle$ we get:

$$\langle\psi|\psi'\rangle = -\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right) + e^{\iota\phi}e^{-\iota\phi}\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right) = 0$$

Thus, the inner product is zero which implies that the states on the opposite ends of the Bloch Sphere are orthogonal.

Note: Such a correspondence does not exist for higher dimensions i.e. No such visualization exists for say Qudits which is d level quantum systems. From figure 2.1 we can see that to convert from polar to cartesian coordinates we use the following relations:

$$x = r\sin\theta\cos\phi$$

$$y = r\sin\theta\sin\phi$$

$$z = r\cos\theta$$

note that here $r = 1$ (from pure states i.e. points on the surface of the sphere). To convert from carteisan to polar coordinates we use the following relations:

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \cos^{-1}(z)$$

$$\phi = \tan^{-1}\left(\frac{y}{x}\right)$$

Let us denote the vector as $\vec{n} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta, \cos\theta)$. Points on a Bloch sphere is alwyas a pure state.

2.2 Single Qubit Gates

As said earlier that quantum gates are unitary operators that act on the state of a quantum system. Recall from Appendix that the unitary operators are the ones that preserve the inner product of the vectors and the norm of the vectors i.e. they only rotate the vectors in the complex vector space (Rotation matrices in $\mathbb{C}^n$). Thus, a state normalized initially remains in the normalized state and does not change its norm upon the action of Unitary gates/operators. The single qubit gates are the quantum gates that act on a single qubit, represented by 2x2 matrices. Since a quantum state is represented by a column vector (in some basis, generally standard basis). A gate is a unitary matrix/transformation that acts on the qubit(column vector) and changes its state to some other state. They are rotations on the Bloch Sphere.

Example 2.2.1. Consider a 2×2 matrix U given as:

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

in the orthonormal input and output basis $\{|0\rangle, |1\rangle\}$, thus, we can write the matrix U as:

$$U = a\,|0\rangle\langle 0| + b\,|0\rangle\langle 1| + c\,|1\rangle\langle 0| + d\,|1\rangle\langle 1|$$

This, will be used to express the Pauli Matrices in the later sections.

2.2.1 Pauli Matrices

Pauli-I Gate

It is used for the measurement of decoherence (Qubits start to lose its Quantum abilities after certain time). The Truth table is given as in table 2.2. It does no rotation of the bloch vector on the Bloch Sphere.

Input	Output
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|1\rangle$

Table 2.2: Truth Table for Pauli-I Gate

The code for the Pauli-I Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')

circuit = QuantumCircuit(qreg_q)

circuit.id(qreg_q[0]) #For Pauli-Identity Gate
circuit.draw(output='mpl')#.savefig('../images/pauli-i.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.3

Figure 2.3: Pauli-I Gate

In operator Notation it is:

$$I|0\rangle = |0\rangle$$
$$I|1\rangle = |1\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$\sigma_I = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

It's Eigen values are 1 and 1. It's Eigen vectors are any vector in the 2-D Space. Note that it's Unitary($II^\dagger = I^\dagger I = I; I = I^{-1}$) and Hermitian Matrix($I = I^\dagger$), thus a Normal Matrix ($II^\dagger = I^\dagger I$) hence Unitarily digonalizable. Thus, it has a Spectral decomposition (outer product representation) which can be written as:

$$I = |0\rangle\langle 0| + |1\rangle\langle 1|$$

Action of Pauli-I Gate on a general qubit $\alpha|0\rangle + \beta|1\rangle$ is:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha|0\rangle + \beta|1\rangle$$

Pauli-X Gate

It is also known as the NOT gate. It is equivalent of Classical NOT Gate. It is thus called Bit Flip Gate. The truth table is as in 2.3. It does anticlockwise π rotation about X axis of Bloch sphere. It flips the state $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$.

Input	Output
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$

Table 2.3: Truth Table for Pauli-X Gate

The code for the Pauli-X Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='Not Gate')

circuit.x(qreg_q[0])

circuit.draw(output='mpl')#.savefig('../images/pauli-x.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.4.

Figure 2.4: Pauli-X Gate

In the operator Notation it is:

$$X|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$\sigma_X = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

It's Eigen values are 1 and -1. It's corresponding Eigen vectors are $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable. It has a Spectral decomposition (outer product representation) which can be written as:

$$X = |+\rangle\langle+| - |-\rangle\langle-|$$

Upon, substituting the values of $|+\rangle$ and $|-\rangle$ we get:

$$\begin{aligned} X &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(\langle 0| + \langle 1|) - \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\frac{1}{\sqrt{2}}(\langle 0| - \langle 1|) \\ &= \frac{1}{2}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| + |1\rangle\langle 1|) + \frac{1}{2}(|0\rangle\langle 0| - |0\rangle\langle 1| - |1\rangle\langle 0| + |1\rangle\langle 1|) \\ &= |0\rangle\langle 1| + |1\rangle\langle 0| \end{aligned}$$

Thus, the Pauli-X Gate can also be written as:

$$X = |0\rangle\langle 1| + |1\rangle\langle 0|$$

Note that the Pauli-X Gate can be written as $X = HZH$ or $\sigma_X = H\sigma_Z H$ where H and Z are the Hadamard and Pauli-Z Gates respectively. Action of Pauli-X Gate on a general qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \beta|0\rangle + \alpha|1\rangle$$

Pauli-Y Gate

It is a anitclockwise π rotation about Y axis of Bloch Sphere. It flips the Qubit and muultiplies a complex amplitude. The truth table is as in table 2.4.

Input	Output
$\|0\rangle$	$i\|1\rangle$
$\|1\rangle$	$-i\|0\rangle$

Table 2.4: Truth Table for Pauli-Y Gate

The code for the Pauli-Y Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='Not Gate')

circuit.y(qreg_q[0]) #Apply the Y gate

circuit.draw(output='mpl')#.savefig('../images/pauli-x.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.5.

Figure 2.5: Pauli-Y Gate

In the operator Notation it is:

$$Y|0\rangle = i|1\rangle$$
$$Y|1\rangle = -i|0\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$\sigma_Y = Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

It's Eigen values are 1 and -1. It's corresponding Eigen vectors are $|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ and $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable. It has a Spectral decomposition (outer product representation) which can be written as:

$$Y = |+i\rangle\langle+i| - |-i\rangle\langle-i|$$

Upon, substituting the values of $|+i\rangle$ and $|-i\rangle$ we get:

$$\begin{aligned} Y &= \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)\frac{1}{\sqrt{2}}(\langle 0| - i\langle 1|) - \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)\frac{1}{\sqrt{2}}(\langle 0| + i\langle 1|) \\ &= \frac{1}{2}(|0\rangle\langle 0| + i|0\rangle\langle 1| - i|1\rangle\langle 0| + i|1\rangle\langle 1|) - \frac{1}{2}(|0\rangle\langle 0| - i|0\rangle\langle 1| + i|1\rangle\langle 0| + i|1\rangle\langle 1|) \\ &= i|0\rangle\langle 1| - i|1\rangle\langle 0| \end{aligned}$$

Thus, the Pauli-Y Gate can also be written as:

$$Y = -i\left|0\right\rangle\left\langle 1\right| + i\left|1\right\rangle\left\langle 0\right|$$

Note that the Pauli-Y Gate can be written as $Y = \iota XZ$ or $\sigma_Y = \iota\sigma_X\sigma_Z$ where X and Z are the Pauli-X and Pauli-Z Gates respectively. This means that the Pauli-Y Gate is a rotation about the Y-axis of the Bloch Sphere by π in anticlocwise direction and is equivalent to a rotation about the X-axis by anticlockwise π followed by a rotation about the Z-axis by anticlockwise π (upto a global phase).

Action of Pauli-Y Gate on a general qubit $\left|\psi\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$ is:

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -i\beta \\ i\alpha \end{bmatrix} = i\alpha\left|1\right\rangle - i\beta\left|0\right\rangle$$

Pauli-Z Gate

It is a anticlockwise π rotation about Z axis of Bloch Sphere. It flips the sign of the $\left|1\right\rangle$ state. It is also called Phase Flip/Phase shoft/Sign flip gate. The truth table is as in table 2.5.

Input	Output
$\left\|0\right\rangle$	$\left\|0\right\rangle$
$\left\|1\right\rangle$	$-\left\|1\right\rangle$

Table 2.5: Truth Table for Pauli-Z Gate

The code for the Pauli-Z Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='Not Gate')

circuit.z(qreg_q[0]) #Apply the Z gate

circuit.draw(output='mpl')#.savefig('../images/pauli-z.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.6.

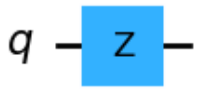


Figure 2.6: Pauli-Z Gate

In the operator Notation it is:

$$Z|0\rangle = |0\rangle$$
$$Z|1\rangle = -|1\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$\sigma_Z = Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

It's Eigen values are 1 and -1. It's corresponding Eigen vectors are $|0\rangle$ and $|1\rangle$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable. It has a Spectral decomposition (outer product representation) which can be written as:

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|$$

In Compact form, we can write Pauli-Z Gate as:

$$Z|j\rangle = (-1)^j |j\rangle$$

where $j = 0, 1$. Action of Pauli-Z Gate on a general qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} = \alpha|0\rangle - \beta|1\rangle$$

Important Note

Note that all the Pauli-gate are Unitary and Hermitian. Since, all the Unitary gates ($U^\dagger = U^{-1}$) are reversible. Thus, from Postulate 2: Evolution, we can apply Pauli Gates. All the Pauli gates are reversible and thus can be applied in Quantum Circuits. The four Pauli gates can also be denoted as $X = \sigma_X$, $Y = \sigma_Y$, $Z = \sigma_Z$ and $I = \sigma_I$. All the Pauli Gates as said earlier are anticlockwise rotations on the Bloch Sphere. The Pauli-X Gate is a π rotation about the X-axis, Pauli-Y Gate is a π rotation about the Y-axis and Pauli-Z Gate is a π rotation about the Z-axis. The Pauli-I Gate does not do any rotation on the Bloch Sphere. Also note that applyin the same gate twice will be equivalent to applying the Identity Gate. Thus, the Pauli Gates are self-inverse. Since they are hermitian hence, $U^\dagger = U$ and they are unitary thus, $U^\dagger U = UU^\dagger = I$. Thus combining, the two we get $U^2 = I$. This, can also be verified as shown below:

$$X^2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

$$Y^2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

$$Z^2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

$$I^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Thus, we can see that all the Pauli Gates are self-inverse. The Pauli gates also form the basis for the single qubit gates. Any single qubit gate can be written as a linear combination of the Pauli Gates. In other words, any 2×2 matrix can be written as a linear combination of the Pauli Matrices.

$$A = \alpha I + \beta X + \gamma Y + \delta Z$$

where $\alpha, \beta, \gamma, \delta$ are complex numbers. The Pauli Matrices, except Identity are also traceless (trace=0) and all the Pauli matrices are Hermitian. Note here it should be presumed that the the rotations are always in anticlockwise direction.

2.2.2 Hadamard Gate

It creates a superposition of Qubits. It is a π rotation about the X+Z axis of Bloch Sphere or a $\pi/2$ rotation about the Y axis followed by a π rotation about the X axis. It is also called as the Square root of NOT gate. The truth table is as in table 2.6.

Input	Output
$\|0\rangle$	$\|+\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
$\|1\rangle$	$\|-\rangle = \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$

Table 2.6: Truth Table for Hadamard Gate

The code for the Hadamard Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='Hadamard Gate')

circuit.h(qreg_q[0]) #Apply the H gate

circuit.draw(output='mpl')#.savefig('../images/hadamard.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.7.

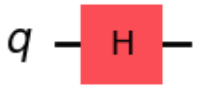


Figure 2.7: Hadamard Gate

In the operator Notation it is:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

Generally in a quantum circuit we intialise a qubit in state $|0\rangle$ and thus Hadamard gate can be used to create a superposition. It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

In compact form we can write the action of Hadamard Gate on one qubit as:

$$H|j\rangle = \frac{1}{\sqrt{2}} \sum_{k=0}^{1} (-1)^{jk} |k\rangle$$

where $j = 0, 1$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable. It has a Spectral decomposition (outer product representation) which can be written as:

$$H = |+\rangle\langle 0| + |-\rangle\langle 1|$$

Action of Hadamard Gate on a general qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is:

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{\alpha+\beta}{\sqrt{2}} \\ \frac{\alpha-\beta}{\sqrt{2}} \end{bmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

Note that the Hadamard gate is also a self-inverse gate. Since its Hermitian as well as Unitary, thus $H^\dagger = H$ and $H^\dagger H = HH^\dagger = I; H = H^{-1}$. Thus, $H^2 = I$. Intuitively this means that applying Hadamard gate twice will bring the quantum state back to the same quantum state. This can also be imagined as doing rotations on bloch sphere and then arriving back at the same state.

$$H^2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1+1 & 1-1 \\ 1-1 & 1+1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Thus, we can see that the Hadamard gate is self-inverse. This means that if I apply Hadamard gate twice on $|0\rangle$ or on $|1\rangle$ then we will arrive back at the same state.

$$H^2|0\rangle = H|+\rangle = |0\rangle$$

$$H^2|1\rangle = H|-\rangle = |1\rangle$$

Hadamard Gates on Multiple Qubits

Hadamard gate can also be applied on multiple qubits. The Hadamard gate on multiple qubits (say n qubits) using tensor products is given as:

$$H^{\otimes n} = \underbrace{H \otimes H \otimes \ldots \otimes H}_{n \text{ times}}$$

One of the way to find the Hadamard matrix is to perform the tensor product n times to get the Hadamard matrix acting on the n qubits. For example, we can find Hadamard gate acting on 2 qubits in matrix form using tensor product as:

$$H^{\otimes 2} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}$$

Clearly this does not scale. Another method is, we can find the action of Hadamard gate on multiple qubits by applying the Hadamard gate on each qubit separately.

$$H^{\otimes n}(|0\rangle \otimes |0\rangle \otimes \ldots \otimes |0\rangle) = H|0\rangle \otimes H|0\rangle \otimes \ldots \otimes H|0\rangle = |+\rangle \otimes |+\rangle \otimes \ldots \otimes |+\rangle$$

This is a better method then writing entire matrix for n qubits. Thus, we can see that the Hadamard gate on multiple qubits creates a superposition of all possible states of n qubits. So, for writing the Hadamard gate on multiple qubits, in a compact form, first we consider the compact form of $H^{\otimes 2}$ as:

$$H^{\otimes 2}|x\rangle = H|x_1\rangle \otimes H|x_2\rangle$$

where $x = x_1x_2$. Now using the fact that $H|x_1\rangle = \frac{1}{\sqrt{2}}\sum_{y_1\in\{0,1\}}(-1)^{x_1y_1}|y_1\rangle$ and $H|x_2\rangle = \frac{1}{\sqrt{2}}\sum_{y_2\in\{0,1\}}(-1)^{x_2y_2}|y_2\rangle$, we get:

$$H^{\otimes 2}|x\rangle = \frac{1}{2}\sum_{y_1,y_2\in\{0,1\}}(-1)^{x_1y_1}(-1)^{x_2y_2}|y_1\rangle \otimes |y_2\rangle = \frac{1}{2}\sum_{y\in\{0,1\}^2}(-1)^{x\cdot y}|y\rangle$$

where $x = x_1x_2$ and $y = y_1y_2$. Thus, it can be generalized and we can write the Hadamard gate on n qubits as:

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle$$

where $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_n$ and $x \cdot y$ is the dot product between x and y. Thus, we can see that the Hadamard gate on n qubits creates a superposition of all possible states of n qubits. In general, we start with state $|0\rangle$ for all the qubits initially, thus Action of hadamard gate on n qubits all in state $|0\rangle$ is:

$$H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} |y\rangle$$

Example 2.2.2. Fast Hadamard Gate using Tabular method :
Consider a state $|\psi\rangle = \frac{|000\rangle + |110\rangle - |100\rangle - |111\rangle}{\sqrt{4}}$. We are suppose to find $H^{\otimes 3} |\psi\rangle$. Thus, on solving using the equation for hadamard gate acting on multiple qubits $H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$ we get,

$$\frac{1}{\sqrt{8}} (|001\rangle - |011\rangle + |100\rangle + |110\rangle + |111\rangle)$$

2.2.3 S and $S^\dagger$ Gate

These are some specialised rotations,

S Gate

It is $\pi/2$ rotation around Z axis of Bloch Sphere. It is also called as the Phase Gate. The truth table is as in table 2.7.

Input	Output
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$i\|1\rangle$

Table 2.7: Truth Table for S Gate

The code for the S Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='S Gate')

circuit.s(qreg_q[0]) #Apply the S gate
```

```
8
9 circuit.draw(output='mpl')#.savefig('../images/s-gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.8.

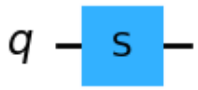


Figure 2.8: S Gate

In the operator Notation it is:

$$S|0\rangle = |0\rangle$$
$$S|1\rangle = i|1\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = e^{i\pi/4} \begin{bmatrix} e^{-i\pi/4} & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

Thus, it is called as $\pi/4$ gate. It's Eigen values are 1 and i. It's corresponding Eigen vectors are $|0\rangle$ and $|1\rangle$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable. It has a Spectral decomposition (outer product representation) which can be written as:

$$S = |0\rangle\langle 0| + i|1\rangle\langle 1|$$

Action of S Gate on a general qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is:

$$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ i\beta \end{bmatrix} = \alpha|0\rangle + i\beta|1\rangle$$

$S^\dagger$ Gate

It is $\pi/2$ rotation around Z axis of Bloch Sphere in the opposite direction. It is also called as the Conjugate Phase Gate. The truth table is as in table 2.8.

Input	Output
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-i\|1\rangle$

Table 2.8: Truth Table for $S^\dagger$ Gate

The code for the $S^\dagger$ Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='T Gate')

circuit.sdg(qreg_q[0]) #Apply the sdag gate

circuit.draw(output='mpl')#.savefig('../images/sdag-gate.png') #Draw
    the circuit
```

The circuit symbol is as shown in figure 2.9.

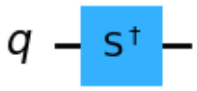


Figure 2.9: $S^\dagger$ Gate

In the operator Notation it is:

$$S^\dagger |0\rangle = |0\rangle$$
$$S^\dagger |1\rangle = -i|1\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$S^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix} = e^{-i\pi/4} \begin{bmatrix} e^{i\pi/4} & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix}$$

It's Eigen values are 1 and $-i$. It's corresponding Eigen vectors are $|0\rangle$ and $|1\rangle$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable.

It has a Spectral decomposition (outer product representation) which can be written as:

$$S^\dagger = |0\rangle\langle 0| - i\,|1\rangle\langle 1|$$

Action of $S^\dagger$ Gate on a general qubit $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ is:

$$\begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ -i\beta \end{bmatrix} = \alpha\,|0\rangle - i\beta\,|1\rangle$$

Note that both S and $S^\dagger$ gates are Unitary but not Hermitian thus they are not self-inverse. Reccall, that the only criteria for a gate is to be a Unitary (from Postulate 2: Unitary evolution) gate and not necessarily Hermitian.

2.2.4 T and $T^\dagger$ Gate

These are some specialised rotations,

T Gate

It is $\pi/4$ rotation around Z axis of Bloch Sphere. It is also called as the $\pi/8$ Gate. The truth table is as in table 2.9.

Input	Output
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$e^{i\pi/4}\,\|1\rangle$

Table 2.9: Truth Table for T Gate

The code for the T Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='T Gate')

circuit.sdg(qreg_q[0]) #Apply the T gate

circuit.draw(output='mpl')#.savefig('../images/sdag-gate.png') #Draw
    the circuit
```

The circuit symbol is as shown in figure 2.10.

Figure 2.10: T Gate

In the operator Notation it is:

$$T|0\rangle = |0\rangle$$
$$T|1\rangle = e^{i\pi/4}|1\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} = e^{i\pi/8} \begin{bmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{bmatrix}$$

Thus, it is called as $\pi/8$ gate. It's Eigen values are 1 and $e^{i\pi/4}$. It's corresponding Eigen vectors are $|0\rangle$ and $|1\rangle$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable. It has a Spectral decomposition (outer product representation) which can be written as:

$$T = |0\rangle\langle 0| + e^{i\pi/4}|1\rangle\langle 1|$$

Action of T Gate on a general qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is:

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ e^{i\pi/4}\beta \end{bmatrix} = \alpha|0\rangle + e^{i\pi/4}\beta|1\rangle$$

$T^\dagger$ Gate

It is $\pi/4$ rotation around Z axis of Bloch Sphere in the opposite direction. It is also called as the $\pi/8$ Gate. The truth table is as in table 2.10.

Input	Output
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$e^{-i\pi/4}\|1\rangle$

Table 2.10: Truth Table for $T^\dagger$ Gate

The code for the $T^\dagger$ Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='T Gate')

circuit.tdg(qreg_q[0]) #Apply the T gate

circuit.draw(output='mpl')#.savefig('../images/tdag-gate.png') #Draw
    the circuit
```

The circuit symbol is as shown in figure 2.11.

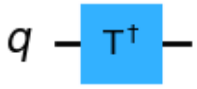


Figure 2.11: $T^\dagger$ Gate

In the operator Notation it is:

$$T^\dagger |0\rangle = |0\rangle$$
$$T^\dagger |1\rangle = e^{-i\pi/4} |1\rangle$$

It is represented by the matrix in the computational basis $(|0\rangle, |1\rangle)$ as:

$$T^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix} = e^{-i\pi/8} \begin{bmatrix} e^{i\pi/8} & 0 \\ 0 & e^{-i\pi/8} \end{bmatrix}$$

It's Eigen values are 1 and $e^{-i\pi/4}$. It's corresponding Eigen vectors are $|0\rangle$ and $|1\rangle$. It is a Unitary and Hermitian Matrix. It is a Normal Matrix hence Unitarily Diagonalizable. It has a Spectral decomposition (outer product representation) which can be written as:

$$T^\dagger = |0\rangle \langle 0| + e^{-i\pi/4} |1\rangle \langle 1|$$

Action of $T^\dagger$ Gate on a general qubit $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ is:

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ e^{-i\pi/4}\beta \end{bmatrix} = \alpha |0\rangle + e^{-i\pi/4}\beta |1\rangle$$

Some relations between S and T Gates:

$$S = T^2$$
$$S^\dagger = T^\dagger T^\dagger$$

This can be verified as follows. Intuitively, one can think as twice $\pi/4$ rotation around Z axis is $\pi/2$ rotation around Z axis. Thus, the reltaions.

$$T^2 = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = S$$
$$T^\dagger T^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\pi/2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix} = S^\dagger$$

2.2.5 Quantum Rotation Gates

These are generalized rotations in the Bloch Sphere denoted as R_X, R_Y, R_Z about the X,Y and Z axis of Bloch Sphere respectively.

R_X Gate

It is a rotation about X axis of Bloch Sphere by an angle θ. This is denoted as R_θ^X. The matrix representation of R_θ^X is:

$$R_\theta^X = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} = \cos\frac{\theta}{2} I - \iota \sin\frac{\theta}{2}\sigma_X = e^{-\iota\frac{\theta}{2}\sigma_X}$$

The code for the R_X Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='U1 Gate')

circuit.rx(pi/2,qreg_q[0]) #Apply the RX gate

circuit.draw(output='mpl')#.savefig('../images/rx-gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.12.

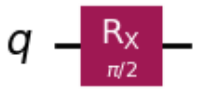


Figure 2.12: R_X Gate

Note that the value $\pi/2$ given in the figure 2.12 denotes the angle by which we rotate around the X Axis of Bloch Sphere.

R_Y Gate

It is a rotation about Y axis of Bloch Sphere by an angle θ. This is denoted as R_θ^Y. The matrix representation of R_θ^Y is:

$$R_\theta^Y = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} = \cos\frac{\theta}{2}I - \iota\cos\frac{\theta}{2}\sigma_Y = e^{-\iota\frac{\theta}{2}\sigma_Y}$$

The code for the R_Y Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='RY Gate')

circuit.ry(pi/2,qreg_q[0]) #Apply the RY gate

circuit.draw(output='mpl')#.savefig('../images/ry-gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.13.

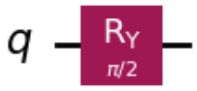


Figure 2.13: R_Y Gate

Note here the value $\pi/2$ denotes the angle by which we rotate around the Y Axis of Bloch Sphere.

R_Z Gate

It is a rotation about Z axis of Bloch Sphere by an angle θ. This is denoted as R_θ^Z. The matrix representation of R_θ^Z is:

$$R_\theta^Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} = \cos\frac{\theta}{2}I - \iota\cos\frac{\theta}{2}\sigma_Z = e^{-\iota\frac{\theta}{2}\sigma_Z}$$

The code for the R_Z Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='RZ Gate')

circuit.rz(pi/2,qreg_q[0]) #Apply the RZ gate

circuit.draw(output='mpl')#.savefig('../images/rz-gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.14.

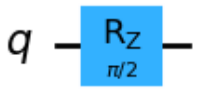


Figure 2.14: R_Z Gate

Note here the value $\pi/2$ denotes the angle by which we rotate around the Z Axis of Bloch Sphere.

This Quantum Rotation Gates can be used in making variational circuit as angles can vary.

2.2.6 Universal Quantum Gates

To construct any single qubit Quantum gates.

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i(\phi+\lambda)}\cos(\theta/2) \end{bmatrix}$$

where $\theta, \phi, \lambda \in \mathbb{R}$ are rotations around x, y and z axis respectively. Here, $0 \leq \theta \leq \pi$, $0 \leq \phi \leq 2\pi$ and $0 \leq \lambda \leq 2\pi$. Similarly, we can fix θ and write U_2 gate as:

$$U_2(\phi, \lambda) = \begin{bmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\phi+\lambda)} \end{bmatrix}$$

Here $\theta = \pi/2$ by default. Similarly, we can fix θ and ϕ and write U_1 gate as:

$$U_1(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}$$

Here $\theta = \phi = 0$ by default.

The code for the U_3 Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(1, 'q')
circuit = QuantumCircuit(qreg_q,name='U3 Gate')

circuit.u(pi/2,pi/2,pi/2,qreg_q[0]) #Apply the U3 gate

circuit.draw(output='mpl')#.savefig('../images/u3-gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.15.

Figure 2.15: U_3 Gate

Note here the values $\pi/2, \pi/2, \pi/2$ denotes the angles θ, ϕ, λ by which we rotate around the Bloch Sphere respectively.

2.3 Multi-Qubit Gates

Quantum Pauli gates and hadmard gate are Unitary and Hermitian, and all the other single qubit gate are Unitary, hence, reversible. In general, we can generalise any single qubit gate to multi-qubit gate. Note that since all the gates can be in general represented as Unitary Matrices, thus the number of qubits in input = number of qubits in output always. Multi-Qubit gates act on multiple qubits at once thus the matrices are of size $2^n \times 2^n$ where n is the number of qubits.

2.3.1 Two-qubit Gates

CNOT/CX Gate

It is a two qubit gate, also called as the Controlled-NOT gate. Here the 1st Qubit is controlled and the 2nd Qubit is target. (It is also possible in the other way that the 2nd Qubit is controlled and the 1st Qubit is target). This gate is used to create entanglement. The truth table is as in table 2.11.

Input	Output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

Table 2.11: Truth Table for CNOT Gate

For the table in 2.11, the 1st Qubit is control and the 2nd Qubit is target. Thus, the CNOT gate flips the target qubit if the control qubit is $|1\rangle$. The code for the CNOT Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

# Add 2 qubits for CNOT gate demo
qreg_q = QuantumRegister(2, 'q')
circuit = QuantumCircuit(qreg_q,name='CNOT Gate')

circuit.cx(qreg_q[0],qreg_q[1]) #Apply the CNOT gate

circuit.draw(output='mpl')#.savefig('../images/cnot-gate.png') #Draw
    the circuit
```

The circuit symbol is as shown in figure 2.16.

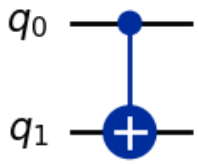


Figure 2.16: CNOT Gate

Here it can be seen that the 1st Qubit is control and the 2nd Qubit is target. In the figure 2.16, the dot denotes the control qubit and the cross denotes the target qubit.

In the operator Notation it is:

$$CX\left|00\right\rangle = \left|00\right\rangle$$
$$CX\left|01\right\rangle = \left|01\right\rangle$$
$$CX\left|10\right\rangle = \left|11\right\rangle$$
$$CX\left|11\right\rangle = \left|10\right\rangle$$

It is represented by the matrix in the computational basis $(\left|00\right\rangle, \left|01\right\rangle, \left|10\right\rangle, \left|11\right\rangle)$ as:

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

In Outer product representation, it can be written as:

$$CX = \left|00\right\rangle\left\langle 00\right| + \left|01\right\rangle\left\langle 01\right| + \left|10\right\rangle\left\langle 11\right| + \left|11\right\rangle\left\langle 10\right|$$

In Compact form we can write the CNOT gate as:

$$CX\left|xy\right\rangle = \left|x(x \oplus y)\right\rangle$$

where $\oplus$ denotes the XOR operation. It is a non-linear operation. It is used in Quantum Error Correction Codes. This is a very important gate which is used in

Teleoprtation protocol, Superdense Coding, Quantum Error Correction Codes etc. Action of CX/CNOT on general two - qubit $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \delta \\ \gamma \end{bmatrix} = \alpha|00\rangle + \beta|01\rangle + \delta|10\rangle + \gamma|11\rangle$$

CY Gate

It is a two qubit gate, also called as the Controlled-Y gate. Here the 1st Qubit is controlled and the 2nd Qubit is target (Pauli - Y Gate) (It is also possible in the other way that the 2nd Qubit is controlled and the 1st Qubit is target).

The truth table is as in table 2.12.

Input	Output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$i\|10\rangle$
$\|11\rangle$	$-i\|11\rangle$

Table 2.12: Truth Table for CY Gate

For the table in 2.12, the 1st Qubit is control and the 2nd Qubit is target. Thus, the CY gate flips the target qubit if the control qubit is $|1\rangle$. The code for the CY Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

# Add 2 qubits for CNOT gate demo
qreg_q = QuantumRegister(2, 'q')
circuit = QuantumCircuit(qreg_q,name='CY Gate')

circuit.cy(qreg_q[0],qreg_q[1]) #Apply the CY gate

circuit.draw(output='mpl')#.savefig('../images/cy-gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.17.

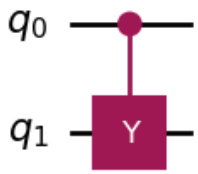


Figure 2.17: CY Gate

In this figure 2.17, the first qubit is control and the second qubit is target. If the first qubit is $|1\rangle$, then Y gate is applied on the second qubit. The Y Gate is a gate which rotates the qubit by π around the Y axis of Bloch Sphere. Thus, if the input is $|0\rangle$, then the output is $|0\rangle$ and if the input is $|1\rangle$, then the output is $-i\,|1\rangle$.

In the operator Notation it is:

$$\begin{aligned} CY\,|00\rangle &= |00\rangle \\ CY\,|01\rangle &= |01\rangle \\ CY\,|10\rangle &= i\,|11\rangle \\ CY\,|11\rangle &= -i\,|10\rangle \end{aligned}$$

It is represented by the matrix in the computational basis $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$ as:

$$CY = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\iota \\ 0 & 0 & \iota & 0 \end{bmatrix}$$

In Outer product representation, it can be written as:

$$CY = |00\rangle\langle 00| + |01\rangle\langle 01| + \iota\,|11\rangle\langle 10| - \iota\,|10\rangle\langle 11|$$

In Compact form we can write the CY gate as:

$$CY\,|xy\rangle = (-1)^y \iota^x\,|x(x \oplus y)\rangle$$

Flipping amplitude of 1 state. Marking of this state can be untilised for marking elements in data base. It is useful in Grover's Algorithm.

Action of CY on general two - qubit $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\iota \\ 0 & 0 & \iota & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \iota\delta \\ -\iota\gamma \end{bmatrix} = \alpha|00\rangle + \beta|01\rangle - \iota\delta|10\rangle + \iota\gamma|11\rangle$$

CZ/CPHASE Gate

It is a two qubit gate, also called as the Controlled-Z gate. Here the 1st Qubit is controlled and the 2nd Qubit is target (Pauli - Z Gate) (It is also possible in the other way that the 2nd Qubit is controlled and the 1st Qubit is target).

The truth table is as in table 2.13.

Input	Output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|10\rangle$
$\|11\rangle$	$-\|11\rangle$

Table 2.13: Truth Table for CZ Gate

For the table in 2.13, the 1st Qubit is control and the 2nd Qubit is target. Thus, the CZ gate flips the target qubit if the control qubit is $|1\rangle$. The code for the CZ Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(2, 'q')
circuit = QuantumCircuit(qreg_q,name='CZ Gate')

circuit.cz(qreg_q[0],qreg_q[1]) #Apply the CZ gate

circuit.draw(output='mpl')#.savefig('../images/CZ_gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.18.

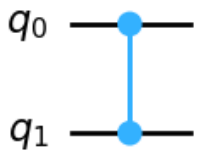


Figure 2.18: CZ Gate

In this figure 2.18, the first qubit is control and the second qubit is target. If the first qubit is $|1\rangle$, then Z gate is applied on the second qubit. The Z Gate is a gate which rotates the qubit by π around the Z axis of Bloch Sphere. Thus, if the input is $|0\rangle$, then the output is $|0\rangle$ and if the input is $|1\rangle$, then the output is $-|1\rangle$.

In the operator Notation it is:

$$\begin{aligned} CZ\,|00\rangle &= |00\rangle \\ CZ\,|01\rangle &= |01\rangle \\ CZ\,|10\rangle &= |10\rangle \\ CZ\,|11\rangle &= -|11\rangle \end{aligned}$$

It is represented by the matrix in the computational basis $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$ as:

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

In Outer product representation, it can be written as:

$$CZ = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| - |11\rangle\langle 11|$$

In Compact form we can write the CZ gate as:

$$CZ\,|xy\rangle = (-1)^{xy}\,|xy\rangle$$

Action of CZ on general two - qubit $|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ -\delta \end{bmatrix} = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle - \delta\,|11\rangle$$

CH/ Controlled Hadamard Gate

It is a two qubit gate, also called as the Controlled-Hadamard gate. Here the 1st Qubit is controlled and the 2nd Qubit is target. The truth table is as in table 2.14.

Input	Output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\frac{1}{\sqrt{2}}(\|10\rangle + \|11\rangle)$
$\|11\rangle$	$\frac{1}{\sqrt{2}}(\|10\rangle - \|11\rangle)$

Table 2.14: Truth Table for CH Gate

For the table in 2.14, the 1st Qubit is control and the 2nd Qubit is target. Thus, the CH gate applies Hadamard gate on the target qubit if the control qubit is $|1\rangle$. The code for the CH Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

# Add 2 qubits for CNOT gate demo
qreg_q = QuantumRegister(2, 'q')
circuit = QuantumCircuit(qreg_q,name='CY Gate')

circuit.ch(qreg_q[0],qreg_q[1]) #Apply the CH gate

circuit.draw(output='mpl')#.savefig('../images/ch-gate.png') #Draw the
    circuit
```

The circuit symbol is as shown in figure 2.19.

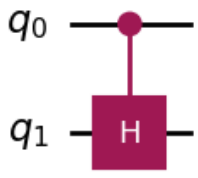


Figure 2.19: CH Gate

In this figure 2.19, the first qubit is control and the second qubit is target. If the first qubit is $|1\rangle$, then Hadamard gate is applied on the second qubit. The Hadamard Gate is a gate which rotates the qubit by π around the X axis of Bloch Sphere. Thus, if the input is $|0\rangle$, then the output is $|0\rangle$ and if the input is $|1\rangle$, then the output is $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

In the operator Notation it is:

$$\begin{aligned} CH\,|00\rangle &= |00\rangle \\ CH\,|01\rangle &= |01\rangle \\ CH\,|10\rangle &= \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle) \\ CH\,|11\rangle &= \frac{1}{\sqrt{2}}(|10\rangle - |11\rangle) \end{aligned}$$

It is represented by the matrix in the computational basis $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$ as:

$$CH = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

In Outer product representation, it can be written as:

$$CH = |00\rangle\langle 00| + |01\rangle\langle 01| + \frac{1}{\sqrt{2}}|10\rangle\langle 10| + \frac{1}{\sqrt{2}}|10\rangle\langle 11|$$

In Compact form we can write the CH gate as:

$$CH\,|xy\rangle = \left(\frac{1}{\sqrt{2}}\right)^x (|x(x \oplus y)\rangle + (-1)^y\,|xy\rangle)$$

Action of CH on general two - qubit $|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \frac{\gamma+\delta}{\sqrt{2}} \\ \frac{\gamma-\delta}{\sqrt{2}} \end{bmatrix} = \alpha\,|00\rangle + \beta\,|01\rangle + \frac{\gamma+\delta}{\sqrt{2}}\,|10\rangle + \frac{\gamma-\delta}{\sqrt{2}}\,|11\rangle$$

Important Note

The CX/CNOT, CY, CZ and CH gates are Unitary as well as Hermitian. In, general we can write any of the controlled gates where the first qubit acts as a control bit and the second qubit acts as a target bit in matrix form as show:

$$\begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix}$$

where I is the Identity matrix of size $2(2^{n-1}-1) \times 2(2^{n-1}-1)$ and U is the Unitary matrix of size 2×2. Now for forming any of the n-qubit control gates where there are n-1 control bits and 1 target bit we replace U with the corresponding gate wich we wish to form. For example, for 2-qubit Control Hadamard gate we replace U with the Hadamard gate matrix.

$$\begin{bmatrix} I & 0 \\ 0 & H \end{bmatrix}$$

Similarly, for any n-qubit control gate we can form the matrix by replacing U with the corresponding gate matrix. For example, for 3-qubit control Z gate we replace U with the Z gate matrix.

$$\begin{bmatrix} I & 0 \\ 0 & Z \end{bmatrix}$$

where I is the Identity matrix of size 6×6 and Z is the Pauli-Z gate matrix of size 2×2. We will later on use this concept to create the CCNOT/Toffoli gate.

SWAP Gate

It is a two Qubit Gate which swaps the states of the two qubits. The truth table is as in table 2.15.

Input	Output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|10\rangle$
$\|10\rangle$	$\|01\rangle$
$\|11\rangle$	$\|11\rangle$

Table 2.15: Truth Table for SWAP Gate

The code for the SWAP Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

# Add 2 qubits for CNOT gate demo
qreg_q = QuantumRegister(2, 'q')
circuit = QuantumCircuit(qreg_q,name='SWAP Gate')

circuit.swap(qreg_q[0],qreg_q[1]) #Apply the SWAP gate

circuit.draw(output='mpl')#.savefig('../images/swap-gate.png') #Draw
    the circuit
```

The circuit symbol is as shown in figure 2.20.

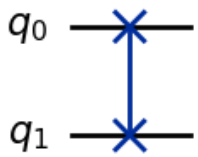


Figure 2.20: SWAP Gate

In the operator Notation it is:

$$SWAP\,|00\rangle = |00\rangle$$
$$SWAP\,|01\rangle = |10\rangle$$
$$SWAP\,|10\rangle = |01\rangle$$
$$SWAP\,|11\rangle = |11\rangle$$

It is represented by the matrix in the computational basis ($|00\rangle, |01\rangle, |10\rangle, |11\rangle$) as:

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

In Outer product representation, it can be written as:

$$SWAP = |00\rangle\langle 00| + |01\rangle\langle 10| + |10\rangle\langle 01| + |11\rangle\langle 11|$$

Action of SWAP on general two - qubit $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} \alpha \\ \gamma \\ \beta \\ \delta \end{bmatrix} = \alpha|00\rangle + \gamma|01\rangle + \beta|10\rangle + \delta|11\rangle$$

SWAP Gate can also be formed using CNOT gates as follows:

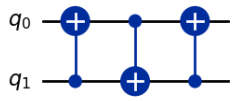


Figure 2.21: SWAP Gate using CNOT Gates

Or using CNOT gates as follows:

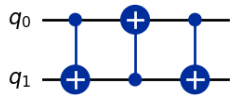


Figure 2.22: SWAP Gate using CNOT Gates

2.3.2 Three-qubit Gates

CCNOT/Toffoli Gate

It is a three qubit gate, also called as the CCNOT gate. Here the 1st and 2nd Qubits are controlled and the 3rd Qubit is target. The truth table is as in table 2.16.

Input	Output
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$\|111\rangle$
$\|111\rangle$	$\|110\rangle$

Table 2.16: Truth Table for Toffoli Gate

For the table in 2.16, the 1st and 2nd Qubits are control and the 3rd Qubit is target. Thus, the Toffoli gate flips the target qubit if the control qubits are $|1\rangle$. The code for the Toffoli Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(3, 'q')
circuit = QuantumCircuit(qreg_q)

circuit.ccx(qreg_q[0], qreg_q[1], qreg_q[2])

circuit.draw(output='mpl')#.savefig('../images/ccx-gate.png')
```

The circuit symbol is as shown in figure 2.23.

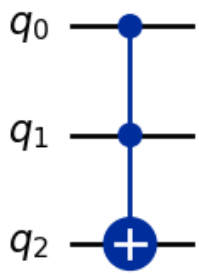


Figure 2.23: Toffoli Gate

In this figure 2.23, the first and second qubits are control and the third qubit is target. If the first and second qubits are $|1\rangle$, then the third qubit is flipped. The Toffoli Gate is a gate which flips the target qubit if the control qubits are $|1\rangle$.

In the operator Notation it is:

$$\begin{aligned}
CCX\,|000\rangle &= |000\rangle \\
CCX\,|001\rangle &= |001\rangle \\
CCX\,|010\rangle &= |010\rangle \\
CCX\,|011\rangle &= |011\rangle \\
CCX\,|100\rangle &= |100\rangle \\
CCX\,|101\rangle &= |101\rangle \\
CCX\,|110\rangle &= |111\rangle \\
CCX\,|111\rangle &= |110\rangle
\end{aligned}$$

It is represented by the matrix in the computational basis $(|000\rangle, |001\rangle, \ldots, |110\rangle, |111\rangle)$

as:

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

In Outer product representation, it can be written as:

$$\begin{aligned} CCX = &|000\rangle\langle 000| + |001\rangle\langle 001| + |010\rangle\langle 010| + |011\rangle\langle 011| \\ &+ |100\rangle\langle 100| + |101\rangle\langle 101| + |110\rangle\langle 111| + |111\rangle\langle 110| \end{aligned}$$

Action of CCX on general three - qubit $|\psi\rangle = \alpha|000\rangle + \beta|001\rangle + \gamma|010\rangle + \delta|011\rangle + \epsilon|100\rangle + \zeta|101\rangle + \eta|110\rangle + \theta|111\rangle$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \epsilon \\ \zeta \\ \eta \\ \theta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \epsilon \\ \zeta \\ \theta \\ \eta \end{bmatrix} = \alpha|000\rangle + \beta|001\rangle + \ldots + \zeta|101\rangle + \theta|110\rangle + \eta|111\rangle$$

CSWAP/Fredkin Gate

It is a three qubit gate, also called as the CSWAP gate. Here the 1st Qubit is control and the 2nd and 3rd Qubits are target. The truth table is as in table 2.17.

Input	Output
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|110\rangle$
$\|110\rangle$	$\|101\rangle$
$\|111\rangle$	$\|111\rangle$

Table 2.17: Truth Table for Fredkin Gate

For the table in 2.17, the 1st Qubit is control and the 2nd and 3rd Qubits are target. Thus, the Fredkin gate swaps the states of the 2nd and 3rd qubits if the control qubit is $|1\rangle$. The code for the Fredkin Gate is as follows:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(3, 'q')
circuit = QuantumCircuit(qreg_q)

circuit.cswap(qreg_q[0], qreg_q[2], qreg_q[1])

circuit.draw(output='mpl')#.savefig('../images/cswap-gate.png')
```

The circuit symbol is as shown in figure 2.24.

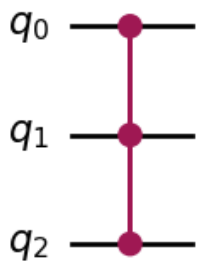


Figure 2.24: Fredkin Gate

In this figure 2.24, the first qubit is control and the second and third qubits are target. If the first qubit is $|1\rangle$, then the second and third qubits are swapped. The Fredkin Gate is a gate which swaps the states of the 2nd and 3rd qubits if the control qubit is $|1\rangle$.

In the operator Notation it is:

$$
\begin{aligned}
CSWAP\,|000\rangle &= |000\rangle \\
CSWAP\,|001\rangle &= |001\rangle \\
CSWAP\,|010\rangle &= |010\rangle \\
CSWAP\,|011\rangle &= |011\rangle \\
CSWAP\,|100\rangle &= |100\rangle \\
CSWAP\,|101\rangle &= |110\rangle \\
CSWAP\,|110\rangle &= |101\rangle \\
CSWAP\,|111\rangle &= |111\rangle
\end{aligned}
$$

It is represented by the matrix in the computational basis $(|000\rangle, |001\rangle, \ldots, |110\rangle, |111\rangle)$

as:

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

In Outer product representation, it can be written as:

$$\begin{aligned} CSWAP = &|000\rangle\langle 000| + |001\rangle\langle 001| + |010\rangle\langle 010| + |011\rangle\langle 011| \\ &+ |100\rangle\langle 100| + |101\rangle\langle 110| + |110\rangle\langle 101| + |111\rangle\langle 111| \end{aligned}$$

Action of CSWAP on general three - qubit $|\psi\rangle = \alpha|000\rangle + \beta|001\rangle + \gamma|010\rangle + \delta|011\rangle + \epsilon|100\rangle + \zeta|101\rangle + \eta|110\rangle + \theta|111\rangle$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \epsilon \\ \zeta \\ \eta \\ \theta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \epsilon \\ \zeta \\ \theta \\ \eta \end{bmatrix} = \alpha|000\rangle + \beta|001\rangle + \ldots + \zeta|101\rangle + \theta|110\rangle + \eta|111\rangle$$

We can construct Fredkin gate using CNOT and Toffoli gate as shown in the figure 2.25.

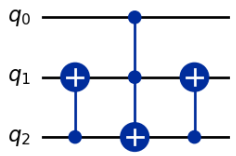


Figure 2.25: Fredkin Gate using CNOT and Toffoli Gates

Important Note

Applying a gate on a superposition state is the same as applying the gate on each of the basis states and then superposing the results. This is because, the gate is a linear operator and the superposition is a linear combination of the basis states. Thus, the gate can be applied on each of the basis states and then the results can be superposed. For example, Consider a superposition state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. If we apply a gate U on this state, then the output will be $U|\psi\rangle = U(\alpha|0\rangle + \beta|1\rangle) = \alpha U|0\rangle + \beta U|1\rangle$. Thus, the gate can be applied on each of the basis states and then the results can be superposed. This, can also be thought that since U is a 2×2 matrix and $|0\rangle$ and $|1\rangle$ are 2×1 column vectors, multiplying $U(\alpha|0\rangle + |1\rangle)$ is same as $\alpha U|0\rangle + \beta U|1\rangle$ Since Matrix multiplication is distributive.
For example, say CNOT gate needs to be applied on a superposition of two bits $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. Then the output will be $CNOT|\psi\rangle = CNOT(\frac{|00\rangle+|11\rangle}{\sqrt{2}}) = \frac{CNOT|00\rangle+CNOT|11\rangle}{\sqrt{2}}$. which will thus be $\frac{|00\rangle+|10\rangle}{\sqrt{2}}$. In the matrix form,

$$CNOT|\psi\rangle = CNOT\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right)$$

$$= \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right)$$

$$= (CNOT\frac{1}{\sqrt{2}}|00\rangle + CNOT\frac{1}{\sqrt{2}}|11\rangle) = \frac{1}{\sqrt{2}}CNOT|00\rangle + \frac{1}{\sqrt{2}}CNOT|11\rangle$$

$$= \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\\0\end{bmatrix}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

Thus, applying a gate on a superposition is the same as applying the gate on the basis states of the superposition. Thus, the gate can be applied on each of the basis states and then the results can be superposed in the same linear combination as that in which the state $|\psi\rangle$ was superposed.

2.4 Quantum Circuits

Example 2.4.1. Consider the quantum circuit as shown in figure 2.26.

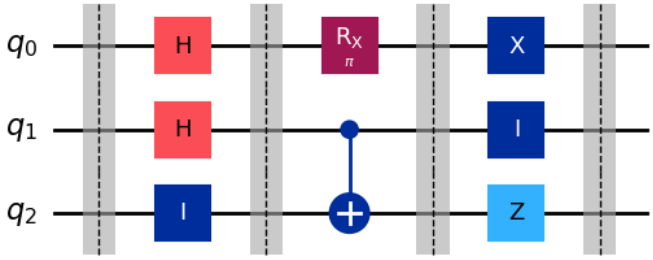


Figure 2.26: Quantum Circuit

1. Make Reverse circuit of the given circuit.
2. Write the output of the circuit for the input $|010\rangle$.
3. Write complete circuit as Unitary Matrix.
4. Write reverse circuit as Unitary Matrix.
5. Compute output using Unitary matrix of circuit.

1. Note that excpet for the rotation gate all the other gates are their own inverses. Thus, the reverse circuit is as shown in figure 2.27.

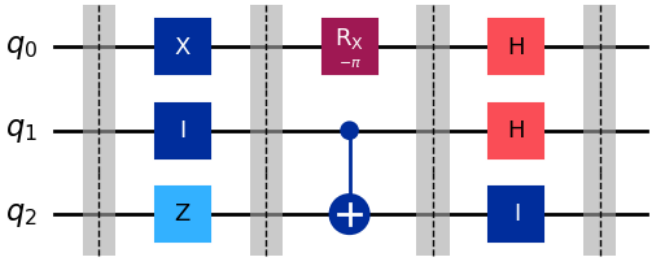


Figure 2.27: Reverse Quantum Circuit

If we place the two circuits side by side, we will get the input back. Thus it will act like an Identity matrix.

2. As shown in the figure 2.26 the circuit has been divded into parts for easy computation. At the first barrier, the input is $|010\rangle$. After applying the gates, at the second stage the output will be:

$$\begin{aligned}(H \otimes H \otimes I)(|010\rangle) &= H|0\rangle \otimes H|1\rangle \otimes I|0\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |0\rangle \\ &= \frac{1}{2}(|000\rangle - |010\rangle + |100\rangle - |110\rangle)\end{aligned}$$

After this, the third stage requires the application of the RX gate on the first qubit and CNOT gate with 2nd qubit as control and 3rd qubit as target. The output after this stage will be:

$$\begin{aligned}(RX \otimes CX)(\frac{1}{2}(|000\rangle - |010\rangle + |100\rangle - |110\rangle)) &= \frac{1}{2}(RX|0\rangle \otimes CX|00\rangle - RX|0\rangle \otimes CX|10\rangle \\ &\quad + RX|1\rangle \otimes CX|00\rangle - RX|1\rangle \otimes CX|10\rangle) \\ &= \frac{|100\rangle - |111\rangle - |000\rangle + |011\rangle}{2}\end{aligned}$$

Here the action of RXx is $RX|0\rangle = |1\rangle$, $RX|1\rangle = -|0\rangle$. Now we apply the last stage which involves applying X gate on the first qubit and Z gate on the 3rd qubits.

$$\begin{aligned}(X \otimes I \otimes Z)\frac{|100\rangle - |111\rangle - |000\rangle + |011\rangle}{2} &= \frac{1}{2}(X|1\rangle \otimes I|0\rangle \otimes Z|0\rangle - X|1\rangle \otimes \\ &\quad I|1\rangle \otimes Z|1\rangle - X|0\rangle \otimes I|0\rangle \otimes Z|0\rangle + X|0\rangle \otimes I|1\rangle \\ &\quad \otimes Z|1\rangle) \\ &= \frac{1}{2}(|000\rangle + |011\rangle - |100\rangle - |111\rangle)\end{aligned}$$

Thus, the state of the qubits after passing through the circuit. In the matrix

form, it can be written as:

$$\frac{1}{2}\begin{bmatrix}1\\0\\0\\1\\-1\\0\\0\\-1\end{bmatrix}$$

3. In order to find the Unitary matrix for the complexte circuit, we are required to find the unitary matrices corresponding to the individual stages and multiply them to get the final Unitary Matrix. For the first stage, we have two Hadamard gates acting on first and second qubit and an Identity gate. Thus, we are required to find,

$$\begin{aligned} S_1 &= H \otimes H \otimes I \\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \end{aligned}$$

Thus, doing this the overall matrix will be:

$$S_1 = \frac{1}{2}\begin{bmatrix}1 & 0 & 1 & 0 & 1 & 0 & 1 & 0\\0 & 1 & 0 & 1 & 0 & 1 & 0 & 1\\1 & 0 & -1 & 0 & 1 & 0 & -1 & 0\\0 & 1 & 0 & -1 & 0 & 1 & 0 & -1\\1 & 0 & 1 & 0 & -1 & 0 & -1 & 0\\0 & 1 & 0 & 1 & 0 & -1 & 0 & -1\\1 & 0 & -1 & 0 & -1 & 0 & 1 & 0\\0 & 1 & 0 & -1 & 0 & -1 & 0 & 1\end{bmatrix}$$

For stage 2, we are required to find,

$$\begin{aligned} S_2 &= RX \otimes CX \\ &= \begin{bmatrix}0 & -1\\1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix} \end{aligned}$$

In the matrix form, we get:

$$S_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For stage 3, we are required to find,

$$\begin{aligned} S_3 &= X \otimes I \otimes Z \\ &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

Finally, we are required to calculate the product of the three matrices to get the overall matrix.

$$\begin{aligned} U &= S_3 S_2 S_1 \\ &= \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

4. The Unitary matrix for the reverse circuit will be the Transpose conjugate of the given circuit i.e. the transpose conjugate of the matrix found in the

previous part.

$$U^\dagger = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & -1 & -1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 & 0 & -1 & -1 & 0 \end{bmatrix}$$

Thus, the matrix for the reverse circuit.

5. The output of the circuit can be found by multiplying the input with the Unitary matrix of the circuit.

$$U\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

which can be verified as the same output from the previous part.

Chapter 3

Entanglement

Given a state of multiple entangled qubit, one cannot express individual qubit seperately. For example, consid the state $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$, cannot be expressed as some $|\phi\rangle \otimes |\chi\rangle$, $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\chi\rangle = \gamma|0\rangle + \delta|1\rangle$ i.e. no value of $\alpha, \beta, \gamma, \delta$ exists.The n the states are called entangled.

On the contrary some states can be seperated into individual qubit states, such states are called separable states. For example, consider the state $|\psi\rangle = \frac{|00\rangle+|01\rangle}{\sqrt{2}}$, can be expressed as $|0\rangle \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, here $\alpha = 1, \beta = 0, \gamma = \frac{1}{\sqrt{2}}, \delta = \frac{1}{\sqrt{2}}$.

Given an Entangled state of multiple qubit, measuring any qubit individuall reveals all other qubits. For example, conisder $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$, thus suppose we measure only one qubit, i.e. say we measure the first qubit. Then the probability of the first qubit being $|0\rangle$ is $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$ and the probability of the first qubit being $|1\rangle$ is $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$. Now suppose we find that the first qubit is $|0\rangle$, then the state of the quantum system after measurement will be $|00\rangle$, thus the state of the second qubit will be $|0\rangle$. Similarly, if we find that the first qubit is $|1\rangle$, then the state of the quantum system after measurement will be $|11\rangle$, thus the state of the second qubit will be $|1\rangle$. This, we can see that measuring one qubit reveals the state of the other qubit. This is called entanglement.

But, in case of seperable qubits measuring one qubit will not reveal the state of the second qubit. For example, consider the state $|\psi\rangle = \frac{|00\rangle+|01\rangle}{\sqrt{2}}$, if we measure the first qubit, then the probability of the first qubit being $|0\rangle$ is $|\frac{1}{\sqrt{2}}|^2 + |\frac{1}{\sqrt{2}}|^2 = 1$ and the probability of the first qubit being $|1\rangle$ is 0. Thus, if we find that the first qubit is $|0\rangle$, then the state of the quantum system after measurement will be $\frac{|00\rangle+|01\rangle}{\sqrt{2}}$, thus the state of the second qubit will be $|0\rangle$ with probability $\frac{1}{2}$ and will be in state $|1\rangle$ with probability $\frac{1}{2}$. Thus, the state of the second qubit cannot be revealed. The

measurement on the first qubit does not reveal any information of the state of the second qubit.

3.1 Bell States/EPR Pairs

We have four Bell states, which are also called as EPR states. These are:

$$|\Phi^+\rangle = |\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$
$$|\Phi^-\rangle = |\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$
$$|\Psi^+\rangle = |\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$
$$|\Psi^-\rangle = |\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

The general formula for Bell states is:

$$|\beta_{xy}\rangle = \frac{|0y\rangle + (-1)^x |1\bar{y}\rangle}{\sqrt{2}}$$

where $\bar{y}$ is the negation of y. Thus, we can find any Bell state using the above formula for any given input $x \in \{0, 1\}$ and $y\ in \{0, 1\}$. We can create these bell states using quantum circuits as shown in figure 3.1.

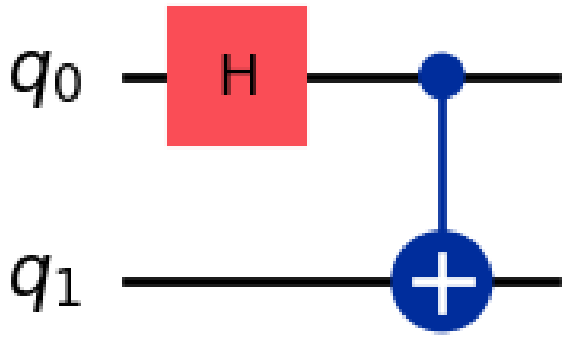


Figure 3.1: Bell States

For example, suppose we wish to apply the circuit on state $|01\rangle$. Then the state after applying the operation $H \otimes I$ will be $\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes |1\rangle = \frac{|01\rangle+|11\rangle}{\sqrt{2}}$. Similarly, the state after applying the operation CX will be $\frac{|01\rangle+|10\rangle}{\sqrt{2}} = \beta_{01}$.

Example 3.1.1. Given the state $|\psi\rangle = \frac{|01\rangle-|10\rangle}{\sqrt{2}}$, find whether the state is entangled or not. We proove this by contradiction, Assume that $|\psi\rangle$ can be written seperately as $\alpha|0\rangle + \beta|1\rangle \otimes \gamma|0\rangle + \delta|1\rangle$. Thus, we get,

$$\begin{aligned}\frac{|01\rangle - |10\rangle}{\sqrt{2}} &= \alpha|0\rangle + \beta|1\rangle \otimes \gamma|0\rangle + \delta|1\rangle \\ &= \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle\end{aligned}$$

Comparing the equations we have,

$$\begin{aligned}\alpha\gamma &= 0\\ \alpha\delta &= \frac{1}{\sqrt{2}}\\ \beta\gamma &= -\frac{1}{\sqrt{2}}\\ \beta\delta &= 0\end{aligned}$$

No solution exists for the above equations, thus the given state is entangled as it cannot be written as a tensor product of two states.

Entanglement has applications in Teleportation and Superdense Coding which will be discussed next. To get back the original input state from the Bell states we can use the following Circuit.

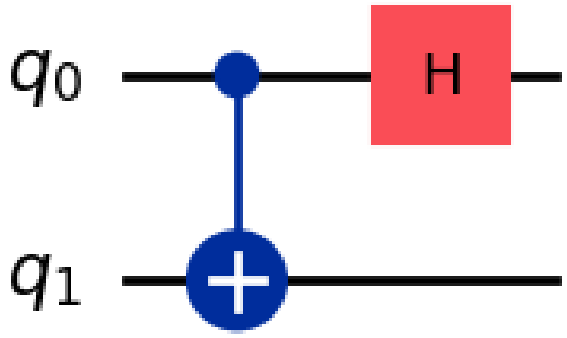


Figure 3.2: Reverse Bell States

Here, we have used a reverse CNOT gate i.e. inverse of CNOT gate which is the CNOT gate since it's a self-inverse gate (Hermitian and Unitary). Then we use the

reverse Hadamard gate which is Hadamard Gate on the first qubit and the circuit will give back the original input state. (Hadmard gate is also self-inverse).

3.2 No Cloning Theorem

One of the counter-intuitive consequences of the laws of Quantum Mechanics is that it is impossible to create an exact copy of an arbitrary unknown quantum state. This is called the No Cloning Theorem. More formally,

Theorem 3.2.1. *Given an arbitrary unknown quantum state $|\psi\rangle$, it is impossible to create an exact copy of the state $|\psi\rangle$ i.e. create the state $|\psi\rangle|\psi\rangle$. More precisely, it is impossible to start with two qubits in the state $|\phi\rangle \otimes |0\rangle$ and transform them to the state $|\phi\rangle \otimes |\phi\rangle$.*

Proof. From the Postulate 2: Evolution Postulate given in section 1.1.2, from Axiom of Quantum Mechanics, for this to be possible there must be a Unitary transformation U such that $U|\phi\rangle \otimes |0\rangle = |\phi\rangle \otimes |\phi\rangle$. Thus, in order to prove that it is impossible to do so, it suffices to prove that no such unitary transformation exists. We start with proving that no such unitary transformation exists that can achieve this simultaneously for two states $|\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ and $|\psi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$.

Recall that a unitary transformation is a rotation of the Hilbert Space, and therefore necessarily preserves inner products. To be more precise, consider two quantum states:

$$|\phi\rangle \otimes |0\rangle = \alpha_0|00\rangle + \alpha_1|10\rangle$$

and

$$|\psi\rangle \otimes |0\rangle = \beta_0|00\rangle + \beta_1|10\rangle$$

Then, the inner product between them is given by

$$\langle\phi 0|\psi 0\rangle = \alpha_0^*\beta_0 + \alpha_1^*\beta_1 = \langle\phi|\psi\rangle$$

Now consider the quantum states,

$$\begin{aligned}|\phi\rangle \otimes |\phi\rangle &= (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\alpha_0|0\rangle + \alpha_1|1\rangle) \\ &= \alpha_0^2|00\rangle + \alpha_0\alpha_1|01\rangle + \alpha_1\alpha_0|10\rangle + \alpha_1^2|11\rangle\end{aligned}$$

and

$$\begin{aligned}|\psi\rangle \otimes |\psi\rangle &= (\beta_0|0\rangle + \beta_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle) \\ &= \beta_0^2|00\rangle + \beta_0\beta_1|01\rangle + \beta_1\beta_0|10\rangle + \beta_1^2|11\rangle\end{aligned}$$

Their inner product is

$$\begin{aligned}\langle\phi\phi|\psi\psi\rangle &= (\langle 00|\,\alpha_0^{*2} + \langle 01|\,\alpha_0^*\alpha_1^* + \langle 10|\,\alpha_1^*\alpha_0^* + \langle 11|\,\alpha_1^{*2})(\beta_0^2\,|00\rangle + \beta_0\beta_1\,|01\rangle + \beta_1\beta_0\,|10\rangle + \beta_1^2\,|11\rangle) \\ &= \alpha_0^{*2}\beta_0^2 + \alpha_0^*\alpha_1^*\beta_0\beta_1 + \alpha_1^*\alpha_0\beta_1\beta_0 + \beta_0^2\beta_1^2 \\ &= (\alpha_0^*\beta_0 + \alpha_1^*\beta_1)^2\end{aligned}$$

Now let us assume that we have a Unitary operator U and the two arbitrary quantum states $|\phi\rangle$ and $|\psi\rangle$ such that the action of U is defined as follows:

$$|\phi\rangle \otimes |0\rangle \xrightarrow{U} |\phi\rangle \otimes |\phi\rangle$$

$$|\psi\rangle \otimes |0\rangle \xrightarrow{U} |\psi\rangle \otimes |\psi\rangle$$

Thus, clearly operator U corresponds to an operator which performs copying arbitrary quantum states. Since we are performing the same unitary transformation on both the states $|\phi\rangle$ and $|\psi\rangle$, thus the inner product before and after transformation must be the same, as a Unitary transformation corresponds to rotations in the complex Hilbert space. Their inner product before transformation is

$$\langle\phi 0|\psi 0\rangle = \alpha_0^*\beta_0 + \alpha_1^*\beta_1$$

Their inner product after the transformation will be:

$$\langle\phi\phi|\psi\psi\rangle = (\alpha_0^*\beta_0 + \alpha_1^*\beta_1)^2 = \langle\phi 0|\psi 0\rangle^2 = \langle\phi|\psi\rangle^2$$

Thus, before transformation we had the inner product as $\langle\phi|psi\rangle$ and after the transformation we have the inner product as $\langle\phi|\psi\rangle^2$. Thus, in general, $\langle\phi|\psi\rangle \neq \langle\phi|\psi\rangle^2$ (It is equal only in the case when $\langle\phi|\psi\rangle = 0$ or 1 i.e. both the quantum states are orthonormal). Since, this is not true for any two arbitrary quantum states, thus our assumption that such a Unitary transformation U exists is wrong. Thus, by contradiction, we have proved that such a Unitary transformation U does not exists, hence, it is impossible to copy an arbitrary unknown quantum state $|\phi\rangle$ or in other words, to be more precise, no such unitary transformation U exists which can perform the following transformation of $|\phi\rangle \otimes |0\rangle \rightarrow |\phi\rangle \otimes |\phi\rangle$. □

3.3 Applications of Entanglement

3.3.1 Quantum Teleportation

Alice and Bob are two friends. They get together and create a Bell state β_{00} and then distribute the quantum system. One qubit stays with Alice (say $|\beta_A\rangle$) and the

other qubit goes to Bob (say $|\beta_B\rangle$). Now Bob moves 100,000 light years away. Now, say Alive wishes to send Bob an Arbitrary state $|\psi\rangle$ to Bob.

The teleportation protocol is as follows:

1. Alice and Bob create a Bell state β_{00}. It is as shown in the figure 3.3 which has both the inputs $q_0 = |0\rangle$ and $q_1 = |0\rangle$.

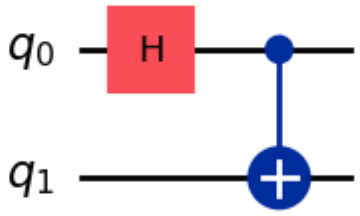


Figure 3.3: Creating Bell States

   The state of the circuit after applying this state is the Bell state which is given as $\beta_{00} = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Now, say the outcome of the first bit be $|\beta_A\rangle$ which is the state of the first qubit goes to Alice and the outcome of the second bit be $|\beta_B\rangle$ which is the state of the second qubit which goes to Bob.

2. Alice applies a CNOT gate with the input state $|\psi\rangle$ as the control qubit and the state $|\beta_A\rangle$ as the target qubit. Then Alice applies a Hadamard gate on the input state $|\psi\rangle$. This is as shown in the figure 3.4 with $q_0 = |\psi\rangle$ and $q_1 = |\beta_A\rangle$.

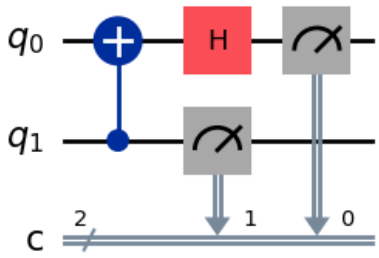


Figure 3.4: Alice's Operations

Thus the state here, is $|\psi\rangle |\beta_{00}\rangle$. Let the state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$. Thus,

$$|\psi\rangle |\beta_{00}\rangle = (\alpha |0\rangle + \beta |1\rangle) \otimes \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{\alpha |000\rangle + \alpha |011\rangle + \beta |100\rangle + \beta |111\rangle}{\sqrt{2}}$$

Now applying CNOT with first bit as control bit and the second bit as the targt bit we get,

$$\frac{\alpha |000\rangle + \alpha |011\rangle + \beta |110\rangle + \beta |101\rangle}{\sqrt{2}}$$

Now, applying Hadamard gate on the first bit we get,

$$\frac{\alpha |000\rangle + \alpha |100\rangle + \alpha |011\rangle + \alpha |111\rangle + \beta |010\rangle - \beta |110\rangle + \beta |001\rangle - \beta |101\rangle}{2}$$

We further simplify this by considering only the first two qubits, we get,

$$\frac{1}{2} [|00\rangle (\alpha |0\rangle + \beta |1\rangle) + |01\rangle (\alpha |1\rangle + \beta |0\rangle) + |10\rangle (\alpha |0\rangle - \beta |1\rangle) + |11\rangle (\alpha |1\rangle - \beta |0\rangle)]$$

Now Alice does measurement of the first two qubits. Remember that the qubit $|\psi\rangle$ is already normalized i.e. $|\alpha|^2 + |\beta|^2 = 1$. The probability of the first two qubits being $|00\rangle$ is $|\frac{\alpha}{2}|^2 + |\frac{\beta}{2}|^2 = \frac{|\alpha|^2+|\beta|^2}{4} = \frac{1}{4}$, similarly, the probability of the first two qubits being $|10\rangle$ is $\frac{1}{4}$, the probability of the first two qubits being $|01\rangle$ is $\frac{1}{4}$ and the probability of the first two qubits being $|11\rangle$ is $\frac{1}{4}$.

Suppose upon measuring the two qubits by Alice they collapsed to $|00\rangle$, then the state of the system will be:

$$\alpha|000\rangle + \beta|001\rangle = |00\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)$$

Suppose upon measuring the two qubits by Alice they collapsed to $|10\rangle$, then the state of the system will be:

$$\alpha|100\rangle - \beta|101\rangle = |10\rangle \otimes (\alpha|0\rangle - \beta|1\rangle)$$

Suppose upon measuring the two qubits by Alice they collapsed to $|01\rangle$, then the state of the system will be:

$$\alpha|011\rangle + \beta|010\rangle = |01\rangle \otimes (\alpha|1\rangle + \beta|0\rangle)$$

Suppose upon measuring the two qubits by Alice they collapsed to $|11\rangle$, then the state of the system will be:

$$\alpha|111\rangle - \beta|110\rangle = |11\rangle \otimes (\alpha|1\rangle - \beta|0\rangle)$$

All of the above are possible with a probability of $\frac{1}{4}$.

3. Now Alice sends the result of the measurement to Bob. Bob applies the necessary operations to get the original state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

   - Suppose Alice measured the first two qubits to be $|00\rangle$, then Bob applies no operation.

     $$\alpha|000\rangle + \beta|001\rangle = |00\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)$$

   - Suppose Alice measured the first two qubits to be $|10\rangle$, then Bob applies the Z gate on his qubit to get the original state.

     $$Z(\alpha|100\rangle - \beta|101\rangle) = \alpha|100\rangle + \beta|101\rangle = |10\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)$$

   - Suppose Alice measured the first two qubits to be $|01\rangle$, then Bob applies the X gate on his qubit to get the original state.

     $$X(\alpha|011\rangle + \beta|010\rangle) = \alpha|010\rangle + \beta|011\rangle = |01\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)$$

   - Suppose Alice measured the first two qubits to be $|11\rangle$, then Bob applies the X and Z gate on his qubit to get the original state.

     $$Z(X(\alpha|111\rangle - \beta|110\rangle)) = \alpha|110\rangle + \beta|111\rangle = |11\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)$$

Thus, we have teleported the unkown state $|\psi\rangle$ from Alice to Bob. Note that during this protocol, the Bell state of the qubits is destroyed upon measurement by Alice and thus, cannot be done again.

The final Teleportation circuit is as shown in the figure 3.5.

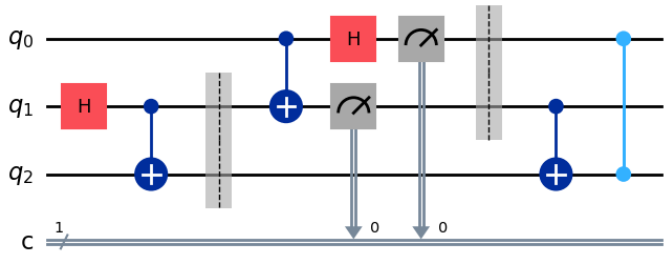


Figure 3.5: Teleportation Circuit

We can see from the figure that the measurement ouput from the Alice's qubits act as control bits for appling Z and X gate. If the first qubit is 1 then we apply the Z gate, if the second qubit is 1 then we apply the X gate. Based on the Alice's output we can decide whether to apply Thus the circuit shown in the figure 3.5, show that after Alice's measurement the bits act as control bits for the operations to be applied by Bob.

3.3.2 Superdence Coding

In Superdense coding we can send from one place to another two classical bits but using only one qubit. Alice and Bob create a bell state β_{00} and then distribute the qubits. Alice takes the first qubit and Bob takes the second qubit. Bob now moves light years away. Alice now wishes to send two classical bits to Bob. The protocol is as follows:

1. **Preparing Bell States:** Alice and Bob create a Bell state β_{00}. The state of the circuit after applying this state is the Bell state which is given as $\beta_{00} = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Now, say the first bit be $|\beta_A\rangle$ which is the qubit that goes to Alice and the second bit be $|\beta_B\rangle$ which is the qubit which goes to Bob.

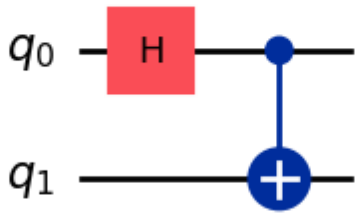


Figure 3.6: Creating Bell States

This is how the Bell state is created shown in the figure 3.6.

2. **Alice Encoding:** Alice applies the necessary operations to send the two classical bits to Bob. These operations are done on the first qubit which is with Alice as follows in the figure 3.7.

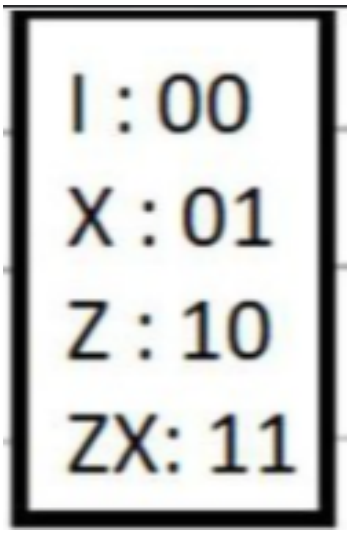


Figure 3.7: Alice's Encoding

- If Alice wishes to send 00, then she applies no operation.

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

- If Alice wishes to send 01, then she applies the Pauli - X gate on her portion of Bell state.

$$X\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \frac{|10\rangle + |01\rangle}{\sqrt{2}}$$

- If Alice wishes to send 10, then she applies the Pauli-Z gate on the qubit.

$$Z\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

- If Alice wishes to send 11, then she applies the Pauli-X and Pauli-Z gate on the qubit.

$$X\left(Z\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right)\right) = \frac{|10\rangle - |01\rangle}{\sqrt{2}}$$

3. **Bob Decoding:** NOw Alice sends her portion of Bell state qubit to Bob. Bob then applies the necessary operations to get the two classical bits. Bob performs the following operations on the qubit he received from Alice: He applies CNOT gate with the first qubit (Qubit sent by Alice) as the control qubit and the second qubit as the target qubit. Then he applies Hadamard gate on the first qubit (Qubit sent by Alice). Then she performs the measurement. The measurement will give the two classical bits that Alice wished to send. This is as shown in the figure 3.8.

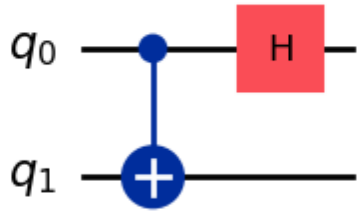


Figure 3.8: Bob's Decoding

- If Alice wished to send 00, then the state of the system is currently $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Thus, after applying the CNOT gate,

$$CNOT\left(\frac{|00\rangle+|11\rangle}{\sqrt{2}}\right)=\frac{|00\rangle+|10\rangle}{\sqrt{2}}$$

  Now, after applying the Hadamard Gate on the first qubit we get,

$$H\left(\frac{|00\rangle+|10\rangle}{\sqrt{2}}\right)=\frac{|00\rangle+|10\rangle+|00\rangle-|10\rangle}{2}=|00\rangle$$

  Thus, on measurement, the two classical bits are 00.

- If Alice wished to send 01, then the state of the system is currently $\frac{|10\rangle+|01\rangle}{\sqrt{2}}$. Thus, after applying the CNOT gate,

$$CNOT\left(\frac{|10\rangle+|01\rangle}{\sqrt{2}}\right)=\frac{|11\rangle+|01\rangle}{\sqrt{2}}$$

  Now, after applying the Hadamard Gate on the first qubit we get,

$$H\left(\frac{|11\rangle+|01\rangle}{\sqrt{2}}\right)=\frac{|01\rangle-|11\rangle+|01\rangle+|11\rangle}{2}=|01\rangle$$

- If Alice wished to send 10, then the state of the system is currently $\frac{|00\rangle-|11\rangle}{\sqrt{2}}$. Thus, after applying the CNOT gate,

$$CNOT\left(\frac{|00\rangle-|11\rangle}{\sqrt{2}}\right)=\frac{|00\rangle-|10\rangle}{\sqrt{2}}$$

  Now, after applying the Hadamard Gate on the first qubit we get,

$$H\left(\frac{|00\rangle-|10\rangle}{\sqrt{2}}\right)=\frac{|00\rangle+|10\rangle-|00\rangle+|10\rangle}{2}=|10\rangle$$

- If Alice wished to send 11, then the state of the system is currently $\frac{|10\rangle-|01\rangle}{\sqrt{2}}$. Thus, after applying the CNOT gate,

$$CNOT\left(\frac{|10\rangle-|01\rangle}{\sqrt{2}}\right)=\frac{|11\rangle-|01\rangle}{\sqrt{2}}$$

  Now, after applying the Hadamard Gate on the first qubit we get,

$$H\left(\frac{|11\rangle-|01\rangle}{\sqrt{2}}\right)=\frac{|01\rangle-|11\rangle-|01\rangle-|11\rangle}{2}=-|11\rangle$$

  Thus, upon measurement the two classical bits are 11.

The final Superdense coding circuit is as shown in the figure 3.9.

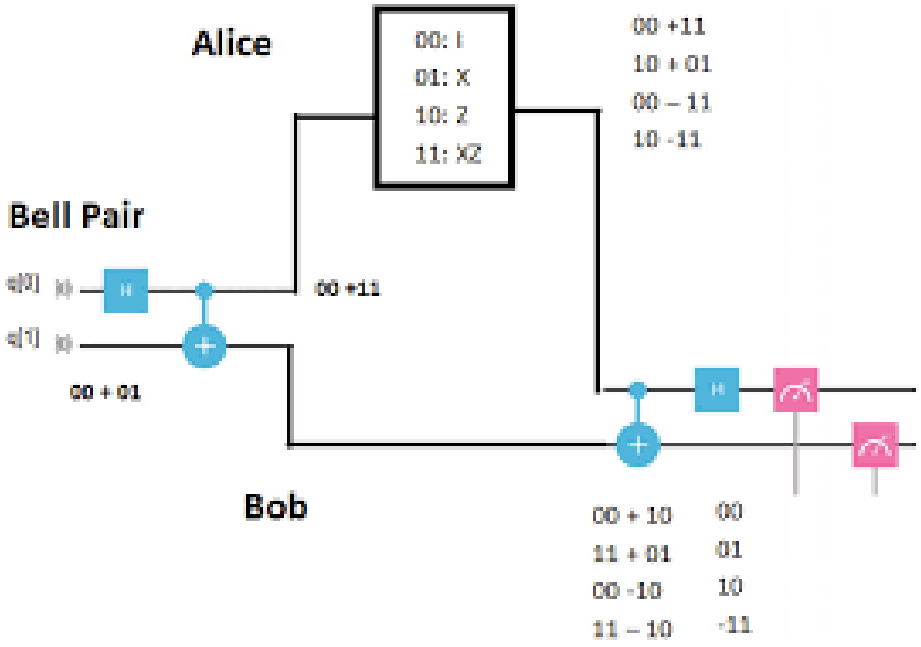


Figure 3.9: Superdense Coding Circuit

Chapter 4

Quantum Algorithms

The following are the Quantum Algorithms are discussed in this section:

1. Elitzur-Vaidman Bomb Detection Algorithm
2. Deutsch's Algorithm
3. Deutsch-Josza Algorithm
4. Simon's Algorithm
5. Shor's Algorithm
6. Bernstein-Vazirani Algorithm
7. Grover's Algorithm
8. Quantum Fourier Transform
9. Quantum Phase Estimation

In order to understand Quantum Algorithms we must understand the Oracles and their working.

4.1 Oracles

Consider a classical circuit representing a classical function f. Consider the classical function $f : \{0,1\}^n \to \{0,1\}^m$ which takes n bits as input and gives m bits as output. **We can represent this classical function using a classical circuit**. (For the

proof of this please refer to Applendix B.2) Thus, any classical boolean function $f : \{0,1\}^n \to \{0,1\}^m$ can be represented using a classical circuit as shown in the figure 4.1.

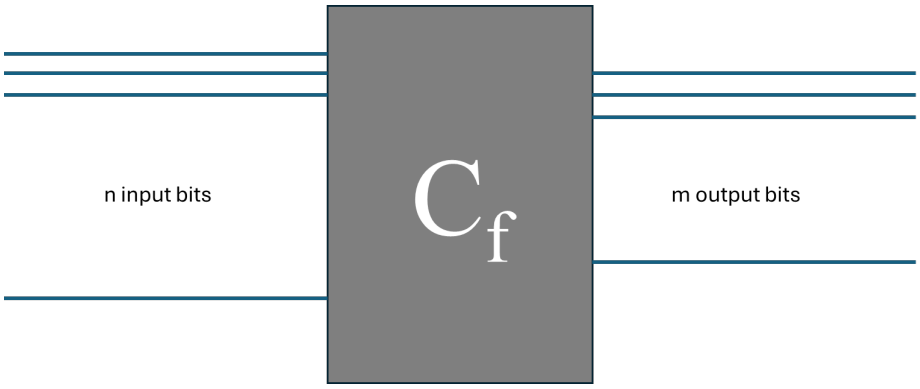


Figure 4.1: Classical Circuit

But we are interested in Quantum Circuits. Thus, we are required to convert this classical circuit into a Quantum Circuit. Recall, that a Quantum Circuit is a sequence of Quantum Gates. Since all the operations using quantum gates done on qubits are untiary opertaions (thus, reversible), and thus the Quantum circuit can be written as a product of Unitary quantum gates which is a Unitary matrix (product of unitary matrices is a unitary matrix). Thus, the Quantum Circuit is itself a Unitary Gate and hence reversible. Thus, the Quantum circuits are reversible. Note that here the meaning of reversibility is that given the output of the circuit we can find the input of the circuit. This is because the Quantum Gates are Unitary and thus reversible. Thus, a Gate is said to be reversible if given the output of the gate we can find the input of the gate.

But, a classical circuit need not be necessarily reversible. Thus, we are required to convert the classical circuit into a reversible circuit in order to implement it on a Quantum Computer as a Quantum Circuit (Unitary Gates i.e. hence always reversible). Thus, we are required to transform the classical circuit in order to make it reversible in order to implement it as a Unitary matrix made up of Quantum Gates (reversible since $U_f^\dagger = U_f^{-1}$).

This, can be done by transforming gates in a classical universal gate set to corresponding reversible quantum gates. Thus, then can we implement any classical circuit on a quantum computer. Note that a universal gate set is the set of gates which can be used to implement any classical circuit. There can be multiple gate set which form the universal gate sets. For example, the $\{NAND\}$ gate is a universal

gate set, $\{AND, NOT\}$ is another universal gate set. Thus, we can create any classical circuit using only the gates in the universal gate set. Now, if we are able to show that the gates in the universal gate set can be transformed into reversible quantum gates, then we can implement any classical circuit on a quantum computer using the corresponding reversible quantum gates. We can convert any classical circuit in terms of the universal gate sets and then replace the gates in the universal gate set with their corresponding reversible quantum gates to get the quantum circuit.

For the consideration let us take the Classical universal gate set $\{AND, NOT\}$. Here, NOT gate (refer Appendix B.1.1) is already reversible becuase if we know the output of the NOT gate we can find the input. For example, if the output of the NOT gate is 1 then we know that the input was 0 and if the output of the NOT gate is 0 then we know that the input was 1. This is because NOT is a single input single output gate (number of inputs=number of outputs) which negates the input. Thus, for the two inputs 0 and 1 we get the outputs 1 and 0 respectively. Thus, for finding the input from the given output we can simply negate that output to get the input (negation twice cancels out). Hence, the NOT gate is already a reversible gate. It's equivalent Quantum Gate is the Pauli-X gate which is also reversible. Thus, we can replace the NOT gate in the classical circuit with the Pauli-X gate in the Quantum Circuit. Recall that the Pauli-X Gate is $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and its action on $|0\rangle$ and $|1\rangle$ are $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$ respectively. Thus, the Pauli-X gate is reversible and the quantum equivalent of the Classical NOT Gate.

Now, we are left with the AND gate. The AND gate is a two input single output gate i.e. It takes two inputs and gives one output. The output is 1 if and only if both the inputs are 1, else the output is 0 (refer Appendix B.1.2). Thus, for the case when the output is 0 the input could be either 00 or 01 or 10 but not 11. For the output of 1 we definitely know that the input was 11. But, for the output of 00 we cannot determine the input. Thus, the AND gate is not reversible. Thus, we are required to convert the AND gate into a reversible gate in order to implement it on a Quantum Computer. Thus, in order to create a reversible AND gate consider the control-Swap Gate (refer section 2.3.2). It takes three different inputs say $|x\rangle$, $|y\rangle$ and $|z\rangle$. Say x acts as the control bit and y and z are the target bits. This, is as shown in the figure 4.2

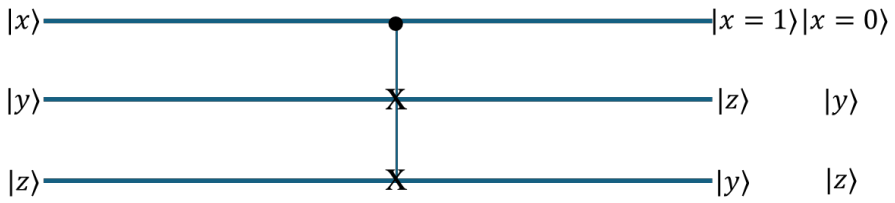


Figure 4.2: Control Swap Gate

Thus, if $x = 1$ then the state of the system is swapped i.e. $|y\rangle$ and $|z\rangle$ are swapped, thus the output is $|x\rangle, |z\rangle, |y\rangle$. If $x = 0$ then the state of the system remains the same $|x\rangle, |y\rangle, |z\rangle$.

Now in order to create reversible AND gate using the control-Swap gate we can do the following. Set the $|z\rangle = |0\rangle$. Then, the output of the control-Swap gate will be $|x\rangle, |0\rangle, |y\rangle$ if $x = 1$ and $|x\rangle, |y\rangle, |0\rangle$ if $x = 0$. This, output can also be written in a more compact form as $|x\rangle, (|0\rangle, |y\rangle), |x \wedge y\rangle$ where $\wedge$ is the AND operation. The second qubit can be either $|0\rangle$ or $|y\rangle$ depending on the value of x. If $x = 1$ then the second qubit will be $|y\rangle$ and if $x = 0$ then the second qubit will be $|0\rangle$. This, output of the second qubit is refered to as junk which is a function of x i.e. $junk(x)$. This is as shown in the figure 4.3.

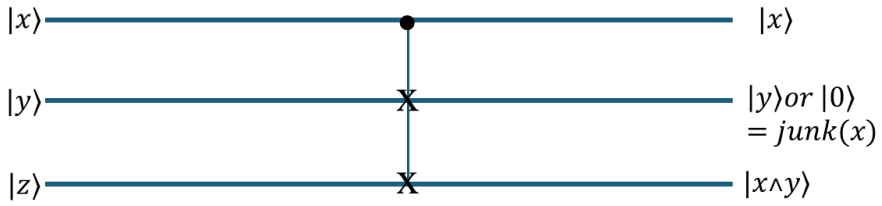


Figure 4.3: Reversible AND Gate

The truth table for the above circuit is as shown in the table 4.1.

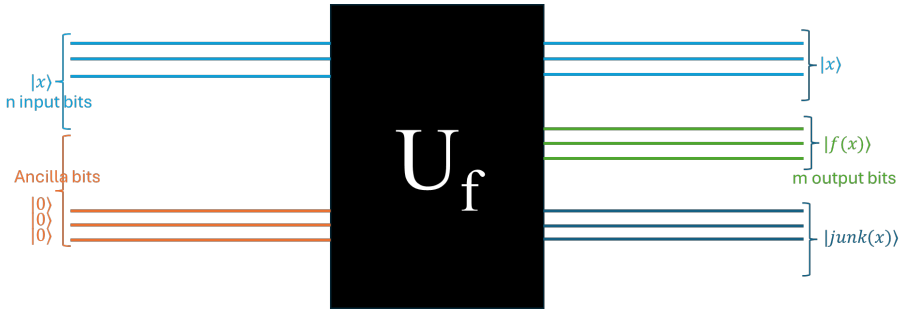


Figure 4.4: Quantum Circuit

x	y	0	$junk(x)$	$x \wedge y$
0	0	0	0	0
0	1	0	1	0
1	0	0	0	0
1	1	0	1	1

Table 4.1: Reversible AND Gate

Hence, we can clearly see that the output of the circuit is the same as the output of the AND gate. Thus, the above circuit is the reversible AND gate. Thus, using CSWAP gate as shown above we can convert the AND gate into a reversible gate.

Thus, we can convert any classical circuit into a reversible circuit by replacing the gates in the universal gate set with their corresponding reversible quantum gates. We replace NOT gate with the Pauli-X gate and the AND gate with the modified CSWAP Gate. Hence, any classical function can be implemented on a classical circuit (in terms of universal gate set $\{AND, NOT\}$) which then can be converted to a reversible circuit by replacing the gates in the universal gate set with their corresponding reversible quantum gates (AND gate with Modified CSWAP and NOT gate with Pauli-X gate). Thus, for a general classical function with corresponding implementation in classical circuit as C_f we would have a corresponding quantum circuit U_f which is the reversible version of the classical circuit C_f thus, implementing the classical function f on a quantum circuit by making it reversible. Here, we have converted the classical function f to a reversible function $\tilde{f}$ which has number of inputs equal to the number of outputs, produces the same output as the classical function f along with the input and some junk values as shown in the figure 4.4. **We can implement any classical function on a quantum circuit.** The above figure shows the conversion of a classical circuit to a quantum circuit.

Thus, a classical circuit when converted to a quantum circuit is as shown in the figure 4.4, it takes input $|x\rangle$ and along with it some ancilla bits (aka junk bits) and produces the output $|x\rangle$, $|y \oplus f(x)\rangle$ along with some junk bits $junk(x)$. We need to remove the junk since it consumes extra amount of qubits which is a waste of quantum resources and also the junk bits might get entangled with the output and any interference with environment may lead to decoherence or incorrect output and errors. Thus, we need to remove the junk bits. This is done using the following circuit 4.5

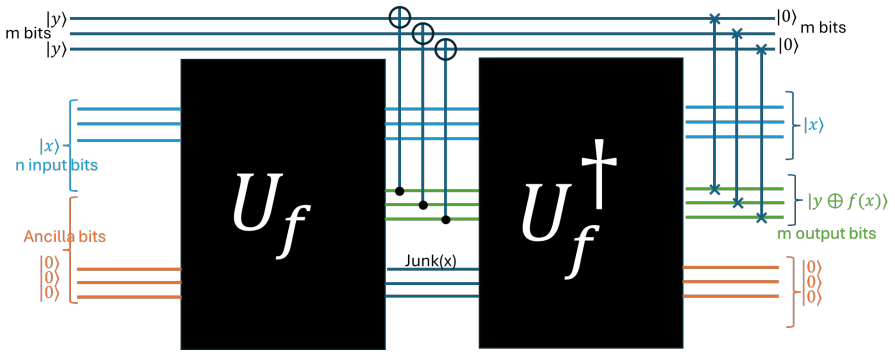


Figure 4.5: Quantum Oracle

Using this circuit it can be seen that the output of the entire circuit is $|x\rangle, |y \oplus f(x)\rangle$ and the junk bits are are now $|0\rangle$ so that they can be used for other purposes in the circuit ahead. The output from the first Quantum Circuit U_f is $|x\rangle, |y \oplus f(x)\rangle, junk(x)$. To remove the junk bits, it was done by first copying the output $|f(x)\rangle$ to the other (working bits) bits and then the output from the U_f circuit was passed through its inverse $U_f^{-1} = U_f^\dagger$. Thus, upon passing through its inverse the output will be $|x\rangle, |0\rangle, |0\rangle$ since $U_f^\dagger$ is a reverse circuit and hence will make the outputs back to the inputs. Then the bits are swapped between as shown in the figure to get the desired output of $|x\rangle, |y \oplus f(x)\rangle, |0\rangle, |0\rangle$ with no junk. Thus, the output of the circuit is the input $|x\rangle$ and the output $|y \oplus f(x)\rangle$. This entire thing is called an Oracle. Thus, we can imagine a Quantum Circuit which is an implementation of a classical fucntion f as a black box which takes input $|x\rangle, |y\rangle, |0\rangle$ and produces the output $|x\rangle, |y \oplus f(x)\rangle, |0\rangle$. For the sake of simpliicity we generally ignore the $|0\rangle$ in inputs and outputs.

Thus, wheverer given a classical function f, we can make it as a reversible quantum function and thus, implmement it as a quantum circuit using reversible quantum gates. This can be visualized as shown in the figure 4.6.

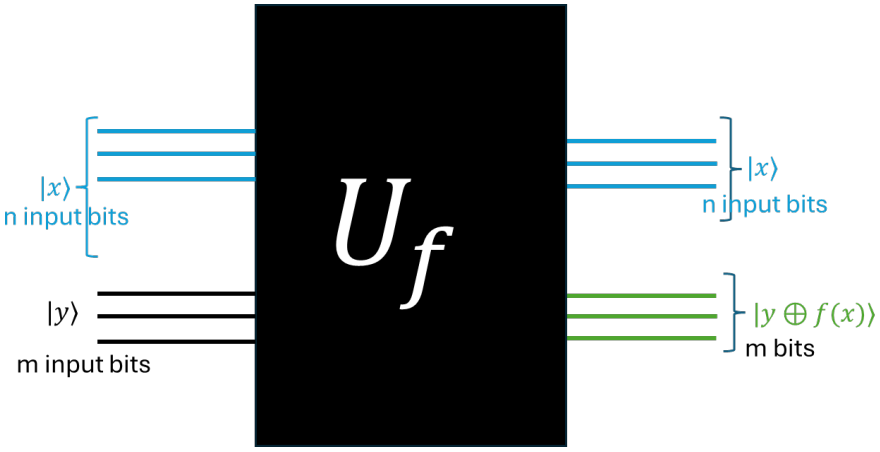


Figure 4.6: Oracle

4.2 Elitzur-Vaidman Bomb Detection Algorithm

We are given a device which can be a working Bomb or dud. The devices takes a Quantum bit as input and produces an output. If the device is a bomb, then the device performs a measurement on that qubit. If the outcome of the measurement is 1, then the bomb explodes, else the bomb does not explode. If the device is dud, then the device does nothing i.e output is same as input. We wish to determine whether the device is a bomb or dud without exploding the bomb. The Elitzur-Vaidman Bomb Detection Algorithm is as follows:

1. **Approach 1: Naive Approach**
   We give the device naively the input of $|0\rangle$ and $|1\rangle$.

   - **Case 1: Device is a Bomb**
     If we give device input of $|0\rangle$ then the output will also be $|0\rangle$ always, becuase the outcome of the measurement will be $|0\rangle$ so the bomb doesn't explode and returns $|0\rangle$. If we give device inputof $|1\rangle$ then upon measurement the outcome will be $|1\rangle$ and the bomb explodes always.
   - **Case 2: Device is a Dud**
     For the input of $|0\rangle$ if the device is a dud then the output is $|0\rangle$ since it does no operation on the input. For the input of $|1\rangle$ if the device is a dud then the output is $|1\rangle$ since it does no operation on the input.

   To summarise the above approach, giving input $|0\rangle$ isof no help as it provides no information about the device since in both the cases if its a bomb or a dud

the output will be $|0\rangle$. Upon giving input $|1\rangle$ if the device is a dud then the output will be $|1\rangle$ and if the device is a bomb then the bomb will explode. Thus, it tells us whether the device is a bomb or a dud but if its a bomb it will explode. Thus, this approach is not useful since we can never deteact a working bomb.

2. **Approach 2: Quantum Superposition Approach**
Here we use a Hadamard Gate before and after the device. The circuit is as shown in the figure 4.7.

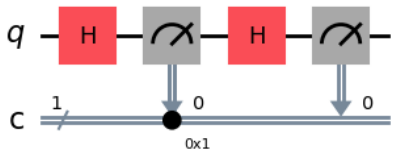


Figure 4.7: Elitzur-Vaidman Bomb Detection Algorithm

Here the first measurement is present if device is a bomb.

- **Case 1: Device is a Bomb**
If we give input of $|0\rangle$ then upon passing thorugh the hadamard gate it will be $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. As it passes through the device, it performs a measurement since its a bomb. Thus, the outcome of the measurement in computational basis will be $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$. Thus, in the cases where the outcome is $|0\rangle$ the bomb will not explode and the output will then pass through another Hadamard gate which will give output as $|+\rangle$. Thus, upon final measurement we will get $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$. In the case where the measurement outcome is $|1\rangle$ the bomb will explode.
- **Case 2: Device is a Dud**
If we give input of $|0\rangle$ then the upon passing through the hadamard gate it will be $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. As it passes through the device, the device does nothing and the output will be $|+\rangle$. Then, we pass through the Hadamard gate which will output $|0\rangle$ and upon final measurement we will get the output as $|0\rangle$.

To summarize, the above approach, suppose there are a 1000 devices out of which probability of being a bomb is 50%, thus, 500 are bombs and 500 are duds. If we given input $|0\rangle$ to all the devices. Then the output will be $|0\rangle$ for all the duds. For the bombs, because of the measurement 50% will detect $|1\rangle$ and will explode while the other 50% will not explode and output $|0\rangle$. Thus, out of 500 bombs, 250 will explode and the rest 250 won't explode. Then, the out of the bombs which didn't explode and which give outcomes as $|0\rangle$, it will pass thorugh a hadamard gate whose output will be $|+\rangle$ and thus upon final measurement we will get $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$. Thus, out of the 250 bombs which didn't explode 50% i.e. 125 of them will output$|1\rangle$ and the other 50% i.e. 125 will output $|0\rangle$.

Thus, out of the 1000 devices we will get 250 bombs which will explode and out of the rest 750 devices we will get output of $|0\rangle$ for 625 (500 duds + 125 bombs which gave output $|0\rangle$) devices and output of $|1\rangle$ for 125 devices. Now, we are specifically interested in the cases where the output was $|1\rangle$ upon input $|0\rangle$. This is possible only in the case where the devices were bomb which didn't explode, since had it been a dud the output would have been $|0\rangle$. Thus the 125 cases where the ouptut was $|1\rangle$ upon input of $|0\rangle$ are the bombs which didn't explode. Thus, out of the 500 bombs 250 of them exploded and we found 125 working bombs, and the other 125 bombs didn't explode but couldn't be identified among the duds. Thus, we found 25% of the working bombs without exploding them. Or, we can say that the we can find a working bomb with a probability of $\frac{1}{8}$ out of all the given devices (or we can find a working bomb witout exploding given that the device is a bomb with probability $\frac{1}{4}$).

What if the input had been $|1\rangle$?: The result will be the same as above. The only difference is that the output will be $|1\rangle$ for the duds and the bombs which didn't explode. Some of the bombs which didn't explode will output $|0\rangle$ and thus, can be detected as working bombs.

Thus, we can find a working bomb with a probability of $\frac{1}{8}$ (i.e. without exploding) out of all the given devices (or we can find a working bomb witout exploding given that the device is a bomb with probability $\frac{1}{4}$). But, recall that in this case $\frac{1}{2}$ (i.e. 250 bombs) of the bombs exploded and out of the other $\frac{1}{2}$ (i.e. 250 bombs) which didn't explode we only found $\frac{1}{2}$ (i.e. 125 bombs) of the working bombs, the rest $\frac{1}{2}$ (i.e. 125 bombs) didn't explode but couldn't be identified either.

3. **Elitzur-Vaidman Bomb Detection Algorithm:**

Recall that we are measuring in orthnormal computational basis i.e. $|0\rangle$ and $|1\rangle$. Consider the Single-Qubit gate R_θ^Y which is defined as:

$$R_\theta^Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

where θ is the angle of rotation. The R_θ^Y gate is a rotation gate about the Y-axis. The R_θ^Y gate is a unitary gate and thus preserves the norm of the vector as well as inner product. It is also a Hermitian gate. Thus, R_θ^Y is a self-inverse gate. $R_\theta^Y = (R_\theta^Y)^\dagger = (R_\theta^Y)^{-1}$. Thus, the action of R_θ^Y gate on the state $|0\rangle$ is as follows:

$$R_\theta^Y |0\rangle = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \end{bmatrix} = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$$

Thus, upon measurement the probability of getting $|0\rangle$ is $|\cos\frac{\theta}{2}|^2$ and the probability of getting $|1\rangle$ is $|\sin\frac{\theta}{2}|^2$. Thus, the R_θ^Y gate is a rotation gate about the Y-axis by an angle θ. Now clearly, for very small θ, the probability of getting $|0\rangle$ is approximately $\cos\frac{\theta}{2}$. The circuit for Elitzur-Vaidman Bomb Detection Algorithm is as shown in the figure 4.8.

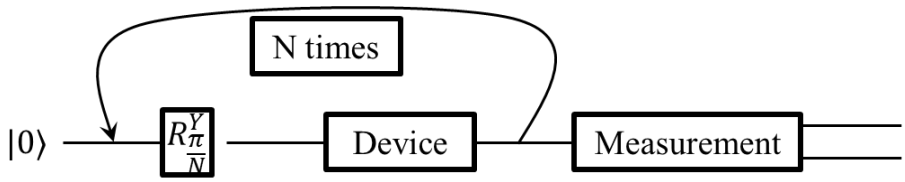


Figure 4.8: Elitzur-Vaidman Bomb Detection Algorithm

Thus, we we pass $|0\rangle$ through the R_θ^Y gate, the output will be $\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$.

- **Case 1:Device is a Dud**
  Now since the device is Dud, it will do nothing and after the first iteration, the output will be $\cos\frac{\pi}{2N}|0\rangle + \sin\frac{\pi}{2N}|1\rangle$. Now, for the second iteration it will be passed through the R_θ^Y gate again and the output will be:

  $$\begin{bmatrix} \cos\frac{\pi}{2N} & -\sin\frac{\pi}{2N} \\ \sin\frac{\pi}{2N} & \cos\frac{\pi}{2N} \end{bmatrix} \begin{bmatrix} \cos\frac{\pi}{2N} \\ \sin\frac{\pi}{2N} \end{bmatrix} = \begin{bmatrix} \cos^2\frac{\pi}{2N} - \sin^2\frac{\pi}{2N} \\ 2\cos\frac{\pi}{2N}\sin\frac{\pi}{2N} \end{bmatrix} = \cos\frac{\pi}{N}|0\rangle + \sin\frac{\pi}{N}|1\rangle$$

  Thus, after the second iteration the output will be $\cos\frac{\pi}{N}|0\rangle + \sin\frac{\pi}{N}|1\rangle$. Thus, after the kth iteration the output will be $\cos\frac{k\pi}{2N}|0\rangle + \sin\frac{k\pi}{2N}|1\rangle$.

Thus, at the last Nth iteration it will be $\cos\frac{\pi}{2}|0\rangle + \sin\frac{\pi}{2}|1\rangle = |1\rangle$. Thus, upon the final measurement the output will be $|1\rangle$. Thus, for the case when the device is a dud we will get $|1\rangle$ as the output.

- **Case 2: Device is a Bomb**
  Now since the device is a bomb, it will perform a measurement. Thus, after passing the input $|0\rangle$ through the rotation gate, the output will be $\cos\frac{\pi}{2N}|0\rangle + \sin\frac{\pi}{2N}|1\rangle$. Now since the device is a bomb, it performs a measurement and the measurement will be $|0\rangle$ with probability $\cos^2\frac{\pi}{2N}$ and $|1\rangle$ with probability $\sin^2\frac{\pi}{2N}$. Now consider for very large N, the probability of getting $|0\rangle$ is $\lim_{N\to\infty}\cos^2\frac{\pi}{2N} \approx \cos 0 \approx 1$ and the probability of getting $|1\rangle$ is $\lim_{N\to\infty}\sin^2\frac{\pi}{2N} \approx \sin 0 \approx 0$. Thus, the output will be $|0\rangle$ with a very high probability and $|1\rangle$ has almost zero probability for very large N. Thus, after the first iteration we will get $|0\rangle$ as the output with very high probabiltiy and this is then again passed through the rotation gate for the next iteration. Now, the second iteration will be the same as the first iteration since the input here is again $|0\rangle$ i.e.e same as in the first iteration and this will repeat for N iterations. After the N iterations the circuit will output $|0\rangle$ provided the bomb doesn't explode even though the probability of the bomb exploding is very low. Thus, the output of the circuit will be $|0\rangle$ for the case when the device is a bomb or it will explode with a very low probability. Note that the chances of the bomb exploding is inversely proportional to the value of N i.e. the number of times we run the circuit. Thus, the chances of the bomb exploding is very low for very large N. Hence, we get the output as $|0\rangle$ for the case when the device is a bomb.

The probability of that the bomb explodes after first iteration is $\sin^2\frac{\pi}{2N}$ and the probability that the bomb explodes after N iterations is $\sin^{2N}\frac{\pi}{2N}$ (this is because for the explosion of bomb is an independent event in each iteration and for independent events $P(A\cap B\cap\ldots\cap Z) = P(A)\cdot P(B|A)\ldots P(Z|A,B,\cdot,Z) = P(A)\cdot P(B)\ldots P(Z)$, thus the probability P(bomb exploding in 1st iteration and bomb exploding in second iteration and ... and bomb exploding in Nth iteration))=P(bomb exploding in first iteration)P(bomb exploding in second iteration)...P(bomb exploding in N iterations). For very large N $\sin\frac{\pi}{2N} \approx \frac{\pi}{2N}$. Hence, the probability of the bomb exploding after N iterations will be $\left(\frac{\pi}{2N}\right)^{2N}$.

N	Probability of Bomb Exploding
2	0.25
10	7.702×10^{-17}
20	6.105×10^{-45}
⋮	⋮
500	1.409×2^{-2503}

Table 4.2: Probability of Bomb Exploding after N iterations

As can be seen in the table 4.2, the probability of the bomb exploding after N iterations drops rapidly for very large N. Thus, the Elitzur-Vaidman Bomb Detection Algorithm is a very efficient algorithm for detecting a working bomb without exploding it.

4.3 Deutsch's Algorithm

It is a simple and good to learn basic algorithm.

Definition 4.3.1. Given a function $f : \{0, 1\} \to \{0, 1\}$, which takes a single bit as input and produces a single bit as output. The function f is said to be constant if $f(0) = f(1)$ and f is said to be balanced if $f(0) \neq f(1)$.

There are four possible functions as shown in the table 4.3. Out of these four possible cases we can see that two are constant and the other two are balanced functions.

Function	Input	Output
f_1	0	0
	1	0
f_2	0	1
	1	1

(a) Constant Function

Function	Input	Output
f_3	0	0
	1	1
f_4	0	1
	1	0

(b) Balanced Function

Table 4.3: Possible Functions

Given a function f, our aim is to determine whether the given function is constant or balanced with as minimum number of queries as possible to the function.

On a Classical Computer: To solve this problem on a classical computer we are required to make two function calls. If both the outputs are the same then the function is constant and if the outputs are different then the function is balanced. Thus, we require **two function calls** to determine whether the function is constant or balanced.

On a Quantum Computer: To solve this problem on a Quantum Computer using deutsch's Algorithm we are required to make only one fuction call. This speedup is because of the superposition.

The circuit for Deutsch's Algorithm is as shown in the figure 4.9.

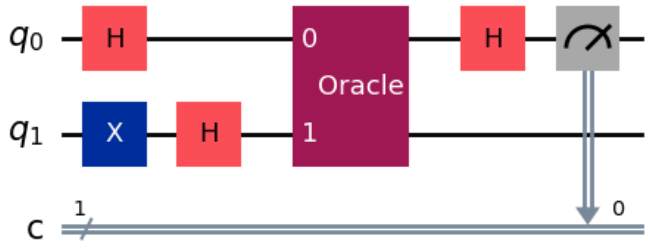


Figure 4.9: Deutsch's Algorithm

Consider the oracle as shown in the figure 4.9. The oracle is a black box which takes inputs $|x\rangle$ and $|y\rangle$ as inputs and outputs $|x\rangle$ and $|y \oplus f(x)\rangle$ as outputs. It is a unitary operator and implements the functions f.

Claim: If upon measurement of the first qubit, if it is $|0\rangle$ then the function is constant or else the function is balanced.

Analysis: At inputs both the qubits $q_0 = |0\rangle$ and $q_1 = |0\rangle$. As q_0 passes though the Hadamard Gate it becomes $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. As the qubit q_1 passes through X gate, its output is $|1\rangle$ and then it is passed through H gate thus, its state at the output of the Hadamard gate will be $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. thus, at the oracle we have $|+\rangle$ and $|-\rangle$ as inputs. Thus, the action of oracle is:

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

Thus, the action of oracle will be as follows:

$$U_f \left|+\right\rangle \left|-\right\rangle = U_f(\frac{\left|0\right\rangle + \left|1\right\rangle}{\sqrt{2}} \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}})$$

$$U_f \frac{1}{2}\left(\left|00\right\rangle - \left|01\right\rangle + \left|10\right\rangle - \left|11\right\rangle\right) = \frac{1}{2}\left[\left|0\right\rangle \left|0 \oplus f(0)\right\rangle - \left|0\right\rangle \left|1 \oplus f(0)\right\rangle + \left|1\right\rangle \left|0 \oplus f(1)\right\rangle - \left|1\right\rangle \left|1 \oplus f(1)\right\rangle\right]$$

Thus, the output of the oracle is as shown above. Now consider two cases:

1. **Case 1: Function is Constant** $f(0) = f(1) = 0$
   If the function is constant then $f(0) = f(1)$. Let $f(0) = f(1) = 0$ and thus the output of the oracle will be:

   $$\frac{1}{2}\left[\left|0\right\rangle \left|0\right\rangle - \left|0\right\rangle \left|1\right\rangle + \left|1\right\rangle \left|0\right\rangle - \left|1\right\rangle \left|1\right\rangle\right]$$

   Now factorizing the above into tensor product of two states as:

   $$\frac{1}{2}\left[\left|0\right\rangle \left|0\right\rangle - \left|0\right\rangle \left|1\right\rangle + \left|1\right\rangle \left|0\right\rangle - \left|1\right\rangle \left|1\right\rangle\right] = \frac{\left|0\right\rangle + \left|1\right\rangle}{\sqrt{2}} \otimes \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}$$

   $$(H \otimes I)(\left|+\right\rangle \otimes \left|-\right\rangle) = \left|0\right\rangle \otimes \left|-\right\rangle$$

   Hence, as the qubit q_0 passes through the Hadamard gate, the action of the Hadamard gate on the first qubit will be $H\left|+\right\rangle = \left|0\right\rangle$. Thus, upon measurement the output will be $\left|0\right\rangle$ with probablity 1. Thus, we get a measurement of 0.

2. **Case 1: Function is Constant** $f(0) = f(1) = 1$
   If the function is constant then $f(0) = f(1)$. Let $f(0) = f(1) = 1$ and thus the output of the oracle will be:

   $$\frac{1}{2}\left[\left|0\right\rangle \left|1\right\rangle - \left|0\right\rangle \left|0\right\rangle + \left|1\right\rangle \left|1\right\rangle - \left|1\right\rangle \left|0\right\rangle\right]$$

   Now factorizing the above into tensor product of two states as:

   $$\frac{1}{2}\left[\left|0\right\rangle \left|1\right\rangle - \left|0\right\rangle \left|0\right\rangle + \left|1\right\rangle \left|1\right\rangle - \left|1\right\rangle \left|0\right\rangle\right] = \frac{\left|0\right\rangle + \left|1\right\rangle}{\sqrt{2}} \otimes - \left(\frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}\right)$$

   $$(H \otimes I)(\left|+\right\rangle \otimes - \left|-\right\rangle) = \left|0\right\rangle \otimes - \left|-\right\rangle$$

   Hence, as the qubit q_0 passes through the Hadamard gate, the action of the Hadamard gate on the first qubit will be $H\left|+\right\rangle = \left|0\right\rangle$. Thus, upon measurement the output will be $\left|0\right\rangle$ with probablity 1. Thus, we get a measurement of 0.

3. **Case 2: Function is Balanced with $f(0) = 0$ and $f(1) = 1$**
   If the function is balanced then $f(0) \neq f(1)$ and let $f(0) = 0$ and $f(1) = 1$. Thus, the output of the oracle will be:

$$\frac{1}{2}\left[|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|1\rangle - |1\rangle|0\rangle\right]$$

   Now factorizing the above into tensor product of two states as:

$$\frac{1}{2}\left[|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|1\rangle - |1\rangle|0\rangle\right] = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$(H \otimes I)(|-\rangle \otimes |-\rangle) = |1\rangle \otimes |-\rangle$$

   Hence, as the qubit q_0 passes through the Hadamard gate, the action of the Hadamard gate on the first qubit will be $H|-\rangle = |1\rangle$. Thus, upon measurement the output will be $|1\rangle$ with probablity 1. Thus, we get a measurement of 1.

4. **Case 2: Function is Balanced with $f(0) = 1$ and $f(1) = 0$** If the function is balanced then $f(0) \neq f(1)$ and let $f(0) = 1$ and $f(1) = 0$. Thus, the output of the oracle will be:

$$\frac{1}{2}\left[|0\rangle|1\rangle - |0\rangle|0\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle\right]$$

   Now factorizing the above into tensor product of two states as:

$$\frac{1}{2}\left[|0\rangle|1\rangle - |0\rangle|0\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle\right] = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \otimes -\frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$(H \otimes I)(|-\rangle \otimes -|-\rangle) = |1\rangle \otimes -|-\rangle$$

   Hence, as the qubit q_0 passes through the Hadamard gate, the action of the Hadamard gate on the first qubit will be $H|-\rangle = |1\rangle$. Thus, upon measurement the output will be $|1\rangle$ with probablity 1. Thus, we get a measurement of 1.

Conclusion:
In order to summarize the above algorithm, note that in case if the function is constant we get a measurement of 0 and if the function is balanced we get a measurement of 1. Thus, we can determine whether the function is constant or balanced using only one function call. This is a speedup over the classical algorithm which requires two function calls to determine whether the function is constant or balanced.

4.4 Deutsch-Josza Algorithm

This is a generalization of the Deutsch Algorithm.

Definition 4.4.1. Given a classical function $f : \{0,1\}^n \rightarrow \{0,1\}$, which takes n input bits and returns a single output bit. The function is called constant if $f(x) = 0$ or $f(x) = 1$ for all $x \in \{0,1\}^n$ and the function is called balanced if $f(x) = 0$ for exactly half of the inputs.

Our aim is to find whether the function f is constant or balanced with as minimum number of queries as possible to the Oracle.

The total number of such possible functions are 2^{2^n} out of which 2 are constant and $\frac{2^n!}{2^{n-1}!2^{n-1}!}$ are balanced functions for n input bits and 1 output bit. Say for the case n=2, the possible functions are as shown in the table 4.4.

Function	Input	Output
f_1	00	0
	01	0
	10	0
	11	0
f_2	00	1
	01	1
	10	1
	11	1

(a) Constant Function

Function	Input	Output
f_3	00	0
	01	0
	10	1
	11	1
f_4	00	0
	01	1
	10	0
	11	1
$\vdots$	$\vdots$	$\vdots$
f_9	00	0
	01	1
	10	1
	11	0

(b) Balanced Function

Table 4.4: Possible Functions

On a Classical Computer: In order to determine whether the function is constant or balanced we need to query the function for just 1 more than half of the possible inputs. Thus, we require $2^{n-1}+1$ queries to determine whether the function is constant or balanced. Thus, the time complexity for this on a classical computer

is $\mathcal{O}(2^n)$ i.e. exponential time complexity. If all the inputs produce the same output then the function is constant else the function is balanced.

On a Quantum Computer: To solve this problem we need to query the function only once on a Quantum Computer using the Deutsch-Josza Algorithm. Thus, the time complexity to solve this problem is $\mathcal{O}(1)$ i.e. constant time complexity. This is an exponential speedup over the classical algorithm. The circuit for Deutsch Josza Algorithm Circuit is as shown in the figure 4.10.

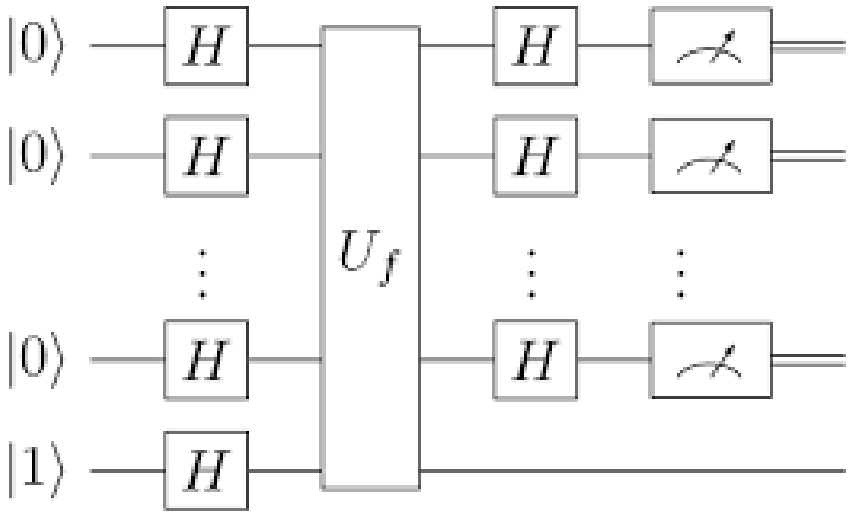


Figure 4.10: Deutsch-Josza Algorithm

The input of this circuit is $q_0 = |0\rangle, q_1 = |0\rangle, q_2 = |1\rangle$. Now at the input we have $|0\rangle^{\otimes n} |1\rangle$ and the action of the Hadamard gate on the input will be:

$$H^{\otimes n} |0\rangle^{\otimes n} H |1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$$

Now the oracle will show the action U_f as $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$ and thus the

action of the oracle will be:

$$U_f\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\frac{(|0\rangle-|1\rangle)}{\sqrt{2}}\right)=U_f\left(\frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{2}}\sum_{x\in\{0,1\}^n}|x\rangle|0\rangle-|x\rangle|1\rangle\right)$$
$$=\frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{2}}\sum_{x\in\{0,1\}^n}U_f|x\rangle|0\rangle-U_f|x\rangle|1\rangle$$
$$=\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\frac{(|0\oplus f(x)\rangle-|1\oplus f(x)\rangle)}{\sqrt{2}}$$

Now $|0\oplus f(x)\rangle=|f(x)\rangle$ and $|1\oplus f(x)\rangle=|\overline{f(x)}\rangle$ where $\overline{f(x)}$ is the complement of $f(x)$. Thus, substituting this in the equation we get,

$$\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\frac{(|f(x)\rangle-|\overline{f(x)}\rangle)}{\sqrt{2}}$$

Now when $f(x)=0$ then $\frac{|f(x)-|\overline{f(x)}\rangle\rangle}{\sqrt{2}}=\frac{|0\rangle-|1\rangle}{\sqrt{2}}=|-\rangle$ and when $f(x)=1$ then $\frac{|f(x)-|\overline{f(x)}\rangle\rangle}{\sqrt{2}}=\frac{|1\rangle-|0\rangle}{\sqrt{2}}=-|-\rangle$. Thus, this can be written as:

$$\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\frac{(|f(x)\rangle-|\overline{f(x)}\rangle)}{\sqrt{2}}=\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle|-\rangle$$

Now we apply the Hadamard gate on the first n qubits and the action of the Hadamard gate on the first n qubits will be:

$$(H^{\otimes n}\otimes I)\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle|-\rangle\right)=H^{\otimes n}\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right)\otimes I|-\rangle$$
$$=\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}H^{\otimes n}|x\rangle\right)\otimes|-\rangle$$

Recall that $H^{\otimes n}|x\rangle=\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle$ for Hadamard gate (refer section 2.2.2) acting on n qubits. Thus, substituting this in the above equation can be written as:

$$\left(\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{f(x)+x\cdot y}|y\rangle\right)\otimes|-\rangle$$

Now, we perform the measurement on the first n qubits. Consider the following two cases:

1. **Case 1: Function is Constant**

   - Let $f(x) = 0$ for all $x \in \{0,1\}^n$. Hence we need to perform measurement on the first n qubits of the following state:

     $$\left(\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}\left|y\right\rangle\right)\otimes\left|-\right\rangle$$

     Upon simplifying we get,

     $$\left(\frac{1}{2^n}2^n\left|0^{\otimes n}\right\rangle\right)\otimes\left|-\right\rangle = \left|0^{\otimes n}\right\rangle\otimes\left|-\right\rangle$$

     Thus, upon measurement the output will be $|0^{\otimes n}\rangle$ with probability 1. Thus, we get a measurement of $|0^{\otimes n}\rangle$.

   - Let $f(x) = 1$ for all $x \in \{0,1\}^n$. Hence we need to perform measurement on the first n qubits of the following state:

     $$\left(\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{1+x\cdot y}\left|y\right\rangle\right)\otimes\left|-\right\rangle$$

     Upon simplifying we get,

     $$\left(\frac{1}{2^n}2^n\left|0^{\otimes n}\right\rangle\right)\otimes\left|-\right\rangle = -\left|0^{\otimes n}\right\rangle\otimes\left|-\right\rangle$$

     Thus, upon measurement the output will be $|0^{\otimes n}\rangle$ with probability 1. Thus, we get a measurement of $|0^{\otimes n}\rangle$.

2. **Case 2: Function is Balanced**

   - Let $f(x)$ be a balanced function. Thus, $f(x) = 0$ for exactly half of the inputs and $f(x) = 1$ for the other half of the inputs. Hence we need to perform measurement on the first n qubits of the following state:

     $$\left(\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{f(x)+x\cdot y}\left|y\right\rangle\right)\otimes\left|-\right\rangle$$

Upon simplifying we get,

$$\left(\frac{1}{2^n}(2^n)(\pm|11\rangle)\right)\otimes|-\rangle = \pm|11\rangle\otimes|-\rangle$$

Depending on the input, for the balanced function the output will either be $|11\rangle$ or $-|11\rangle$. In either cases, upon measurement the output will be $|11\rangle$ with probability 1. Thus, we get a measurement of $|11\rangle$.

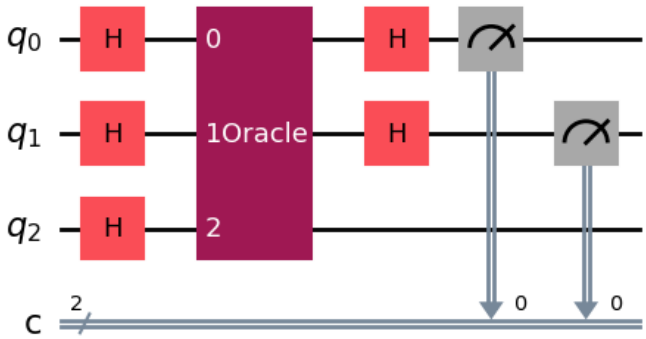


Figure 4.11: Deutsch-Josza Algorithm for n=2

Example 4.4.1. Consider the Deutsch-Josza algorithm for n=2. The possible functions are as shown in the table 4.4 and the circuit for the Deutsch-Josza algorithm is as shown in the figure 4.10. The input to the circuit is $|00\rangle|1\rangle$. The action of the Hadamard gate on the input will be:

$$H^{\otimes 2}|00\rangle H|1\rangle = \frac{1}{2}\sum_{x\in\{0,1\}^2}|x\rangle\frac{|0\rangle-|1\rangle}{\sqrt{2}}$$
$$= \frac{1}{2\sqrt{2}}(|000\rangle+|010\rangle+|100\rangle+|110\rangle-|001\rangle-|011\rangle-|101\rangle-|111\rangle)$$

The action of the oracle will be:

$$U_f\left(\frac{1}{2\sqrt{2}}(|000\rangle+|010\rangle+|100\rangle+|110\rangle-|001\rangle-|011\rangle-|101\rangle-|111\rangle)\right)=$$

$$\frac{1}{2\sqrt{2}}(U_f|000\rangle+U_f|010\rangle+U_f|100\rangle+U_f|110\rangle-U_f|001\rangle-U_f|011\rangle-U_f|101\rangle-U_f|111\rangle)$$

$$= \frac{1}{2\sqrt{2}}(|00\rangle|f(00)\rangle + |01\rangle|f(01)\rangle + |10\rangle|f(10)\rangle + |11\rangle|f(11)\rangle - |00\rangle|\overline{f(00)}\rangle$$

$$- |01\rangle|\overline{f(01)}\rangle - |10\rangle|\overline{f(10)}\rangle - |11\rangle|\overline{f(11)}\rangle)$$

Now consider the following cases;

1. **Case 1: $f(x)$ is constant**

   - Let $f(x) = 0$ for all $x \in \{0,1\}^2$. Thus, the output of the oracle will be:

     $$\frac{1}{2\sqrt{2}}(|00\rangle|0\rangle+|01\rangle|0\rangle+|10\rangle|0\rangle+|11\rangle|0\rangle-|00\rangle|1\rangle-|01\rangle|1\rangle-|10\rangle|1\rangle-|11\rangle|1\rangle)$$

     $$= \frac{1}{2\sqrt{2}}(|000\rangle + |010\rangle + |100\rangle + |110\rangle - |001\rangle - |011\rangle - |101\rangle - |111\rangle)$$

     $$= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

     Now we apply Hadamard on the first n qubits and we get:

     $$(H^{\otimes 2} \otimes I)\frac{1}{2}((|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes |-\rangle)$$

     $$= \frac{1}{2}(H^{\otimes 2}|00\rangle + H^{\otimes 2}|01\rangle + H^{\otimes 2}|10\rangle + H^{\otimes 2}|11\rangle) \otimes |-\rangle$$

     $$= \frac{1}{2}[\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

     $$\frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)] \otimes |-\rangle$$

     Upon simplifying we get,

     $$\frac{1}{4}(4|00\rangle) \otimes |-\rangle = |00\rangle \otimes |-\rangle$$

     Thus, upon measurement the output will be $|00\rangle$ with probability 1. Thus, we get a measurement of $|00\rangle$.

- Let $f(x) = 1$ for all $x \in \{0,1\}^2$. Thus, the output of the oracle will be:

$$\frac{1}{2\sqrt{2}}(|00\rangle|1\rangle+|01\rangle|1\rangle+|10\rangle|1\rangle+|11\rangle|1\rangle-|00\rangle|0\rangle-|01\rangle|0\rangle-|10\rangle|0\rangle-|11\rangle|0\rangle)$$

$$= \frac{-1}{2\sqrt{2}}(|000\rangle + |010\rangle + |100\rangle + |110\rangle - |001\rangle - |011\rangle - |101\rangle - |111\rangle)$$

$$= \frac{-1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Now we apply Hadamard on the first n qubits and we get:

$$(H^{\otimes 2} \otimes I)\frac{-1}{2}((|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes |-\rangle)$$

$$= \frac{-1}{2}(H^{\otimes 2}|00\rangle + H^{\otimes 2}|01\rangle + H^{\otimes 2}|10\rangle + H^{\otimes 2}|11\rangle) \otimes |-\rangle$$

$$= \frac{-1}{2}[\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

$$\frac{-1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)] \otimes |-\rangle$$

Upon simplifying we get,

$$\frac{-1}{4}(4|00\rangle) \otimes |-\rangle = -|00\rangle \otimes |-\rangle$$

Thus, upon measurement the output will be $|00\rangle$ with probability 1. Thus, we get a measurement of $|00\rangle$.

2. **Case 2: $f(x)$ is balanced**

- Let $f(x)$ be a balanced function. Thus, $f(x) = 0$ for exactly half of the inputs and $f(x) = 1$ for the other half of the inputs. Then it can be shown that based upon the input chosen the output will be either of the first n qubits, $|11\rangle$ or $-|11\rangle$. Thus, upon measurement the output will be $|11\rangle$ with probability 1. Thus, we get a measurement of $|11\rangle$.

Conclusion:
From the above algorithm, we can clearly see that for determining whether the function $f : \{0,1\}^n \to \{0,1\}$ is constant or balanced we need to input the function

with n $|0\rangle$ and 1 $|1\rangle$ and query the function only once. Then, in the final measurement we measure the first n qubits and if the function is constant we get $|0^{\otimes n}\rangle$ and if the function is balanced we get $|11\rangle$. Thus, we can determine whether the function is constant or balanced with just one query to the Oracle. This is an exponential speedup over the classical algorithm which requires $2^{n-1}+1$ queries to determine whether the function is constant or balanced. Thus, the time complexity of this algorithm is $\mathcal{O}(1)$ i.e. constant time complexity. Hence, an exponential speedup over the classical algorithm.

4.5 Simon's Algorithm

Definition 4.5.1. Given a function $f : \{0,1\}^n \to \{0,1\}^n$ which takes n input bits and returns n output bits. The function is a two - to - one function, such that $f(x) = f(x \oplus s)$ for $s \in \{0,1\}^n$. We are required to find s.

Example 4.5.1. Consider the function $f : \{0,1\}^3 \to \{0,1\}^3$ for n=3, and let $s =$ 111. Then, since the function is a two-to-one function i.e. produces the same output for two distinct inputs such that $f(x) = f(x \oplus s)$. Thus, $f(000) = f(000 \oplus 111) = f(111), f(001) = f(001 \oplus 111) = f(110), f(010) = f(010 \oplus 111) = f(101), f(011) = f(011 \oplus 111) = f(100), f(100) = f(100 \oplus 111) = f(011), f(101) = f(101 \oplus 111) = f(010), f(110) = f(110 \oplus 111) = f(001), f(111) = f(111 \oplus 111) = f(000)$. Thus, we have:

$$\begin{aligned} f(000) &= f(111) \\ f(001) &= f(110) \\ f(010) &= f(101) \\ f(011) &= f(100) \end{aligned}$$

Consider the follwing functions as an example with outputs given in tabular form:

Input	Output
000	111
001	000
010	110
011	010
100	010
101	110
110	000
111	111

Table 4.5: Example of Simon's Algorithm

It can be clearly seen that the above function is a valid function as it follows the constraints of being a two-to-one function with the above stated constraints. We can also clearly see from the table 4.5 that the function is a two-to-one function and the function is periodic with period $s = 111$. Thus, its a valid example function for Simon's algorithm. Now given that the function is a two-to-one function we are required to find s.

On a Classical Computer: In order to determine the period of the function we need to query the function for $2^{n-1} + 1$ times to determine the period of the function. This is because the total number of possible inputs are 2^n and since the function is a two-to-one function with a period of s, it produces the same output for half of the inputs. Thus, in order to find s calssically, in the worst case scenario we are required to query the function for $2^{n-1} + 1$ times to determine the period of the function. Thus, the time complexity of the algorithm is $\mathcal{O}(2^n)$ i.e. exponential time complexity. So, the best way to solve this problem is using randomized algorithms using some probabilistic approach. In that case, the running time is $\mathcal{O}(2^{n/2})$, which is still exponential. Note that once we know two inputs (say x,y) with the same output. Then, we can use $x \oplus y = s$ to find s (because $y = x \oplus s \implies x \oplus y = x \oplus (x \oplus s) = 0 \oplus s = s$. Because $x \oplus x = 0$).

On a Quantum Computer: To solve this problem we need to query the function $\mathcal{O}(n)$ times on a Quantum Computer using the Simon's Algorithm which is a linear time complexity. The circuit for Simon's Algorithm is as shown in the figure 4.12.

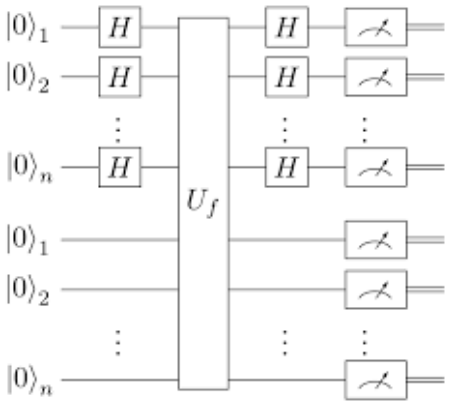


Figure 4.12: Simon's Algorithm

Analysis of Simon's Algorithm:
The input to the circuit is $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$. Then the action of hadamard gates on the input will be:

$$(H^{\otimes n} \otimes I^{\otimes n})(|0\rangle^{\otimes n} \otimes |0\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle |0\rangle^{\otimes n}$$

Now the classical function f takes n inputs and produces n outputs and its a two-to-one function with a period s. Thus, in order to implement it on quantum computer, we use a quantum oracle U_f which implements the function f as $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. Here $|y\rangle = |0\rangle^{\otimes n}$, thus the action of the oracle will be $U_f|x\rangle|0\rangle^{\otimes n} = |x\rangle|0 \oplus f(x)\rangle = |x\rangle|f(x)\rangle$. Thus, the action of the oracle will be:

$$U_f\left(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle |0\rangle^{\otimes n}\right) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle |f(x)\rangle$$

Now we perform a measurement on the second registers. Now the second registers are in some superposition and upon measurement will collapse to some classical values. Let the output of the second register be $f(z)$. Becuase of this the first register will have only those values which are mapped to $f(z)$.

Thus, the state of the system will be:

$$\frac{|z\rangle + |z \oplus s\rangle}{\sqrt{2}} |f(z)\rangle$$

Now applying hadamard gate on the first n qubits will be:

$$(H^{\otimes n} \otimes I)\left(\frac{|z\rangle + |z \oplus s\rangle}{\sqrt{2}} |f(z)\rangle\right) = \frac{1}{\sqrt{2}}[(H^{\otimes n}|z\rangle + H^{\otimes n}|z \oplus s\rangle)|f(z)\rangle]$$

$$= \frac{1}{\sqrt{2}}[\frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y}|y\rangle + \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{(z \oplus s) \cdot y}|y\rangle]|f(z)\rangle$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} ((-1)^{z \cdot y} + (-1)^{(z \oplus s) \cdot y})|y\rangle|f(z)\rangle$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y}(1 + (-1)^{s \cdot y})|y\rangle|f(z)\rangle$$

Now, there are two possible cases:

- **Case 1:** $y \cdot s = 1$. Then the amplitude of y will be zero and thus that value won't be taken in the superposition.
- **Case 2:** $y \cdot s = 0$.Then the amplitude of that corresponding y will be 2 and thus will be in the superposition.

Thus, the supoerposition will consist of only those values of y for which $y \cdot s = 0$. We can use this fact to run the circuit n times and attain a system of n linearly independent equations. We can then solve these equations to find s.

Example 4.5.2. Consider the function $f : \{0,1\}^4 \to \{0,1\}^4$ as shown in the table 4.6.

Input	Output
0000,1001	1111
0001, 1000	0001
0010,1011	1110
0011,1010	1101
0100,1101	0000
0101,1100	0101
0110,1111	1010
0111,1110	1001

Table 4.6: Function

Here, $s = 1001$. Now, say the above shown function has been implemented in a quantum oracle. We are required to find s for the function using Simon's algorithm.

Now at the input the state of the system will be:

$$|0^4\rangle |0^4\rangle = |0000\rangle |0000\rangle$$

Now we apply hadamard gate on the first register. Thus, the state of the system will now be:

$$(H^{\otimes 4} \otimes I^{\otimes 4})(|0000\rangle \otimes |0000\rangle) = \frac{1}{\sqrt{2^4}} \sum_{x\in\{0,1\}^4} |x\rangle |0000\rangle$$

$$= \left(\frac{1}{4} \sum_{x\in\{0,1\}^4} |x\rangle\right) \otimes |0000\rangle$$

$$= \left(\frac{1}{4} \sum_{x\in\{0,1\}^4} |x\rangle |0000\rangle\right)$$

Now, we apply the oracle on the above state ($U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$). Thus, the state of the system will be:

$$U_f \left(\frac{1}{4} \sum_{x\in\{0,1\}^4} |x\rangle |0000\rangle\right) = \frac{1}{4} \sum_{x\in\{0,1\}^4} U_f |x\rangle |0000\rangle$$

$$= \frac{1}{4} \sum_{x\in\{0,1\}^4} |x\rangle |0000 \oplus f(x)\rangle = \frac{1}{4} \sum_{x\in\{0,1\}^4} |x\rangle |f(x)\rangle$$

$$\frac{(|0000\rangle |f(0000)\rangle + |0001\rangle |f(0001)\rangle + \ldots + |1110\rangle |f(1110)\rangle + |1111\rangle |f(1111)\rangle)}{4}$$

Thus, upon substituting the corresponding values from the table 4.6 we get, since n=4 we will get 16 terms:

$$\frac{1}{4}[|0000\rangle |1111\rangle + |0001\rangle |0001\rangle + |0010\rangle |1110\rangle + |0011\rangle |1101\rangle + |0100\rangle |0000\rangle + |0101\rangle |0101\rangle$$
$$|0110\rangle |1010\rangle + |0111\rangle |1001\rangle + |1000\rangle |0001\rangle + |1001\rangle |1111\rangle + |1010\rangle |1101\rangle + |1011\rangle |1110\rangle$$
$$+ |1100\rangle |0101\rangle + |1101\rangle |0000\rangle + |1110\rangle |1001\rangle + |1111\rangle |1010\rangle]$$

Now, we perform a measurement on the second register thus the superposition state of the second register will collapse to some classical state. Let the output of the

second register be $f(z)$ which comes out as, say $f(z) = 1010$. Thus, the state of the system now will be only those values which produce 1010 in the second register as output. Thus, the state of the system will be:

$$\frac{|0110\rangle + |1111\rangle}{\sqrt{2}} \otimes |1010\rangle$$

Now we apply the Hadamard gate on the first register. Thus, the state of the system will be:

$$(H^{\otimes 4} \otimes I^{\otimes 4})(\frac{|0110\rangle + |1111\rangle}{\sqrt{2}} \otimes |1010\rangle)$$

Using the property that the Hadamard gate when applied to any input gives $H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$, we get:

$$= \frac{H^{\otimes 4} |0110\rangle + H^{\otimes 4} |1111\rangle}{\sqrt{2}} \otimes |1010\rangle$$

Again applying the general formula for Hadamard gate $H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$, and simplifying (reacll that we are required to consider only those values where $y \cdot s = 0$ and the amplitude in that case is 2. For the cases where the $y \cdot s = 1$ the amplitude for that corresponding $|y\rangle$ will be zero. Thus, we will get only those value where $y \cdot s = 0$), we get:

$$\frac{|0000\rangle - |0010\rangle - |0100\rangle + |0110\rangle + |1001\rangle - |1011\rangle - |1101\rangle + |1111\rangle}{\sqrt{2^3}} \otimes |1010\rangle$$

We can also verify that $y \cdot s = 0$, (recall that here $s = 1001$ and thus, only those terms in the equation survive that have $y \cdot s = 0$) using the result from the above. Now when we perform the measurement on the first register we will get one of the above state with equal probability ($= \frac{1}{8}$).

Now we are required to get $n - 1 = 4 - 1 = 3$ linearly independent y to find s by forming a system of linear equations. Say we measure and upon the measurement we get the state $|0000\rangle$. Now state $|0000\rangle$ is a null vector and thus we can't use it to form the system of linear equations.

Now we rerun the circuit and for the next time upon measurement we get, say $|0010\rangle$ which works for the system of linear equations. Thus, our set of vectors for making a linear system of equation contains $\{|0010\rangle\}$.

Now we rerun the circuit and for the next time upon measurement we get, say $|0100\rangle$ which works for the system of linear equations since it is independent of the

previous vectors $|0010\rangle$. Thus, our set of vectors now contain $\{|0010\rangle, |0100\rangle\}$. Now we need one more linearly independent vector to find s.

Now we rerun the circuit and for the next time upon measurement we get, say $|0110\rangle$ which does not work for the system of linear equations since it is dependent on the previous two vectors ($|0110\rangle$ can be made using $0100 \oplus 0010 = 0110$).

Now we rerun the circuit and for the next time upon measurement we get, say $|1001\rangle$ which works for the system of linear equations since it is independent of the previous vectors $|0010\rangle$ and $|0100\rangle$. Thus, our set of vectors now contains $\{|0010\rangle, |0100\rangle, |1001\rangle\}$. Now we have 3 linearly independent vectors and thus we can form a system of linear equations to find s.

Now, we know that each of the linearly indpendent vector in the set $\{|0010\rangle, |0100\rangle, |1001\rangle\}$ is a solution to the equation $y \cdot s = 0$. Thus, we can form a system of linear equations as shown below:

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Upon, solving this system of lienear equations we get:

$$\begin{aligned} s_3 &= 0 \\ s_2 &= 0 \\ s_1 + s_4 &= 0 \end{aligned}$$

Now, we can choose one of the values of s_1 or s_4. Let us choose s_4. Now, $s_4 = 0$ or $s_4 = 1$. Let us choose $s_4 = 0$ then we get, $s_1 = 0$ which gives a trivial solution os $s = 0000$. Let $s_4 = 1$ then we get, $s_1 = -1 \implies (-1) mod 2 = 1$. Thus, we get $s = 1001$ which is the correct solution.

4.6 Bernstein-Vazirani Algorithm

4.7 Shor's Algorithm

Definition 4.7.1. Given a number N which is a composite number. We are required to find the prime factors of the number N.

Chapter 5

Grover's Search Algorithm

It is the second most famous quantum algorithm after Shor's algorithm. It doesn't prove exponential speed up, but only a quadratic speedup, yet it is much more widely applicable than Shor's algorithm.

5.1 The Problem Definition

Definition 5.1.1. The Search Problem For $N = 2^n$, we are given an arbitrary $x \in \{0,1\}^n$. Suppose we have a function

$$f : \{0,1\}^n \to \{0,1\}$$

that is implemented by a reversible transformation (Oracle) B_f in the usual way:

$$B_f |x\rangle |a\rangle = |x\rangle |a \oplus f(x)\rangle$$

for all $x \in \{0,1\}^n$ and $a \in \{0,1\}$. (We can build a quantum circuit for B_f given a Boolean circuit for f at a cost linear in size of boolean circuit). The problem of search is to simply 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.

This is also called as unstructured search problem because f is arbitrary i.e. there is no promise and we can't rely on it have a structure that makes finding solutions easy.

Remark. Here, we assume that the we can easily evaluate this function f (function can be computed efficiently). That is, maybe we have a boolean circuit whose size is poly(n). This does not restrict the function to be simple. It can be complex and may not have a simple form. This is a *NP-complete* problem.

5.2 The Classical Solution

Note that the searching problem is completely unstructured. There are no promises on the function f, so it is not possible to use binary search or any other fast searching methods to efficiently solve the problem classically. The best classical algorithm for solving the above search problem is Linear search (a **deterministic algorithm**) and in the worst case it would require $N = 2^n$ queries to the black box (to distinguish the case where f is identically 0 from any of the cases where there is a single x for which $f(x) = 1$). Thus, $\mathcal{O}(N)$ time complexity. (Here we have assumed that the cost of evaluating f is the one that dominates the computation and hence we talk about the umber of queries to the function f in order to evaluate the time complexity.)

Probabilistically, a best strategy for an algorithm that makes k queries is to simply choose k distinct values of x and to query the black-box at these k values. In the case that there is a single value of x for which $f(x) = 1$, and we require that our algorithm succeeds in finding this x with probability at least $1 - \epsilon$, then we must have have $1 - \frac{k}{2^n} \leq \epsilon$. This implies $k \geq (1 - \epsilon)2^n$, so for constant error we need $k = \Omega(2^n) = \omega(N)$ queries to solve the problem.

In contrast, the Grover's algorithm will solve the problem using $\mathcal{O}(\sqrt{2^n})$ i.e. $\mathcal{O}(\sqrt{N})$ queries and $\mathcal{O}(\sqrt{N}\log N)$ other gates (the number of gates can be reduced a bit further).

Important Note

The problem can be thought of as a table of size N, where exactly one element has value 1, and all others are 0. Searching an item in an unsorted table or array of size N costs a classical computer $\mathcal{O}(N)$ running time. If N is large, this is like searching for a needle in a haystack. This Quantum algorithm for search was proposed by Lov Grover in 1995 that consults the table only $\mathcal{O}(\sqrt{N})$ times.
In contrast to algorithms like quantum factoring which provide exponential speedups, the search algorithm only provides a quadratic improvement. The same technique used in this algorithm can be used to speedup algorithms for NP-complete problems.
It is natural to wonder whether there are even faster quantum algorithms for search. However, it turns out that the quadratic speedup is optimal. This was proved in 1994. Any quantum algorithm for search must consult the table at least some constant times $\sqrt{N}$ times. In this, we have assumed *unique search problem* i.e. This is all assuming that there is only one element that has value 1 and all others have value 0. In case of *multiple solutions* i.e. if there are more than one elements with value 1, then the Grover's algorithm can be modified, and in that case the Complexity becomes $\mathcal{O}(\sqrt{\frac{N}{M}})$, where M is the number of elements with value 1 and N is the total number of elements ($N = 2^n$).

5.3 The Quantum Solution: Grover's Algorithm

Here, we assume the problem to be a Query problem. Thus, the boolean function f implemented on a classical circuit can be converted to a Quantum circuit fro implementing a Query operation using the concepts of reversible computation as explained in section 4.1.

5.3.1 Idea of the Algorithm

The best way to describe the Grover's Algorithm is to describe it geometrically. Say we define two sets A and B as follows:

$$A = \{x \in \{0,1\}^n : f(x) = 1\}$$
$$B = \{x \in \{0,1\}^n : f(x) = 0\}$$

Think of A as the set of "good" strings $x \in \{0,1\}^n$; the goal of the algorithm is to find one of these strings. The set B cont contains all of the "bad" strings $x \in \{0,1\}^n$ that do not satisfy the search criterion. Let

$$a = |A| \quad b = |B|; \quad a + b = N$$

where $|\cdot|$ denotes cardinality of the sets (i.e. number of elements in the set), with $a \neq 0$ and $b \neq 0$ (The case with $a = 0$ or $b = 0$ will be considered separately). Although the analysis is for the general case, it might be easier to imagine an interesting case of the problem where a is very small (say $a = 1$) and therefore b is very large (close to $N = 2^n$, say $b = N - 1$). Note that for each $x \in \{0,1\}^n$ it will be either in A or in B but not both since the output corresponding to the input to the function is either 0 or 1. Thus, an arbitrary x will be either in A or in B. In other words, the sets A and B are mutually exclusive sets. Next we define,

$$|A\rangle = \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle$$
$$|B\rangle = \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle$$

which means $|A\rangle$ is the superposition of all the input states to the function f which yield output 1 and similarly, $|B\rangle$ is the superposition of all the input states to the function f which output 0. Now since sets A and B were mutually exclusive and each x lies either in A or B. Clearly $|A\rangle$ and $|B\rangle$ are orthogonal i.e. $\langle A|B\rangle = 0$ because each of the basis state $|x\rangle$ is either in A or in B, thus its corresponding component in the other set is 0, thus their inner product will result in 0. Hence, $|A\rangle$ and $|B\rangle$ are orthogonal.

The two vectors $|A\rangle$ and $|B\rangle$ form a two-dimensional subspace geometrically. Now consider an equal superposition state of all the possible $|x\rangle$ i.e. $|h\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$. Note that the two-dimensional subspace formed by the two orthogonal vectors $|A\rangle$ and $|B\rangle$ contains $|h\rangle$ i.e. $|h\rangle$ can be expressed as a linear combination of the basis

vectors of the two-dimensional subspace as follows:

$$
\begin{aligned}
|h\rangle &= \frac{1}{\sqrt{N}} \sum_{x} |x\rangle \\
&= \frac{1}{\sqrt{N}} \left(\sum_{x \in A} |x\rangle + \sum_{x \in B} |x\rangle \right) \\
&= \frac{\sqrt{a}}{\sqrt{N}} \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle + \frac{\sqrt{b}}{\sqrt{N}} \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle \\
|h\rangle &= \frac{\sqrt{a}}{\sqrt{N}} |A\rangle + \frac{\sqrt{b}}{\sqrt{N}} |B\rangle
\end{aligned}
$$

Considering for the purpose of imagination one could think of $a << b$ so that the search problem is like finding a needle in a haystack. In pother words, the component of $|h\rangle$ on $|A\rangle$ is much smaller than the component of $|h\rangle$ on $|B\rangle$. Thus, geometrically in space one could imagine this as $|h\rangle$ being very close to $|B\rangle$ then $|A\rangle$ as depicted in the following figure 5.1

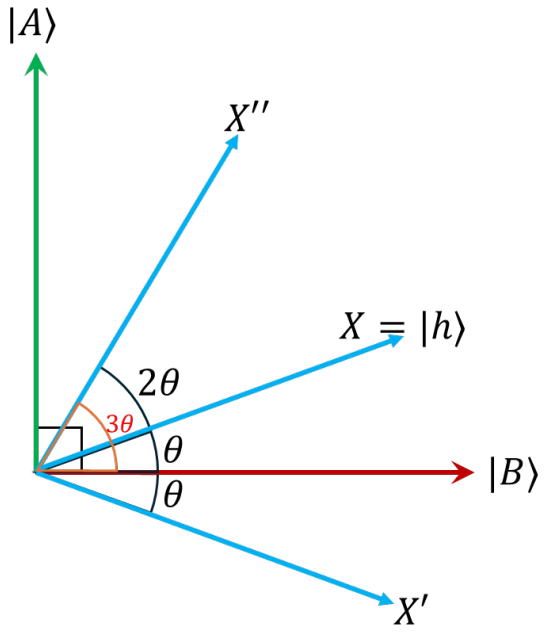


Figure 5.1: Geometric Visualization

In the figure 5.1 it can be clearly seen that the$|A\rangle$ and $|B\rangle$ are orthogonal. Say

$X = |h\rangle$ makes an angle θ with $|B\rangle$ (Note that for the purpose of easier imagination as said in the case where $a << b$, (say $a = 1, b = N - 1$) which is mostly the case, i.e. when the number of good strings to be searched for is much less than the bad strings, the component of $|h\rangle$ on $|A\rangle$ ($= \frac{\sqrt{a}}{\sqrt{N}}$) will be much less than the component of $|B\rangle$ on $|h\rangle$ ($= \frac{\sqrt{b}}{\sqrt{N}}$) since $a << b$. Thus, as shown in the figure 5.1, $|h\rangle$ is much closer to $|B\rangle$ than $|A\rangle$.)

Note that it is easy to create the state $|h\rangle$ since it is simply an equal superposition of all the states. Thus can be created by applying $H^{\otimes n}$ to $|0^n\rangle$. From figure 5.1 we are required to make a series of rotations/unitary transformations such that eventually $|h\rangle$ gets closer to $|A\rangle$ (or becomes $|A\rangle$), thus achieving the purpose of finding the superposition of strings $x \in \{0,1\}^n$ whose input to the function yields output 1. Then, upon measurement of this state we get one of the required strings whose input to the function yields output 1.

The Unitary transformations (recall that they are simply rotations or reflections in the complex Plane i.e. Hilbert Space) required to perform this are a series of reflections as follows (Starting with $|X\rangle = |h\rangle$):

1. Reflect X about $|B\rangle$ which as shown in the figure 5.1 becomes $|X'\rangle$. Say this is denoted by the Unitary matrix $R_{|B\rangle}$. Thus, $|X'\rangle = R_{|B\rangle} |X\rangle$

2. Now reflect $|X'\rangle$ about X which as shown in the figure 5.1 becomes $|X''\rangle$. Say this is denoted by the Unitary matrix $R_{|X\rangle}$. Thus, $|X''\rangle = R_{|X\rangle} |X'\rangle$

After each iteration we put $|X\rangle = |X''\rangle$ and repeat the above two steps. Thus, the angle of $|X''\rangle$ with $|B\rangle$ is 3θ after doing this reflections for the first time. Hence, the $|X''\rangle$ obtained in the second step is now clearly closer to $|A\rangle$. Thus, performing the above steps repeatedly will bring the $|X\rangle$ closer and closer to $|A\rangle$ with each iteration as it moves farther and farther from $|B\rangle$. Once the state $|X''\rangle$ is close enough to $|A\rangle$, measuring the state will result in the required search string with a very high probability. Thus, till now what we have done is the following:

$$|0^n\rangle \xrightarrow{H^{\otimes n}} |h\rangle$$

Then we repeat the following unitary transformation a finite number of times ($\mathcal{O}(\sqrt{N})$ times) starting with $|X\rangle = |h\rangle$

$$|X\rangle \xrightarrow{R_{|B\rangle}} |X'\rangle \xrightarrow{R_{|X\rangle}} |X''\rangle$$

Then, finally we make a measurement on $|X''\rangle$ (since it gets very close to A as shown above) and the result of that measurement is one the strings whose input to the

function yields output 1. Thus, the required algorithm. This is the idea behind the Grover's algorithm. It would suffice if we could find the two unitary transformation matrices corresponding to the reflections and show that it does perform the task of reflection about $|B\rangle$ and reflection about $|X\rangle$ respectively and along with that we are required to find the optimal number of times to repeat the process. All of this is answered in the next section.

5.3.2 Algorithm

Reflection about $|B\rangle$

Recall that $|h\rangle$ can be written as linear superposition of the basis $|A\rangle$ and $|B\rangle$ as follows (where both $|A\rangle$ and $|B\rangle$ are orthogonal):

$$|h\rangle = \frac{\sqrt{a}}{\sqrt{N}}|A\rangle + \frac{\sqrt{b}}{\sqrt{N}}|B\rangle$$

In order to reflect $|h\rangle$ about $|B\rangle$, see figure 5.1, we are required to reverse the component of projection of $|h\rangle$ on $|A\rangle$. Thus, the reflected vector should have the following linear superposition:

$$|X'\rangle = -\frac{\sqrt{a}}{\sqrt{N}}|A\rangle + \frac{\sqrt{b}}{\sqrt{N}}|B\rangle$$

In other words, all we need to do is flip the phase of the component in the direction of $|A\rangle$. For a general $|X\rangle$ which arises in some intermediate step of the iteration, (which will be a linear superposition of $|A\rangle$ and $|B\rangle$) we are required to reverse the phase on the components in the direction of $|A\rangle$ keeping the components in the direction of $|B\rangle$ unchanged. In order to achieve this, we use **Phase Query Gate**. Consider the following unitary transformation on n qubits:

$$Z_f|x\rangle = (-1)^{f(x)}|x\rangle$$

This is also called a *phase kickback*. This transformation would take in the superposition state and reverse the phase of only those components which are in $|A\rangle$ (i.e. whose $f(x) = 1$) using linearity property of distributive action of a matrix on sum of vectors (superposition state). We now study the corresponding circuit as shown in the figure 5.2 which performs this transformation.

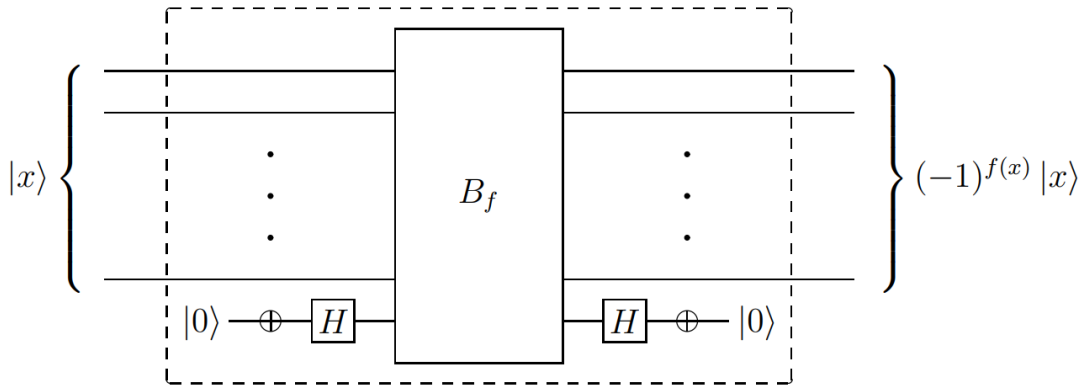


Figure 5.2: Reflection about B circuit

Here, recall that the transformation B_f is the quantum implementation of the classical function f i.e. $B_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. We now analyse the circuit. The input to the circuit is $|x\rangle|0\rangle$. Here $|0\rangle$ is the ancilla bit. We then apply NOT-gate on the second register:

$$|x\rangle|0\rangle \xrightarrow{I^{\otimes n}\otimes X} |x\rangle|1\rangle$$

Then we apply a Hadamard gate on the output on the second register, thus we get,

$$|x\rangle|1\rangle \xrightarrow{I^{\otimes n}\otimes H} |x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Now we use the Oracle B_f on this,

$$B_f|x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} B_f|x\rangle|0\rangle - \frac{1}{\sqrt{2}} B_f|x\rangle|1\rangle = \frac{1}{\sqrt{2}}|x\rangle|0 \oplus f(x)\rangle - \frac{1}{\sqrt{2}}|x\rangle|1 \oplus f(x)\rangle$$

This can be further simplified as follows:

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

Then we perform the uncomputation on the second register by applying a Hadamard gate and then a NOT gate as shown in the figure 5.2.

$$(-1)^{f(x)}|x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \xrightarrow{I^{\otimes n}\otimes H} (-1)^{f(x)}|x\rangle|1\rangle \xrightarrow{I^{\otimes n}\otimes X} (-1)^{f(x)}|x\rangle|0\rangle$$

Now, ignoring the ancilla bit we have thus achieved the following transformation

$$Z_f |x\rangle = (-1)^{f(x)} |x\rangle$$

which was the required transformation for the reflection of $|X\rangle$ over $|B\rangle$ done by reversing the phase of the components in the direction of $|A\rangle$. Note that we were required to perform one evaluation of B_f implemented in X_f to reflect about $|B\rangle$. Thus, what we have achieved through this Oracle as a Unitary operator can be seen from the fact that consider a $|X\rangle$ as some linear superposition of $|A\rangle$ and $|B\rangle$ (recall, $|A\rangle$ and $|B\rangle$ are orthogonal and $|X\rangle$ lies in the two-dimensional subspace spanned by $|A\rangle$ and $|B\rangle$) , say $|X\rangle = r|A\rangle + s|B\rangle$ for some r, s. Then upon applying the operator/Oracle Z_f on this $|X\rangle$ we get,

$$\begin{aligned}
Z_f |X\rangle &= rZ_f |A\rangle + rZ_f |B\rangle \\
&= rZ_f \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle + sZ_f \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle \\
&= r\frac{1}{\sqrt{a}} \sum_{x \in A} Z_f |x\rangle + s\frac{1}{\sqrt{b}} \sum_{x \in B} Z_f |x\rangle \\
&= r\frac{1}{\sqrt{a}} \sum_{x \in A} -|x\rangle + s\frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle \\
&= -r|A\rangle + s|B\rangle
\end{aligned}$$

As can be seen that the component of $|X\rangle$ along $|A\rangle$ is reversed. Thus, we achieve the reflection about $|B\rangle$.

Reflection about $|X\rangle$

For the reflection about $|X\rangle$, we use the Diffusion operator D (assume $N = 2^n$), which works as follows. First, apply $H^{\otimes n}$, which maps $|X\rangle \to |00\ldots0\rangle$. then reflect around $|00\ldots0\rangle$ this is accomplished by the circuit U_g, where g is a function such that $g(00\ldots0) = 0$ and $g(x) = 1$ for $x \neq 00\ldots0$. Finally, apply $H^{\otimes n}$ to return to the original basis. (Note that this is simply a reflection around the zero vector in the Hadamard basis). This is the general idea of Reflection about $|X\rangle$. In order to understand this operation better we get into the details as follows. Consider an operator Z_0 as follows:

$$Z_0 = I - 2|0^n\rangle\langle 0^n|$$

Thus,

$$D = H^{\otimes n} Z_0 H^{\otimes n} = H^{\otimes n}(I - 2|0^n\rangle\langle 0^n|)H^{\otimes n} = I - 2|h\rangle\langle h|$$

This is also the same as having a phase query gate Z_{OR} $(= -Z_0)$ for the n-bit OR function:

$$OR(x) = \begin{cases} 0, & x = 0^n \\ 1, & x \neq 0^n \end{cases}$$

for all $x \in \{0,1\}^n$.

$$Z_{OR}(x) = \begin{cases} |x\rangle & x = 0^n \\ -|x\rangle & x \neq 0^n \end{cases}$$

for all $x \in \{0,1\}^n$. Note that $Z_{OR} = -Z_0 = 2|0^n\rangle\langle 0^n| - I$. Note that this does not depend on function f and thus requires no queries. To implement this on a quantum circuit we convert the classical implementation of a n-bit OR boolean circuit. We now make the following propositions:

Proposition 5.3.1. *The Diffusion operator D has two properties:*

1. *It is Unitary and can be efficiently realised.*
2. *It can be seen as an "inversion about the mean".*

Proof. 1. To prove that D is unitary it suffices to prove that $H^{\otimes n}$ and Z_0 is Unitary. Clearly, $H^{\otimes n}$ is unitary since it is a Hadamard Gate tensored product n times. Now for proving that Z_0 is unitary. We do as follows:

$$\begin{aligned} Z_0^\dagger &= (I - 2|0^n\rangle\langle 0^n|)^\dagger \\ &= I^\dagger - 2(|0^n\rangle\langle 0^n|^\dagger \\ &= I - 2|0^n\rangle\langle 0^n| = Z_0 \end{aligned}$$

Thus, Z_0 is Hermitian. Now doing $Z_0^\dagger Z_0$, we get,

$$\begin{aligned} Z^\dagger Z_0 = Z_0 Z_0^\dagger &= (I - 2|0^n\rangle\langle 0^n|)(I - 2|0^n\rangle\langle 0^n|) \\ &= I - 2|0^n\rangle\langle 0^n| - 2|0^n\rangle\langle 0^n| + 4|0^n\rangle\langle 0^n| \\ &= I \end{aligned}$$

Thus, Z_0 is Unitary. Hence $D = H^{\otimes n} Z_0 H^{\otimes n}$ is a product of Unitary matrices and hence itself is a Unitary Matrix. Observe that D is expressed as the product of three Unitary matrices (two hadamard Matrices separated by a conditional phase shift matrix). Therefore, D is also Unitary. regarding the implementation, both the Hadamard and the conditional phase shift transforms can be efficiently realized within $\mathcal{O}(n)$ gates.

2. Consider D operating on a vector $|\alpha\rangle$ to generate another vector $|\beta\rangle$.

$$D\begin{bmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_i\\ \vdots\\ \alpha_N\end{bmatrix} = \begin{bmatrix}\beta_1\\ \beta_2\\ \vdots\\ \beta_i\\ \vdots\\ \beta_N\end{bmatrix}$$

Let $\mu = \frac{1}{N}\sum_j \alpha_j$ be the mean amplitude, then the expression $2\mu - \alpha_i$ describes a reflection of α_i about mean. This might be easier to see by writing it as $\mu + (\mu - \alpha_i)$. In order to realise that the diffusion operator does perform this task, let us write D in matrix form ($N = 2^n$):

$$\begin{aligned} D &= I - 2\,|h\rangle\langle h| \\ &= \begin{bmatrix}1 & 0 & \dots & 0 & 0\\ 0 & 1 & \dots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \dots & 1 & 0\\ 0 & 0 & \dots & 0 & 1\end{bmatrix} - \frac{2}{N}\begin{bmatrix}1 & 1 & \dots & 1 & 1\\ 1 & 1 & \dots & 1 & 1\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 1 & 1 & \dots & 1 & 1\\ 1 & 1 & \dots & 1 & 1\end{bmatrix} \\ D &= \begin{bmatrix}1-\frac{2}{N} & -\frac{2}{N} & \dots & -\frac{2}{N} & -\frac{2}{N}\\ -\frac{2}{N} & 1-\frac{2}{N} & \dots & -\frac{2}{N} & -\frac{2}{N}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ -\frac{2}{N} & -\frac{2}{N} & \dots & 1-\frac{2}{N} & -\frac{2}{N}\\ -\frac{2}{N} & -\frac{2}{N} & \dots & -\frac{2}{N} & 1-\frac{2}{N}\end{bmatrix} \end{aligned}$$

Thus, the amplitude of $\beta_i = -\frac{2}{N}\sum_j \alpha_j + \alpha_j = -2\mu + \alpha_i$ can be considered an "inversion about the mean" with respect to α_i.

□

Now, in order to understand how we achieve the reflection about $|h\rangle$, let us understand that the operator $D = I - 2\,|h\rangle\langle h|$ is a reflection operator about a hyperplane orthogonal to $|h\rangle$. Consider the vector $|X'\rangle$. Now we can write $|X'\rangle$ as a linear superposition of components along $|h\rangle$ and components orthogonal to $|h\rangle$ which is denoted by $|h^\perp\rangle$ (we can do this because recall that $|X'\rangle$ and $|h\rangle$ both lie in the two-dimensional subspace spanned by the orthogonal basis $|A\rangle$ and $|B\rangle$. Thus, when we say component orthogonal to $|h\rangle$, we mean that $|h^\perp\rangle$ which lies in this

two-dimensional subspace, since, the component of $|X'\rangle$ which is orthogonal to $|h\rangle$ will obviously lie in the plane spanned by $|A\rangle$ and $|B\rangle$). Thus, $|X'\rangle = p\,|h\rangle + q\,|h^{\perp}\rangle$ for some p, q. Now we apply the operator D on $|X'\rangle$, we get,

$$\begin{aligned} D\,|X'\rangle &= pD\,|h\rangle + qD\,|h^{\perp}\rangle \\ &= p(I - 2\,|h\rangle\,\langle h|)\,|h\rangle + q(I - 2\,|h\rangle\,\langle h|)\,|h^{\perp}\rangle \\ &= p(|h\rangle - 2\,|h\rangle\,\langle h|h\rangle) + q(|h^{\perp}\rangle - 2\,|h\rangle\,\langle h|h^{\perp}\rangle \\ &= p\,|h\rangle - 2p\,|h\rangle + q\,|h^{\perp}\rangle \\ &= -p\,|h\rangle + q\,|h^{\perp}\rangle \end{aligned}$$

Thus, the component along $|h\rangle$ is reversed while the component along the hyperplane orthogonal to $|h\rangle$ (i.e. along $|h^{\perp}\rangle$ is preserved. Hence, it's a reflection of $|X'\rangle$ about a plane orthogonal to $|h\rangle$. Now, we simply reverse the sign i.e. use $-D$ as operator. Thus, we get,

$$-D\,|X'\rangle = p\,|h\rangle - q\,|h^{[}\perp\rangle$$

which preserves the component along $|h\rangle$ and reverses the component along the plane orthogonal to $|h\rangle$ (i.e. along $|h^{\perp}\rangle$). Hence, this operation is a reflection of $|X'\rangle$ about $|h\rangle$. Thus, the operator $-D$ is the required unitary transformation operator which performs the required reflection of $|X'\rangle$ operation about $|h\rangle$.

Grover's Rotation Operator

We have seen that using unitary operator/matrix Z_f (as an oracle) we can perform the required reflection transformation about $|B\rangle$ and using $-D$ operator/matrix we can perform the required reflection transformation about $|h\rangle$. Now, let us combine the two operations into one and define the Grover's Rotation operator as follows:

$$G = -DZ_f = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$$

We are now interested in the action of G on elements in the two-dimensional subspace spanned by $\{|A\rangle, |B\rangle\}$. The action of any matrix on any element in the subspace can be determined by examining its effect on the basis vectors of the subspace. Thus, in order to determine the action of Grover's operator G, it is sufficient to examine its effects on the basis vectors $|A\rangle$ and $|B\rangle$ of the two-dimensional subspace. Thus, the

action of G on $|A\rangle$ is given as follows:

$$\begin{aligned}
G|A\rangle &= -H^{\otimes n} Z_0 H^{\otimes n} Z_f |A\rangle \\
&= (I - 2|h\rangle\langle h|)(-Z_f)|A\rangle \\
&= (I - 2|h\rangle\langle h|)(-Z_f)\frac{1}{\sqrt{a}}\sum_{x\in A}|x\rangle \\
&= (I - 2|h\rangle\langle h|)\frac{1}{\sqrt{a}}\sum_{x\in A}-Z_f|x\rangle \\
&= (I - 2|h\rangle\langle h|)\frac{1}{\sqrt{a}}\sum_{x\in A}|x\rangle \\
&= (I - 2|h\rangle\langle h|)|A\rangle \\
&= |A\rangle - 2|h\rangle\langle h|A\rangle
\end{aligned}$$

Upon Substituting the value of $|h\rangle = \sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle$, and simplifying we get,

$$\begin{aligned}
&= |A\rangle - 2\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right)\left(\sqrt{\frac{a}{N}}\langle A|A\rangle + \sqrt{\frac{b}{N}}\langle B|A\rangle\right) \\
&= |A\rangle - 2\sqrt{\frac{a}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) \\
&= \left(1 - \frac{2a}{N}\right)|A\rangle - \frac{2\sqrt{ab}}{N}|B\rangle
\end{aligned}$$

Note that action of $G|A\rangle$ results in a vector in the same two-dimensional subspace spanned by $|A\rangle$ and $|B\rangle$. and similarly action of operator G on $|B\rangle$ is as follows:

$$\begin{aligned}
G|B\rangle &= -H^{\otimes n} Z_0 H^{\otimes n} Z_f |B\rangle \\
&= -(I - 2|h\rangle\langle h|)|B\rangle \\
&= -(|B\rangle - 2|h\rangle\langle h|B\rangle) \\
&= -|B\rangle + 2\sqrt{\frac{b}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) \\
&= \frac{2\sqrt{ab}}{N}|A\rangle - \left(1 - \frac{2b}{N}\right)|B\rangle
\end{aligned}$$

which is also a vector in the two-dimensional subspace spanned by $|A\rangle$ and $|B\rangle$. Thus, action of G on some $|X\rangle$ in the two dimensional space spanned by $|A\rangle$ and $|B\rangle$ (say $|X\rangle = \alpha|A\rangle + \beta|B\rangle$) is $G|X\rangle = \alpha G|A\rangle + \beta G|B\rangle$. From the above results, we an see that $|X\rangle$ after applying G remains in the subspace spanned by $|A\rangle$ and $|B\rangle$ with real coefficients. Let us represent the action of G on the subspace spanned by $\{|A\rangle, |B\rangle\}$ as a matrix in $|B\rangle, |A\rangle$ basis (both input and output basis) will be as follows:

$$M = \begin{bmatrix} -\left(1 - \frac{2b}{N}\right) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \left(1 - \frac{2a}{N}\right) \end{bmatrix} = \begin{bmatrix} \frac{b-a}{N} & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \frac{b-a}{N} \end{bmatrix}$$

using $a + b = N$. Also, here $|B\rangle$ is the first column and $|A\rangle$ is the second column. Notice that

$$\begin{bmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{A}{N}} & \sqrt{\frac{b}{N}} \end{bmatrix}^2 = \begin{bmatrix} \frac{b-a}{N} & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \frac{b-a}{N} \end{bmatrix} = M$$

Thus, M is a rotation matrix. Now, for $\theta \in (0, \pi/2)$ is the angle that satisfies

$$\sin\theta = \sqrt{\frac{a}{N}} \quad \text{and} \quad \cos\theta = \sqrt{\frac{b}{N}}$$

then

$$R_{2\theta} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^2 = \begin{bmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{A}{N}} & \sqrt{\frac{b}{N}} \end{bmatrix}^2 = M$$

In other words, operator G causes a rotation by an angle 2θ in the space spanned by $\{|A\rangle, |B\rangle\}$ for

$$\theta = \sin^{-1}\sqrt{\frac{a}{N}}$$

Now, we can write the state $|h\rangle$ as follows show in figure 5.3:

$$|h\rangle = \sqrt{\frac{b}{N}}|B\rangle + \sqrt{\frac{a}{N}} = \cos\theta|B\rangle + \sin\theta|A\rangle$$

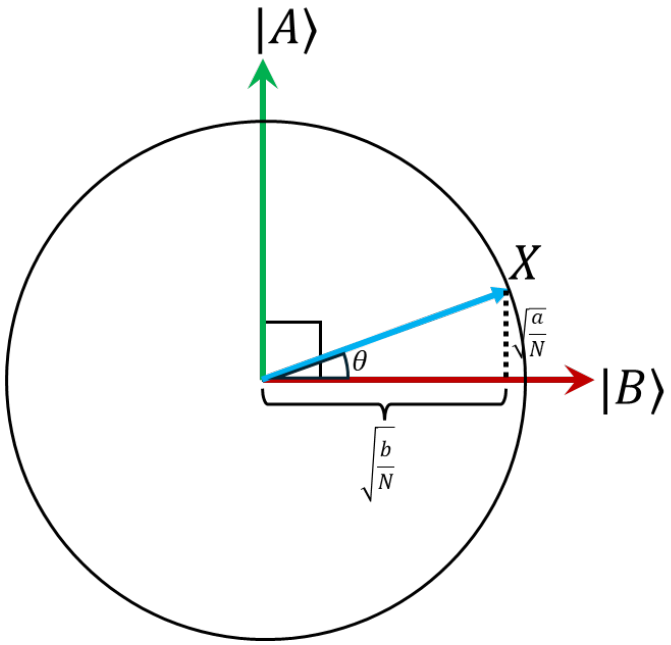


Figure 5.3: Action of G

Thus, the action of operator G on $|X\rangle$ for k times (i.e. after k iterations), the state $|X\rangle$ will be,

$$|X\rangle = \cos((2k+1)\theta)\,|B\rangle + \sin((2k+1)\theta)\,|A\rangle$$

We take $k = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ for the case assuming $a = 1$. Note that Measurement in the

Algorithm 1 Grover's Algorithm

1: **Input:** Oracle B_f, number of qubits n
2: **Output:** x such that $f(x)) = 1$
3: **Initialize:** $|0^n\rangle$
4: Apply Hadamard gate to each qubit in $|0^n\rangle$, $|X\rangle = H^{\otimes n}\,|0^n\rangle = \frac{1}{N}\sum_{x=0}^{N-1}|x\rangle = |h\rangle$
5: **for** $i = 1$ to $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ **do**
6: Apply Grover's operator G on $|X\rangle \implies G\,|X\rangle$
7: Set $|X\rangle = G\,|X\rangle$
8: **end for**
9: Measure the final state $|X\rangle$ in Standard Basis
10: **return** x of the measured state

standard basis may reveal one of the candidate solutions based on the number of times we iterate the for loop. The Grover's circuit corresponding to above implementation is as shown in the figure 5.4.

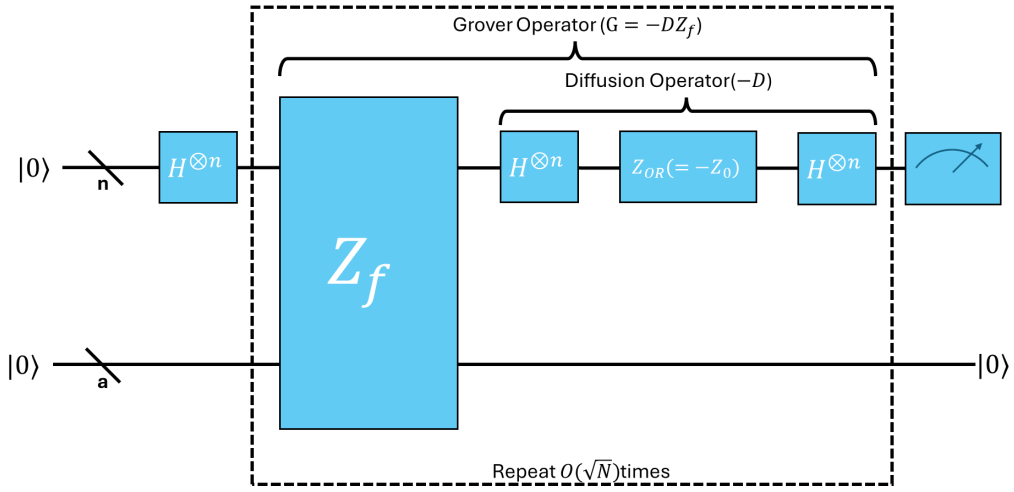


Figure 5.4: Grover's Circuit

Note that both the unitary operations Z_f which is a reflection about $|B\rangle$ and reflection about $|h\rangle$ are reflection operations/transformation and hence, the determinant of those two unitary operations is -1. When we compose two reflections, we obtain a rotation by twice the angle between the lines of reflection.

5.3.3 Example

For example, consider the function $f : \{0,1\}^3 \to \{0,1\}$ as shown in the table 5.1.

Input	Output
000	0
001	0
010	0
011	1
100	1
101	0
110	0
111	0

Table 5.1: Example of Grover's Algorithm

Thus, the set $A = \{011, 100\}$ and $B = \{000, 001, 010, 101, 110, 111\}$. Let the Cardinality i.e. the number of elements in the set represented as $|A| = a = 2$ and $|B| = b = 6$. Here, in this example, $a = 2$ and $b = 6$ and $n = 3$. Now, we define the vector $|A\rangle$ as:

$$|A\rangle = \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle = \frac{1}{\sqrt{2}}(|011\rangle + |100\rangle)$$

Similarly, we define the vector $|B\rangle$ as:

$$|B\rangle = \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle = \frac{1}{\sqrt{6}}(|000\rangle + |001\rangle + |010\rangle + |101\rangle + |110\rangle + |111\rangle)$$

Thus, it can be clearly seen that $|A\rangle$ and $|B\rangle$ are orthogonal as their inner product $\langle A|B\rangle = 0$.

STEP1: INITIALIZATION:

Now, we define the vector $|h\rangle$ as an equal superposition of all the inputs by applying the Hadamard gate on $|000\rangle$ as follows:

$$\begin{aligned}
H^{\otimes 3}|000\rangle &= \frac{1}{\sqrt{2^3}} \sum_{x \in \{0,1\}^3} |x\rangle \\
&= \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle) \\
|h\rangle &= \sqrt{\frac{2}{8}}\left(\frac{1}{\sqrt{2}}(|011\rangle + |100\rangle)\right) + \sqrt{\frac{6}{8}}\left(\frac{1}{\sqrt{6}}(|000\rangle + |001\rangle + |010\rangle + |101\rangle + |110\rangle + |111\rangle)\right) \\
|h\rangle &= \sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle
\end{aligned}$$

STEP2: REFLECTION ABOUT $|B\rangle$:
Now we perform the reflection about $|B\rangle$ by the action of $R_{|B\rangle} = Z_f$ on the state $|h\rangle$ (recall that $Z_f |x\rangle = (-1)^{f(x)} |x\rangle$) which gives:

$$\begin{aligned}
|X'\rangle = Z_f |h\rangle &= Z_f \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle) \\
&= \frac{1}{\sqrt{8}}(Z_f |000\rangle + Z_f |001\rangle + Z_f |010\rangle + Z_f |011\rangle + Z_f |100\rangle + Z_f |101\rangle + Z_f |110\rangle + Z_f |111\rangle) \\
&= \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle - |010\rangle + |011\rangle - |100\rangle + |101\rangle + |110\rangle + |111\rangle) \\
&= -\sqrt{\frac{2}{8}}\left(\frac{1}{\sqrt{2}}(|011\rangle + |100\rangle)\right) + \sqrt{\frac{6}{8}}\left(\frac{1}{\sqrt{6}}(|000\rangle + |001\rangle + |010\rangle + |101\rangle + |110\rangle + |111\rangle)\right) \\
|X'\rangle &= -\sqrt{\frac{a}{N}} |A\rangle + \sqrt{\frac{b}{N}} |B\rangle
\end{aligned}$$

Thus, we can clearly see that the component in the direction of $|A\rangle$ are reversed while the components in the direction of $|B\rangle$ are preserved and hence the resulting $|X'\rangle$ is a reflection about $|B\rangle$.
STEP3: REFLECTION ABOUT $|X'\rangle$:
Now, we are required to perform a reflection of $|X'\rangle$ about $|h\rangle$ by the application of the Diffusion operator. Thus, applying $-D = -(I - 2 |h\rangle \langle h|)$ operator on $|X'\rangle$ gives

$|X''\rangle$ as:

$$\begin{aligned}
|X''\rangle = -D\,|X'\rangle &= -(I - 2\,|h\rangle\,\langle h|)\,|X'\rangle = -(|X'\rangle - 2\,|h\rangle\,\langle h|X'\rangle) \\
&= 2\,\langle h|X'\rangle\,|h\rangle - |X'\rangle \\
&= 2\left(\sqrt{\frac{a}{N}}\,\langle A| + \sqrt{\frac{b}{N}}\,\langle B|\right)\left(-\sqrt{\frac{a}{N}}\,|A\rangle + \sqrt{\frac{b}{N}}\,|B\rangle\right)|h\rangle - |X'\rangle \\
&= 2\left(\frac{b}{N} - \frac{a}{N}\right)|h\rangle - |X'\rangle \\
&= \frac{2(b-a)}{N}\sqrt{\frac{a}{N}}\,|A\rangle + \frac{2(b-a)}{N}\sqrt{\frac{b}{N}}\,|B\rangle + \sqrt{\frac{a}{N}}\,|A\rangle - \sqrt{\frac{b}{N}}\,|B\rangle \\
&= \left(\frac{3b-a}{N}\right)\sqrt{\frac{a}{N}}\,|A\rangle + \left(\frac{b-3a}{N}\right)\sqrt{\frac{b}{N}}\,|B\rangle \\
&= \frac{3(6)-2}{8}\sqrt{\frac{2}{8}}\,|A\rangle + \frac{6-3(2)}{8}\sqrt{\frac{6}{8}}\,|B\rangle \\
&= \frac{16}{8}\frac{1}{2}\,|A\rangle + 0 \\
&= |A\rangle \\
&= \frac{1}{\sqrt{2}}(|100\rangle + |011\rangle)
\end{aligned}$$

Initially the angle between $|h\rangle$ and $|B\rangle$ was $\langle h|B\rangle = \sqrt{\frac{b}{N}} = \cos\theta \implies \theta = \cos^{-1}\sqrt{\frac{6}{8}} = \frac{\pi}{6}$. After the transformation we can clearly see that the angle between $|X''\rangle$ and $|B\rangle$ is $\langle X''|B\rangle = 0 = \cos\theta \implies \theta = \cos^{-1} 0 = \frac{\pi}{2}$. Thus, the angle changed from $\frac{\pi}{6}$ to $\frac{\pi}{2}$. Thus, we can clearly see that the angle changed from θ to 3θ. Now, we need to repeat this reflection transformation $k = \lfloor \frac{\pi}{4}\sqrt{8} \rfloor = 2$. Thus, we need to perform this once again. But note thatwe have already arrived at $|A\rangle$ and doing this once agina will bring us farther from $|A\rangle$. This is the problem with the grover's algorithm. Since, we don't know the number of possible outcomes i.e. $a = 1$ or $a \geq 1$, we assumed that $a = 1$ and decided to perform the iterations based on $a = 1$ which comes out as $k = \lfloor \frac{\pi}{4} \rfloor = 2$. But, if we perform the operation once again then the angle between $|X''\rangle$ and $|B\rangle$ would become $5\theta = 5\frac{\pi}{6} > \frac{\pi}{2}$, thus it overshoots, decreasing the probability of outcomes being from state $|A\rangle$. In order to improve over this we introduce modified grover's algorithm which states to set m=1, pick $k \in 1, \ldots, m+1$ uniformly and perform the grover's algorithm k times and then perform the measurement, and stop only if we get x such that $f(x) = 1$. Else, if

$m < \sqrt{N}$ set $m = \lfloor (8/7)m \rfloor$, else fail.
STEP4: MEASUREMENT:
Thus, supposed we pick $k = 1$ and then we perform the Grover's algorithm $k = 1$ times, we would get the state $|X''\rangle = |A\rangle$,

$$|X''\rangle = |A\rangle = \frac{1}{\sqrt{2}}(|011\rangle + |100\rangle)$$

thus upon performing measurements we would get the required search strings $x = |100\rangle$ and $x = |011\rangle$ with the property that $f(x) = 1$ with equal probability. The Modified Grover's Algorithm is further explained in the following complexity analysis section (refer section 5.4).

5.4 Complexity Analysis

Here we answer why the number of times to iterate is $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ and thus the time complexity of the Grover's algorithm is $\mathcal{O}(\sqrt{N})$. Recall, that after k iterations of applying operator G on $|X\rangle$, the state $|X\rangle$ is:

$$\cos((2k+1)\theta)\,|B\rangle + \sin((2k+1)\theta)\,|A\rangle$$

The goal is to measure some element $x \in A$, so we would like the state $|X\rangle$ to be as close to $|A\rangle$ as possible. In other words, we want the probability of measuring $|A\rangle$ to be as high as possible which consequently means that the probability of measuring $|B\rangle$ to be as low as possible. Thus, we want,

$$\sin^2((2k+1)\theta) \approx 1$$

then we require the minimum value of k with which we can achieve this,

$$(2k+1)\theta \approx \frac{\pi}{2}$$

will suffice, so we should choose

$$k \approx \frac{\pi}{4\theta} - \frac{1}{2}$$

Of course k must be an integer, which is why we can only hope to approximate the quantity. Thus, putting $\theta = \sin^{-1}\sqrt{\frac{a}{N}}$, we get,

$$k \approx \frac{\pi}{4\sin^{-1}\sqrt{\frac{a}{N}}} - \frac{1}{2}$$

Thus, since k must be an integer,

$$k = \lfloor \frac{\pi}{4 \sin^{-1} \sqrt{\frac{a}{N}}} \rfloor$$

Note that k depends on angle θ which in turn depends on a which is the number of solutions. In general, we might not know how many solutions there are so we can't pick the value of a and thus, don't know the number of iterations k to perform.

5.4.1 Unique Search Problem

This is the case which Lov Grover considered in his original paper in 1996 where he introduced Grover's Algorithm. Suppose for simplicity, which also turns out to be the worst case we would have $a = 1$ i.e. only one unique solution string x satisfying such that $f(x) = 1$ and hence, $\sin^{-1} \theta \approx \theta \implies \sin^{-1} \frac{1}{\sqrt{N}} \approx \frac{1}{\sqrt{N}}$.

$$k \approx \frac{\pi}{4}\sqrt{N} - \frac{1}{2}$$

$$k \approx \frac{\pi}{4}\sqrt{N}$$

Thus, we use $k = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ for our algorithm. Now, recall that the operator G includes the operator Z_f which queries the black box (oracle) B_f. Thus, the number of queries needed by the algorithm is equal to the number of times we use the Grover operator which is also k and thus, the queries required for the algorithm is $\mathcal{O}(\sqrt{N})$.

Note that $a = 1$ represents only a special case which is coincidentally also the worst case for the searching algorithm. Thus, with the choice of $k = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$, the probability of finding the single x such that $f(x) = 1$ is

$$\sin^2 \left(\left(2 \lfloor \frac{\pi}{4}\sqrt{N} \rfloor + 1 \right) \sin^{-1} \frac{1}{\sqrt{N}} \right)$$

Thus, the limit of this probability is 1 as N goes to infinity, and it is at least 1/2. This is as shown in the following table 5.2

N	Probability
2	0.5000000000
4	1.0000000000
8	0.9453125000
16	0.9613189697
32	0.9991823155
64	0.9965856808
128	0.9956198657
256	0.9999470421
512	0.9994480262
1024	0.9994612447
2048	0.9999968478
4096	0.9999453461
8192	0.9999157752
16384	0.9999997811
32768	0.9999868295
65536	0.9999882596
131072	0.9999992587
262144	0.9999978382
524288	0.9999997279

Table 5.2: Unique Search ($a = 1$): Success Probability at various N

So even in the worst case, repeating the algorithm some small constant number of times and evaluating f at the output each time will find the unique x such that $f(x) = 1$ with very high probability.

Notice that these probabilities are not strictly increasing. In particular, we have an interesting anomaly when $N = 4$, where we get a solution with certainty. It can be proved analytically that

$$p(N, 1) \geq 1 - \frac{1}{N}$$

where $p(N, 1)$ is the probability of finding the required search string x upon measurement. Notice that even a weak bound such as $p(N, 1) \geq 1/2$ establishes the utility of Grover's algorithm. For whatever measurement outcome x we obtain from running the procedure, we can always check to see if $f(x) = 1$ using a single query to f. And if we fail to obtain the unique string x for which $f(x) = 1$ with probability at most $1/2$ by running the procedure once, then ater m independent runs of the procedure we will have failed to obtain this unique string x with probability at most 2^{-m}, That is using $\mathcal{O}(m\sqrt{N})$ queries to f, we will obtain the unique solution x with probability

at least $1-2^{-m}$. Using the better bound $p(N,1) \geq 1-\frac{1}{N}$ reveals that the probability to find $s \in A$ using this method is actually at least $1-N^{-m}$. Thus, as can be seen that the probabilities for the case $a=1$ approaches 1 as we increase N.

5.4.2 Multiple Solutions

Now we discuss the general case. In the case when we do not know that $a=1$, the situation is more challenging. For example, if $a=4$ but we still choose $k=\lfloor\frac{\pi}{4}\sqrt{N}\rfloor$, the probability of success is given as:

$$\sin^2\left(\left(2\lfloor\frac{\pi}{4}\sqrt{N}\rfloor+1\right)\sin^{-1}\sqrt{\frac{a}{N}}\right)$$

thus, the success probability goes to 0 in the limit of large N as seen in table 5.3.

N	Probability
4	1.0000000000
8	0.5000000000
16	0.2500000000
32	0.0122070313
64	0.0203807689
128	0.0144530758
256	0.0000705058
512	0.0019310741
1024	0.0023009083
2048	0.0000077506
4096	0.0002301502
8192	0.0003439882
16384	0.0000007053
32768	0.0000533810
65536	0.0000472907
131072	0.0000030066
262144	0.0000086824
524288	0.0000010820

Table 5.3: Multiple Solutions: Success Probability at various N

This is because we are effectively rotation twice as far as we should. Generalizing what was claimed earlier, it can be proved that

$$p(N,a) \geq 1-\frac{a}{N}$$

This lower bound of $1 - \frac{a}{N}$ on the probability of success is slightly peculiar in that more solutions implies a worse lower bound - but under the assumption that a is significantly smaller than N, nevertheless conclude that the probability of success is reasonably high. As before, the mere fact that $p(N, a)$ is reasonably large implies the algorithm's usefulness. It is also the case that

$$p(N, a) \geq \frac{a}{N}$$

This lower bound describes the probability that a string $x \in \{0, 1\}^n$ selected uniformly at random is a solution - so Grover's algorithm always does at least as well as random guessing. In fact, Grover's algorithm is random guessing when $k = 0$. As the number of elements a varies, so does he angle θ, which can have a significant effect on the algorithm's probability of success. In case we know a in advance and we choose the corresponding value of k as

$$k = \lfloor \frac{\pi}{4 \sin^{-1} \sqrt{\frac{a}{N}}} \rfloor$$

then the probability $p(N, a)$ for finding one of the solutions upon measuring is given by

$$p(N, a) \geq max\{1 - \frac{a}{N}, \frac{a}{N}\}$$

thus, as a/N goes smaller and smaller the probability approaches 1. We now need to see how does the number of queries k depend on N and a.

$$sin^{-1}(x) \geq x \quad \text{for every } x \in [0, 1]$$

$$\begin{aligned} \theta &= \sin^{-1}\left(\sqrt{\frac{a}{N}}\right) \geq \sqrt{\frac{a}{N}} \\ \implies k &\leq \frac{\pi}{4\theta} \leq \frac{\pi}{4}\sqrt{\frac{N}{a}} \\ k &= \mathcal{O}\left(\sqrt{\frac{N}{a}}\right) \end{aligned}$$

In the case when the number of solutions is unknown i.e. we don't know the value of a. When we don't know the number of solutions, there are different strategies for dealing with this problem. One possibility is simply choose a random value uniformly for

$$k \in \{1, \ldots, \lfloor \frac{\pi}{4}\sqrt{N} \rfloor\}$$

The idea is that we are giving the state vector $|X\rangle$ a random rotation. This does not make the probability of getting a solution close to 1 but the probability of finding a solution (if one exists) will be at least 40%. We can now boost the probability of getting the correct solution exponentially by repeating this process choosing k independently each time. The chance of not finding a solution decreases exponentially in the number of time we repeat. For example to increase probability to 99% we can run this algorithm 10 times with k chosen independently each time. The number of queries is still $\mathcal{O}(\sqrt{N})$ (or evaluations of f) even when we don't know the number of solutions in advance. Still operating under the assumption $a \geq 1$, the algorithm will fail to find an x with $f(x) = 1$ with probability at most 3/4. Repeating some constant number of times until a "good" x is found quickly decreases error.

A somewhat better strategy is to apply the method above for successively larger values. Specifically, instead of choosing a random k in the range $1, \ldots, \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$, do the following:

Algorithm 2 Modified Grover's Algorithm for $a \geq 1$

1: Set $m = 1$.
2: Choose $k \in \{1, \ldots, m+1\}$ uniformly at random 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 "fail". Else set $m = \lfloor (8/7)m \rfloor$ and goto step 2.

This will succeed in finding $x \in A$ with probability at least 1/4 after $\mathcal{O}(\sqrt{\frac{N}{a}})$ queries. By repeating step 2 a constant number of times during each iteration of the loop, the error decreases exponentially. (the number 8/7 just happens to be a number that works for the analysis, which we will not discuss. The point is that m increases exponentially, but not too fast to cause the algorithm to fail). Another choice is to use $\lfloor \frac{5}{4}k \rfloor$. The rate of increase of k must be carefully balanced; slower rates require more queries, higher rates decrease success probability. If the number of solutions is $a \geq 1$, then the number of queries (or evaluations of f) required is $\mathcal{O}(\sqrt{N/a})$. If there are no solutions $\mathcal{O}(\sqrt{N})$ queries are required.

5.4.3 Trivial Case

Finally we need to deal with the special cases when $a = 0$ or $b = 0$. Obviously if $a = 0$, the output x of the algorithm will never satisfy $f(x) = 1$ because there are no such values of x. It is easy to check directly that Grover's Algorithm will output a choice of $x \in \{0, 1\}^n$ that is uniformly distributed in this case. The situation where

one of A and B is empty happens when f is constant; A is empty when $f(x) = 0$ for every $x \in \{0,1\}^n$, and B is empty when $f(x) = 1$ for every $x \in \{0,1\}^n$. This means that

$$Z_f |h\rangle = \pm |h\rangle$$

and therefore

$$\begin{aligned} G |h\rangle &= (2 |h\rangle \langle h| - I) Z_f |h\rangle \\ &= \pm(2 |h\rangle \langle h| - I) |h\rangle \\ &= \pm |h\rangle \end{aligned}$$

So, irrespective of the number of iterations k we perform in these cases, the measurement always reveals a uniform random string $x \in \{0,1\}^n$. Supposing that we do not know a, if we run Grover's Algorithm as described above and always measure a value of x for which $f(x) = 0$, we can reasonable conclude with high confidence that $a = 0$. It is also the case when $b = 0$ that Grover's Algorithm will output a uniformly distributed choice of x. Of course in this case $f(x) = 1$ for all x, so the problem is trivially solved.

5.5 Conclusion

Grover's algorithm is asymptotically optimal within the query model. It means that it is not possible to come with an algorithm for the search problem (unique or multiple solutions) that uses asymptotically fewer than $\mathcal{O}(\sqrt{n})$ queries in the worst case. This has been proved rigorously and was known even before Grover's algorithm was discovered. This algorithm is broadly applicable as it can be used as a subroutine in other quantum algorithms to use its quadratic speedup on top of other algorithms. The technique used in Grover's algorithm has been generalized and this is called Amplitude Amplification which is then applied to another quantum algorithms to essentially boost its success probability quadratically then it would have been possible classically.

5.6 Amplitude Amplification

5.7 Applications of Grover's Algorithm

5.7.1 Application: Satisfiability

5.8 Implementation using Qiskit

5.8.1 Background

Usage estimate: 4 seconds (NOTE; This is an estimate only. Your runtime may vary)
Amplitude amplification is a general purpose quantum algorithm, or subroutine, that can be used to obtain a quadratic speedup over a handful of classical algorithms. Grover's algorithm was the first to demonstrate this speedup on unstructured search problems. Formulating a Grover's search problem requires an oracle function that marks one or more computational basis states as the states we are interested in finding, and an amplification circuit that increase the amplitude of marked states, consequently suppressing the remaining states.

Here, we demonstrate how to construct Grover oracles and use the *GroverOperator* from the Qiskit circuit library to easily set up a Grover's search instance. The runtime *Sampler* primitive allows seamless execution of Grover circuits.

5.8.2 Requirements

Before starting this tutorial, enusre that you have the following installed:

- Qiskit SDK 1.0 or later, with visualization support (pip install 'qiskit[visualization]')
- Qiskit Runtime (pip install qiskit-ibm-runtime) 0.22 or later

5.8.3 Setup

Here we import the small number of tools we need for this tutorial.

```
# Built-in modules
import math

# Imports from Qiskit
from qiskit import QuantumCircuit
```

```
from qiskit.circuit.library import GroverOperator, MCMT, ZGate
from qiskit.visualization import plot_distribution

# Imports from Qiskit Runtime
from qiskit_ibm_runtime import QiskitRuntimeService
from qiskit_ibm_runtime import SamplerV2 as Sampler
```

```
# To run on hardware, select the backend with the fewest number of jobs
    in the queue
service = QiskitRuntimeService(channel="ibm_quantum")
backend = service.least_busy(operational=True, simulator=False)
backend.name
```

This output one of the ibm quantum computers (say 'ibm_kyoto').

1. **Step 1: Map classical inputs to a quantum problem**:
   Grover's algorithm requires an oracle that specifies one or more marked computational basis states, where "marked" means a state with a phase of -1, A controlled-Z gate, or its multi-controlled generation over N qubits, marks the $2^N - 1$ state ('1' * N bit-string). Marking basis states with one or more '0' in the binary representation requires applying X-gates on the corresponding qubtis before and after the controlled-Z gate; equivalent to having an open-control on that qubit. In the following code, we define an oracle tht does just that, marking one or more input basis states defined through their bit-string representation. The MCMT gate is used to implement the multi-controlled Z gate.

   ```
   def grover_oracle(marked_states):
   """Build a Grover oracle for multiple marked states

   Here we assume all input marked states have the same number of
       bits

   Parameters:
       marked_states (str or list): Marked states of oracle

   Returns:
       QuantumCircuit: Quantum circuit representing Grover oracle
   """
   if not isinstance(marked_states, list):
       marked_states = [marked_states]
   # Compute the number of qubits in circuit
   num_qubits = len(marked_states[0])

   qc = QuantumCircuit(num_qubits)
   ```

```
# Mark each target state in the input list
for target in marked_states:
    # Flip target bit-string to match Qiskit bit-ordering
    rev_target = target[::-1]
    # Find the indices of all the '0' elements in bit-string
    zero_inds = [ind for ind in range(num_qubits) if rev_target.
    startswith("0", ind)]
    # Add a multi-controlled Z-gate with pre- and post-applied X-
    gates (open-controls)
    # where the target bit-string has a '0' entry
    qc.x(zero_inds)
    qc.compose(MCMT(ZGate(), num_qubits - 1, 1), inplace=True)
    qc.x(zero_inds)
return qc

```

Specific Grover's Instance

Now that we have the oracle function, we can define a specific instance of Grover search. In this example we will mark two computational state out of the eight available in a three-qubit computational space:

```
marked_states = ["011", "100"]

oracle = grover_oracle(marked_states)
oracle.draw(output="mpl", style="iqp")
```

This output is as shown in the following figure 5.5

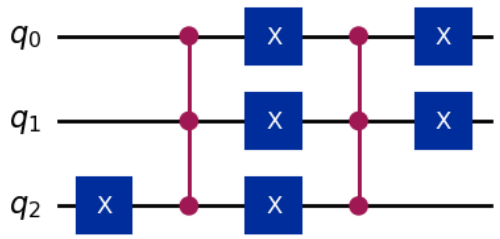


Figure 5.5: Specific Grover's Instance

GroverOperator

The built-in Qiskit GroverOperator takes an oracle circuit and returns a circuit that is composed of the oracle circuit and a circuit that amplifies the states

marked by the oracle. Here, we decompose the circuit to see the gates within the operator:

```
grover_op = GroverOperator(oracle)
grover_op.decompose().draw(output="mpl", style="iqp")
```

This outputs as shown in the figure 5.6.

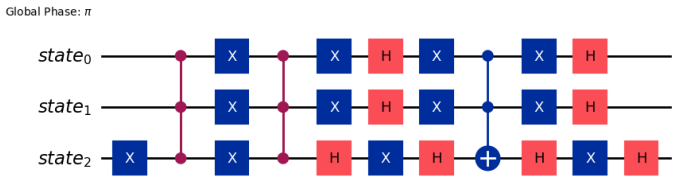


Figure 5.6: Grover Operator

Repeated application of this grover_op circuit amplify the marked states, making them the most probable bit-strings in the output distribution from the circuit. There is an optimal number of such application that is determined by the ratio of marked states to the total number of possible computational states:

```
optimal_num_iterations = math.floor(
math.pi / (4 * math.asin(math.sqrt(len(marked_states) / 2**
    grover_op.num_qubits)))
)
```

Full grover circuit

A complete Grover experiment starts with a Hadamard gate on each qubit; creating an even superposition of all computational basis states, followed the Grover operator (grover_op) repeated the optimal number of times. Here we make use of the QuantumCircuit.power(INT) method to repeatedly apply the Grover operator.

```
qc = QuantumCircuit(grover_op.num_qubits)
# Create even superposition of all basis states
qc.h(range(grover_op.num_qubits))
# Apply Grover operator the optimal number of times
qc.compose(grover_op.power(optimal_num_iterations), inplace=True)
# Measure all qubits
qc.measure_all()
qc.draw(output="mpl", style="iqp")
```

This output the following figure 5.7

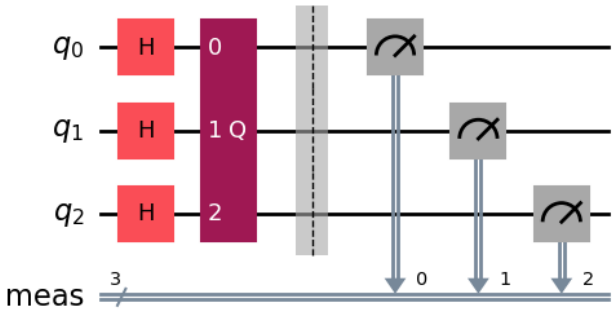


Figure 5.7: Full Grover Circuit

2. **Step 2: Optimize problem for Quantum Execution**

```
from qiskit.transpiler.preset_passmanagers import
    generate_preset_pass_manager

target = backend.target
pm = generate_preset_pass_manager(target=target,
    optimization_level=3)

circuit_isa = pm.run(qc)
circuit_isa.draw(output="mpl", idle_wires=False, style="iqp")
```

This outputs the following 5.8.

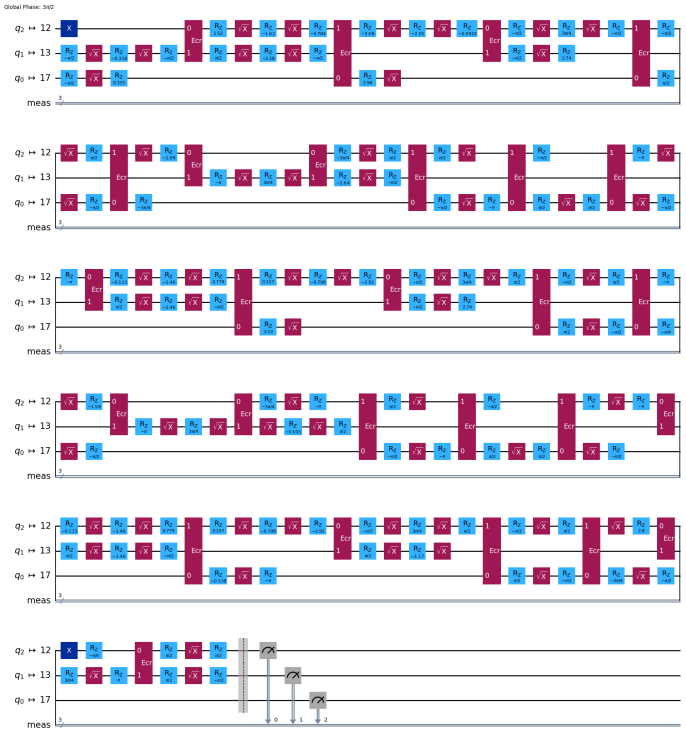


Figure 5.8: Optimized Circuit

3. **Step 3: Execute using Qiskit Primitives**
   Amplitude amplification is a sampling problem that is suitable for execution with the Sampler runtime primitive.

   Note that the run() method of Qiskit Runtime SampleV2 takes an iterable of primitive unified blocs (PUBs). For sampler, each PUB is an iterable in the format (circuit, parameter_values). However, at a minimum, it takes a list of quantum circuit(s).

```
# To run on local simulator:
#   1. Use the SatetvectorSampler from qiskit.primitives instead
sampler = Sampler(backend=backend)
sampler.options.default_shots = 10_000
result = sampler.run([circuit_isa]).result()
dist = result[0].data.meas.get_counts()
```

4. **Step 4: Post-Process, return result in classical format**

```
plot_distribution(dist)
```

The result upon measurement is as shown in the figure 5.9

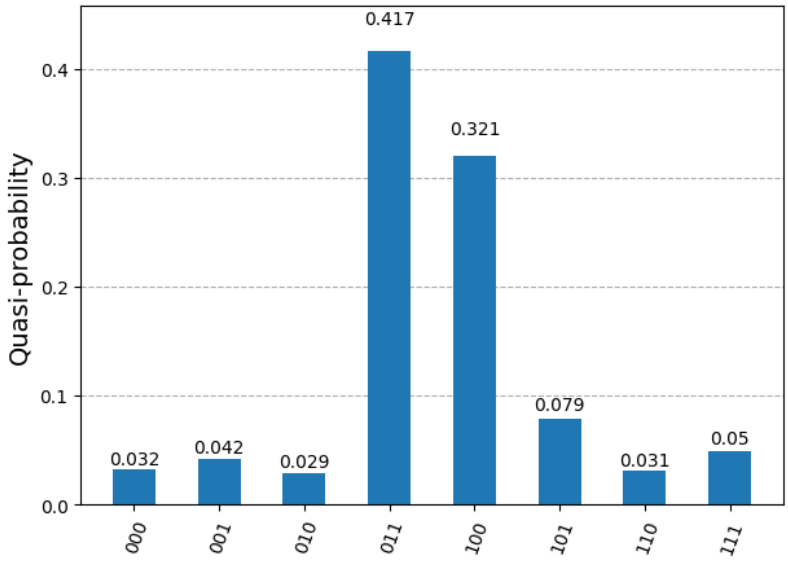


Figure 5.9: Probability plot

Chapter 6

Phase Estimation

We know that any classical Boolean circuit can be converted to a reversible (therefore unitary) circuit that efficiently implements the function computed by the original circuit. For example, if the original Boolean circuit computes the Function

$$f : \{0,1\}^n \to \{0,1\}^m$$

then the reversible circuit efficiently implements the transformation

$$U_f : |x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$$

for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$, possibly using some ancilla qubits that we do not mention explicity. Moreover, if

$$f : \{0.,1\}^n \to \{0,1\}^n$$

is invertible and we have a Boolean ciruict for efficiently computing both f and f^{-1}, then we can construct an efficient reverisble circuit that performs the transformation

$$P_f |x\rangle = |f(x)\rangle$$

for all $x \in \{0,1\}^n$.

6.1 Phase Estimation

Suppose we are given a Quantum circuit Q which acts on n qubits (i.e. takes input as n qubits and obvviously because of reversibility and unitary transformation outputs n qubits). Thus, corresponding to that Circuit's Unitary transformation we have a

Unitary matrix U of dimensions $2^n \times 2^n$. Now, recall that because U is unitary, we know that it has a complete, orthonormal collection of eigenvectors

$$|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_{2^n}\rangle$$

with corresponding eigenvalues (Recall, that the eigen values of a Unitary matrix are complex numbers of unit Modulus (refer Appendix A.1.6) and thus can be written in the form $e^{2\pi i\theta}$ where θ is a real number.)

$$e^{2\pi i\theta_1}, e^{2\pi i\theta_2}, \ldots, e^{2\pi i\theta_{2^n}}$$

where $\theta_1, \theta_2, \ldots, \theta_{2^n}$ are real numbers. This means that

$$U|\psi_j\rangle = e^{2\pi\iota\theta_j}|\psi_j\rangle$$

for all $j = 1, 2, \ldots, 2^n$. Thus, now we define the **PHASE ESTIMATION PROBLEM** as follows:
PHASE ESTIMATION PROBLEM

Problem 6.1.1. *Input: A quantum circuit Q that performs a unitary operation U, along with wa quantum state $|\psi\rangle$ that is promised to be an eigenvector of U:*

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$$

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

Remark. Note that we have made no specific requirements on the precision to which θ must be approximated. It will urn out that for an arbitrary Quantum circuit Q and eigen vector $|\psi\rangle$, the number θ can be efficiently approximated by the procedure that we will describe, but only to low precision (to a logarithimic number of bits in the circuit size). However, for certain choices of U it will be possible to achieve much higer precision, and when we apply our methods to factoring this is the case in which we will be inteested.

6.1.1 Clasical Method

Using a classical computer, we can estimate θ using $U|\psi\rangle \oslash |\psi\rangle$, where $\oslash$ stands for the elemnt-wise division operation. Specifically, if $|\psi\rangle$ is indeed an eigen vector and $\langle j|\psi\rangle \neq 0$, for any j in the computational basis, then we can extract the phase from

$$\langle j|U|\psi\rangle / \langle j|\psi\rangle = e^{2\pi i\theta}$$

Unfortunately, such an element-wise division operation cannot be efficiently implemented on a quantum computer. In this chapter we look at some of the most basic variants.

6.1.2 Hadamard Test

For computing the expectation value of an Unitary operator with respect to a general state (not necessarily an eigen vector state) i.e. $\langle\psi| U |\psi\rangle$. Note that U is Unitary and not necessarily Hermitian, this doe snot correspond to the measurmenet of a physical observable. Instead the real and imaginary part of the expectation value need to be measured separately. The (real) Hadamard test is a quantum algorithm that computes the real part of the expectation value of a unitary operator with respect to a state. The circuit is as shown in the figure 6.1

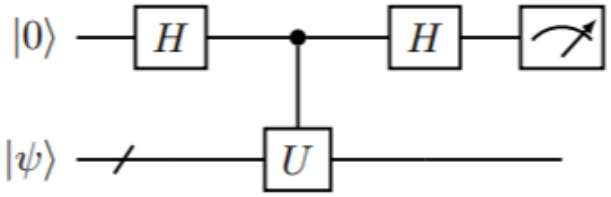


Figure 6.1: Hadamard Test

Note that the cirucit transforms $|0\rangle |\psi\rangle$ as:

$$(H \otimes I)(|0\rangle |\psi\rangle) \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) |\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle |\psi\rangle) + \frac{1}{\sqrt{2}}(|1\rangle |\psi\rangle)$$

Then perfoming a Controlled Unitary operation:

$$(I \otimes c - U)(\frac{1}{\sqrt{2}}(|0\rangle |\psi\rangle) + \frac{1}{\sqrt{2}}(|1\rangle |\psi\rangle)) = \frac{1}{\sqrt{2}}(|0\rangle |\psi\rangle + |1\rangle U |\psi\rangle)$$

Now applying the Hadamard transformation ont the first qubit we get:

$$(H \otimes I)\frac{1}{\sqrt{2}}(|0\rangle |\psi\rangle + |1\rangle U |\psi\rangle) \rightarrow \frac{1}{\sqrt{2}}(H |0\rangle |\psi\rangle + H |1\rangle U |\psi\rangle)$$

$$\frac{1}{\sqrt{2}}(H |0\rangle |\psi\rangle + H |1\rangle U |\psi\rangle) = \frac{1}{2}(|0\rangle (|\psi\rangle + U |\psi\rangle) + |1\rangle (|\psi\rangle - U |\psi\rangle))$$

Because the second qubit is not normalized as $\| |\psi\rangle + U |\psi\rangle \| \neq 1$, thus we normalize it as shown (Recall that for normalizing we divide by the norm of the vector given

as $\sqrt{\langle\psi|\psi\rangle}$). Thus, here we use $\sqrt{\langle(I+U^\dagger)(I+U)\,|\psi\rangle|}$ and upon solving for this we get the following:

$$\frac{\sqrt{2+2Re(\langle\psi|\,U\,|\psi\rangle)}}{2}\left(|0\rangle\otimes\frac{(|\psi\rangle+U\,|\psi\rangle)}{\sqrt{2+2Re(\langle\psi|\,U\,|\psi\rangle)}}\right)+$$
$$\frac{\sqrt{2-2Re(\langle\psi|\,U\,|\psi\rangle)}}{2}\left(|1\rangle\otimes\frac{(|\psi\rangle-U\,|\psi\rangle)}{\sqrt{2-2Re(\langle\psi|\,U\,|\psi\rangle)}}\right)$$

Now upon measurement of the first qubit, the probabiltiy of measuring the first qubit to be 0 is:

$$p(0)=\frac{2+2Re(\langle\psi|\,U\,|\psi\rangle)}{4}=\frac{1}{2}(1+Re(\langle\psi|\,U\,|\psi\rangle))$$

which is very well defined since $-1\leq Re(\langle\psi|\,U\,|\psi\rangle)\leq 1$.

To obtain the imaginary part we can use the circuit as shown in the figure 6.2 called the (imaginary) Hadamard test.

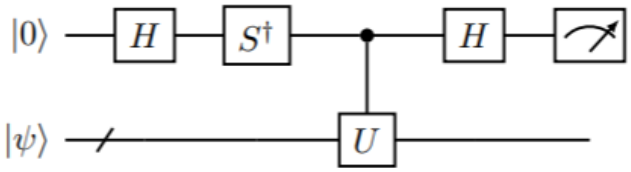


Figure 6.2: Hadamard Test

Similarly doing similar calculations as done above the circuit transforms $|0\rangle\,|\psi\rangle$ to the state:

$$\frac{1}{2}\,|0\rangle\,(|\psi\rangle\,\iota U\,|\psi\rangle)+\frac{1}{2}(|\psi\rangle+\iota U\,|\psi\rangle)$$

Therefore the probability of measuring the first qubit in the state $|0\rangle$ is:

$$\frac{1}{2}(1+Im(\langle\psi|\,U\,|\psi\rangle))$$

Thus, combining the results of the two circuits, we obtain the estimate to $\langle\psi|\,U\,|\psi\rangle$.

Overlap Estimate using Swap Test

it is an application of the Hadamard test called the swap test, which is used to estimate the overlap of two quantum states $\langle\phi|\psi\rangle$. The quantum circuit for the swap test is as shown in the figure 6.3.

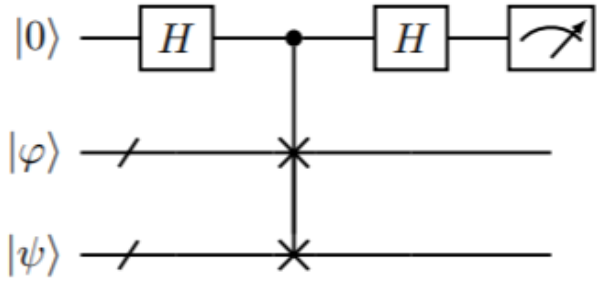


Figure 6.3: Swap Test

Note that it is exactly the Hadamard test with U being the n-swap gate. Using the sme calculations as done previously, the state of the system after the first hadamard gate acting on $|0\rangle |\phi\rangle |\psi\rangle$ is:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) |\phi\rangle |\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle |\phi\rangle |\psi\rangle + |1\rangle |\phi\rangle |\psi\rangle)$$

Then the controlled swap gate acts on the state as:

$$\frac{1}{\sqrt{2}}(|0\rangle |\phi\rangle |\psi\rangle + |1\rangle |\psi\rangle |\phi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle |\phi\rangle |\psi\rangle + |1\rangle |\psi\rangle |\phi\rangle)$$

Then the second Hadamard gate acts on the first qubit as:

$$\frac{1}{2}(|0\rangle |\phi\rangle |\psi\rangle + |0\rangle |\psi\rangle |\phi\rangle - |1\rangle |\psi\rangle |\phi\rangle + |1\rangle |\phi\rangle |\psi\rangle)$$

$$= \frac{\sqrt{2+2|\langle\psi|\phi\rangle|^2}}{2}\left(|0\rangle \otimes \frac{|\phi\rangle |\psi\rangle + |\psi\rangle |\phi\rangle}{\sqrt{2+2|\langle\psi|\phi\rangle|^2}}\right) + \frac{\sqrt{2-2|\langle\psi|\phi\rangle|^2}}{2}\left(|1\rangle \otimes \frac{|\phi\rangle |\psi\rangle - |\psi\rangle |\phi\rangle}{\sqrt{2-2|\langle\psi|\phi\rangle|^2}}\right)$$

Now, upon measurement of the first qubit, the probability of measuring the first qubit to be 0 is:

$$p(0) = \frac{1}{2}(1 + |\langle\phi|\psi\rangle|^2)$$

Overlap estimate with Relative phase information

In the swap test, the quantum states $|\phi\rangle$ and $|\psi\rangle$ can be black-box states, and in such a scenario obtaining an estimate to $|\langle\phi|\psi\rangle|$ is the best one can do. In order to

retrieve the relative phase information and to obtain $\langle\phi|\psi\rangle$, we need to have access to the unitary circuit preparing $|\phi\rangle$ and $|\psi\rangle$.

$$U_\phi |0^n\rangle = |\phi\rangle\,, \quad U_\psi |0^n\rangle = |\psi\rangle$$

Then we have $\langle\phi|\psi\rangle = \langle 0^n| U_\phi^\dagger U_\psi^\dagger |0^n\rangle$. Now putting $U = U_\phi U_\psi$ in the (real) and (imegaingary) Hadamard test and putting the state $|0^n\rangle$ in the second register, thus calculating the expectation value, we can obtain the real and imaginary parts of $\langle\phi|\psi\rangle$.

Single Phase Estimtation

The hadamard test can be also be used to derive the simplest version of the phase estimation based on success probabilities. Apply the Hadamrad test as shown in the figure 6.1. with U, ψ satisfying the equation $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$. Then the probability of measuring the first qubit to be in state $|1\rangle$ is:

$$p(1) = \frac{1}{2}(1 - Re(e^{2\pi i\theta})) = \frac{1}{2}(1 - cos(2\pi\theta))$$

Therefore,

$$\theta = \pm\frac{arccos(1-2p(1))}{2\pi} \pmod 1$$

In order to quantify the efficiency of this procedure recall that if $p(1)$ is far away from 0 (or close to 1) i.e. $(2\theta \bmod 1)$ is far away from 0, in order to approimate $p(1)$ (and hence θ) to additive precision ϵ, the number of samples need is $\mathcal{O}(1/\epsilon^2)$ using the formula dervied from statistics (If we know that the count X of "successes" in a group of n observations with sucess probability p has a binomial distribution with mean np and variance np(1-p), then we are able to derive information about the distribution of the sample proportion, the count of successes X divided by the number of observations n. By the multiplicative properties of the mean, the mean of the distribution of X/n is equal to the mean of X divided by n, or np/n = p. This proves that the sample proportion is an unbiased estimator of the population proportion p. The variance of X/n is equal to the variance of X divided by n^2, or $(np(1-p))/n^2 = (p(1-p))/n$. This formula indicates that as the size of the sample increases, the variance decreases.):

$$N \geq \frac{p(1-p)}{\epsilon^2}$$

Now assume that θ is very close to 0 and we would like to estimate θ to additive precision ϵ. Note that

$$p(1) \approx (2\pi\theta)^2 = \mathcal{O}(\epsilon^2)$$

Then $p(1)$ needs to be estimated to precision $\mathcal{O}(\epsilon^2)$, and again the number of samples needed is $\mathcal{O}(1/\epsilon^2)$. The case when θ is very close to $1/2$ or 1 is similar.

Note that the circuit in figure 6.1 cannot distinguish the sign of θ (or whether $\theta \geq 1/2$ when restricted to the interval $[0, 1)$). To do this, we can use the circuit in figure 6.2 by replacing $S^\dagger$ with S, so that the success probability of measuring 1 in the computational basis is

$$p(1) = \frac{1}{2}(1 + \sin(2\pi\theta))$$

This gives

$$\theta = \begin{cases} \in [0, 1/2), & p(1) \geq \frac{1}{2} \\ \in (1/2, 1), & p(1) < \frac{1}{2} \end{cases}$$

Unlike previous estimate, in order to correctly estimate the sign, we only require $\mathcal{O}(1)$ accuracy, and run figure 6.2 for a constant number of times (unlesss θ very close to 0 or π).

6.1.3 Kitaev's Method - for Quantum Phase Estimation

The number of measurments needed to estimate θ to precision ϵ is $\mathcal{O}(1/\epsilon^2)$. A quadractic improvement in precision (i.e.$\mathcal{O}(1/\epsilon)$) can be achieved by means of the quantum phase estimation using Kitaev's method. Assuming that the eigen value can be respresented using d bits i.e.,

$$\theta = (.\theta_{d-1}...\theta_0)$$

In the simplest scenario, we assume $d = 1$, and $\theta = .\theta_0, \theta_0 \in \{0, 1\}$. Thus, if $\theta_0 = 0$ then $e^{2\pi\iota 0} = e^{\pi\iota 0} = 1$ and if $\theta_0 = 1$ then $e^{2\pi\iota 0.5} = e^{\pi\iota}$. Then $e^{2\pi\iota\theta} = e^{\pi\iota\theta_0}$. Performing the real Hadamard test, we have $p(1) = \frac{1}{2}(1 - Re(e^{\pi\iota\theta_0})) = \frac{1}{2}(1 - \cos(\pi\theta_0))$.

Now, $p(1) = 0$ if $\theta_0 = 0$ and $p(1) = 1$ if $\theta_0 = 1$. In, either case the result is deteministic, and one measurment is sufficient to detemine the value of θ_0.

Next, consider $\theta = 0.\underbrace{0...0\theta_0}_{dbits}$. To determine the value of θ_0 we need to reach precision of $\epsilon < 2^{-d}$. The method in Single Phase qubit estimation requires $\mathcal{O}(1/\epsilon^2) = \mathcal{O}(2^{2d})$ repeated measusmrents or number of queries to U. The observation from Kitaev's method is that if we can have access to U^j for a suitable power of

j, then the number of queries to U can be reduced. More specifcally, if we can query $U^{2^{d-1}}$, then the circuit in figure 6.4 with $j = d-1$ gives

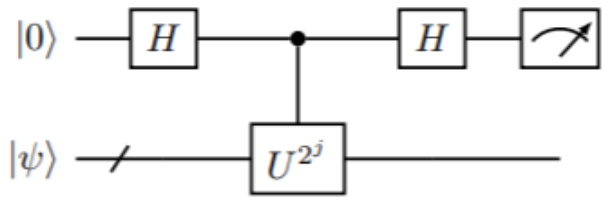


Figure 6.4: Kitaev's Method

$$p(1) = \frac{1}{2}(1 - Re(\langle\psi| U^{2^{d-1}} |\psi\rangle) = \frac{1}{2}(1 - Re(e^{2\pi\iota 2^{d-1}\theta}))$$

Now the value of $2^{d-1}0.0..0\theta_0 = \theta_0/2 = 0.\theta_0$(in binary):

$$p(1) = \frac{1}{2}(1 - \cos(2\pi.\theta_0)) = \begin{cases} 0, & \theta_0 = 0 \\ 1, & \theta_0 = 1 \end{cases}$$

This is the basic idea behind Kitaev's method: use a more complex circuit (and in particular, with a larger circuit depth) to reduce the total number of queries. As a general strategy, instead of estimating θ from a single number, we assume access to U^{2^j}, and estimate θ bit-by-bit.

$$2^j\theta = \theta_{d-1}...\theta_{d-j}.\theta_{d-j-1}...\theta_0 = .\theta_{d-j-1}...\theta_0 \pmod 1$$

Note that now upon running the circuit we would get $p(1)$ thus we can estimate $.\theta_{d-j-1}...\theta_0 \pmod 1$. But, for values such as 0.0111 and 0.1000 we need to be careful in order to distinguish, in case of bit-by-bit estimation, and the two numbers can be arbitrarily close to each other. Here, we first describe the algorithm and then analyze its performance. The algorithm works for any θ and then the goal is to estimate its d bits. For simplicity of the ananlysis we assum that θ is exactly represented by d bits.

Now applying circuit in figure 6.4 for $j = 0, 1, ..., d-3$ and the corresponding circuit to determine its sign. For each j we can estimate $p(0)$,so that the error in $2^j\theta$ is less than 1/16 for all j. (This can happen with a sufficient high success probability. For simplicity, we assume that this happens with certaininty). The measured result is denoted by α_j. This means that any perturbation must

be due to the 5th digit in binary representation. For example, if $2^j\theta = 0.11100$, then $\alpha_j = 0.11011$ is an acceptable result with an error of $0.00001 = 1/32$, but $\alpha_j = 0.11110$ is not acceptable since the error is $0.0001 = 1/16$. We then round α_j (mod 1) by its closest 3-bit estimate denoted by β_j, i.e. β_j is taken from the set $\{0.000, 0.001, 0.010, 0.011, 0.100, 0.101, 0.110, 0.111\}$.

Consider the example of $2^j\theta = 0.11110$, if $\alpha_j = 0.11101$ then $\beta_j = 0.111$. But, if $\alpha_j = 0.11111$, then $\beta_j = 0.000$ is not required to approximate. Another example is $2^j\theta = 0.11101$, if $\alpha_j = 0.11110$ then both $\beta_j = 0.111$ (rounded down) and $\beta_j = 0.000$ (rounded up) are acceptable. We can pick one of them at random. We will show later that the uncertainity in α_j, β_j is not detrimental to the success of the algorithm.

Second, we perform post-processing. Start from $j = d - 3$, we can estimate $.\theta_2\theta_1\theta_0$ to accuracy $1/16$, which reconvers these three bits exactly. The values of these three bits will be taken from β_{d-3} directly. Then we proceed with the iteration for $j = d - 4, ..., 0$, we assign

$$\theta_{d-j-1} = \begin{cases} 0, & |.0\theta_{d-j-2}\theta_{d-j-3} - \beta_j| < 1/4 \\ 1, & |.1\theta_{d-j-2}\theta_{d-j-3} - \beta_j| > 1/4 \end{cases}$$

6.1.4 Phase Estimation Procedure

Suppose that U is a unitary transformation acting on n qubits (recall that dimensions of U will be $2^n \times 2^n$), and suppose m is any positive integer. Then we can define another unique unitary transformation $\Lambda_m(U)$ acting on $n + m$ qubits (thus of dimesnions $2^{m+n} \times 2^{m+n}$) as follows:

$$\Lambda_m(U) |k\rangle |\phi\rangle = |k\rangle (U^k |\phi\rangle)$$

for all choices of $k \in \{0, \ldots, 2^m - 1\}$. Here, $|k\rangle$ will be of dimensions 2^m and $|\phi\rangle$ will be of dimensions 2^n. Thus, $|k\rangle |\phi\rangle$ will be of dimensions $2^m 2^n = 2^{m+n}$ and thus the matrix $\Lambda_m(U)$ of dimensions $2^{m+n} \times 2^{m+n}$ acting on $|k\rangle |\phi\rangle$. In other words, the first m qubits specificy the number of times that U is to be applied to the remaining n qubits. This can be seen in the following circuit figure 6.5:

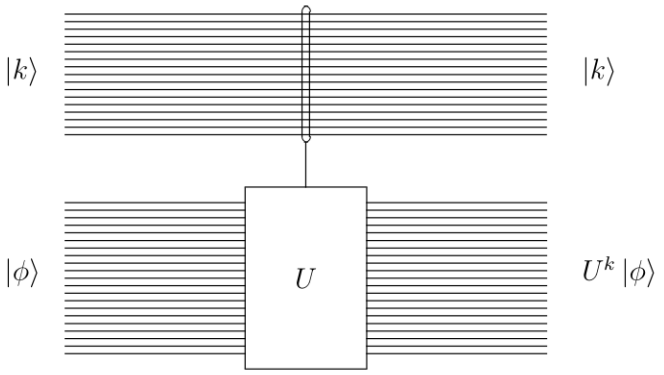


Figure 6.5: circuit

Important Note

Note that it is possible that n amd m differ significantly. Now, if given a Quantum circuit Q that performs the Unitary operation U, there is no gurantee that you will be able to build a quantum circuit that efficiently implements $\Lambda_m(U)$ for your choice of m. For example, if $m \approx n$, you would probably require that U has some special properties that allow an efficient implmementation. This is because the number of times (k) that U may effectively need to be performed can be exponential in m (since $k = \mathcal{O}(2^m)$), If $m = \mathcal{O}(\log n)$, you can always construct an efficient implementation of $\Lambda_m(U)$ (assuming that you have a circuit Q that efficiently implements U).
We are not concerned with the details of how one would construct $\Lambda_m(U)$ for small values of m given a circuit Q implementing U, because our interest will be foucsed on a particular choice of U where it is easy to implement $\Lambda_m(U)$, even for $m = \Theta(n)$. **Specifically, the transofrmation U will correpond to modular multiplication by a fixed number a,and so $\Lambda_m(U)$ will correspond to modular exponentation, which we have already show is effiently implementable.** Let us forget about the specifices and suppose that we have a quantum circuit implementing $\Lambda_m(U)$ for some particular choice of , (which may be large or small).

Consider the following circuit diagram 6.6:

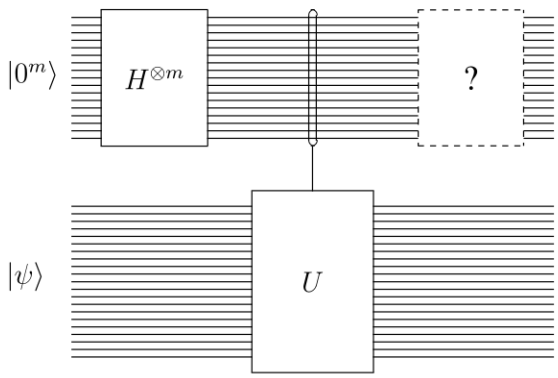


Figure 6.6: circuit

Note that the input to the second collection of qubits is the state $|\psi\rangle$, which is promised to be an eigenvector of U. Now consider the state after the $\Lambda_m(U)$ operation is performed. The initial state is $|0^m\rangle |\psi\rangle$, and after the hadamard transformations are performed the state becomes:

$$\frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} |k\rangle |\psi\rangle$$

Then the $\Lambda_m(U)$ transformation is performed, which transforms the state to

$$\frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} |k\rangle |U^k |\psi\rangle\rangle$$

Now, let us use the fact that $|\psi\rangle$ is an eigenvector of U to simplify this expression. Specificall we assume that

$$U |\psi\rangle = e^{2\pi\iota\theta} |\psi\rangle$$

for some real number $\theta \in [0, 1)$, which is the value that we are trying to approximate. Applying U^k to $|\psi\rangle$ is equivalent to applying U to $|\psi\rangle$ a total of k times, so

$$U^k |\psi\rangle = \left(e^{2\pi\iota\theta}\right)^k |\psi\rangle = e^{2\pi\iota k\theta} |\psi\rangle$$

Thus, the state after the $\Lambda_m(U)$ operation is performed is

$$\frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} |k\rangle e^{2\pi\iota k\theta} |\psi\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi\iota k\theta} |k\rangle |\psi\rangle$$

This is called **Phase kickback**. The first m qubits and the last n qubits are uncorrelated at this point, given that they are in tensor product state:

$$\left(\frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} e^{2\pi\iota k\theta}\,|k\rangle\right)\otimes|\psi\rangle$$

So, if we discard the lat n qubits we are left with the state:

$$\frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} e^{2\pi\iota k\theta}\,|k\rangle$$

Example 6.1.1. A simple case: Recall that our goal is to approximate θ. Suppose that θ happends to have a special form:

$$\theta = \frac{j}{2^m}$$

for some intger $j \in \{0,\ldots,2^m-1\}$. (This makes θ take only certain distinct values thus, in general, it is not correct to assume that θ has this form, but it is helpful to consider this case first.) Then the above statement can be rewritten as:

$$\frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} e^{2\pi\iota k\theta}\,|k\rangle = \frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} e^{2\pi\iota kj/2^m}\,|k\rangle$$

Now, let $\omega = e^{2\pi\iota/2^m}$, then the above expression can be written as:

$$\frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} \omega^{kj}\,|k\rangle$$

Now let us define:

$$|\phi_j\rangle = \frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1} \omega^{kj}\,|k\rangle$$

for each $j \in \{0,\ldots,2^m-1\}$. we know that the state of the first m qubtis of our circuit is one of the states $\{|\phi_j\rangle : j = 0,\ldots,2^m-1\}$ and the goal is to determine which one. **Recall that given a known set of states, an unknown state from that collection is given to you, and your goal is to determine which of the possible states it is. It is possible to solve the problem perfectly is the set**

of states is orthonormal.. Once we know j, we know θ as well (because we are still considering the special case $\theta = j/2^m$). We have

$$\langle \phi_j | \phi_{j'} \rangle = \frac{1}{2^m} \sum_{k=0}^{2^m-1} \omega^{k(j-j')} = \frac{1}{2^m} \sum_{k=0}^{2^m-1} \left(\omega^{j-j'}\right)^k$$

Note that when we take the inner product in the above equation only the terms for which $|k\rangle$ is same survive since the $|k\rangle$ forms an orthonormal basis. Thus, $\langle k|k'\rangle = \delta_{kk'}$. Also, for $\langle \phi_j|$ recall that because of the conjugate transpose operation the ω will be conjugated and thus $\omega^* = \left(e^{2\pi\iota\theta j/2^m}\right)^* = e^{-2\pi\iota\theta j/2^m} = \omega^{-j}$. Using the formula for geometric series for the term $\omega^{j-j'}$ we get:

$$\sum_{k=0}^{n-1} = \frac{x^n - 1}{x - 1}$$

Thus, we get:

$$\langle \phi_j | \phi_{j'} \rangle = \frac{1}{2^m} \frac{\omega^{(j-j')2^m} - 1}{\omega^{j-j'} - 1}$$

Now, since $\omega^{(2^m)l} = 1$ for any integer $l = j - j'$ (subtraction of two integers is an integer) we get:

$$\langle \phi_j | \phi_{j'} \rangle = \begin{cases} 1, & \text{if } j = j' \\ 0, & \text{if } j \neq j' \end{cases}$$

Thus, the set $\{|\phi_0\rangle, \ldots, |\phi_{2^m-1}\rangle\}$ is indeed orthonormal. Recall that we transformed from the basis $\{|0\rangle, \ldots, |2^m - 1\rangle\}$ to the basis $\{|\phi_j\rangle : j = 0, \ldots, 2^m - 1\}$. Now both the basis are orthonormal basis. Thus, this is a rotation in complext plane and thus a corresponding Unitary matrix exists. There is therefore a Unitary transformation F that satisfies

$$F|j\rangle = |\phi_j\rangle$$

for all $j \in \{0, \ldots, 2^m - 1\}$. Thus, the transformation F is the one that performs the rotation in the complex plane. We can describe this matrix explicitly by allowing the vecotrs $|\phi_j\rangle$ to determine the columns of D:

$$F = \begin{pmatrix} |\phi_0\rangle & |\phi_1\rangle & \ldots & |\phi_{2^m-1}\rangle \end{pmatrix}$$

$$F = \frac{1}{\sqrt{2^m}} \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{2^m-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(2^m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^m-1} & \omega^{2(2^m-1)} & \ldots & \omega^{(2^m-1)(2^m-1)} \end{pmatrix}$$

This is the matrix that defines a linear transformation in the complex plane that rotates the basis $\{|0\rangle, \ldots, |2^m - 1\rangle\}$ to the basis $\{|\phi_0\rangle, \ldots, |\phi_{2^m-1}\rangle\}$ (Note that the matrix is not Hermitian but symmetric. Also, it is Unitary since the columns are orthonormal, thus, $QFTQFT^\dagger = QFT^\dagger QFT = I \implies QFT^\dagger = QFT^{-1}$). $QFT^\dagger = QFT^{-1}$ since the matrix is also Unitary) . This is called the *Discrete Fourier Transform*. In the context of Quantum Computing we call it Quantum Fourier Transform, and for that reason it is also some times denoted as QFT_{2^m}. Thus, for convinence we can write it explicitly as:

$$QFT_{2^m} |j\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi\iota jk/2^m} |k\rangle = |\phi_j\rangle$$

Thus, plugging the inverse of this transfomration into our circuit from before, we obtain (using $QFT^\dagger = QFT^{-1}$):

$$QFT^\dagger_{2^m} |\phi_j\rangle = |j\rangle$$

The circuit for this is as shown in the figure 6.7

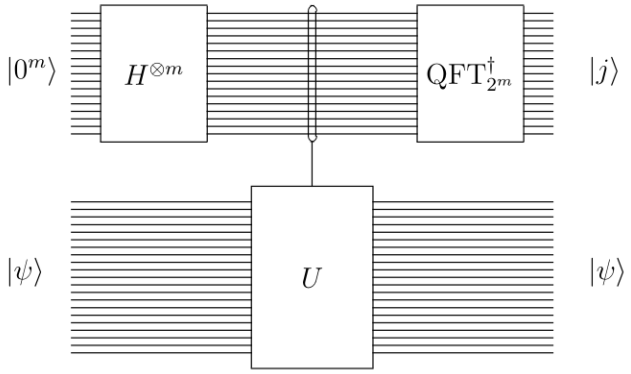


Figure 6.7: circuit

Thus, measuring the first m qubits and dividing by 2^m tell us precisely the value of θ. **But this assumes that:**

$$U |\psi\rangle = e^{2\pi\iota j/2^m} |\psi\rangle$$

We still need to prove this for the more general case that θ does not have the form $j/2^m$ for some integer j. The exact same circuit is used to deal with the general case, but the analysis will become more complicated (**and the answer will only be an approximation to** θ). The two major issues to deal with are:

1. What can be said about the measurement in the case that θ does not have the form $j/2^m$ for some integer j, and

2. Can the Quantum Fourier transform be implemented efficiently?

For the general values of θ

From the above analysis we know that for an arbitrary value of θ, the state of the above circuitimmediately before the inverse of the QFT is applied is

$$\frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi\iota k\theta} |k\rangle |\psi\rangle$$

Ignoring the second collection of qubits since it is uncorrelated with the first m qubits, so we are free to disregard these qubits and consider just the first m qubits. Applying the transformation $QFT_{2^m}^\dagger$ to these qubits results in the state:

$$\frac{1}{2^m} \sum_{k=0}^{2^m-1} e^{2\pi\iota k\theta} \sum_{j=0}^{2^m-1} e^{-2\pi\iota jk/2^m} |j\rangle$$

Note that in the above equation we have expanded the $QFT^\dagger |k\rangle$ in terms of the basis $\{|j\rangle\}$. Thus, upon simplifying we get:

$$\frac{1}{2^m} \sum_{k=0}^{2^m-1} \sum_{j=0}^{2^m-1} e^{2\pi\iota(k\theta - jk/2^m)} |j\rangle$$

Now rearranging and grouping the terms with respect to j we can rearrange the summations to get:

$$\sum_{j=0}^{2^m-1} \sum_{k=0}^{2^m-1} \frac{1}{2^m} e^{2\pi\iota(k\theta - jk/2^m)} |j\rangle$$

$$\sum_{j=0}^{2^m-1} \left(\sum_{k=0}^{2^m-1} \frac{1}{2^m} e^{2\pi\iota(k\theta - jk/2^m)} \right) |j\rangle$$

Thus, upon measurement the probability that the measurement result in outcome j is therefore.

$$p_j = \left| \frac{1}{2^m} \sum_{k=0}^{2^m-1} e^{2\pi\iota k(\theta - j/2^m)} \right|^2$$

for each $j \in \{0, \ldots, 2^m - 1\}$. Agian using the summation of geometric series formula to evaluate the above expression we get:

$$p_j = \left| \frac{1}{2^m} \frac{e^{2\pi\iota 2^m(\theta - j/2^m)} - 1}{e^{2\pi\iota(\theta - j/2^m)} - 1} \right|^2 = \frac{1}{2^{2m}} \left| \frac{e^{2\pi\iota(2^m\theta - j)} - 1}{e^{2\pi\iota(\theta - j/2^m)} - 1} \right|^2$$

Now we are required to show that the probabiltiy p_j is large for values of j that satisfy $j/2^m \approx \theta$ and small otherwise.

Recall that the all of this is with respect to the eigen vector state $|\psi\rangle$ and we have assumed that the corresponding eigen value associated with the eigen vector of the Unitary matrix U, $|\psi\rangle$ is $e^{2\pi\iota\theta}$. We are required to approximate $\theta \in [0, 1)$. We have showed that it works perfectly for $\theta = j/2^m$ for some integer $j \in \{0, \ldots, 2^m - 1\}$, where upon measurement the results in oucom j with probability 1. Now, we are required to show that the probability p_j is large for value of j that satisfy $j/2^m \approx \theta$ and small otherwise. Thus, when θ does not have the form $j/2^m$ for some integer j. We had determined that the probabiliyy associated with each possible outcome $j \in \{0, \ldots, 2^m - 1\}$ of the measurement is:

$$p_j = \frac{1}{2^{2m}} \left| \frac{e^{2\pi\iota(2^m\theta - j)} - 1}{e^{2\pi\iota(\theta - j/2^m)} - 1} \right|^2$$

Because, we already dealt with the case that $\theta = j/2^m$ for some integer j, We assume now that $\theta \neq j/2^m$. Now to prove that the probabiltiy p_j is highest when $\theta \approx j/2^m$, we assume that $\theta = j/2^m + \epsilon$. Thus,

$$e^{2\pi\iota\theta} = e^{2\pi\iota(j/2^m + \epsilon)}$$

for some real number ϵ with $|\epsilon| \leq \frac{1}{2^{m+1}}$, where equality means that the fractional parts of the two sides of the equations agree. This is equivalent to saying:

$$\theta = \frac{j}{2^m} + \epsilon \bmod 1$$

Now we prove a lowerbound on p_j as follows: Let

$$a = |e^{2\pi\iota(2^m\theta - j)} - 1| = |e^{2\pi\iota\epsilon 2^m} - 1|$$

$$b = |e^{2\pi\iota(\theta - j/2^m)} - 1| = |e^{2\pi\iota\epsilon} - 1|$$

Thus, we can write the probability p_j as (using the properties of the modulus of a complex numbers):

$$p_j = \frac{1}{2^{2m}} \left| \frac{a}{b} \right|^2 = \frac{1}{2^{2m}} \frac{a^2}{b^2}$$

Now, to get a lower bound on p_j we need to get a lower bound on a and an upper bound on b. To get a lower bound on a consider the following figure 6.8:

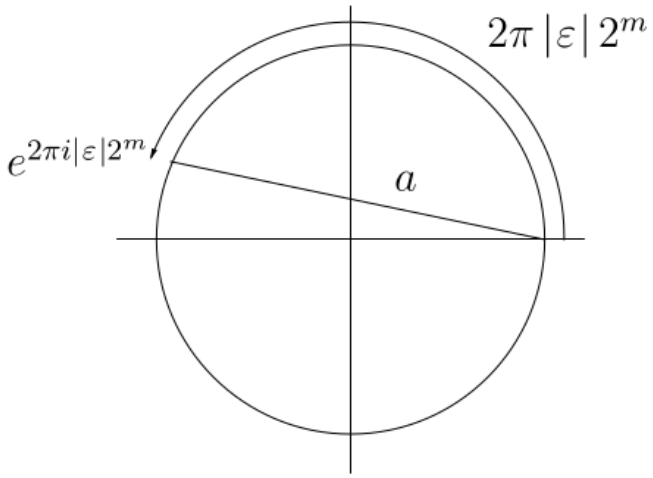


Figure 6.8: Lower bound of a

Now from the figure recall that the ratio of the minor arc length to the chord length can be written as

$$\frac{R\theta}{2R\sin\frac{\theta}{2}} = \frac{\theta}{2\sin\frac{\theta}{2}} \leq \frac{\pi}{2}$$

with equality at $\theta = \pi$. Thus, from the figure 6.8 we can say that: $\frac{2\pi|\epsilon|2^m}{a} \leq \frac{\pi}{2} \implies a \geq 4|\epsilon|2^m$. Along the similar lines, we may consider b along with the fact that the ratio of the arc length to chord length is at least 1:

$$\frac{\theta}{2\sin\frac{\theta}{2}} = \frac{1}{\frac{\sin\frac{\theta}{2}}{\frac{\theta}{2}}} \geq 1$$

Thus, as $\theta \to 0$ the ratio $\frac{\theta}{2\sin\frac{\theta}{2}}$ approaches 1. Thus, we can say that $b \geq 1$. Thus we can say that the ratio is always greater than 1. Thus,

$$\frac{2\pi|\epsilon|}{b} \geq 1 \implies b \leq 2\pi|\epsilon|$$

Thus, we can say that the probability p_j is lower bounded by:

$$p_j = \frac{1}{2^{2m}}\frac{a^2}{b^2} \geq \frac{1}{2^{2m}}\frac{(4|\epsilon|2^m)^2}{(2\pi|\epsilon|)^2} = \frac{1}{2^{2m}}\frac{16|\epsilon|^2 2^{2m}}{4\pi^2|\epsilon|^2} = \frac{4}{\pi^2} = 0.405284$$

Thus, the probability is atleast 40.5% of measuring the value of j when θ is close to $j/2^m$. This is the probability that every single one of the bits you measure is correct, so that your approximation to θ is good to m bits of precision.

Similarly we can use the above analysis to put upper bounds on the probability of obtaining inaccurate results. Suppose now that for a given value of j we have

$$e^{2\pi\iota\theta} = e^{2\pi\iota(j/2^m+\epsilon)}$$

for some real number ϵ with $\frac{\alpha}{2^m} \leq |\epsilon| < 1/2$. Here α is an arbitrary positive number that we can choose later. Thus, we have

$$p_j = \frac{1}{2^{2m}}\frac{a^2}{b^2}$$

for

$$a = |e^{2\pi\iota(2^m\theta - j)} - 1| = |e^{2\pi\iota\epsilon 2^m} - 1|$$
$$b = |e^{2\pi\iota(\theta - j/2^m)} - 1| = |e^{2\pi\iota\epsilon} - 1|$$

Now, we can put upper bounds on a and lower bounds on b. From the figure 6.8, we can say that $a \leq 2$ and $b \geq 4|\epsilon|$. Now we have,

$$p_j \leq \frac{4}{2^{2m}(4|\epsilon|^2)} = \frac{1}{4\alpha^2}$$

This implies that highly inaccurate results are very unlikely. For example, if we consider $\alpha =$, measning that our assumption is only that $|\epsilon| \geq 2^{-m}$, the probability of obtaining the corresponding value of j is atmost 1/4. For worse approximations, implying a larger bound on $|\epsilon|$, the probability of obatining the corresponding value of j quickly becomes very small.

Thus, to have a better result than a $4/\pi^2$ probability of obtaining an approximation of θ upto k bits of precision. Set $k = m + 2$ bits, run the phase estimation procedure several times, and to look for the most commonly appearing outcome. At least one outcomes, which is accurate to $k + 2$ bits of precision, occurs with probabiltiy at least $4/\pi^2$. Outcome with fewer than k bits of precision are much less likely as argued above. If you now take the most commonly occuring outcome and round it to k bits of precision, the probability of correctneess approaches 1 exponentially fast in the number of times the procedure is repeated. Notice also that you do not need multiple copies of the state $|\psi\rangle$ to perform this process, because the state $|\psi\rangle$ remains on the lower collection of qubits each time the procedure is performed and can be simply fed into the next iteration.

6.1.5 Quantum Fourier Transform

Quantum fourier transform is a quanutum implementation of Discrete Fourier Transform. Fourier transform has applications in signal processing, linear algebra, and many more.In short, **Fourier Analysis is a tool to describe the internal frequencies of a function.** Here, shown is the **Quantum Algorithm for computing the discrete fourier transform which is exponentially faster than the famous Fast Fourier Transform (FFT) algorithm fo classical computers.**

But, it has certain problems related to measurement. The QFT algorithm to transform the n qubit state vector $|\alpha\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \ldots + \alpha_{2^n-1} |2^n - 1\rangle$ to the state vector $|\beta\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle + \ldots + \beta_{2^n-1} |2^n - 1\rangle$, a measurment of $|\beta\rangle$ will only return one of its n components, and we are not able to recover all the information of the Fourier transform. For this reason, we describe the algorithm as **quantum fourier sampling**.

The Quantum Fourier transform is the generalization of the Hadmard transform. it is very similar, with the exception that QFT introduces phase./ The specific kinds of phases introduced are what we call primitive roots of unity, ω. To know more about the primitive roots of unity (refer Appendix A.3.8).

Quantum Fourier Transform with quantum gates

The strength of the FFT is that we are able to use the symmetry of the discrete Fourier transform to our advantage. The circuit application of QFt uses the same principle, but because of the power os superposition QFT is even faster.

The QFT is motivated by the FFT so we will follow the same steps, but because this is a quantum algorithm the implmentation fo the steps will be different. We first take the Fourier transform of the odd and even parts, then multiply the odd terms by the phase ω^j. Consider the figure 6.9.

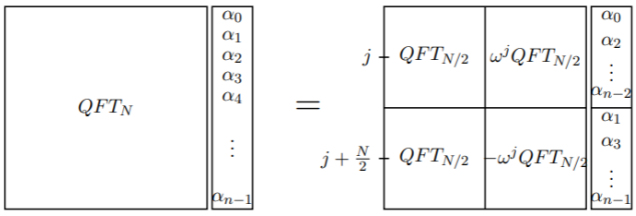


Figure 6.9: circuit

In quantum algorithm, the first step is that the odd and even terms are together

in superposition: the odd terms ae those whose least significant bit is 1, and even with 0. Therefore, we can apply $QFT_{N/2}$ to both the odd and even terms together. We do this by simply applying the $QFT_{N/2}$ to the m-1 most significant bits, and recombine the odd and even appropriately by applying the Hadmard to the least significant bit as shown in the figure **??**.

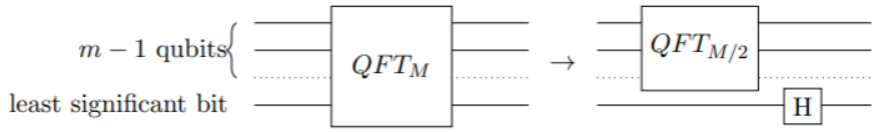


Figure 6.10: circuit

Now to carry out the phase multiplication, we need to multiply each odd term j by the phase ω_j. But remeber, an odd number in binary ends with a 1, while an even ends with 0. Thus, we can use the controlled phase shift, where the least significant bits is the control, to multiply only the odd terms by the phase without doing anything to the even terms. Recall that the controlled phase shift is similar to the CNOT gate in that it only applies a phase to the target if the control bit is one.

The phase associated with each controlled phase shift should be equal to ω^j where j is associated to the kth bit by $j = 2^k$. Thus, apply the controlled phase shift to each of the first $m-1$ qubits , with the least significant bit as the control. With the controlled phase shift and the Hadmard transform, QFT_N has been reduced to $QFT_{N/2}$ as shown in the figure 6.11.

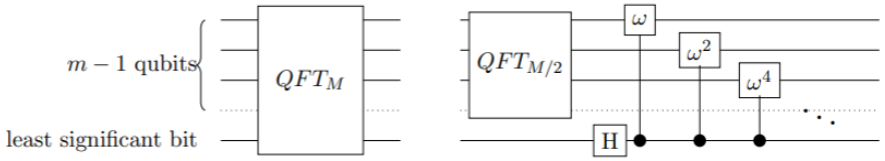


Figure 6.11: circuit

QFT_N is reduced to $QFT_{N/2}$ and N additional gates

Example 6.1.2. Let us contrsuct QFT_8. Following the algorithm, we will turn QFT_8 into QFT_4 and a few quantum gates. Then continuing on this way we turn QFT_4 into QFT_2 (which is just a Hadamard gate) and another few gates. Controlled phase gates will be represented by R_ϕ. **number of gates necessary to carry out**

QFT_N **is exactly** $\sum_{i=1}^{\log N} i = \log N(\log N + 1)/2 = \mathcal{O}(\log^2 N)$**.** Thus, the circuit is as shown in the figure 6.12.

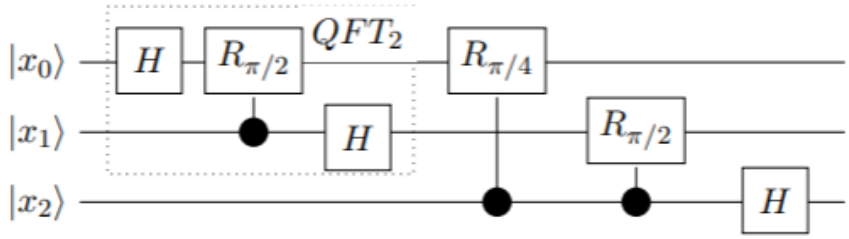


Figure 6.12: circuit

How can quantum Fourier transoform may be implemented by quantum circuits. Recall that

$$QFT_{2^m} |j\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi\iota jk/2^m} |k\rangle$$

Let the notations be ($N = 2^m$):

$$\omega_N = e^{2\pi\iota/N}$$

for any positive integer N. Let us also define a unitary mapping $Q\tilde{F}T_{2^m}$ to be the same as QFT_{2^m} except with the output bits in reverse order. Specifically, if an integer $k \in \{0, \ldots, 2^m - 1\}$ is written in binary notation as $k_{m-1}, k_{m-2}...k_0$ then we define

$$Q\tilde{F}T_{2^m} |j_{m-1}j_{m-2}...j_0\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} \omega_{2^m}^{jk} |k_0k_1...k_{m-1}\rangle$$

If we can come up with an efficient implementation of $Q\tilde{F}T_{2^m}$ follows - just reverse the order of the output qubits after performing $Q\tilde{F}T_{2^m}$. The reason why we consider $Q\tilde{F}T_{2^m}$ rather than QFT_{2^m} is simply for convinenece.

Our description of quantum circuits for performing $Q\tilde{F}T_{2^m}$ for any given value of m is essentially recursive. Starting with base case m=1. The Transnformation is just $Q\tilde{F}T_2$ is just a fany name for a Hadamard transformation:

$$Q\tilde{F}T_2 |j\rangle = \frac{1}{\sqrt{2}} \sum_{k=0}^{1} \omega_2^{jk} |k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \omega_2^j |1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^j |1\rangle)$$

For general case $m \geq 2$, the following circuit computers $Q\tilde{F}T_{2^m}$:

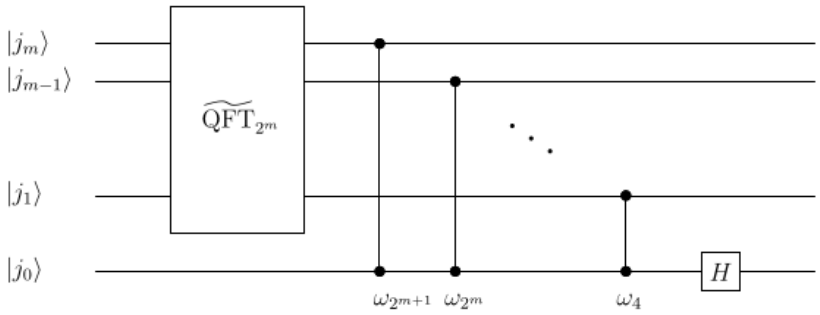


Figure 6.13: QFT circuit

This diagram assume you know how to inmplement the transformation $Q\tilde{F}T_{2^m}$, but usuing the fact that $Q\tilde{F}T_2$ is the same as Hadamrd transform we can easily unwind the recursion if we want an explicit description of a circuit.

To show that the circuit works correctly, it suffices as usual to show that it works correctly on classical states. We wish to show that

$$Q\tilde{F}T_{2^m}\,|j_m j_{m-1} j_{m-2} ... j_0\rangle = \frac{1}{\sqrt{2^{m+1}}} \sum_{k=0}^{2^{m+1}-1} \omega_{2^m}^{jk}\,|k_0 k_1 ... k_{m-1} k_m\rangle$$

for each $j \in \{0, \ldots, 2^{m+1} - 1\}$. Let us write

$$j' = j_m j_{m-1} ... j_1 = \lfloor j/2 \rfloor$$

$$k' = k_{m-1} k_{m-2} ... k_0 = k - k_m 2^m$$

Chapter 7

The HHL Algorithm

7.1 Introduction

The Harrow-Hassidim-Lloyd (HHL) [2] algorithm for solving large systems of linear equations. Linear system of equations arise in almost any real-life applications from Physics, Maths and Engineering to Finanical and Biological Sciences. Typical examples of real-life problems involving solving system of linear equations computationally include, solutions of Partial Differential Equations, the calibration of financial models, fluid simulation of numerical field calculation, optimization, machine learning etc. Most real-life problems generally boils down to a set of Linear Equations. We even approximate non-linear terms in non-linear equations, to linear terms in order to approximate the non-linear equations to linear equations in few cases.

7.2 The Linear-System Problem (LSP)

The classical linear-system problem is defined as follows:

Definition 7.2.1. Given a matrix $A \in \mathbb{C}^{N \times N}$ and a vector $\vec{b} \in \mathbb{C}^N$, find $\vec{x} \in \mathbb{C}^N$ satisfying $A\vec{x} = \vec{b}$.

In most of the applications it sufficies to find $\vec{x}$ such that $\|\vec{x} - \vec{x}_0\| < \epsilon$ for some $\epsilon > 0$, where $\vec{x}_0$ is the exact solution. Assuming that A is invertible (or $rank(A) = N$ i.e. full rank) for simplicity, the solution to the above problem is given by $\vec{x} = A^{-1}\vec{b}$. In case, A is not invertible, we can use the Moore-Penrose Pseudo-Inverse of A to get the solution which is $\vec{x} = (A^\dagger A)^{-1}A^\dagger\vec{b}$, where $(A^\dagger A)^{-1}A^\dagger$ is called the Moore-Penrose Psuedo-Inverse of A.

A system of linear equations is called **s-sparse if A has at most s non-zero entries per row or column**. Solving an s-sparse system of size N with a classical computer requires $O(Ns\kappa \log(1/\epsilon))$ time complexity using the conjugate gradient method [6]. Here, κ is the condition number of the system and ϵ is the accuracy of the approximation. Overall, the solution on a classical computer is simply $x = A^{-1}b$. Note that here the solution x is not necessarily a normalized vector, and hence cannot be stored as a quantum state. Thus, for solving the problem on a Quantum Computer we define the corresponding QLSP which is the quantum equivalent of the classical LSP.

The Quantum Linear-System Problem (QLSP) is defined as follows:

Definition 7.2.2. Given a matrix $A \in \mathbb{C}^{N\times N}$ and $|b\rangle \in \mathbb{C}^N$, find $|\tilde{x}\rangle \in \mathbb{C}^N$ such that $\| |x\rangle - |\tilde{x}\rangle \| \leq \epsilon$ for some $\epsilon > 0$ where $|x\rangle$ is the exact solution satisfying $A|x\rangle = |b\rangle$.

For simplicity, we assume $|b\rangle$ is normalized and can be efficiently implemented using a transformation U_b (has a efficient quantum circuit) as a n_b qubit quantum state. Here, the goal is to produce an n-qubit state whose amplitude-vector (up to normalization and up to ϵ error) is a solution to the linear system. Thus,

$$\| |x\rangle - |\tilde{x}\rangle\| \leq \epsilon, \quad |x\rangle = \frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|}$$

We are required to find the normalization constant $\|A^{-1}|b\rangle\|$ separately. Note that here $|\tilde{x}\rangle$ will be in a superposition state, thus in order to estimate the amplitude/-coefficients it is required to do a lot of experiments.

The HHL algorithm requires A to be Hermitian. We can create any given general matrix A to a Hermitian Matrix using the following method called **Dilation method**.

Without the loss of generality we can assume that A (through this we can convert the general matrix A given in classical LSP to a Hermitian matrix for QLSP) is Hermitian since any matrix A can be made Hermitian using the following method. We define A' as:

$$A' = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix} = |1\rangle\langle 0| \otimes A + |0\rangle\langle 1| \otimes A^\dagger$$

and thus, the problem now can be defined as:

$$\begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix} \begin{bmatrix} |0\rangle \\ |x\rangle \end{bmatrix} = \begin{bmatrix} |b\rangle \\ |0\rangle \end{bmatrix}$$

thus preserving the original problem $A|x\rangle = |b\rangle$. Thus, $A'|x\rangle' = |b\rangle'$ where $|x\rangle' = \begin{bmatrix}|0\rangle \\ |x\rangle\end{bmatrix}, |b\rangle' = \begin{bmatrix}|b\rangle \\ |0\rangle\end{bmatrix}$ and A' is Hermitian ($A'^{\dagger} = A'$). Thus for a given problem in classical LSP as $A|x\rangle = |b\rangle$, we can convert it to a QLSP problem as $A'|x\rangle' = |b\rangle'$ where A' is Hermitian using the above method. Thus, without the loss of generality for the application of HHL algorithm we can assume that the matrix A is Hermitian.

The HHL algorithm estimates a function of the solution with time complexity of $O(log(N)s^2\kappa^2/\epsilon)$. The **assumptions/restrictions** on the Linear system in order to make the HHL algorithm work are:

1. **Assume that b is normalized: (hence can be stored in a quantum state)** We have a unitary matrix U_b which can prepare $|b\rangle$ as an n-qubit quantum state $|b\rangle = \frac{1}{\|b\|}\sum_i b_i|i\rangle$ using a circuit of B 2-qubit gates. We also assume for simplicity that $\|b\| = 1$. Thus, it is possible that using some work registers $|0\rangle_{n_b}$ to transform it to $|b\rangle$ using the transformation as shown:

$$|0^{n_b}\rangle \xrightarrow{U_b} |b\rangle_{n_b}$$

2. The matrix A is s-sparse and we have sparse access to it. Such sparsity is not essential to the algorithm, and could be replaced by other properties that enable an efficient block-encoding of A.

3. The matrix A is well-conditioned: (A is invertible, thus κ is some finite value) The ratio between its largest and smallest singular value is at most some κ. For simplicity, assume the smallest singular value is $\geq 1/\kappa$ while the largest is ≤ 1. In other words, all eigenvalues of A lie in the interval $[-1, -1/\kappa] \cup [1/\kappa, 1]$. The smaller the condition number κ is, the better it will be for the algorithm. Let's assume our algorithm knows κ, or at least knows a reasonable upper bound on κ.

7.3 The HHL Algorithm

7.3.1 Some Mathematical Preliminaries

To encode the problem on a quantum computer we need to rescale the system, by assuming b and x to be normalized and map them to the respective quantum states

$|b\rangle$ and $|x\rangle$. The ith component of $|b\rangle$ corresponds to the amplitude of the ith basis state of the quantum state $|b\rangle$. Thus, we now focus on the rescaled problem

$$A|x\rangle = |b\rangle$$

Since A is Hermitian ($A^\dagger = A$) thus, a Normal Matrix ($AA^\dagger = A^\dagger A$), hence Unitarily diagonalizable as:

$$A = U\Lambda U^\dagger$$

where U is a unitary matrix made up of the eigenvectors of A and Λ is a diagonal matrix with the eigenvalues of A on the diagonal. It's spectral decomposition can be written in outer product form as:

$$A = \sum_{j=0}^{N-1} \lambda_j |\lambda_j\rangle \langle\lambda_j|, \quad \lambda_j \in \mathbb{R}$$

where $|\lambda_j\rangle$ are the eigenvectors of A and λ_j are the corresponding eigenvalues. Then,

$$\begin{aligned} A^{-1} &= (U\Lambda U^\dagger)^{-1} \\ &= U\Lambda^{-1}U^\dagger \end{aligned}$$

Thus spectral decomposition of A^{-1} is:

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |\lambda_j\rangle \langle\lambda_j|$$

Now, since A is unitarily diagonalizable thus, it forms a complete basis set of orthonormal eigenvectors $\{|\lambda_j\rangle\}$ which span the $\mathbb{C}^N$ space. Thus, any vector $|v\rangle$ can be written as a linear combination of the eigenvectors of A. Thus, we write the right hand side of the system $|b\rangle$ as:

$$|b\rangle = \sum_{j=0}^{N-1} b_j |\lambda_j\rangle, \quad b_j \in \mathbb{C}$$

The goal of the HHL algorithm is to exit with the readout register in the state

$$\begin{aligned} |x\rangle = A^{-1}|b\rangle &= \left(\sum_{j=0}^{N-1} \lambda_j^{-1} |\lambda_j\rangle \langle\lambda_j|\right) \left(\sum_{k=0}^{N-1} b_k |\lambda_k\rangle\right) \\ &= \left(\sum_{j=0}^{N-1} \sum_{k=0}^{N-1} b_k \lambda_j^{-1} |\lambda_j\rangle \langle\lambda_j|\lambda_k\rangle\right) \end{aligned}$$

Now $\langle\lambda_j|\lambda_k\rangle = \delta_{jk}$ (because A is a Hermitian matrix, thus Unitarily diagonalizable and hence has orthonormal eigenvectors), thus the above expression simplifies to:

$$|x\rangle = \sum_{j=0}^{N-1} b_j \lambda_j^{-1} |\lambda_j\rangle$$

which is the required solution to the linear system $A|x\rangle = |b\rangle$.

7.3.2 Algorithm

Consider the following circuit as shown in the figure 7.1. The first qubit show is initialised to $|0\rangle$ which is also called the signal qubit. The algorithm uses three quantum register, all set to $|0\rangle$ at the beginning of the algorithm. One register, which we will denote with the subindex n_l, is used to store a binary representation of the eigen values of A. A second register, denoted by n_b, contains the vector solution. Note that $N = 2^{n_b}$. There is an extra register, which we will denote with n_a for auxillary qubits, used for intermediate steps in the computation. We can ignore any auxillary in the following description as they are $|0\rangle$ at the beginning of each computation, and are restored back to $|0\rangle$ at the end of each individual operation.

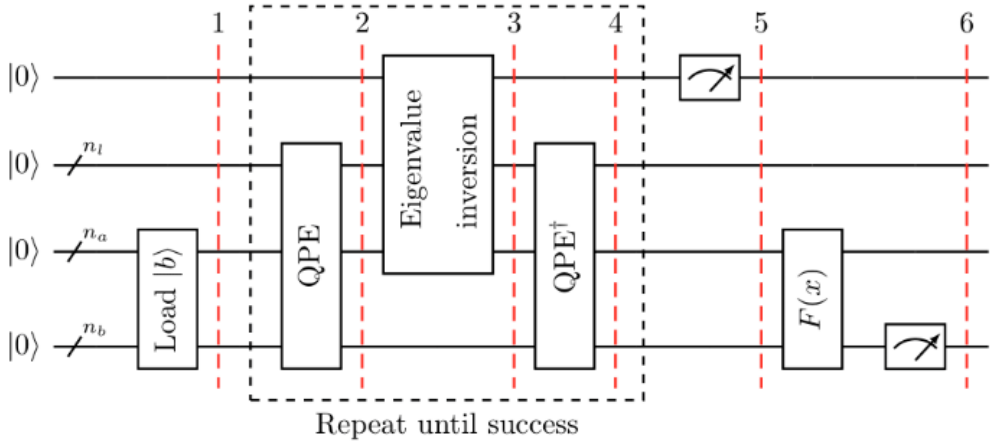


Figure 7.1: HHL Circuit

Analysis:
Let us now analyze the working of the above circuit (note that we ignore the ancilla registers throughout the analysis):

1. Load the data $|b\rangle \in \mathbb{C}$. That is, perform the transformation

$$|0\rangle |0\rangle_{n_l} |0\rangle_{n_b} \xrightarrow{I\otimes I\otimes U_b} |0\rangle |0\rangle_{n_l} |b\rangle_{n_b}$$

2. Now we are required to apply QPE. Ignoring the Ancilla bits, recall that in QPE given a Unitary matrix U and one of its eigen vector $|\psi\rangle$, we can find the eigenvalue $e^{\iota 2\pi\theta}$ corresponding to $|\psi\rangle$ (or specifically approximate θ i.e. $\tilde{\theta}$ upto d bits of accuracy) by applying the QPE algorithm, where $\theta \approx \tilde{\theta} \in [0, 1)$. Thus, we have the following transformation:

$$|0^d\rangle |\psi\rangle \xrightarrow{U_{QPE}} |\tilde{\theta}\rangle |\psi\rangle$$

where $|\tilde{\theta}\rangle$ is a d-bit (in binary) basis state in the computational basis corresponding to the closest binary representation of the actual phase $\theta \in [0, 1)$ that corresponds to the eigen vector $|\psi\rangle$. Thus, first we require a Unitary matrix, here we are required to form a Unitary matrix using A which is a Hermitian matrix. Let us define $U = e^{\iota 2\pi At}$ where t is some scalar prescaling factor and U is a Unitary matrix (exponential of a Hermitian matrix results in a unitary matrix). Now,

$$e^{\iota 2\pi At} = I + \iota 2\pi At - \frac{1}{2}(2\pi)^2 A^2 t^2 - \frac{\iota}{6}(2\pi)^3 A^3 t^3 + \dots$$

We know that $A = Q\Lambda Q^\dagger$, since A is unitarily diagonalizable for some orthonormal matrix Q, upon substituting and simplifying, we get, $e^{\iota 2\pi At} = Qe^{\iota 2\pi\Lambda t}Q^\dagger$. Upon writing this in outer product/spectral decomposition form we get,

$$U = e^{\iota 2\pi At} = \sum_{j=0}^{N-1} e^{\iota 2\pi\lambda_j t} |\lambda_j\rangle \langle\lambda_j|$$

Thus, U and A share the same eigen vectors but the eigen values of At are $2\pi\lambda_j t$ and the eigen values of U are $e^{\iota 2\pi\lambda_j t}$, thus we can apply QPE to find the eigen values of A. Also recall that since A is Hermitian matrix, thus, the eigen values of A i.e. $\lambda_j \in \mathbb{R} \forall j$. Using QPE, we can find approximate values of the phases and thus, we can write, $e^{\iota 2\pi\tilde{\theta_j}} \approx e^{\iota 2\pi\lambda_j t} \implies \tilde{\theta_j} \approx \lambda_j t$ (recall that $\tilde{\theta_j} \in [0, 1)$ which is an approximation to the exact phase θ_j), Upon rearranging the terms we get,

$$\lambda_j \approx \frac{\tilde{\theta_j}}{t}$$

for each $j \in \{0, \ldots, N-1\}$. Recall, that here t is not known to us, thus we need to estimate it. We use this prescaling factor in order to map the eigen values of A to the range $[0, 1)$. There are various methods to estimate t so that the Eigenvalues of A are in the range $[0, 1)$. Without getting involved on how to find t, we assume that we know the value of t so that the eigen values of A are in $[0, 1)$. Let this new eigen values be $\tilde{\theta_j} \approx \lambda'_j = \lambda_j t$ such that $0 \leq lambda'_0 \leq \lambda'_1 \leq \lambda'_2 \leq \ldots \leq \lambda'_{N-1} < 1$. Thus,

$$\tilde{\theta_j} \approx \lambda'_j \quad \forall \quad j \in \{0, \ldots, N-1\}$$

Thus, the phases are approximations to the shifted eigenvalues to be in range $[0, 1)$. Here, the QPE problem is as shown in the figure 7.2.

Figure 7.2: QPE

Note that the input to the QPE is not an eigen state but as shown in the figure 7.2, it is a superposition of the eigen states of A, i.e. $|b\rangle = \sum_{j=0}^{N-1} b_j |\lambda_j\rangle$. Thus, the resulting output will be a superposition of $\tilde{\theta_j}$ i.e. $\sum_{j=0}^{n-1} b_j |\tilde{\theta_j}\rangle |\lambda_j\rangle$. Here, $|\tilde{\theta_j}\rangle$ is one of the basis states in the computational basis corresponding to the closest binary representation of the actual phase $\theta_j = \lambda'_j = \lambda_j t$ that corresponds to the eigen vector $|\lambda_j\rangle$. In other words, say the actual phase corresponding to a eigen state $|\lambda_j\rangle$ is $\theta_j \in [0, 1)$ Thus, $\theta_j \approx 0.\theta_{j_{n_l-1}}\theta_{j_{n_l-2}} \ldots \theta_{j_0} = \tilde{\theta_j}$ is the n_l bit binary approximation of θ_j. Thus, the output of the QPE will be a superposition of the n_l bit binary approximations of the actual phases θ_j. At the end of this transformation we get,

$$|0\rangle |0\rangle_{n_l} |0\rangle_{n_b} \xrightarrow{I \otimes QPE} |0\rangle \sum_{j=0}^{N-1} b_j |\tilde{\theta_j}\rangle |\lambda_j\rangle$$

where $\tilde{\theta_j}$ are binary approximations of the eigen values of A in the range $[0, 1)$ corresponding to the eigen vector $|\lambda_j\rangle$.

3. **Implementation of the controlled rotations:** Now we are suppose to inverse the eigen values in order to achieve the aim of getting

$$\sum_{j=0}^{N-1} \lambda_j^{-1} b_j \left|\tilde{\theta_j}\right\rangle \left|\lambda_j\right\rangle$$

from

$$\left|0\right\rangle \sum_{j=0}^{N-1} b_j \left|\tilde{\theta_j}\right\rangle \left|\lambda_j\right\rangle$$

In order to do this we are required to implement controlled rotations. Let us take a slight diversion from the algorithm to learn about the implementation of controlled rotation in order to implement it efficiently for the algorithm. Consider the following proposition.

Proposition 7.3.1. *(Controlled rotation given rotation angles). Let* $0 \leq \theta < 1$ *has exact d-bit fixed point representation* $\theta = 0.\theta_{d-1} \dots \theta_0$ *be its d-bit fixed point representation. Then there is a (d+1)qubit unitary* U_θ *such that*

$$U_\theta : \left|0\right\rangle \left|\theta\right\rangle \rightarrow \left(\cos(\pi\theta)\left|0\right\rangle + \sin(\pi\theta)\left|1\right\rangle\right)\left|0\right\rangle$$

Proof. Recall that R_Y gate is as follows:

$$R_\theta^Y = R_Y(\theta) = e^{-\iota\frac{\theta}{2}\sigma_Y} = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

Now putting $\theta = 2\tau$ we get

$$R_Y(2\tau) = e^{\iota\tau\sigma_Y} = \begin{bmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\tau) & \cos(\tau) \end{bmatrix}$$

Here $R_Y(.)$ performs a single-qubit rotation around the y-axis. For any $j \in \{0, \dots, 2^d - 1\}$ with its binary representation $j = j_{d-1} \dots j_0$, we have

$$\frac{j}{2^d} = (0.j_{d-1} \dots j_0)$$

So we choose $\tau = \pi(0.j_{d-1} \dots j_0)$, which can be further simplified putting $j_k = \theta_k \forall k \in \{d-1, \dots, 0\}$. Thus, we define

$$\begin{aligned} U_\theta &= \sum_{j=0}^{N-1} e^{-\iota\pi(0.j_{d-1}\dots j_0)\sigma_Y} \otimes \left|j\right\rangle\left\langle j\right| \\ &= \sum_{\theta_{d-1},\dots,\theta_0} e^{-\iota\pi(0.\theta_{d-1}\dots\theta_0)\sigma_Y} \otimes \left|\theta_{d-1}\dots\theta_0\right\rangle\left\langle\theta_{d-1}\dots\theta_0\right| \end{aligned}$$

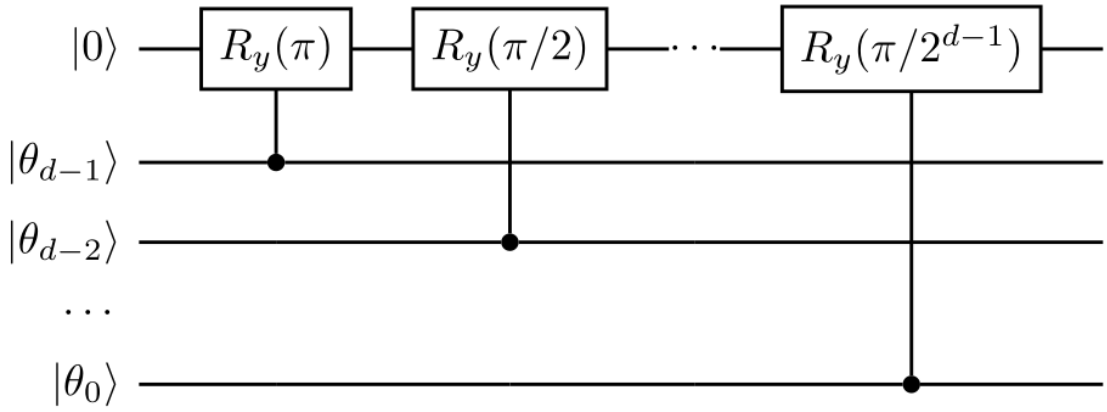


Figure 7.3: Controlled Rotation

$$
\begin{aligned}
U_\theta &= \sum_{\theta_{d-1},\ldots,\theta_0} e^{-\iota\pi\frac{\theta_{d-1}}{2}\sigma_Y} e^{-\iota\pi\frac{\theta_{d-2}}{4}\sigma_Y} \ldots e^{-\iota\pi\frac{\theta_0}{2^d}\sigma_Y} \otimes |\theta_{d-1}\ldots\theta_0\rangle \langle\theta_{d-1}\ldots\theta_0| \\
&= \sum_{\theta_{d-1},\ldots,\theta_0} R_Y(\pi\theta_{d-1}) R_Y(\pi\frac{\theta_{d-2}}{2}) \ldots R_Y(\pi\frac{\theta_0}{2^{d-1}}) \otimes |\theta_{d-1}\ldots\theta_0\rangle \langle\theta_{d-1}\ldots\theta_0| \\
&= \sum_{\theta_{d-1},\ldots,\theta_0} R_Y(\pi\theta_{d-1}) R_Y(\frac{\pi}{2}\theta_{d-2}) \ldots R_Y(\frac{\pi}{2^{d-1}}\theta_0) \otimes |\theta_{d-1}\ldots\theta_0\rangle \langle\theta_{d-1}\ldots\theta_0|
\end{aligned}
$$

Thus, applying U_θ to $|0\rangle|\theta\rangle$ gives the desired result since corresponding to a particular value of θ only one of the terms from the summation with which its inner product results in 1 will survive (since all the basis states are orthonormal). The resulting quantum circuit for the controlled rotation is as shown in the figure 7.3. □

Thus, as shown in the circuit in figure 7.3 based upon the input binary state of $\theta_{d-1}\ldots\theta_0$ the corresponding R_Y gates will be acted on the signal bit (which was initialised to $|0\rangle$) where each θ_k will act as the control bit for the corresponding $R_Y(\pi/2^{d-k-1})$.

In other words, the controlled rotation implementation will perform a transformation as:

$$U_\phi |0\rangle |\phi\rangle \rightarrow (\cos(\pi\phi)|0\rangle + \sin(\pi\phi)|1\rangle)|\phi\rangle$$

where $\phi \in [0, 1)$. This is a sequence of single-qubit rotations on the signal qubit, each controlled by a single qubit.

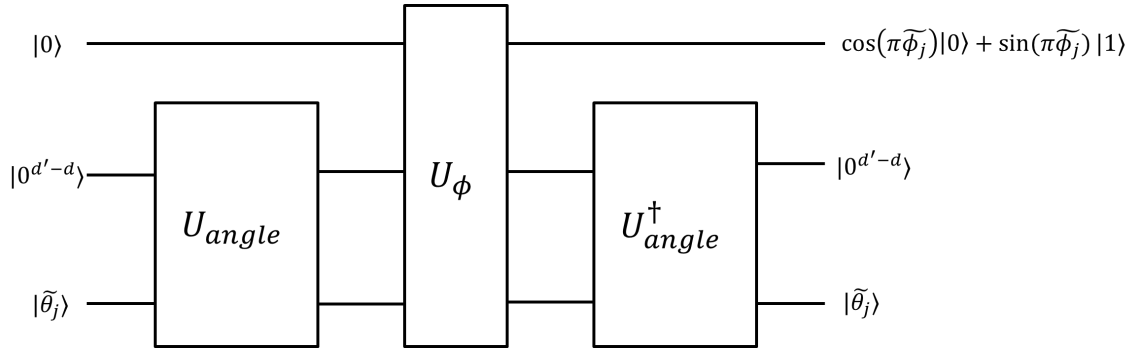


Figure 7.4: Entire Controlled Rotation

Back to the problem, we are required to inverse the eigen values which can be done using implementation of controlled rotation operation. Say we define

$$\phi_j = \frac{1}{\pi}\sin^{-1}(C/\tilde{\theta}_j)$$

where $C > 0$ be a lower bound to λ'_0, so that $0 < C/\tilde{\theta}_j < 1$ for all j. Recall that $\tilde{\theta}_j \approx \lambda'_j$ and $\tilde{\theta}_j \in [0, 1)$. Writing all of this in d' bit binary representation we get,

$$\tilde{\phi_j} = \frac{1}{\pi}\sin^{-1}(C/\tilde{\theta'_j}) \approx \phi_j = \frac{1}{\pi}\sin^{-1}(C/\tilde{\theta}_j)$$

For simplicity, we assume that d' is large enough so that the error of the fixed point representation is negligible in this step. The mapping

$$U_{angle}\,|0^{d'-d}\rangle\,|\tilde{\theta}_j\rangle = |\tilde{\phi_j}\rangle$$

can be implemented using *classical arithmetics circuits*, which may require poly(d') gates and an additional working register of poly(d') qubits, which is not displayed here. Therefore the entire controlled rotation operation needed for the HHL algorithm is as given by the circuit 7.4. Therefore through the uncomputation $U^{\dagger}_{angle}$, the d'-d ancilla qubits also become a working register. Discard the working register we obtain a unitary U_{CR} satisfying

$$U_{CR}\,|0\rangle\,|\tilde{\theta}_j\rangle = (cos(\pi\tilde{\phi_j})\,|0\rangle + \sin(\pi\tilde{\phi_j})\,|1\rangle)\,|\tilde{\theta}_j\rangle = \left(\sqrt{1 - \frac{C^2}{\tilde{\theta'^2_j}}}\,|0\rangle + \frac{C}{\tilde{\theta'_j}}\,|1\rangle\right)|\tilde{\theta}_j\rangle$$

This is the unitary circuit used in the HHL algorithm. Now recall the state of

the system after applying the QPE was

$$|0\rangle \sum_{j=0}^{N-1} b_j \, |\tilde{\theta_j}\rangle \, |\lambda_j\rangle$$

Thus, now we apply U_{CR} to this state we get the following transformation:

$$|0\rangle \sum_{j=0}^{N-1} b_j \, |\tilde{\theta_j}\rangle \, |\lambda_j\rangle \xrightarrow{U_{CR}\otimes I} \sum_{j=0}^{N-1} b_j \left(\sqrt{1 - \frac{C^2}{\tilde{\theta_j'^2}}} \, |0\rangle + \frac{C}{\tilde{\theta_j'}} \, |1\rangle \right) |\tilde{\theta_j}\rangle \, |\lambda_j\rangle$$

where each $\theta_j' \approx \lambda_j'$.

4. Finally, we perform uncomputation by applying $U_{QPE}^{\dagger}$, we convert the information from the ancilla register $|\tilde{\theta_j}\rangle$ back to $|0\rangle_{n_l}$. Thus, through the uncomputation, the n_l ancilla qubits for storing the eigenvalues also becomes a working register. Discarding all the working registers, the resulting unitary denoted by U_{HHL} satisfies

$$U_{HHL} \, |0\rangle \, |b\rangle = \sum_{j=0}^{N-1} \left(\sqrt{1 - \frac{C^2}{\tilde{\theta_j'^2}}} \, |0\rangle + \frac{C}{\tilde{\theta_j'}} \, |1\rangle \right) b_j \, |\lambda_j\rangle$$

Finally, measuring the signal qubit, if the outcome is 1, we obtain the (unnormalized) vector

$$\tilde{x} = \sum_j \frac{C b_j}{\tilde{\theta_j'}} \, |\lambda_j\rangle$$

stored as a normalized state in the system register is

$$|\tilde{x}\rangle = \frac{\tilde{x}}{\|\tilde{x}\|} \approx |x\rangle$$

which is the desired approximate solution to QLSP. In particular, the constant C does not appear in the solution.

Recovering the norm of the solution: The HHL algorithm returns the solution to QLSP in the form of a normalized state $|x\rangle$ stored in the quantum computer. In order to recover the magnitude of the unnormalized solution $\|\tilde{x}\|$, we note that the success probability of measuring the signal qubit in 1 is

$$p(1) = \| \sum_j \frac{C b_j}{\tilde{\theta_j'}} \|^2 = \|\tilde{x}\|^2$$

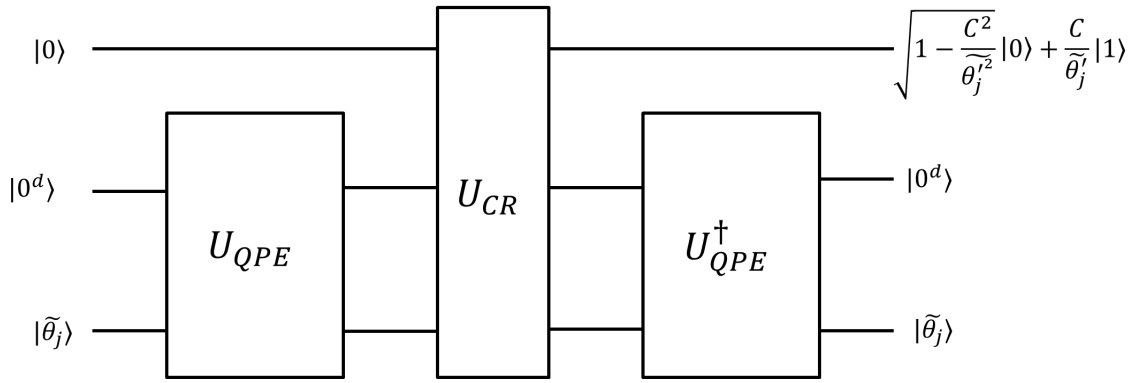


Figure 7.5: HHL Circuit

Therefore sufficient repetitions of running the circuit and estimate $p(1)$, we can obtain an estimate of $\|\tilde{x}\|$. The general HHL circuit is as shown in the figure 7.5.

7.3.3 Example

Example 7.3.1. (4-qubit HHL) Consider an example to illustrate the algorithm. Consider

$$A = \begin{bmatrix} 1 & -1/3 \\ -1/3 & 1 \end{bmatrix} \quad and \quad |b\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Note that b is normalized and $b \in \mathbb{C}^2$. Thus, we need $n_b = 1$ qubit to represent b. Also, $A \in \mathbb{C}^{2\times 2}$ is Hermitian, and hence will have two real eigen values ($\lambda_1, \lambda_2 \in \mathbb{R}$), thus we use $n_l = 2$ qubits to store the binary representation of the eigen values and finally 1 auxiliary qubit for the conditioned rotation. Here, $N = 2$.

1. We first prepare the state $|b\rangle$. Note that here in this example $|b\rangle = |0\rangle$, trivial.

$$|0\rangle_{n_b} \xrightarrow{U_b} |b\rangle_{n_b}$$

2. Now we apply QPE, with inputs as $|b\rangle_{n_b}$ and $|0\rangle_{n_l}$.

   (Note that the eigen values of A are 2/3 and 4/3. and suppose we choose $t = 3/8$, for simplicity. Thus, the eigen values of scaled A are: $(2/3)t = (2/3)(3/8) = 2/8 = 1/4 = 0.25 = (0.01), (4/3)t = (4/3)(3/8) = 4/8 = 1/2 = 0/5 = (0.10)$, thus the eigen values of scaled A in binary are 0.01, 0.10. The

corresponding eigen vectors of A are

$$|\lambda_1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} \quad and \quad |\lambda_2\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$$

Thus, we can write $|b\rangle$ in the eigen basis of A as

$$|b\rangle_{n_b} = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}\right) = \frac{1}{\sqrt{2}}|\lambda_1\rangle + \frac{1}{\sqrt{2}}|\lambda_2\rangle$$

$b_1 = \frac{1}{\sqrt{2}}, b_2 = \frac{1}{\sqrt{2}}$. **Note that all of this done here is not required since the algorithm will itself output the eigen values. This is done just for the sake of understanding and simplicity.**)

Thus, QPE will output,

$$|0\rangle|0\rangle_{n_l}|b\rangle_{n_b} \xrightarrow{I\otimes QPE} |0\rangle\sum_{j=0}^{1} b_j|\tilde{\theta_j}\rangle|\lambda_j\rangle = \frac{1}{\sqrt{2}}|0\rangle|01\rangle|\lambda_!\rangle + \frac{1}{\sqrt{2}}|0\rangle|10\rangle|\lambda_2\rangle$$

3. Now we do controlled rotations. (Suppose we use $C = 1/8$ that is less than the smallest (rescaled) eigen values of A i.e.1/4, recall, C should be chosen such that it is less than the smallest (rescaled) eigenvalue of 1/4 but as large as possible so that when the auxiliary qubit is measured, the probability of being in the state $|1\rangle$ is large.) Thus, as output of the controlled rotations we get,

$$|0\rangle\sum_{j=0}^{N-1} b_j|\tilde{\theta_j}\rangle|\lambda_j\rangle \xrightarrow{U_{CR}\otimes I} \sum_{j=0}^{N-1} b_j\left(\sqrt{1-\frac{C^2}{\tilde{\theta_j'^2}}}|0\rangle + \frac{C}{\tilde{\theta_j'}}|1\rangle\right)|\tilde{\theta_j}\rangle|\lambda_j\rangle$$

$$= \frac{1}{\sqrt{2}}\left(\sqrt{1-\frac{(1/8)^2}{(1/4)^2}}|0\rangle + \frac{1/8}{1/4}|1\rangle\right)|01\rangle|\lambda_1\rangle + \frac{1}{\sqrt{2}}\left(\sqrt{1-\frac{(1/8)^2}{(1/2)^2}}|0\rangle + \frac{1/8}{1/2}|1\rangle\right)|10\rangle|\lambda_2\rangle$$

$$= \frac{1}{\sqrt{2}}\left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle\right)|01\rangle|\lambda_1\rangle + \frac{1}{\sqrt{2}}\left(\frac{\sqrt{15}}{4}|0\rangle + \frac{1}{4}|1\rangle\right)|10\rangle|\lambda_2\rangle$$

4. Now we apply the uncomputation step $QPE^\dagger$, it will convert $\tilde{\theta_j}$ to $|0\rangle_{n_l}$,(thus discarding the working registers as they will not contribute anything now to the algorithm) we will get the final state of the system to be in,

$$U_{HHL}|0\rangle|b\rangle = \sum_{j=0}^{N-1}\left(\sqrt{1-\frac{C^2}{\tilde{\theta_j'^2}}}|0\rangle + \frac{C}{\tilde{\theta_j'}}|1\rangle\right)b_j|\lambda_j\rangle$$

Thus, the state of the system after the uncomputation will be:

$$\frac{1}{\sqrt{2}}\left(\frac{\sqrt{3}}{2}|0\rangle+\frac{1}{2}|1\rangle\right)|\lambda_1\rangle+\frac{1}{\sqrt{2}}\left(\frac{\sqrt{15}}{4}|0\rangle+\frac{1}{4}|1\rangle\right)|\lambda_2\rangle$$

Upon substituting, $|\lambda_1\rangle=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$

$|\lambda_2\rangle=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}=\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$, we get,

$$=\frac{1}{\sqrt{2}}\left(\frac{\sqrt{3}}{2}|0\rangle+\frac{1}{2}|1\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)+\frac{1}{\sqrt{2}}\left(\frac{\sqrt{15}}{4}|0\rangle+\frac{1}{4}|1\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle$$

$$=\frac{\sqrt{3}}{4}|00\rangle-\frac{\sqrt{3}}{4}|01\rangle+\frac{1}{4}|10\rangle-\frac{1}{4}|11\rangle+\frac{\sqrt{15}}{8}|00\rangle+\frac{\sqrt{15}}{8}|01\rangle+\frac{1}{8}|10\rangle+\frac{1}{8}|11\rangle$$

Thus, upon partial measurement of the auxiliary qubit (i.e. the first qubit here) to outcome 1, the state of the system will be:

$$\frac{\frac{1}{4}|10\rangle-\frac{1}{4}|11\rangle+\frac{1}{8}|10\rangle+\frac{1}{8}|11\rangle}{\sqrt{\frac{5}{32}}}=\frac{\frac{3}{8}|10\rangle-\frac{1}{8}|11\rangle}{\sqrt{\frac{5}{32}}}=|1\rangle\otimes\frac{\frac{3}{8}|0\rangle-\frac{1}{8}|1\rangle}{\sqrt{\frac{5}{32}}}$$

which is the value of normalized $|x\rangle$ in the second register. Thus,

$$\frac{|x\rangle}{\||x\rangle\|}=\frac{\frac{3}{8}|0\rangle-\frac{1}{8}|1\rangle}{\sqrt{\frac{5}{32}}}$$

5. In order to calculate the norm of $|x\rangle$, we can perform several runs of the circuit and get the approximate $p(1)$. Here, the probability $p(1)$ we get,

$$P(|1\rangle)=\left(\frac{3}{8}\right)^2+\left(\frac{-1}{8}\right)^2=\frac{5}{32}=\|x\|^2$$

In each of the run we can also measure the second register to get the approximate values of the components of x i.e. $p(0)\approx\left(\frac{3/8}{\sqrt{5/32}}\right)^2=\frac{9}{10}$ and $p(1)\approx\left(\frac{-1/8}{\sqrt{5/32}}\right)^2=\frac{1}{10}$. Thus, the components will be the root of the probabilities and then multiplying this by the norm to recover the components of

the original vector x, we get,

$$\sqrt{\frac{5}{32}}\left(\sqrt{\frac{9}{10}}\right) = \frac{3}{8}; \quad \sqrt{\frac{5}{32}}\left(\sqrt{\frac{1}{10}}\right) = \frac{1}{8}$$

Now recall that we chose $t = 3/8$ and $C = 1/8$, hence the current x is the solution of At. Thus, in order to recover the original x, we are required to multiply $\frac{t}{C}$ to each of the components. Thus, we get,

$$\frac{3/8}{1/8}\left(\frac{3}{8}\right) = \frac{9}{8}; \quad \frac{3/8}{1/8}\left(\frac{1}{8}\right) = \frac{3}{8}$$

Thus, we get,

$$x = \begin{bmatrix} 9/8 \\ 1/8 \end{bmatrix}$$

Now, checking the value of x by substituting in Ax, we get,

$$\begin{bmatrix} 1 & -1/3 \\ -1/3 & 1 \end{bmatrix} \begin{bmatrix} 9/8 \\ 3/8 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |b\rangle$$

Thus, clearly, we have recovered the original solution x.

Note that here we multiply the components by $\frac{t}{C}$ can be explained as follows:

1. **Multiplication by $\frac{1}{C}$, because recall that we are required to find**

$$|x\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |\lambda_j\rangle$$

but using the algorithm we obtained

$$|x\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} C b_j |\lambda_j\rangle$$

Thus, upon multiplying the components by $\frac{1}{C}$, we recover the original components.

2. **Multiplying by t: Recall that we are not finding the solution of A but solution of scaled A i.e. At.**

$$At\,|x'\rangle = |b\rangle$$

Thus, the original solution will be $|x\rangle = t\,|x'\rangle$. Thus, the multiplication by t.

Hence, upon multiplying the components of the square root of the approximate probabilities obtained from the algorithm by $\frac{t}{C}$, we recover the original components of x.

7.4 Complexity Analysis

Recall that, in the final step of the algorithm we get,

$$\tilde{x} = \sum_{j=0}^{N-1} \frac{Cb_j}{\tilde{\theta'}_j}\,|\lambda_j\rangle$$

Thus, in the register it will be stored as:

$$|\tilde{x}\rangle = \frac{\tilde{x}}{\|\tilde{x}\|} \approx |x\rangle$$

We get the probability of getting outcome 1 is $p(1)$ as:

$$\begin{aligned} p(1) = \|\tilde{x}\|^2 &= \|\sum_{j=0}^{N-1} \frac{Cb_j}{\tilde{\theta'}_j}\,|\lambda_j\rangle\,\|^2 \\ &= C^2\|\sum_{j=0}^{N-1} \frac{b_j}{\tilde{\theta'}_j}\,|\lambda_j\rangle\,\|^2 \\ &\approx C^2\|A^{-1}\,|b\rangle\,\|^2 \end{aligned}$$

Therefore the success probability is determined by

1. The choice of the normalization constant C
2. The norm of the true solution $\|x\| = \|A^{-1}\,|b\rangle\,\|$

To maximize the success probability, C should be chosen to be as large as possible (without exceeding λ_0). So assuming that the exact knowledge of λ_0, we can choose $C = \lambda_0$. For a Hermitian positive definite matrix A, (recall that 2-norm (induced norm is spectral norm) of any matrix A is given as $\|A\|_2 = \sigma_{max} = \sqrt{\lambda_{max}(A^\dagger A)}$, thus, for a Hermitian matrix$(A^\dagger = A; A = Q\Lambda Q^\dagger \implies \|A\|_2 = \sqrt{\lambda_{max}(A^2)} = \sqrt{\lambda_{max}(Q\Lambda Q^\dagger Q\Lambda Q^\dagger)} = \sqrt{\lambda_{max}(Q\Lambda^2 Q^\dagger)} = |\lambda_{max}|$ where λ_{max} is the eigen value with the largest magnitude (irrespecitve of sign)) which is also positive definite (has all the eigen values ≥ 0) A, $\|A\| = \lambda_{N-1}$, and $\|A^{-1}\| = \lambda_0^{-1}$. For simplicity, assume the largest eigenvalue of A is $\lambda_{N-1} = 1$. Then the condition number of A is

$$\kappa = \|A\|\|A^{-1}\| = \frac{\lambda_{N-1}}{\lambda_0} = C^{-1}$$

Also, using the definition of norms ($\|A^{-1}\,|b\rangle\,\| \geq \|A^{-1}\|\|\,|b\rangle\,\|$)

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

Therefore,

$$p(1) = C^2\|A^{-1}\,|b\rangle\,\|^2 \geq \frac{1}{\kappa^2}$$
$$\implies p(1) = \Omega(\kappa^{-2})$$

Thus, in the worst case we need to run the HHL algorithm for $\mathcal{O}(\kappa^2)$ times to obtain the outcome 1 in the signal bit.

Assuming the number of qubits n is large, the circuit depth and the gate complexity of U_{HHL} is mainly determined by those of U_{QPE} because we assumed that it is easy to make $|b\rangle$ and the only other part in the circuit is controlled rotation which has less number of gates than QFT in QPE. Thus, the majority of circuit depth and gate complexity is dominated by QPE of the HHL circuit. Therefore we can measure the complexity of the HHL algorithm in terms of the number of queries to $U = e^{\iota 2\pi A}$.

In order to solve QLSP to precision ϵ, we need to estimate the eigen values to multiplicative accuracy ϵ instead of the standard additive accuracy. **An ϵ-additive precision is an algorithm that for a true solution x returns $\tilde{x} \in [x-\epsilon, x+\epsilon]$. For a ϵ-multiplicative precision algorithm, the algorithm returns $\tilde{x} \in [x(1+\epsilon), x(1-\epsilon)]$.** Assume $\tilde{\theta'_j} = \lambda_j(1+e_j)$ and $|e_j| \leq \frac{\epsilon}{4} \leq \frac{1}{2}$. Then the unnormalized solution satisfies

$$\|\tilde{x} - x\| = \|\sum_j b_j \left(\frac{1}{\tilde{\theta'_j}} - \frac{1}{\lambda_j}\right)|\lambda_j\rangle\,\| \leq \|\sum_j \frac{b_j}{\lambda_j}\left(\frac{-e_j}{1+e_j}\right)|\lambda_j\rangle\,\| \leq \frac{\epsilon}{2}\|x\|$$

Hence, using the property that $|\|\vec{x}\| - \|\vec{y}\|| \leq \|\vec{x} + \vec{y}\|$

$$|1 - \frac{\|\tilde{x}\|}{\|x\|}| \leq \frac{\epsilon}{2}$$

Then the normalized solution satisfies

$$\| \, |\tilde{x}\rangle - |x\rangle \, \| = \| \frac{\tilde{x}}{\|\tilde{x}\|} - \frac{x}{\|x\|} \| \leq |1 - \frac{\|\tilde{x}\|}{\|x\|}| + \frac{\|\tilde{x} - x\|}{\|x\|} \leq \epsilon$$

The discussion requires the QPE to run to additive precision $\epsilon' = \lambda_0 \epsilon = \epsilon/\kappa$. Therefore the query complexity of QPE is $\mathcal{O}(\kappa/\epsilon)$. Counting the number of times needed to repeat the HHl circuit, the worst case query complexity of the HHL algorithm is $\mathcal{O}(\kappa^3/\epsilon)$.

The above ananlysis is the worst case analysis, because we assume $p(1)$ attains the lower bound $\Omega(\kappa^{-2})$. In practical applications, the result may not be so pessimistic. For instance, if b_j centrates around the smallest eigen values of A, then we may have $\|x\| \approx \Theta(\lambda_0^{-1} = \Theta(\kappa^{-1})$. Then $p(1) = \Theta(1)$. IN such a case, we only need to repeat the HHl algorithm for a constant number of times to yield outcome 1 in the ancilla qubit. this does not reduce the query complexity of each run of the algorithm. Then in this best case, the query complexity is $\mathcal{O}(\kappa/\epsilon)$.

Additional Considerations: Below we discuss a few more aspects of the HHL algorithm. The first observation is that the asymptotic worst-case complexity of the HHL algorithm can be generally improved using amplitude estimation.

Remark. (HHL with amplitude amplification).

$$U_{HHL} \, |0\rangle \, |b\rangle = \sqrt{p(1)} \, |1\rangle \, |\psi_{good}\rangle + \sqrt{1 - p(1)} \, |0\rangle \, |\psi_{bad}\rangle \, , \quad |\psi_{good}\rangle = |\tilde{x}\rangle$$

Since $|\psi_{good}\rangle$ is marked by a signal qubit, we may construct a signal qubit to construct a reflection operator with respect to the signal qubit. This is simply given by

$$R_{good} = Z \otimes I_n$$

The reflection with respect to the initial vector is

$$R_{\psi_0} = U_{\psi_0}(2 \, |01 + n\rangle \, \langle 0^{1+n}| - I)U_{\psi_0}^{\dagger}$$

where $U_{\psi_0} = U_{HHL}(I_1 \otimes U_b)$. Let $G = R_{\psi_0} R_{good}$ be the Grover iterate. Then amplitude amplification allows us to apply G for $\Theta(p(1)^{-\frac{1}{2}})$ times to boost the success probability of obtaining ψ_{good} with constant success probability. Therefore in the worst case when $p(1) = \Theta(\kappa^{-2})$, the number of repetitions is reduced to $\mathcal{O}(\kappa)$, and

the total runtime is $\mathcal{O}(\kappa^2/\epsilon)$. This query complexity is the commonly referred query complexity for the HHL algorithm. Note that as usual, amplitude amplification increases the circuit depth. However, the tradeoff is that the circuit depth increases from $\mathcal{O}(\kappa/\epsilon)$ to $\mathcal{O}(\kappa^2/\epsilon)$.

So far our analysis, especially that based on QPE relies on the assumption that all λ_j eigen values have exact d-bit representation. Such an assumption is unrealistic and causes theoretical and practical difficulties. Refer [10] for a more detailed explanation.

Remark. (Comparison with classical iterative linear system solvers). Let us now compare the cost of the HHL algorithm to that of classical iterative algorithms. If A is n-bit Hermitian positive definite with condition number κ, and is d-sparse (i.e. each row/column of A has at most d nonzero entries), then each matrix vector multiplication Ax costs $\mathcal{O}(dN)$ floating point operations. The number of iterations for the steepest descent (SD) algorithm is $\mathcal{O}(\kappa \log \epsilon^1)$, and this number can be significantly reduced to $\mathcal{O}(\sqrt{\kappa} \log \epsilon^{-1})$ by the renowned conjugate gradient (CG) method. Therefore the total cost (or wall clock time) of SD and CD is $\mathcal{O}(dN\kappa \log \epsilon^{-1})$ and $\mathcal{O}(dN\sqrt{\kappa} \log \epsilon^{-1})$, respectively.

On the other hand, the query complexity of the HHL algorithm, even after using the Amplitude amplification algorithm, is still $\mathcal{O}(\kappa^2/\epsilon)$. Such a performance is terrible in terms of both *kappa* and ϵ. Hence the power of the HHL algorithm, and other QLSP solvers is based on that each application of A (in this case, using Unitary U) is much faster. In particular, if$U$ can be implemented with poly(n) gate complexity (also can be measured by the wall clock time), then the total gate complexity of the HHL algorithm (with Amplitude amplification) is $\mathcal{O}(poly(n)\kappa^2/\epsilon)$. When n is large enough, we expect that $poly(n) << N = 2^n$ and the HHL algorithm would eventually yield an advantage. Nonethelesss, for a realistic problem, the assumption that U can be implemented with $poly(n)$ cost should be taken with a grain of salt and carefully examined.

Remark. (Readout problem of QLSP). By solving the QLSP, the solution is stored as a quantum state in the quantum computer, Sometimes the QLSP is only a subproblem of a larger application, so it is sufficient to treat the HHL algorithm (or other QLSP solvers) as a "quantum subroutine", and leave $|x\rangle$ in the quantum computer. However, in many applications (such as the solution of the Poisson's equation) the goal is to solve the linear system. Then the information in $|x\rangle$ must be recovered to a measurable classical output.

The most common case is to compute the expectation of some observable $\langle \mathcal{O} \rangle = \langle x|\mathcal{O}|x\rangle \approx \langle \tilde{x}|\mathcal{O}|\tilde{x}\rangle$. Assuming $\langle \mathcal{O} \rangle = \Theta(1)$. Then to reach additive precision ϵ of the

observable, the number of samples needed is $mathcalO(\epsilon^{-2})$. On the other hand, in order to reach precision ϵ, the solution vector $|\tilde{x}\rangle$ must be solved to precision ϵ. Assuming the worst case analysis for the HHL algorithm, the total query complexity needed is

$$\mathcal{O}(\kappa^2/\epsilon) \times \mathcal{O}(\epsilon^{-2}) = \mathcal{O}(\kappa^2/\epsilon^3)$$

Remark. (Query complexity lower bound). The cost of a quantum algorithm for solving a generic QLSP scales at least as $\Omega(\kappa(A))$ where $\kappa(A) = \|A\|\|A^{-1}\|$ is the condition number of A. The proof is based on converting the QLSP into a Hamiltonian simulation problem, and the lower bound with respect to κ is proved via the no-fast-forwarding" theorem for simulating generic Hamiltonians [2]. Nonetheless, for speicific classes of hamiltonians, it may still be possible to develop fast algorithm to overcome this lower bound.

7.5 Improvements in HHL algorithm

7.6 Implementation using Qiskit

7.7 Applications

7.7.1 Poisson's problem

Poisson's equation is a partial differential equation that describes the distribution of a potential field in a region. The equation is given as:

$$\nabla^2 u + f = 0$$

where u is the potential field, and f is the source term. The Poisson's equation can be solved using the finite difference method. The finite difference method approximates the derivatives in the equation using finite difference schemes and discretizes the region into a grid, and the potential field is calculated at the grid points. The finite difference method converts the Poisson's equation into a linear system of equations, which can be solved using the HHL algorithm. Consider a toy problem of solving the Poisson's equation in one dimension with Dirichlet Boundary conditions as:

$$-u''(x) = f(x) \quad x \in [0,1] \quad u(0) = u(1) = 0$$

Using the central finite difference formula for discretizing the Laplacian operator on a uniform grid.

$$-\frac{\partial^2 u}{\partial x^2} = f(x) \quad x \in [0,1] \quad u(0) = u(1) = 0$$

Now recall that the central difference formula for discretizing second order derivative is

$$\frac{\partial^2 u}{\partial x^2} = \left(\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}\right)$$

Substituting in the equation we get the discretized form of the above Poisson's equation as:

$$-\left(\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}\right) = f(x_i)$$

Since we already know the solution at the boundary points from dirichlet conditions, thus, ignoring the values at the boundary points and writing equations at the N interior points we get,

$$A = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \dots & 0 & 0 & 0 \\ -1 & 2 & -1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 2 & -1 \\ 0 & 0 & 0 & \dots & 0 & -1 & 2 \end{bmatrix}$$

where $h = \frac{1}{N+1}$, $x_i \; \forall \; i \in \{1, \dots, N\}$. Let $u_i = u(x_i)$ and $f_i = f(x_i)$. We assume that vector b is already normalized so that $|f\rangle = f$.

Proposition 7.7.1. *(Diagonalization of tridiagonal matrices) Consider the following Hermitian toeplitz tridiagonal matrix:*

$$A = \begin{bmatrix} a & \bar{b} & 0 & \dots & 0 & 0 & 0 \\ b & a & \bar{b} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & b & a & \bar{b} \\ 0 & 0 & 0 & \dots & 0 & b & a \end{bmatrix} \in \mathbb{C}^{N \times N}$$

can be analytically diagonalized as

$$Av_k = \lambda_k v_k, \quad k = 1, \dots, N.$$

where $(v_k)_j = v_{j,k}$ denotes the jth component of v_k eigen vector, $j = 1, \dots, N, b = |b|e^{\iota\theta}$, and

$$\lambda_k = a + 2|b| \cos \frac{k\pi}{N+1}; \quad v_{j,k} = \sin \frac{jk\pi}{N+1} e^{\iota j\theta}$$

Proof. Now $v_{j,k} = \sin\frac{jk\pi}{N+1}e^{\iota j\theta}$. Thus, note that $v_{0,k} = v_{N+1,k} = 0$. Then direct matrix vector multiplication above shows that for any $j = 1, \ldots, N$,

$$(Av_k)_j = bv_{k,j-1} + av_{k,j} + \overline{b}v_{k,j+1}$$

Upon substituting the value of $v_{k,j}$ we get,

$$\begin{aligned}
(Av_k)_j &= b\sin\frac{(j-1)k\pi}{N+1}e^{\iota(j-1)\theta} + a\sin\frac{jk\pi}{N+1}e^{\iota j\theta} + \overline{b}\sin\frac{(j+1)k\pi}{N+1}e^{\iota(j+1)\theta} \\
&= a\sin\frac{jk\pi}{N+1}e^{\iota j\theta} + |b|e^{\iota\theta}\sin\frac{(j-1)k\pi}{N+1}e^{\iota(j-1)\theta} + |b|e^{-\iota\theta}\sin\frac{(j+1)k\pi}{N+1}e^{\iota(j+1)\theta} \\
&= a\sin\frac{jk\pi}{N+1}e^{\iota j\theta} + |b|\left(\sin\frac{(j-1)k\pi}{N+1} + \sin\frac{(j+1)k\pi}{N+1}e^{\iota j\theta}\right) \\
&= a\sin\frac{jk\pi}{N+1}e^{\iota j\theta} + 2|b|\sin\frac{jk\pi}{N+1}\cos\frac{k\pi}{N+1}e^{\iota j\theta} \\
&= \left(a + 2|b|\cos\frac{k\pi}{N+1}\right)\sin\frac{jk\pi}{N+1}e^{\iota j\theta} \\
&= \lambda_k v_{j,k} \\
&= \lambda_k (v_k)_j
\end{aligned}$$

□

Using this proposition and substituting with $a = 2/h^2$, $b = -1/h^2$, the largest eigen value of A is $\lambda_{max} = \|A\| \approx \frac{4}{h^2}$, and the smallest eigen value $\lambda_{min} = \|A^{-1}\|^{-1} \approx \pi^2$. So,

$$\kappa(A) = \|A\|\|A^{-1}\| \approx \frac{4}{h^2\pi^2} = \mathcal{O}(N^2)$$

Recall that the circuit depth of the HHL algorithm is $\mathcal{O}(N^2/\epsilon)$, and the worst case query complexity (using AA) is $\mathcal{O}(N^4/\epsilon)$. So when N is large, there is little benefit in employing the quantum computer to solve this problem.

Let us now consider solving a d-dimensional Poisson's equation with Dirichlet boundary conditions

$$-\nabla^2 u(r) = b(r), \quad r = \Omega = [0,1]^d, \quad u|_{\partial\Omega} = 0$$

The grid is the Cartesian product of the uniform gird in 1D with N grid points per dimension and $h = 1/(N+1)$. The total number of grid points is $N = N^d$. After discretization we will obtain a linear system $Au = b$, where

$$A = A \otimes I \otimes I \ldots \otimes I + I \otimes A \otimes I \ldots \otimes I + \ldots + I \otimes I \ldots \otimes A$$

where I is the identity matrix of size N. Since A is Hermitian and positive definite, we have $\|A\| \approx 4d/h^2$, and $\|A^{-1}\|^{-1} = d\pi^2$. So $\kappa(A) \approx 4/(h^2\pi^2) \approx \kappa(A)$. There for the condition number is independent of the spatial dimension d.

The worst case query complexity of the HHL algorithm is $\mathcal{O}(N^2/\epsilon)$. So when the number of grid points per dimension N is fixed and the spatial dimension d increases, and if $U = e^{\iota A\tau}$ can be implemented efficiently for some $\tau\|A\| < 1$ with poly(d) cost, then the HHL algorithm will have an advantage over classical solvers, for which each matrix-vector multiplication scales linearly with N and is therefore exponential in d.

7.7.2 Solve Linear Differential Equations

Consider the solution of a time-dependent linear differential equation

$$\frac{d}{dt}x(t) = A(t)x(t) + b(t), \quad t \in [0, T]$$

$$x(0) = x_0$$

Here $b(t), x(t) \in \mathbb{C}^d$ and $A(t) \in \mathbb{C}^{d\times t}$. The above differential equation can be used to represent a very large class of ordinary differential equations (ODEs) and spatially discretized partial differential equations (PDEs). For example, if $A(t) = -\iota H(t)$ for some Hermitian Matrix $H(t)$ and $b(t) = 0$, this is the time-dependent Schrodinger equation

$$\iota\frac{d}{dt}x(t) = H(t)x(t)$$

A special case when $H(t) = H$ is often called the **Hamiltonian Simulation Problem**, and the solution can be written as

$$x(T) = e^{\iota HT}x(0)$$

which can be viewed as the problem of evaluating the matrix functions $e^{\iota HT}$.

In this section we consider the general case, and for simplicity discretize the equation in time using the forward Euler method with a uniform grid $t_k = k\Delta t$ where $\Delta t = T/N, k = 0, \ldots, N$. Let $A_k = A(t_k), b_k = b(t_k)$. The resulting discretized system become

$$x_{k+1} - x_k = \Delta t(A_k x_k + b_k), \quad k = 1, \ldots, N$$

$$x_{k+1} = (A_k\Delta t + I)x_k + b_k\Delta t$$

$$x_{k+1} - (I + A_k\Delta t)x_k = b_k\Delta t$$

$$-(I + A_k\Delta t)x_k + x_{k+1} = b_k\Delta t$$

Note that for $t = 0$, since we know that $x(0) = x_0$, thus the above equation reduces to

$$x_1 = (I + A_0\Delta t)x_0 + \Delta t b_0$$

which can be rewritten as

$$\begin{bmatrix} I & 0 & 0 & \dots & 0 & 0 \\ -(I+\Delta t A_1) & I & 0 & \dots & 0 & 0 \\ 0 & -(I+\Delta t A_2) & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -(I+\Delta t A_{N-1}) & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} (I+\Delta t A_0)x_0 + b_0\Delta t \\ b_1\Delta t \\ b_2\Delta t \\ \vdots \\ b_{N-1}\Delta t \end{bmatrix}$$

or more compactly as a linear system of equations

$$\mathcal{A}x = b$$

Here I is the identity matrix of size d, and $x \in \mathbb{R}^{Nd}$ encodes the entire history of the states.

To solve the equation as a QLSP, the right hand side b needs to be a normalized vector. This means we need to properly normalize x_0, b_k so that

$$\|b\|^2 = \|(I + \Delta t A_0)x_0 + \Delta t b_0\|^2 + \sum_{k\in[N-1]} (\Delta t)^2 \|b_k\|^2 = 1$$

In the limit $\Delta t \to 0$, this can be written as

$$\|b\|^2 = \|x_0\|^2 + \int_0^T \|b(t)\|^2 dt = 1$$

which is not difficult to satisfy as long as $x_0, b(t)$ can be prepared efficiently using unitary circuits and $\|x_0\|, \|b(t)\| = \Theta(1)$.

To solve this equation using the HHL algorithm we need to estimate the condition number of $\mathcal{A}$. Note that $\mathcal{A}$ is a block-diagonal matrix and in particular is not Hermitian. So we need to use the dilation method and solve the corresponding Hermitian problem.

Scalar case

In order to estimate the condition number, for simplicity we first assume $d = 1$ (i.e., this is a scalar ODE problem), and $A(t) = a \in \mathbb{C}$ is a constant. Then

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -(1+a\Delta t) & 1 & 0 & \dots & 0 & 0 \\ 0 & -(1+a\Delta t) & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -(1+a\Delta t) & 1 \end{bmatrix} \in \mathbb{C}^{N\times N}$$

Recall that matrix $\mathcal{A}$ should be invertible. Thus, let $\zeta = 1 + a\Delta t$. When $Re(a) \leq 0$, the absolute stability condition of the forward Euler method requires $|1 + a\Delta t| = |\zeta| < 1$. In general we are interested in the regime $\Delta t|a| << 1$, and in particular $|1 + a\Delta t| = |\zeta| < 2$.

Proposition 7.7.2. *For any $A \in \mathbb{C}^{N\times N}$, $\|A\|^2, (1/\|A^{-1}\|)^2$ are given by the largest and smallest eigen values of $A^\dagger A$ respectively.*

We first need to compute

$$\mathcal{A}^\dagger\mathcal{A} = \begin{bmatrix} 1+|\zeta|^2 & -\overline{\zeta} & \dots & \dots & 0 & 0 \\ -\zeta & 1+|\zeta|^2 & -\overline{\zeta} & \dots & 0 & 0 \\ 0 & -\zeta & 1+|\zeta|^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1+|\zeta|^2 & -\overline{\zeta} \\ 0 & 0 & 0 & \dots & -\zeta & 1 \end{bmatrix}$$

Theorem 7.7.3. *(Greshgorin Disk Theorem) Let $A \in \mathbb{C}^{N\times N}$ with entries a_{ij}. For each $i = 1, \dots, N$, define*

$$R_i = \sum_{j\neq i} |a_{ij}|$$

Let $D(a_{ii}, R_i) \subseteq \mathbb{C}$ be a closed disc centered at a_{ii} with radius R_i, which is called a Greshgorin disk. Then every eigen value of A lies within at least one of the Greshgorin discs $D(a_{ii}, R_i)$.

Since $\mathcal{A}^\dagger A$ is Hermitian, we can restrict the Greshgorin discs to the real line so that $D(A_{ii}, R_i) \subseteq \mathbb{R}$. Then Greshgorin discs of the matrix $\mathcal{A}^\dagger\mathcal{A}$ satisfy the bound

$$D(a_{11}, R_1) \subseteq [1 + |\zeta|^2 - |\zeta|, 1 + |\zeta|^2 + |\zeta|]$$

$$D(a_{ii}, R_i) \subseteq [1 + |\zeta|^2 - 2|\zeta|, 1 + |\zeta|^2 + 2|\zeta|], \quad i = 2, \ldots, N-1$$
$$D(a_{NN}, R_N) \subseteq [1 - |\zeta|, 1 + |\zeta|]$$

Now using the proposition we have, for $D(a_{ii}, R_i) \subseteq [1+|\zeta|^2-2|\zeta|, 1+|\zeta|^2+2|\zeta|], \quad i = 2, \ldots, N-1$. Thus, the maximum eigen value is

$$\lambda_{max}(\mathcal{A}^\dagger \mathcal{A}) \leq (1 + |\zeta|)^2 < 9$$

for all values of a such that $|\zeta| < 2$.

To obtain a meaningful lower bound of $\lambda_{min}(\mathcal{A}^\dagger \mathcal{A})$, we need $Re(a) < 0$ and hence $|\zeta| < 1$. Use the inequality

$$\sqrt{1+x} \leq 1 + \frac{1}{2}x, \quad x > -1$$
$$\implies -\sqrt{1+x} \geq -1 - \frac{1}{2}x, \quad x > -1$$
$$\implies 1 - \sqrt{1+x} \geq -\frac{1}{2}x. \quad x > -1$$

we have

$$1-|\zeta| = 1-\sqrt{(1 + \Delta t Re(a))^2 + (\Delta t)^2 (Im(a))^2} \geq -\frac{1}{2}((\Delta t)^2 (Re(a))^2 + 2Re(a)\Delta t + (\Delta t)^2 (Im(a))^2)$$

$$1 - |\zeta| \geq -Re(a)\Delta t - \frac{1}{2}(\Delta t)^2 |a|^2 \geq -\frac{\Delta t}{2} Re(a)$$

when $(\Delta t)^2 |a|^2 < (-\Delta t Re(a))$ is satisfied. Then we have

$$1 > 1 - |\zeta| \geq -\frac{\Delta t}{2} Re(a)$$

Now using the proposition we have, for $D(a_{ii}, R_i) \subseteq [1+|\zeta|^2-2|\zeta|, 1+|\zeta|^2+2|\zeta|], \quad i = 2, \ldots, N-1$. Thus, the minimum eigen value is

$$\lambda_{min}(\mathcal{A}^\dagger \mathcal{A}) \geq (1 - |\zeta|)^2 \geq \frac{(\Delta t Re(a))^2}{4}$$

Therefore

$$\|A\| = \sqrt{\lambda_{max}(\mathcal{A}^\dagger \mathcal{A})} \leq \sqrt{1 + |\zeta|^2 + 2|\zeta|} < 3$$

and

$$\|A^{-1}\|^{-1} = \sqrt{\lambda_{min}(\mathcal{A}^\dagger \mathcal{A})} \geq \frac{\Delta t(-Re(a))}{2}$$

As a result, when $(-Re(a)) = \Theta(1)$, the condition number satisfies

$$\kappa(\mathcal{A}) = \|\mathcal{A}\|\|\mathcal{A}^{-1}\| = \mathcal{O}(1/\Delta t)$$

In summary, for the scalar problem $d = 1$ and $Re(a) < 0$, the query complexity of the HHL algorithm is $\mathcal{O}((\Delta t)^{-2}\epsilon^{-2})$.

Remark. It may be tempting to modify the $(N.N)$ th entry of $\mathcal{A}$ to be $1 + |\zeta|^2$ to obtain

$$\mathcal{G} = \begin{bmatrix} 1+|\zeta|^2 & -\overline{\zeta} & 0 & \dots & 0 & 0 \\ -\zeta & 1+|\zeta|^2 & -\overline{\zeta} & \dots & 0 & 0 \\ 0 & -\zeta & 1+|\zeta|^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1+|\zeta|^2 & -\overline{\zeta} \\ 0 & 0 & 0 & \dots & -\zeta & 1+|\zeta|^2 \end{bmatrix}$$

Here $\mathcal{G}$ is a Toeplitz tridiagonal matrix satisfying the requirements of Proposition 7.7.1. The eigen values of $\mathcal{G}$ take the form

$$\lambda_k = 1 + |\zeta|^2 + 2|\zeta| \cos \frac{k\pi}{N+1}; \quad k = 1, \dots, N$$

If we take the approximation $\lambda_{min}(\mathcal{A}^\dagger \mathcal{A}) \approx \lambda_{min}(\mathcal{G})$, we would find that the above equation holds even when $Re(a) > 0$. This behavior is however incorrect, despite that the matrices $\mathcal{A}^\dagger \mathcal{A}$ and $\mathcal{G}$ only differ by a single entry!

Vector Case

Here we consider a general $d > 1$, but for simplicity assume $A(t) = A \in \mathbb{C}^{d \times d}$ is a constant matrix. We also assume A is diagonalizable with eigenvalue decomposition

$$A = V \Lambda V^{-1}$$

and $\Lambda = diag(\lambda_1, \dots, \lambda_N)$. We only consider the case $Re(\lambda_k) < 0$ for all k.

Proposition 7.7.4. *For any diagonalizable $A \in \mathbb{C}^{N \times N}$ with eigenvalue decomposition $Av_k = \lambda v_k$, we have*

$$\|A^{-1}\|^{-1} \leq \min_k |\lambda_k| \leq \max_k |\lambda_k| \leq \|A\|$$

Proof. Use the Schur form $A = QTQ^\dagger$, where Q is an unitary matrix and T is an upper triangular matrix. The diagonal entries of T encodes all eigenvalues of A, and the eigenvalues can appear in any order along the diagonal of T. The proposition follows by arranging the eigenvalue of A with the smallest and largest absolute values to the (N,N)th entry, respectively. □

The absolute stability condition of the forward Euler method requires $\Delta t\|A\| < 1$, and we are interested in the regime $\Delta t\|A\| << 1$. Therefore $\Delta t|\lambda_k| << 1$ for all k. Let I be the identity matrix of size d, and denote $B = -(I + \Delta tA)$, then

$$\mathcal{A}^\dagger\mathcal{A} = \begin{bmatrix} I + B^\dagger B & B & \dots & 0 & 0 & 0 \\ B & I + B^\dagger B & B^\dagger & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I + B^\dagger B & B^\dagger \\ 0 & 0 & 0 & \dots & B & I \end{bmatrix}$$

Note that

$$\|B\| \leq \|I + B^\dagger B\| \leq 1 + (1 + \Delta t\|A\|)^2 \leq 5$$

Bibliography

[1] Aram W. Harrow, Avinatan Hassidim, Seth Lloyd, *Quantum algorithm for linear systems of equations*, Physical Review Letters 103, 150502 (2009), https://arxiv.org/abs/0811.3171

[2] various authors. (2023). Qiskit Textbook. Github. https://github.com/Qiskit/textbook https://github.com/Qiskit/textbook

[3] De Wolf, R. (2019). Quantum computing: Lecture notes. ArXiv Preprint ArXiv:1907.09415. https://doi.org/10.48550/arXiv.1907.09415

[4] Lin, L. (2022). Lecture notes on quantum algorithms for scientific computation. ArXiv Preprint ArXiv:2201.08309. https://doi.org/10.48550/arXiv.2201.08309

[5] Shewchuk, J. R., others. (1994). An introduction to the conjugate gradient method without the agonizing pain. https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf

[6] Dewes, A., Ong, F. R., Schmitt, V., Lauro, R., Boulant, N., Bertet, P., Vion, D., Esteve, D. (2012). Characterization of a two-transmon processor with individual single-shot qubit readout. Physical Review Letters, 108(5), 057002. https://doi.org/10.1103/PhysRevLett.108.057002

[7] Stamatopoulos, N., Egger, D. J., Sun, Y., Zoufal, C., Iten, R., Shen, N., Woerner, S. (2020). Option pricing using quantum computers. Quantum, 4, 291. https://doi.org/10.22331/q-2020-07-06-291

[8] Iten, R., Reardon-Smith, O., Malvetti, E., Mondada, L., Pauvert, G., Redmond, E., Kohli, R. S., Colbeck, R. (2019). Introduction to universalqcompiler. ArXiv Preprint ArXiv:1904.01072. https://doi.org/10.48550/arXiv.1904.01072

[9] https://quantum.ibm.com/

[10] Dervovic, D., Herbster, M., Mountney, P., Severini, S., Usher, N., Wossnig, L. (2018). Quantum linear systems algorithms: a primer. `https://arxiv.org/abs/1802.08227`

Chapter 8

Block Encoding

In order to perform matrix computations, we must first address the problem of the input model: how to get access to information in matrix $A \in \mathbb{C}^{N\times N}(N = 2^n)$ which is generally a non-unitary matrix, into the quantum computer? One possible input model is given via the unitary $e^{\iota\tau A}$ (if A is not Hermitian, in some scenarios we an consider its Hermitian version via the dilation method). This is particularly useful when $e^{\iota\tau A}$ can be constructed using simple circuits, e.g. Trotter Splitting,

A more general input model, as will be discussed in this chapter is called "block encoding". Of course, if A is a dense matrix without obvious structures, any input model will be very expensive (e.g. exponential in n) to implement. Therefore a commonly assumed input model is s-sparse there are at most s nonzero entries in each row/column of the matrix. Furthermore, we have an efficient procedure to get access to the location, as well as the value of the nonzero entries. This in general can again be a difficult task given that the number of nonzero entries can still be exponential in n for a sparse matrix. Some dense matrices may also be efficiently block encoded on quantum computers. This chapter will illustrate the block encoding procedure via a number of detailed examples.

8.1 Query model for matrix entries

The query model for sparse matrices is based on certain quantum oracles. In some scenarios, these quantum oracles can be implemented all the way to the elementary gate level. Throughout the discussion we assume A is an n-qbuit, square matrix, and

$$\|A\|_{max} = \max_{ij} |A_{ij}| < 1$$

If the $\|A\|_{max} \geq 1$, we can simply consider the rescaled matrix $\tilde{A}/\alpha$ for some $\alpha > \|A\|_{max}$. To query the entries of a matrix, the desired oracle takes the following general form

$$O_A |0\rangle |i\rangle |j\rangle = \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2} |1\rangle \right) |i\rangle |j\rangle$$

In other words, given $i, j \in [N]$ and a signal bit 0, O_A performs a controlled rotation (controlling on i,j) of the signal bit, which encodes the information in terms of amplitude of $|0\rangle$.

However, the classical information in A is usually not stored natively in terms of such an oracle O_A. Sometimes it is more natural to assume that there is an oracle

$$\tilde{O}_A |0^{d'}\rangle |i\rangle |j\rangle = |\tilde{A}_{ij}\rangle |i\rangle |j\rangle$$

where $\tilde{A}_{ij}$ is a d'-bit fixed point representation of A_{ij}, and the value of $\tilde{A}_{ij}$ is either computed on the fly with a quantum computer, or obtained through an external database. In either case, the implementation of $\tilde{O}_A$ may be challenging, and we will only consider the query complexity with respect to this oracle.

Using classical arithmetic operations, we can convert this oracle into an oracle

$$O'_A |0^d\rangle |i\rangle |j\rangle = |\tilde{\theta}_{ij}\rangle |i\rangle |j\rangle$$

where $0 \leq \tilde{\theta}_{ij} < 1$, and $\tilde{\theta}_{ij}$ is a d-bit representation of $\theta_{ij} = arccos(A_{ij})/\pi$. This step may require some additional work registers not shown here.

Now using the controlled rotation, the information of $\tilde{A}_{ij}$, $\tilde{\theta}_{ij}$ has now been transferred to the phase of the signal bit. We should then perform uncomputation and free the work work register storing such intermediate information $\tilde{A}_{ij}$, $\tilde{\theta}_{ij}$. The procedure is as follows:

$$\begin{aligned} |0\rangle |0^d\rangle |i\rangle |j\rangle &\xrightarrow{O'_A} |0\rangle |\tilde{\theta}_{ij}\rangle |i\rangle |j\rangle \\ &\xrightarrow{CR} \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2} |1\rangle \right) |\tilde{\theta}_{ij}\rangle |i\rangle |j\rangle \\ &\xrightarrow{(O'_A)^{-1}} \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2 x} |1\rangle \right) |0^d\rangle |i\rangle |j\rangle \end{aligned}$$

This can be implemented as shown in the circuit 8.1

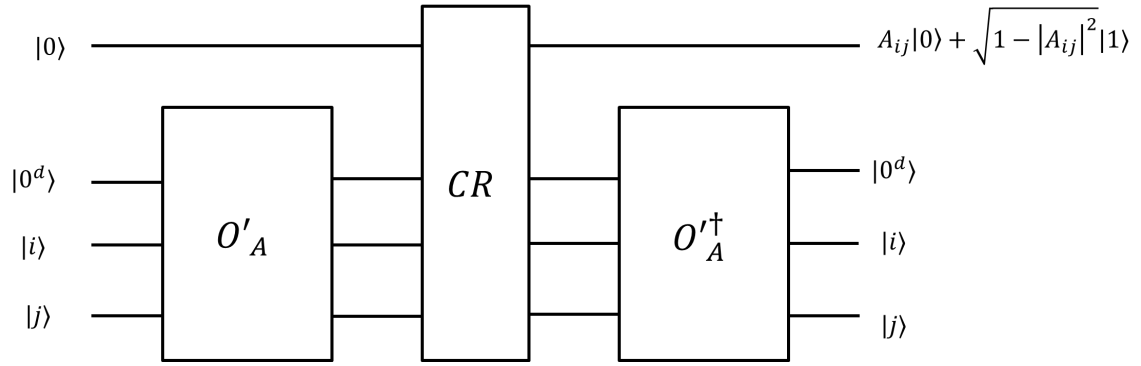


Figure 8.1: Query model for matrix entries

From now on, we will always assume that the matrix entries of A can be queried using the phase oracle O_A or its variants.

8.2 Block Encoding

The simplest example of block encoding is the following: assume we can find a (n+1)qubit unitary U (i.e. $U \in \mathbb{C}^{2N\times 2N}$ such that

$$U_A = \begin{bmatrix} A & * \\ * & * \end{bmatrix}$$

here * means that the corresponding entries are irrelevant, then for any n-qubit quantum state $|b\rangle$, we can consider the state

$$|0,b\rangle = |0\rangle\,|b\rangle = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

and

$$U_A\,|0,b\rangle = \begin{bmatrix} Ab \\ * \end{bmatrix} = |0\rangle\,A\,|b\rangle + |\perp\rangle$$

$$U_A\,|0,b\rangle = \|A\,|b\rangle\,\| \left(|0\rangle \otimes \frac{A\,|b\rangle}{\|A\,|b\rangle\,\|}\right) + \|\,|\psi\rangle\,\| \left(|1\rangle \otimes \frac{|\psi\rangle}{\|\,|\psi\rangle\,\|}\right)$$

Here the unnormalized state $|\perp\rangle$ can be written as $|1\rangle\,|\psi\rangle$ for some unnormalized state $|\psi\rangle$, that is irrelevant to the computation of $A\,|b\rangle$. In particular, it satisfies the orthogonality relation.

$$(\langle 0| \otimes I_n)\,|\perp\rangle = 0$$

in order to obtain $A|b\rangle$, we need to ensure measure the qubit 0 and only keep the state if it returns 0. This can be summarised into the following quantum circuit as shown in the figure 8.2.

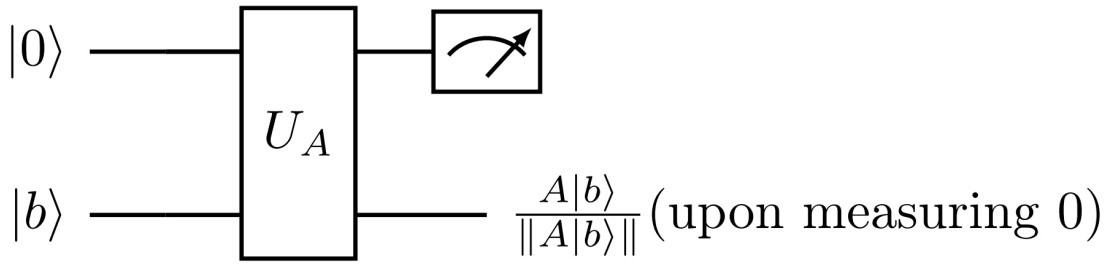


Figure 8.2: Circuit for Block Encoding A using one Ancilla qubit

Note that the output state is normalized after the measurement takes place. The success probability of obtaining 0 from the measurement can be computed as

$$p(0) = \|A|b\rangle\|^2 = \langle b|A^\dagger A|b\rangle\rangle$$

So the missing information of norm $\|A|b\rangle\|$ can be recovered via the success probability $p(0)$ if needed. We find that the success probability is only determined by $A|b\rangle$, and is independent of other irrelevant components of U_A.

Note that we may not need to restrict the matrix U_A to be a $(n+1)$-qubit matrix. if we can find any (n+m)-qubit matrix U_A so that

$$U_A = \begin{bmatrix} A & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots \\ * & * & \dots & * \end{bmatrix}$$

Here each $*$ stands for an n-qubit matrix, and there are 2^m block rows/columns in U_A. The relation above can be written compactly using the braket notation as

$$A = (\langle 0^m| \otimes I_n) U_A (|0^m\rangle \otimes I_n)$$

since $\langle 0^m| \otimes I_n$ will be a block row vector with Identity matrix (of size $2^n \times 2^n$) at the first entry and 0 everywhere else. Similarly, $|0^m\rangle \otimes I_n$ will be a block column

vector with Identity matrix at the first entry and 0 everywhere else. Thus, the above expression can be thought of as being simplified to

$$\langle 0^m| \otimes I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} I_n & 0 & 0 & \dots & 0 \end{bmatrix}$$

where I_n is the n-qubit identity matrix of size $2^n \times 2^n$ and 0 indicates block matrices of size $2^n \times 2^n$ with all entries 0. Thus, $\langle 0^m| \otimes I_n$ is a matrix of size $2^n \times 2^{n+m}$. Similarly for $|0^m\rangle \otimes I_n$, we get,

$$|0^m\rangle \otimes I_n = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} I_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Thus, on putting it all together we get,

$$(\langle 0^m| \otimes I_n) U_A (|0^m\rangle \otimes I_n) = \begin{bmatrix} I_n & 0 & 0 & \dots & 0 \end{bmatrix} U_A \begin{bmatrix} I_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = A$$

A necessary condition for the existence of U_A is $\|A\| \leq 1$. (Note: $\|A\|_{max} \leq 1$ does not guarantee that $\|A\| \leq 1$). However, if we can find sufficiently large α and U_A so that

$$A/\alpha = (\langle 0^m| \otimes I_n) U_A (|0^m\rangle \otimes I_n)$$

Measuring the m ancilla qubits and given that all m-qubits return 0, we still obtain the normalized state $\frac{A|b\rangle}{\|A|b\rangle\|}$. The number α is hidden in the success probability:

$$p(0^m) = \frac{1}{\alpha^2} \|A\,|b\rangle\,\|^2 = \frac{1}{\alpha^2} \langle b|A^\dagger A\,|b\rangle\rangle$$

So if α is chosen to be too large, the probability of obtaining all 0s from the measurement can be vanishingly small.

Finally, it can be difficult to find U_A to encode A exactly. This is not a problem, since it is sufficient if we can find U_A to block encode A upto some error ϵ. We are now ready to give the definition of block encoding.

Definition 8.2.1. (Block encoding) Given an n-qubit matrix A, if we can find $\alpha, \epsilon \in \mathbb{R}_+$, and an $(m+n)$-qubit unitary matrix U_A so that

$$\|A - \alpha(\langle 0^m| \otimes I_n)U_A(|0^m\rangle \otimes I_n)\| \leq \epsilon$$

then U_A is called an (α, m, ϵ)-block encoding of A. When the block encoding is exact with $\epsilon = 0$, U_A is called an (α, m)-block encoding of A. The set of all (α, m, ϵ)-block-encoding of A is denoted by $BE_{\alpha,m}(A, \epsilon)$, and we define $BE_{\alpha,m}A = BE(A, 0)$.

Assume we know each matrix element of the n-qubit matrix A_{ij}, and we are given an (n+m)-qubit unitary U_A. In order to verify that $U_A \in BE_{1,m}(A)$ we only need to verify that

$$\langle 0^m| \langle i| U_A |0^m\rangle |j\rangle = A_{ij}$$

Here, $|0^m\rangle |j\rangle$ will be a column vector of size $(2^m \times 1 \otimes 2^n \times 1 = 2^{m+n} \times 1)$ which will be 1 in the jth row and 0 everywhere else. Similar arguments can be used for establishing $\langle 0^m| \langle i|$ which will be a row vector with 1 in the ith column entry and 0 everywhere else. Thus, upon multiplying the matrices it will pick the i, j th entry of the matrix U_A which will be A_{ij}. U_A applied to the vector $|0^m, b\rangle$ can be obtained via the superposition principle.

Therefore we may first evaluate the state $U_A |0^m, j\rangle$, and perform inner product with $|0^m, i\rangle$ and verify the resulting the inner product is A_{ij}. We will also use the following technique frequently. Assume $U_A = U_B U_C$, and then

$$\langle 0^m, i|U_A |0^m, j\rangle\rangle = \langle 0^m, i|U_B U_C |0^m, j\rangle\rangle = (U_B^\dagger |0^m, i\rangle)^\dagger (U_C |0^m, j\rangle)$$

So we can evaluate the states $U_B^\dagger |0^m, i\rangle, U_C |0^m j\rangle$ independently, and then verify the inner product is A_{ij}. Such a calculation amounts to running the circuit shown in fig 8.3, and if the ancilla qubits are measured to be 0^m, the system qubits return the normalized state $\sum_i A_{ij} |i\rangle / \| \sum_i A_{ij} |i\rangle \|$.

Example 8.2.1. ((1,1)-block-encoding in general). For any n-qubit matrix A with $\|A\|_2 \leq 1$, the singular value decomposition (SVD) of A is denoted by $W\Sigma V^\dagger$, where all singular values in the diagonal matrix Σ belong to $[0, 1]$. Then we may construct an (n+1)-qubit unitary matrix

$$\begin{aligned} U_A &= \begin{bmatrix} W & 0 \\ 0 & I_n \end{bmatrix} \begin{bmatrix} \Sigma & \sqrt{I_n - \Sigma^2} \\ \sqrt{I_n - \Sigma^2} & -\Sigma \end{bmatrix} \begin{bmatrix} V^\dagger & 0 \\ 0 & I_n \end{bmatrix} \\ &= \begin{bmatrix} A & W\sqrt{I_n - \Sigma^2} \\ \sqrt{I_n - \Sigma^2}V^\dagger & -\Sigma \end{bmatrix} \end{aligned}$$

which is a $(1, 1)$-block-encoding of A.

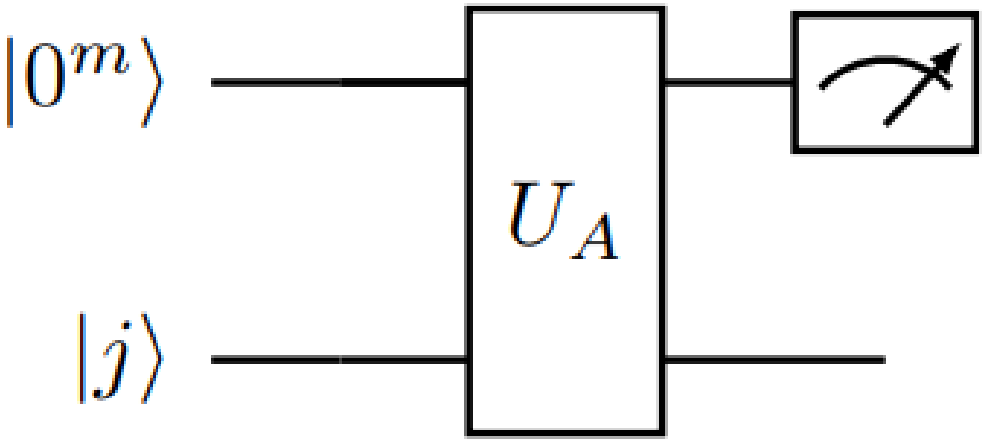


Figure 8.3: Circuit for general block encoding of A

Example 8.2.2. (Random circuit block encoded matrix). In some scenarios, we may want to construct a psuedo-random non-unitary matrix on quantum computers. Note that it would be highly inefficient if we first generate a dense psuedo-random matrix A classically and then feed it into the quantum computer using e.g. quantum-random-access memory (QRAM). Instead we would like to work with matrices that are inherently easy to generate on quantum computers. This inspires the random circuit based block encoding matrix (RACBEM) model. Instead of first identifying A and then finding its block encoding U_A, we reverse this thought process: we first identify a unitary U_A that is easy to implement on a quantum computer, and then ask which matrix can be block encoded by U_A.

This example shows that in principle, any matrix A with $\|A\|_2 \leq 1$ can be accessed via a $(1,1,0)-block-encoding$. In other words, A can be block encoded by an $(n+1)$-qubit random unitary U_A, and U_A can be constructed using only one-qubit unitaries and CNOT gates. The layout of the two-qubit operations can be designed to be compatible with the coupling map of the hardware.

Example 8.2.3. (Block encoding of a diagonal matrix). As a special case, let us consider the block encoding of a diagonal matrix. Since the row and column indices

are the same, we may simplify the oracle into

$$O_A |0\rangle |i\rangle = \left(A_{ii} |0\rangle + \sqrt{1 - |A_{ii}|^2} |1\rangle \right) |i\rangle$$

In the case when the oracle $\tilde{O}_A$ is used, we may assume accordingly

$$\tilde{O}_A |0^{d'}\rangle |i\rangle = |\tilde{A}_{ii}\rangle |i\rangle$$

Let $U_A = O_A$. Direct calculation shows that for any $i, j \in [N]$,

$$\langle 0| \langle i|U_A|0\rangle |j\rangle = A_{ii}\delta_{ij}$$

This proves that $U_A \in BE_{1,1}(A)$i.e., U_A is a $(1,1)$-block-encoding of the diagonal matrix A.

8.3 Block Encoding of s-sparse matrix

We now give a few examples of block encodings of more general sparse matrices. We start from a 1-sparse matrix, i.e., there is only one nonzero entry in each row or column of the matrix. This means that for each $j \in [N]$, there is a unique $c(j) \in [N]$ such that $A_{c(j),j} \neq 0$, and the mapping c is a permutation. Then there exists a unitary O_c such that

$$O_c |j\rangle = |c(j)\rangle$$

The implementation of O_c may require the usage of some work registers that are omitted here. We also have

$$O_c^\dagger |c(j)\rangle = |j\rangle$$

We assume the matrix entry $A_{c(j),j}$ can be queried via

$$O_A |0\rangle |j\rangle = \left(A_{c(j),j} |0\rangle + \sqrt{1 - |A_{c(j),j}|^2} |1\rangle \right) |j\rangle$$

Now we construct $U_A = (I \otimes O_c)O_A$, (tensor product of two unitary matrices is unitary) and compute

$$\langle i| \langle 0| U_A |0\rangle |j\rangle = \langle 0| \langle i| \left(A_{c(j),j} |0\rangle + \sqrt{1 - |A_{c(j),j}|^2} |1\rangle \right) |c(j)\rangle = A_{c(j),j}\delta_{i,c(j)}$$

This proves that $U_A \in BE_{1,1}(A)$.

For a more general s-sparse matrix, WLOG we assume each row and column ha exactly s nonzero entries (otherwise we can always treat some zero entries as

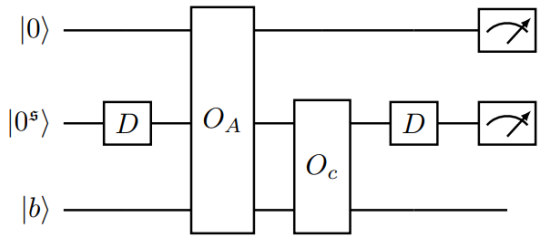


Figure 8.4: Quantum circuit for block encoding an s-sparse matrix

nonzeros). For each column j, the row index for the lth nonzero entry is denoted by $c(j,l) = c_{j,l}$. For simplicity, we assume hat there exists a unitary O_c such that

$$O_c |l\rangle |j\rangle = |l\rangle |c(j,l)\rangle$$

Here we assume $s = 2^p$ and the first register is an p-qubit register. A necessary condition for this model is that O_C is reversible, i.e., we can have $O_c^\dagger |l\rangle |c(j,l)\rangle = |l\rangle |j\rangle$. This means that for each row index $i = c(j,l)$, we can recover the column index j given the value of l. This can be satisfied e.g.

$$c(j,l) = j + l - l_0 \quad (mod N)$$

where l_0 is a fixed number. This corresponds to a banded matrix. This assumption is ofcourse somewhat restrictive. We shall discuss more general query models.

Corresponding to the equation, the matrix entries can be queried via

$$O_A |0\rangle |l\rangle |j\rangle = \left(A_{c(j,l),j} |0\rangle + \sqrt{1 - |A_{c(j,l),j}|^2} |1\rangle \right) |l\rangle |j\rangle$$

In order to construct a unitary that encodes all row indices at the same time ,we define $D = H^{\otimes p}$ (sometimes called a diffusion operator, which is a term originated from Grover's search) satisfying

$$D |0^p\rangle = \frac{1}{\sqrt{p}} \sum_{l \in [p]} |l\rangle$$

Consider U_A given by the circuit in figure 8.4. The measurement means that to obtain $A |b\rangle$, the ancilla register should return the value 0.

Proposition 8.3.1. *The circuit given in figure 8.4 defines $U_A \in BE_{s,s+1}(A)$.*

Proof. We call $|0\rangle|0^p\rangle|j\rangle$ the source state, and $|0\rangle|0^p\rangle|i\rangle$ the target state. In order to compute the inner product $\langle 0|\langle 0^s|\langle i|U_A|0\rangle|0^p\rangle|i\rangle$, we apply D, O_A, O_C to the source state accordingly as

$$\begin{aligned}
|0\rangle|0^p\rangle|j\rangle &\xrightarrow{D} \frac{1}{\sqrt{s}}\sum_{l\in[s]}|0\rangle|l\rangle|j\rangle \\
&\xrightarrow{O_A} \frac{1}{\sqrt{s}}\sum_{l\in[s]}\left(A_{c(j,l),j}|0\rangle + \sqrt{1-|A_{c(j,l),j}|^2}|1\rangle\right)|l\rangle|j\rangle \\
&\xrightarrow{O_C} \frac{1}{\sqrt{s}\sum_{l\in[s]}}\left(A_{c(j,l),j}|0\rangle + \sqrt{1-|A_{c(j,l),j}|^2}|1\rangle\right)|l\rangle|c(j,l)\rangle
\end{aligned}$$

Since we are only interested in the final state when all ancilla qubits are the 0 state, we may apply D to the target state $|0\rangle|0^p\rangle|i\rangle$ as (note that D is Hermitian)

$$|0\rangle|0^p\rangle|i\rangle \xrightarrow{D} \frac{1}{\sqrt{s}}\sum_{l'\in[s]}|0\rangle|l'\rangle|i\rangle$$

Hence the inner product

$$\langle 0|\langle 0^p|\langle i|U_A|0\rangle|0^p\rangle|j\rangle = \frac{1}{s}\sum_l A_{c(j,l),j}\delta_{i,c(j,l)} = \frac{1}{s}A_{ij}$$

□

8.4 Hermitian Block Encoding

So far we have considered general s-sparse matrices. Note that if A is Hermitian matrix, its (α, m, ϵ)-block-encoding U_A does not need to be Hermitian. Even if $\epsilon = 0$, we only have that the upper-left n-qubit block of U_A is Hermitian. For instance, even the block encoding of a Hermitian diagonal matrix may not be Hermitian. On the other hand,there are cases when $U_A^\dagger = U_A$ is indeed a Hermitian matrix, and hence the definition:

Definition 8.4.1. (Hermitian Block Encoding). Let U_A be an (α, m, ϵ)-block-encoding of A. If U_A is also Hermitian, then it is called an (α, m, ϵ)-Hermitian-block-encoding of A. When $\epsilon = 0$, it is called an (α, m)-Hermitian-block-encoding. The set of all (α, m, ϵ)-Hermitian-block-encoding of A is denoted by $HBE_{\alpha,m}(A, \epsilon)$, and we define $HBE_{\alpha,m}(A) = HBE(A, 0)$.

The Hermitian block encoding provides the simplest scenario of the qubitization process.

8.5 Query models for general sparse matrices

If we query the oracle, the assumption that for each l the value of $c(j,l)$ is unique for all j seems unnatural for constructing general sparse matrices. So we consider an alternative method for construct the block encoding of a general sparse matrix as below.

Again without loss of generality we assume that each row/column has at most $p = 2^s$ non zero entires, and that we have access to the following two (2n)-qubit oracles.

$$O_r |l\rangle |i\rangle = |r(i,l)\rangle |i\rangle$$
$$O_c |l\rangle |j\rangle = |c(j,l)\rangle |j\rangle$$

Here $r(i,l)$,$c(j,l)$ gives the l-th nonzero entry in the i-th row and j-th column, respectively. It should be noted that although the index $l \in [p]$, we should expand it into an n-qubit state (e.g. let l take the last s qubits of the n-qubit register following the binary representation of integers). The reason for such an expansion, and that we need two oraclesO_c, O_r will be seen shortly.

Similar to the discussion before, we need a diffusion operator satisfying

$$D |0^n\rangle = \frac{1}{\sqrt{s}} \sum_{l \in [p]} |l\rangle$$

This can be implemented using Hadamard gates as

$$D = I_{n-s} \otimes H^{\otimes s}$$

We assume that the matrix entries are queried using the following oracle using controlled rotations

$$O_A |i\rangle |j\rangle = \left(A_{ij} |0\rangle + \sqrt{1 - |A_{ij}|^2} |1\rangle \right) |i\rangle |j\rangle$$

where the rotation is controlled by both row and column indices. However, if $A_{ij} = 0$ for some i, j, the rotation can be arbitrary, as there will be no contribution due to the usage of O_r, O_c.

Proposition 8.5.1. *The circuit shown in figure 8.5 defines $U_A \in BE_{s,n+1}(A)$.*

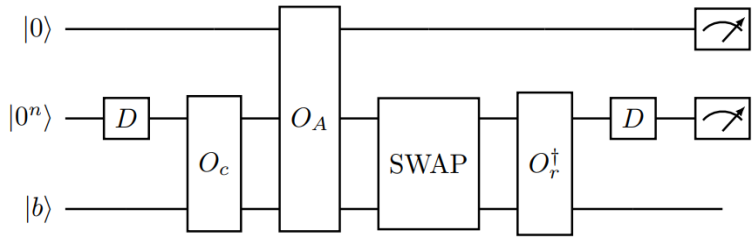


Figure 8.5: Quantum circuit for block encoding of general sparse matrices

Proof. We apply the first four gate sets to the source state

$$|0\rangle|0^n\rangle|j\rangle \xrightarrow{D}\xrightarrow{O_c}\xrightarrow{O_A} \frac{1}{\sqrt{p}}\sum_{l\in[p]}\left(A_{c(j,l),j}|0\rangle + \sqrt{1-|A_{c(j,l),j}|^2}|1\rangle\right)|c(j,l)\rangle|j\rangle$$
$$\xrightarrow{SWAP} \frac{1}{\sqrt{p}}\sum_{l\in[p]}\left(A_{c(j,l),j}|0\rangle + \sqrt{1-|A_{c(j,l),j}|^2}|1\rangle\right)|j\rangle|c(j,l)\rangle$$

we then apply D and O_r to the target state

$$|0\rangle|0^n\rangle|i\rangle \xrightarrow{D}\xrightarrow{O_r} \frac{1}{\sqrt{p}}\sum_{l'\in[p]}|0\rangle|r(i,l')\rangle|i\rangle$$

Then the inner product gives

$$\begin{aligned}\langle 0|\langle 0^n|\langle i|U_A|0\rangle|0^n\rangle|j\rangle &= \frac{1}{p}\sum_{l,l'}A_{c(j,l),j}\delta_{i,c(j,l)}\delta_r(j,l'),j\\ &= \frac{1}{p}\sum_{l}A_{c(j,l),j}\delta_{i,c(j,l)} = \frac{1}{s}A_{ij}\end{aligned}$$

Here we have used that there exists a unique l such that $i = c(j,l)$, and a unique l' such that $j = r(i,l')$. □

We remark that the quantum circuit in figure 8.5 is essentially the construct, which gives a $(s, n+3)$-block-encoding. The construct above slightly simplifies the procedure and saves two extra qubits (used to marke whether $l \geq s$).

Next we consider the Hermitian block encoding of a s-sparse Hermitian matrix. Since A is Hermitian, we only need one oracle to query the location of th nonzero entries.

$$O_c|l\rangle|j\rangle = |c(j,l)\rangle|j\rangle$$

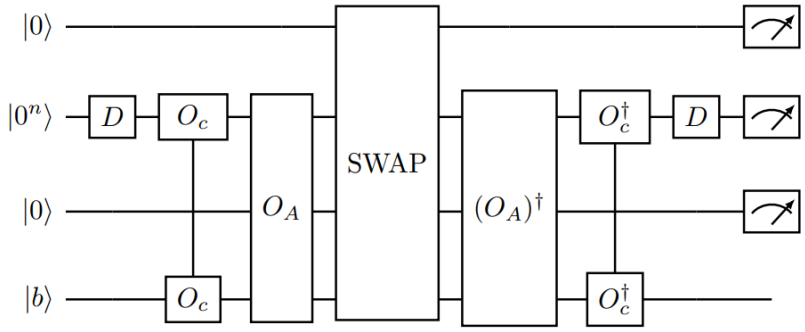


Figure 8.6: Quantum circuit for Hermitian Block encoding of a general Hermitian matrix

Here $c(j,l)$ gives the l-th nonzero entry in the j-th column. It can also be interpreted as the l-th nonzero entry in the ith column. Again the first register needs to be interpreted as an n-qubit register. The diffusion operator is the same as in the equation.

Unlike all discussions before, we introduce two signal qubits and a quantum state in the computational basis take the form $|a\rangle|i\rangle|b\rangle|j\rangle$, where $a,b \in \{0,1\, i,j \in [N]$. In other words, we may view $|a\rangle|i\rangle$ as the first register, and $|b\rangle|j\rangle$ as the second register. The $(n+1)$-qubit SWAP gate is defined as

$$SWAP\,|a\rangle|i\rangle|b\rangle|j\rangle = |b\rangle|j\rangle|a\rangle|i\rangle$$

To query matrix entries, we need access to the square root of A_{ij} as (note that act on the second single-qubit register)

$$O_A\,|i\rangle|0\rangle|j\rangle = |i\rangle\left(A_{ij}\,|0\rangle + \sqrt{1-|A_{ij}|}\,|1\rangle\right)|j\rangle$$

The square root operation is well defined if $A_{ij} \geq 0$ for all entries. If A has negative (or complex) entries, we first write $A_{ij} = |A_{ij}e^{\iota\theta_{ij}}, \theta_{ij} \in [0,2\pi)$, and the square root is uniquely defined as $\sqrt{A_{ij}} = \sqrt{|A_{ij}|}e^{\iota\theta_{ij}/2}$.

Proposition 8.5.2. *Figure 8.6 defines* $U_A \in HBE_{s,n+2}(A)$.

Proof. Apply the first four gate sets to the source state gives

$$|0\rangle|0^n\rangle|0\rangle|j\rangle \xrightarrow{D}\xrightarrow{O_c}\xrightarrow{O_A} \frac{1}{\sqrt{s}}\sum_{l\in[s]}|0\rangle|c(j,l)\rangle\left(\sqrt{A_{c(j,l),j}}|0\rangle\, sqrt1-|A_{c(j,l),j}|1\rangle\right)|j\rangle$$
$$\xrightarrow{SWAP} \frac{1}{\sqrt{s}}\sum_{l\in[s]}\left(\sqrt{A_{c(i,l'),i}}|0\rangle+\sqrt{1-|A_{c(i,l'),i}|}|1\rangle\right)|i\rangle$$

Apply the last three gate sets to the target state

$$|0\rangle|0^n\rangle|0\rangle|i\rangle \xrightarrow{D}\xrightarrow{O_c}\xrightarrow{O_A} \frac{1}{\sqrt{s}}\sum_{l\in[s]}|0\rangle|c(i,l')\rangle\left(\sqrt{A_{c(i,l'),i}}|0\rangle+\sqrt{1-|A_{c(i,l'}|}|1\rangle\right)|i\rangle$$

Finally, take the inner product as

$$\begin{aligned}\langle 0|\langle 0^n|\langle 0|\langle i|U_A|0\rangle|0^n\rangle|0\rangle|j\rangle & \\ &= \frac{1}{s}\sum_{l,l'}\sqrt{A_{c(j,l'),j}}\sqrt{A^*_{c(i,l'),i}}\delta_{i,c(j,l)}\delta_{c(i,l'),j} \\ &= \frac{1}{s}\sqrt{A_{ij}A^*_{ji}}\sum_{l,l'}\delta_{i,c(j,l)}\delta_{c(i,l'),j}=\frac{1}{s}A_{ij}\end{aligned}$$

In this equality, we have used that A is Hermitian: $A_{ij}=A^*_{ji}$, and there exists a unique l such that $i=c(j,l)$, as well as a unique l' such that $j=c(i,l')$.

The quantum circuit in figure 8.6 is essentially construction in. The relation will quantum walks will be discussed further.

□

Chapter 9

Advanced Quantum Computing

9.1 Linear Growth and Duhamel's Principle

Consider a Quantum Algorithm deonted by a Unitary U (since every transformation in Quantum Computing is Unitary, thus a quantum algorithm can be thought of as a black box which performs a Untiary on some qubits and outputs those qubits (note that the number of qubits in input = number of qubits in the output)). Now, we know that the product of Unitary Matrices is a Unitary matrix. Thus we can decompose the Unitary U into a product of Unitary matrices $U = U_1 U_2 \dots U_k$. **Assumption: Suppose we can implement each U_i to precision ϵ then the following *Hybrid Argument***

Proposition 9.1.1. *Hybrid Argument: Given Unitaries $U_1, \tilde{U_1}, \dots, U_K, \tilde{U_K} \in \mathbb{C}^{N \times N}$ satisfying,*

$$\|U_i - \tilde{U_i}\|_2 \leq \epsilon \quad \forall i = 1, \dots, K$$

we have,

$$\|U_K \dots U_2 U_1 - \tilde{U_K} \dots \tilde{U_2} \tilde{U_1}\| \leq K\epsilon$$

Proof. Using a telescoping series, we can write:

$$\begin{aligned} U_K \dots U_2 U_1 - \tilde{U_K} \dots \tilde{U_1} =& U_K \dots U_2 U_1 - U_K \dots U_2 \tilde{U_1} + U_K \dots U_2 \tilde{U_1} \\ & - U_K \dots \tilde{U_2} \tilde{U_1} + \dots + U_K \tilde{U_{K-1}} \dots \tilde{U_1} - \tilde{U_K} \dots \tilde{U_1} \\ & = U_K \dots U_2 (U_1 - \tilde{U_1}) + U_K \dots U_3 (U_2 - \tilde{U_2}) \tilde{U_1} + \dots \\ & + U_K (U_{K-1} - \tilde{U_{K-1}}) \dots \tilde{U_1} + (U_K - \tilde{U_K}) \dots \tilde{U_1} \end{aligned}$$

Note that here we define the norm of a matrix as operator (spectral) norm $\|A\| = \max_{\|x\|_2=1} \|Ax\|_2$. Now, using the property that, $\|A+B\| \leq \|A\| + \|B\|$ for matrices, we get:

$$\begin{aligned}\|U_K \dots U_2 U_1 - \tilde{U_K} \dots \tilde{U_2}\tilde{U_1}\| &\leq \|U_K \dots U_2(U_1 - \tilde{U_1})\| \\ &+ \|U_K \dots U_3(U_2 - \tilde{U_2})\tilde{U_1}\| + \dots + \|U_K(U_{K-1} - \tilde{U_{K-1}}) \dots \tilde{U_1}\| \\ &+ \|(U_K - \tilde{U_K}) \dots \tilde{U_1}\|\end{aligned}$$

Now using the property that, $\|AB\| \leq \|A\|\|B\|$ for matrices and using the property that the $\|A\| = 1$ for a Unitary matrix since Unitary matrix preserves norm of the vector and only rotates it, Thus, we get:

$$\begin{aligned}\|U_K \dots U_2 U_1 - \tilde{U_K} \dots \tilde{U_2}\tilde{U_1}\| &\leq \|U_K \dots U_2\|\|U_1 - \tilde{U_1}\| \\ &+ \|U_K \dots U_3\|\|U_2 - \tilde{U_2}\|\|\tilde{U_1}\| + \dots + \|U_K\|\|U_{K-1} - \tilde{U_{K-1}}\| \dots \|\tilde{U_1}\| \\ &+ \|U_K - \tilde{U_K}\| \dots \|\tilde{U_1}\| \\ &\leq \|U_1 - \tilde{U_1}\| + \|U_2 - \tilde{U_2}\| \dots \|U_K - \tilde{U_K}\| \\ &\leq \sum_{i=1}^{K} \|U_i - \tilde{U_i}\| \\ &\leq K\epsilon\end{aligned}$$

Thus, we say that if we can implement each local unitary to precision ϵ, the global error grows at most linearly with respect to the number of gates and is bounded by $K\epsilon$. *The above proof can be seen as a discrete analogue of the variation of constants method (also called Duhamel's principle).* □

Proposition 9.1.2. *(Duhamel's principle for Hamiltonian Simulation) Let $U(t), \tilde{U}(t) \in \mathbb{C}^{N\times N}$ satisfy*

$$\iota\partial_t U(t) = HU(t), \quad \iota\partial_t \tilde{U}(t) = H\tilde{U}(t) + B(t), \quad U(0) = \tilde{U}(0) = I$$

where $H \in \mathbb{C}^{N\times N}$ is a Hermitian matrix and $B(t) \in \mathbb{C}^{N\times N}$ is an arbitrary matrix. Then

$$\tilde{U}(t) = U(t) - \iota \int_0^t U(t-s)B(s)ds$$

and

$$\|\tilde{U}(t) - U(t)\| \leq \int_0^t \|B(s)\|ds$$

Proof. Substituting the equation,

$$\tilde{U}(t) = U(t) - \iota \int_0^t U(t-s)B(s)ds$$

into the equation,

$$\iota \partial_t \tilde{U}(t) = H\tilde{U}(t) + B(t)$$

we see that it satisfies. NOe consider a special case where $B(t) = E(t)\tilde{U}(t)$, then

$$\tilde{U}(t) = U(t) - \iota \int_0^t U(t-s)E(s)\tilde{U}(s)ds$$

Now, taking the norm of the above equation we get:

$$\|\tilde{U}(t) - U(t)\| \leq \int_0^t \|E(s)\|ds$$

This is a direct analogue o the hybrid argument in the continuous setting. □

9.2 Universal gate sets and Reversible Computation

Universal gate sets are the set of gates which can be used to represent any gate as a combination of the gates from the set. In case of Classical Computation, the NAND gate is a universal gate set. The "NOR" gate is also a universal gate set.

In case of Quantum Computations, any Unitary operator on n qubits can be implemented using 1- and 2- qubit gates. A set of universal quantum gates S is universal if given any unitary operator U and desired precision ϵ we can find $U_1, \cdots, U_m \in S$ such that

$$\|U - U_m \dots U_1\| \leq \epsilon$$

Some of the possible choices of the universal gate sets are {H,T,CNOT}, {H,Toffoli}

Theorem 9.2.1. *(Solovay-Kitaev Theorem) Let $\mathcal{S}, \mathcal{T}$ be two universal gate sets that are closed under inverses. Then any m-gate circuit using the gate set S can be implemented to precision ϵ using a circuit of $\mathcal{O}(m \cdot polylog(m/\epsilon))$ gates from the gate set T, and there is a classical alogirhtm for finding thsi circuit in time $\mathcal{O}(m \cdot polylog(m\epsilon))$. (Here polylog is the polynomial of the logarithm of the argument).* ***All choices of Universal gates are equivalent***

Any classical circuit can also be asymptotically efficiently implemented using a quantum circuit. (Classical gates can be efficiently simulated using quantum circuits.)

Step1: Note that NAND and other classical gates (such as AND, OR, etc..) are not reversible gates! Thus, we need to convert the classical gates to reversible gates. Any irreversible classical gate $x \to f(x)$ can be converted to a reversible gate by adding an extra bit called the "garbage bit" and then applying the reversible gate. The reversible gate is then defined as $x, y \to x, f(x) \oplus y$. Thus, in particular we have $(x, 0) \to (x, f(x))$. the key idea here is to store all the intermediate steps of the computation. On the quantum computer, storing all the intermediate computational steps is problematic:

- Waste of Quantum Resources
- Interediate bits stored in some extra qubits are still entangled to the quantum state of interest. (If environment interferes with the intermeditate result, then the quantum state of interest also gets affected.)

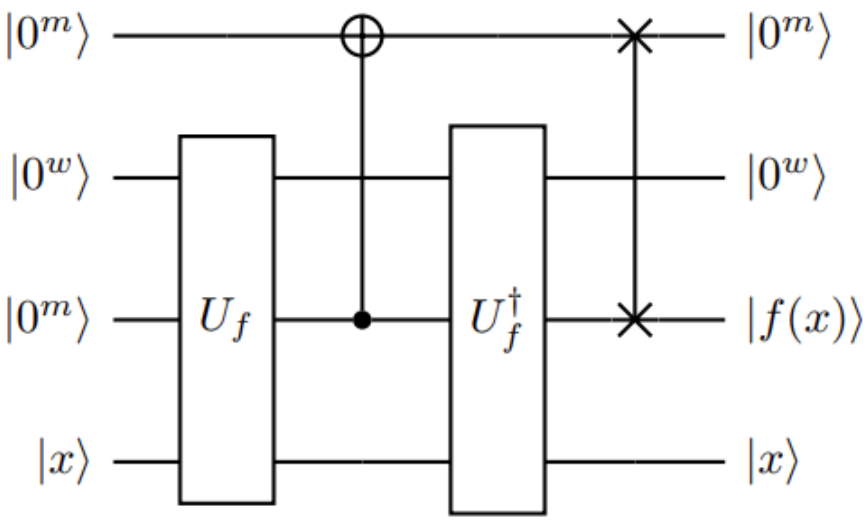


Figure 9.1: Uncomputation of a classical gate

Step 2: **Uncomputation:** To implement a boolean function $f : \{0,1\}^n \to \{0,1\}^m$ assume that there is an oracle.

$$|0^m\rangle |x\rangle \to |f(x)\rangle |x\rangle$$

where $|0^m\rangle$ comes from a m-qubit output register. The oracle is often further implemented with the help of a working register (aka garbage register - they are ancilla registers which can be freed after uncomputation) such that

$$U_f : |0^w\rangle |0^m\rangle |x\rangle \rightarrow |g(x)\rangle |f(x)\rangle |x\rangle$$

From no deletion theorem, there is no generic unitary operator that can set a black box-state to $|0^w\rangle$. In order to set the working register back to $|0^w\rangle$ while keeping the input and output state, we introduce another set of m-qubit ancillar registers initialised as $|0^m\rangle$. Then we use n-qubit CNOT controlled on the output register and obtain

$$|0^m\rangle |g(x)\rangle |f(x)\rangle |x\rangle \rightarrow |f(x)\rangle |g(s)\rangle |f(x)\rangle |x\rangle$$

Note that the multi-qubit CNOT gate only performs the classical copying operation in the computational basis and thus does not violate the *No- Cloning Theorem*. Now we know, $U_f^\dagger = U_f^{-1}$,

$$(I_m \otimes U_f^\dagger) |f(x)\rangle |g(x)\rangle |f(x)\rangle |x\rangle = |f(x)\rangle |g(x)\rangle |f(x)\rangle |x\rangle = |f(x)\rangle |0^w\rangle |0^m\rangle |x\rangle$$

The, we apply a SWAP operation on the ancilla and output registers to obtain

$$|f(x)\rangle |0^w\rangle |0^m\rangle |x\rangle \rightarrow |0^m\rangle |0^w\rangle |f(x)\rangle |g(x)\rangle$$

Thus, not the ancilla and the working register are set to the initial state. They are no longer entangled to the input or output register and can be reused for other purposes. This is called **Uncomputation**.

(Discarding Working Registers). Thus after the uncomputation step the first two registers as shown are uncahnged before and after the application of the circuit, though they are changed during the intermediate steps. Therefore 9.1 effectively implements a unitary operator

$$(I_{m+w} \otimes V_f) |0^m\rangle |0^w\rangle |0^m\rangle |x\rangle = |0^m\rangle |0^w\rangle |f(x)\rangle |x\rangle$$

or

$$V_f |0^m\rangle |x\rangle = |f(x)\rangle |x\rangle$$

NOte hter the definition of V_f all the working registers have been discarded allowing us to simplify notation and focus on the essence of Quantum Algorithm.

Thus, using the technique of uncomputation, if the map $x \rightarrow f(x)$ can be efficiently implemented on a classical computer, then we can implement the map efficiently on a quantum computer as well. All reversible single-bit and two-bit classical gates can be implemented using single-bit and

two-bit quantum gates. Thus, the reversible map can be made into a unitary opertaion as

$$U_f : |x, y\rangle \to |x, y \oplus f(x)\rangle$$

This proves that quantum computers are at least as powerful as classical computers. U_f can be applied to any superposition of the states in the computational basis. Thus, the quantum computer can perform the computation on all possible inputs simultaneously.

$$U_f : \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x, 0\rangle \to \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x, f(x)\rangle$$

Note: This does not mean that we can efficiently implement the map $|x\rangle \to |f(x)\rangle$. If f is a bijection, and we have access tot he inverse of the reversible circuit for computing f^1 then we may use the technique of uncomputation to implement such a map.

Example 9.2.1. Suppose there are three bits and we wish to apply AND gate twice and get the result in the classical setting as:

9.3 Fixed Point Number Representation and Classical Arithmetic operations

Any integer $k \in [N] = \{0, 1, \cdots, N-1\}$ where $N = 2^n$ can be expressed as n n-bit string as $k = k_{n-1} \cdots, k_0$ with $k_i \in \{0, 1\}$. This is called binary representation of integer k.

$$k = \sum_{i\in[n]} k_i 2^i$$

The number k divided by 2^m $(0 \le m \le n)$ (note that the decimal is shifted to be after k_m).

$$a = \frac{k}{2^m} = \sum_{i\in[n]} k_i 2^{i-m} = (k_{n-1} \cdots k_m.k_{m-1} \cdots k_0)$$

when m=n,

$$a = \frac{k}{2^n} = \sum_{i\in[n]} = \sum_{i\in[n]} k_i 2^{i-n} = 0.k_{n-1} \cdots k_0$$

$a = 0.k_{n-1} \cdots k_0$ so that k_i is the ith decimal of a in the binary representation. Note that here $0 \le a \le 1$. The number $(0.k_{n-1} \cdots k_i)$ is called n-bit fixed point

representation of a. Therefore to represent a the additive precision ϵ, $n = \lceil \log_2(\epsilon) \rceil$ qubits are required. **We may add one extra bit to indicate its sign**.

Thus, we can perform classical arithmetic operations, such as $(x, y) \rightarrow x + y, (x, y) \rightarrow xy, x \rightarrow x^{\alpha}, x \rightarrow \cos\alpha$ etc. using reversible quantum circuits. **The number of ancilla qubits, and the number of elementary gates needed for implementing such circuits is $O(poly(n))$.** Thus, quantum computer is theoretically as powerfule as classical computers, there is a very significant overhead in implementing reversblie classcial circuits on quantum devices, both in terms of the number of ancilla qubits and the circuit depth.

9.4 Fault tolerant Computation

We assumed that quantum operations can be perfectly performed and all errors come from either approximation errors at the mathematical level, or Monte Carlo errors in the readout process due to its probabilistic nature of the measurement process. Note that because of technical difficult quantum gates, and measurements involve significant erros on near-term quantum devices. **Threshold theorem: If the noise in individual quantum gates is below a certian constant threshold (around 10^{-4} or above) it is possible to perform an arbitrarily large computaion with any desired precision. This requries quantum error correction protocols.**

9.5 Complexity of Quantum Algorithms

Let n be the number of imput bits. **Efficiency of Quantum Algorithm:** A Quantum Algorithm is efficient if the number of gates in the quantum circuit is $\mathcal{O}(poly(n))$. Because of Probabilistic nature of measurements it is required for probability of correct answer to be higher. Generally, $p > 2/3$ is considered to be a good probability of success or at least $p > 1/2 + 1/poly(n)$. Consdier a case when Probability of getting a correct answer is $p = 1/2 + \epsilon$ where $\epsilon > 0$ is a constant. Then the best method of deciding the correct answer is based on the frequency of the outcomes, the higher frequency outcome is the correct answer. But how many repititions should be performed? Then the number of repetitions of the algorithm can be found using Chernoff bound as (say for binary outcome $X_i = 0$ or $X_i = 1$):

$$P(\sum X_i \leq n/2) \leq e^{-2n\epsilon^2}$$

where n is the number of times. Thus the error gets exponentially small. In Quantum Algorithms, Computational cost = Query Complexity. **Goal: Perform a given task using as few queries as possible to oracle (U_f).** A Quantum Oracle (black box) is a Unitary operator that implements the function f.

Example 9.5.1. (Query access to boolean function) $f : \{0,1\}^n \to \{0,1\}$ be a boolean function and oracle be defined as:

$$U_f |x\rangle = (-1)^{f(x)} |x\rangle$$

where $x \in \{0,1\}^n$. This is used in Grover's Algorithm. This is called phase kickback as $f(x)$ is returned as a phase factor. Thus, U_f can be applied to superposition of the states in computational basis. Having query access to $f(x)$ does not mean we know $f(x)$.

Example 9.5.2. (Partially specified oracles) Sometimes it is possible we might not be interested in entire U_f but only U_f applied to certain vectors say for example, as shown

$$U_f |0\rangle |x\rangle = |0\rangle A |x\rangle + |1\rangle B |x\rangle$$

$$U_f = \begin{bmatrix} A & * \\ B & * \end{bmatrix}$$

where A, B are arbitrary $N \times N$ unitary matrices. This is called as partially specified oracle.

Query complexity hides the implementation details of U_f. For some cases, we can prove lower bounds on number of queries to solve a certain problems (example: Grover's Algorithm) but it's a difficult problem. Once we know number of elementary gates need to implement U_f we know gate complexity of the total algorithm. However, some queries can be (provably) difficult to implement, and there can be a large gap between the query complexity and gate complexity. Quantum Algorithm should not dominate the total gate complexity.

Circuit depth is the maximum number of gates on any path from input to output. It is equivalent of "Wall clock time" in classical computation. Quantum states can be preserved only for a short amount of time = Coherence time. Thus circuit depth can be used to check whether the coherence time is exceeded by a Quantum Computer. **Quantum Algorithm should be depth efficient.** Thus, try to reduce the circuit time even if we need to run it multiple times.

A Quantum Algorithm consist of set of qubits (system regiters - storing quantum states of interest and ancilla registers - auxillary registers needed to implement the

unitary operations acting on system registers) starting from an inital state , apply a series of one/two qubit states, and perform measurements, perform uncomputation whenever possible. Within ancilla register if a register can be freed by uncomputation then it is called a working register. Working register can be reused for other purposes. **We do not factor the cost of working registers into asymptoptic cost analysis in the literature.**

Appendix A

Linear Algebra and Calculus Prerequisites

The following content assumes a high school level understanding of Linear Algebra, Complex numbers, Probability, single-variable Calculus (Multi-Variable Calculus will be a plus) and Differential Equations. It contain all the pre-requisities required for understanding the contents of the given Book. The following references [4], [1], [5] and [3] have been extensively used for the preparation for this appendix. Note that this content assumes that the reader has only high school level exposure of the Quantum Mechanics (De broglie Hypothesis, Schrodinger Equation, Young's Double Slit Experiment, Heisenberg's Uncertaininty Principle) but still the important concepts of Quantum Mechanics are explained upto the required depth whenever it is found to be necessary for the explaination of Quantum computing concepts. A college level exposure of Quantum Mechanics will be a plus.

For a quick revision of the concepts of Linear Algebra the reader is suggested to watch youtube lectures by 3 Blue 1 Brown. If it still feels difficult to follow the material, lectures by MIT Prof. Gilbert Strang Lectures on Linear Algebra are highly recommended. For basic Quantum Mechanics concepts, the reader is suggested to watch the lectures by MIT Prof. Allan Adams and for the mathematics of Quantum Mechanics, the lectures by MIT Prof. Barton Zwiebach are highly recommended. For the mathematics of Quantum Computing the short youtube lectures by Quantum Sense and Advanced Maths will be helpful. Hope that this notes help in understanding the basics of Quantum Computing. For the coding part, you can also follow lectures by Qiskit and John Watrous. Many more such youtube lectures are available, lectures by John preskill where he speaks in a monotonous voice along with his notes on Quantum Computing. Short youtube lectures covering the maths

by David Deutsch and Michael Nielsen. Some good high level view introductory lectures by Scott Aaronson and some amazing videos by Seth Lloyd on the importance of Quantum Computers. Lectures by CERN and many more a lecture series on Advanced Quantum Computing and on Quantum Simulations by APCTP.

A.1 Linear Algebra

Definition A.1.1. Linear Transformation: A function $T : V \to W$ is called a linear transformation if for all $|v\rangle, |w\rangle \in V$ and $a \in \mathbb{F}$ the following properties hold:

1. $T(|v\rangle + |w\rangle) = T(|v\rangle) + T(|w\rangle)$
2. $T(a\,|v\rangle) = aT(|v\rangle)$

Example A.1.1. Let V be a vector space over a field $\mathbb{F}$ and let A be a matrix in $\mathbb{F}^{m\times n}$. The function $T : \mathbb{F}^n \to \mathbb{F}^m$ defined by $T(|v\rangle) = A\,|v\rangle$ is a linear transformation.

Example A.1.2. Differentation, Integration, Laplace transform are examples of linear transformations.

Definition A.1.2. Composition Transformation: Let $T : V \to W$ and $U : W \to X$ be linear transformations. The composition transformation $UT : V \to X$ is defined by $(UT)(|v\rangle) = U(T(|v\rangle))$ If the composition transformation is also a linear transformation then it is called a composite linear transformation. Here, if T and U are linear transformations then UT is also a linear transformation.

Definition A.1.3. Inverse Transformation: Let $T : V \to W$ be a linear transformation. A linear transformation $T^{-1} : W \to V$ is called the inverse transformation of T if $TT^{-1} = T^{-1}T = I$ where I is the identity transformation.

A.1.1 Basics of Linear Algebra

Definition A.1.4. Field ($\mathbb{F}$): A field is a set of elements that is closed under two operations, addition and multiplication, and satisfies the following axioms: Let $a, b, c \in \mathbb{F}$

1. Commutative Property of Addition: $a + b = b + a$
2. Associative Property of Addition: $(a + b) + c = a + (b + c)$
3. Distributive Property of Addition: $a(b + c) = ab + ac$

4. Additive Identity: There exists an element 0 in the field such that $a + 0 = a$
5. Additive Inverse: For every element a in the field, there exists an element $-a$ such that $a + (-a) = 0$
6. Commutative Property of Multiplication: $ab = ba$
7. Associative Property of Multiplication: $(ab)c = a(bc)$
8. Distributive Property of Multiplication: $(a + b)c = ac + bc$
9. Multiplicative Identity: There exists an element 1 in the field such that $a1 = a$
10. Multiplicative Inverse: For every element a in the field, there exists an element a^{-1} such that $aa^{-1} = 1$

Note that it is upto us how we define the addition and multiplication operations on the field $\mathbb{F}$ but for the purpose of this notes we will assume that the operations are the usual addition and multiplication operations unless stated otherwise. A more formal definition of Field can be defined using group theory but the above definition is sufficient for the purpose of this notes. The complications and more precise definition is left to Mathematicians which can be found in any standard textbook on Abstract Algebra.

Example A.1.3. Set of Rational Numbers ($\mathbb{Q}$), Set of Real Numbers ($\mathbb{R}$), Set of Complex Numbers ($\mathbb{C}$) are examples of fields.

Example A.1.4. Set of Integers ($\mathbb{Z}$) is not a field as it does not satisfy the multiplicative inverse axiom. Set of Natural Numbers and Whole numbers is not a filed because it does not satisfy the additive inverse axiom.

Generally, we have represented vectors as: $\vec{v}$, or with bold letter $\mathbf{v}$. but for the purpose of Quantum Mechancis and Quantum Computing we prefer dirac notation which represents vectors as $|v\rangle$ solely for the purpose of simplifying the notation. The dirac notation is also called as bra-ket notation. The symbol $|v\rangle$ represents a column vector v called as ket-v and the symbol $\langle v|$ represents a row vector called as bra-v.

Definition A.1.5. Vector Space: A vector space over a field $\mathbb{F}$ is a set V of elements called vectors, that is closed under two operations, addition and scalar multiplication:

1. Vector Addition: For every pair of vectors $|u\rangle, |v\rangle \in V$, there exists a vector $|u\rangle + |v\rangle \in V$ called the sum of $|u\rangle$ and $|v\rangle$

2. Scalar Multiplication: For every vector $|u\rangle \in V$ and every scalar $a \in \mathbb{F}$, there exists a vector $a\,|u\rangle \in V$ called the scalar multiple of $|u\rangle$ by a

and satisfies the following axioms: Let $|u\rangle, |v\rangle, |w\rangle \in V$ and $a, b \in \mathbb{F}$

1. Closure Property over Vector Addition: $|u\rangle + |v\rangle \in V$
2. Commutative Property of Addition: $|u\rangle + |v\rangle = |v\rangle + |u\rangle$
3. Associative Property of Addition: $(|u\rangle + |v\rangle) + |w\rangle = |u\rangle + (|v\rangle + |w\rangle)$
4. Additive Identity: There exists a vector $|0\rangle$ in V such that $|u\rangle + |0\rangle = |u\rangle$
5. Additive Inverse: For every vector $|u\rangle$ in V, there exists a vector $-|u\rangle$ in V such that $|u\rangle + (-|u\rangle) = |0\rangle$
6. Closure Property over Scalar Multiplication: $a\,|u\rangle \in V$
7. Distributive Property of Scalar Multiplication over Vector Addition: $a(|u\rangle + |v\rangle) = a\,|u\rangle + a\,|v\rangle$
8. Distributive Property of Scalar Multiplication over Field Addition: $(a+b)\,|u\rangle = a\,|u\rangle + b\,|u\rangle$
9. Associative Property of Scalar Multiplication: $a(b\,|u\rangle) = (ab)\,|u\rangle$
10. Multiplicative Identity: There exists a scalar 1 in $\mathbb{F}$ such that $1\,|u\rangle = |u\rangle$

The above axioms imply

1. $|0\rangle$ is unique i.e. if $|0'\rangle$ has all the properties of $|0\rangle$ then $|0'\rangle = |0\rangle$
2. $|-u\rangle = -|u\rangle$
3. $|-u\rangle$ is the unique additive inverse of $|u\rangle$

Example A.1.5. The set of Real numbers $\mathbb{R}$ is a vector space over the field of Real numbers $\mathbb{R}$. It is called Real Vector Space. The set of Complex numbers $\mathbb{C}$ is a vector space over the field of Complex numbers $\mathbb{C}$. It is called Complex Vector Space.

Note the adjective Real and Complex before a vector space is used to denote the field over which the vector space is defined. In this notes we will be mostly dealing with the Complex Vector Space since the Real Vector Space is a special case of the Complex Vector Space.

Any element that satisfies the above said axioms (A.1.5) of a vector space is called a vector. When we generally refer to a vector, picutre of arrow comes to our mind but arrows are just one of the manifetation of a representation of a Real Vector Space. Thus to make the reader digress from thinking of vectors as arrows, and be more general, below given are few examples of manifestations of vectors in a vector space.

Example A.1.6. Matrices are vectors in the vector space of $n \times n$ matrices over a field $\mathbb{F}$. The vector addition is defined as the element-wise addition of the matrices and the scalar multiplication is defined as the element-wise multiplication of the matrix by the scalar. It satisfies the axioms of vector space (refer definition A.1.5). Thus, they are elements of vector space. Note that matrices have no notion of length (magnitude) and direction unlike arrows.

Example A.1.7. Functions are vectors in the vector space of functions over a field $\mathbb{F}$. The vector addition and scalar multiplication is defined as the usual addition of two functions and scalar multiplication is simply multiplying the function by the scalar respectively. It satisfies the above said axioms of vector space. Thus, they are elements of vector space. Note that functions have no notion of length (magnitude) and direction unlike arrows. The functions are an example of infinite dimensional vector space unlike discrete dimensional vector space (such as matrices, vectors).

Thus when we say vector we refer to any element of a vector space and not just the arrows. The arrows are just one of the manifestations of vectors in a vector space and the other examples include matrices, functions. In Quantum mechanics we generally deal with Hilbert Space (Complex Vector Space with inner product defined, more about it later). In Quantum Computing, we deal with finite dimensional complex vector spaces. Thus, it could not be imagined as arrows since they deal with real vector spaces (but imagining as arrows helps in general).

Definition A.1.6. Subspace: A subset W of a vector space V over a field $\mathbb{F}$ is called a subspace of V if W is itself a vector space over $\mathbb{F}$ with the same operations of addition and scalar multiplication defined on V. Note that a subspace must always contain zero vector $|0\rangle$ of the vector space V to satisfy the additive identity axiom.

Example A.1.8. Set of all real symmetric matrices is a subspace of the vector space of all real matrices.

Definition A.1.7. Linear Combination: Let V be a vector space over a field $\mathbb{F}$ and let $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ be vectors in V. A vector of the form

$$a_1 |v_1\rangle + a_2 |v_2\rangle + \ldots + a_n |v_n\rangle \tag{A.1}$$

where $a_1, a_2, \ldots, a_n \in \mathbb{F}$ is called a linear combination of $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$

Example A.1.9. $3x^2 + 2x + 1 = 0$ is a linear combination of $|v_1\rangle = x^2$, $|v_2\rangle = x$ and $|v_3\rangle = 1$ in the vector space of polynomials with coefficients in $\mathbb{R}$ as the field.

$$3x^2 + 2x + 1 = 3|v_1\rangle + 2|v_2\rangle + 1|v_3\rangle$$

Example A.1.10. Consider the matrix $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ in the vector space of 2×2 matrices over the field $\mathbb{R}$. It can be written as a linear combination of the matrices $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ as

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = 1\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + 3\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + 4\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Definition A.1.8. Span: Let V be a vector space over a field $\mathbb{F}$ and let $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ be vectors in V. The set of all linear combinations of $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ is called the span of $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ and is denoted by $Span(|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle)$

Definition A.1.9. Linear Independence: Let V be a vector space over a field $\mathbb{F}$ and let $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ be vectors in V. The vectors $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ are said to be linearly independent if the only solution to the equation

$$a_1|v_1\rangle + a_2|v_2\rangle + \ldots + a_n|v_n\rangle = |0\rangle \tag{A.2}$$

is trivial solution $a_1 = a_2 = \ldots = a_n = 0$

If the vector equation has non-trivial solution (e.e infinitely many solutions or at least one of the $a_i \neq 0$) then the vectors are said to be linearly dependent.

Corollary A.1.1. *A set S with two or more vectors is not linearly independent i.e. linearly dependent iff at least one vector is expressible as a linear combination of other vectors in S*

Corollary A.1.2. *If finite set S contains zero vector $|0\rangle$ then the set S is linearly dependent.*

Definition A.1.10. Basis: Let V be a vector space over a field $\mathbb{F}$. A set of vectors $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ in V is called a basis of V if

1. $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ are linearly independent
2. $Span(|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle) = V$

Example A.1.11. A collection of n linearly independent vector alswas spans $\mathbb{R}^n$ and is called a basis of $\mathbb{R}^n$.

Definition A.1.11. Dimension: Let V be a vector space over a field $\mathbb{F}$. The number of vectors in a basis of V is called the dimension of V and is denoted by $dim(V)$ Note that this is because the number of vectors in a basis of any Vector space is unique.

Theorem A.1.3. *The coefficients of expansion v_i of vector $|v\rangle$ in terms of linearly independent basis $|i\rangle$ are called components of the vector in that basis. It is always a unique expansion in a given basis since the basis is linearly independe and only a trivial linear relation between them can exist. Given a basis components are unique, but if we change the basis, the components will change.*

Note that $|v\rangle$ is an abstract which satisfies various relations. When we choose a basis the vectors assume a concrete form and the relation between vectors is satisfied by the components. On changing basis, the components will change in numerical value, but the relation between them expressing the equality still holds between the set of new components.

Addition of two vectors in a basis (both the vectors are expressed in the same basis):

$$|v\rangle = \sum_{i=1}^{n} v_i |i\rangle, |w\rangle = \sum_{i=1}^{n} w_i |i\rangle$$

$$|v\rangle + |w\rangle = \sum_{i=1}^{n} (v_i + w_i) |i\rangle$$

To add two vectors, add their corresponding components in the same basis.
Scalar multiplication of a vector in a basis:

$$a |v\rangle = a \sum_{i=1}^{n} v_i |i\rangle = \sum_{i=1}^{n} (a v_i) |i\rangle$$

To multiply a vector by a scalar, multiply each component by the scalar.

A.1.2 Inner Product

Similar to the definition of dot product defined in $\mathbb{R}^n$ we define inner product in a vector space. The inner product is a generalization of the dot product in $\mathbb{C}^n$. The inner product is a function that takes two vectors as input and returns a scalar as output. The inner product is denoted by $\langle u|v\rangle$ and is defined as follows:

Definition A.1.12. Inner Product: Let V be a vector space over a field $\mathbb{F}$. An inner product on V is a function that assigns to each pair of vectors $|u\rangle, |v\rangle \in V$ a scalar $\langle u|v\rangle \in \mathbb{F}$ such that for all $|u\rangle, |v\rangle, |w\rangle \in V$ and $a, b \in \mathbb{F}$ the following properties hold:

1. Conjugate Symmetry: $\langle u|v\rangle = \langle v|u\rangle^*$
2. Linearity in the second argument: $\langle v|au + bw\rangle = a\langle v|u\rangle + b\langle v|w\rangle$
3. Positive Definite: $\langle v|v\rangle \geq 0$ and $\langle v|v\rangle = 0$ iff $|v\rangle = |0\rangle$

A vector space with an inner product defined is called an inner product space.

Example A.1.12. In $\mathbb{R}^n$ as Field the inner product is defined as $\langle u|v\rangle = \sum_{i=1}^{n} u_i v_i$ and holds the following properties:

1. Symmetry: $\langle u|v\rangle = \langle v|u\rangle$
2. Linearity: $\langle au + bw|v\rangle = a\langle u|v\rangle + b\langle w|v\rangle = \langle v|au + bw\rangle$
3. Positive Definite: $\langle v|v\rangle = \sum_{i=1}^{n} v_i^2 \geq 0$ and $\langle v|v\rangle = 0$ iff $|v\rangle = |0\rangle$

Here the properties of inner product are reduced since it is defined on Real numbers. The conjugate symmetry Property is reduced to Symmetry Property. The Linearity in first argument reduces to Linearity in both arguments. The Positive Definite property remains the same.

Antilinearity: The inner product is linear in the second argument and antilinear in the first argument. The antilinearity in the first argument is defined as follows:

$$\begin{aligned}\langle av + bw|u\rangle &= \langle u|av + bw\rangle^* \quad \text{(Using Property 1: Conjugate Symmetry)}\\ &= (a\langle v|u\rangle + b\langle w|u\rangle)^* \quad \text{(using Property 2: Linearity in second argument)}\\ &= a^*\langle v|u\rangle + b^*\langle w|u\rangle \quad \text{(Using the properties of complex conjugate)}\end{aligned}$$

where a^*, b^* is the complex conjugate of a and b respectively.

Inner products of a linear superposition with another vector is the corresponding superposition of inner products if the superposition occus in second factor, while it is the superposition with all coefficients conjugated if the superposition occurs in the first factor.

Another way in which inner product can be thought of is as projections. The inner product $\langle u|v\rangle$ is the projection of $|v\rangle$ onto $|u\rangle$. Similarly, we can say that the inner product $\langle v|u\rangle$ is the projection of $|u\rangle$ onto $|v\rangle$. Note that unlike in the case of real vector spaces here the projections of a on b is not the same as the projection of b on a. here, the projections are complex conjugates of each other as can be clearly observed since the inner product is conjugate symmetric ($\langle u|v\rangle = \langle v|u\rangle^*$). Thus, the projection of u on v is a complex conjugate of the projection of v on u.

Definition A.1.13. Norm (length) of a vector: Let V be a vector space over a field $\mathbb{F}$. The norm of a vector $|v\rangle \in V$ is defined as $\| |v\rangle \| = \sqrt{\langle v|v\rangle}$. A vector which has a norm of 1 is called a unit vector.

Definition A.1.14. Orthogonality: Let V be a vector space over a field $\mathbb{F}$. Two vectors $|u\rangle, |v\rangle \in V$ are said to be orthogonal if $\langle u|v\rangle = 0$

Definition A.1.15. Orthonormal: Let V be a vector space over a field $\mathbb{F}$. Two vectors $|u\rangle, |v\rangle \in V$ are said to be orthonormal if they are orthogonal and have unit norm.

Definition A.1.16. Orthonormal Basis: Let V be a vector space over a field $\mathbb{F}$. A basis $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ of V is called an orthonormal basis if

1. $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ are orthogonal
2. $\| |v_i\rangle \| = \sqrt{\langle v|v\rangle} = 1$ for all $i = 1, 2, \ldots, n$

Example A.1.13. A general formula of Inner Product: Let $|u\rangle = \sum_{i=1}^{n} u_i |i\rangle$ and $|v\rangle = \sum_{j=1}^{n} v_j |j\rangle$ be two vectors in a vector space V over a field $\mathbb{F}$. Then the inner product of $|u\rangle$ and $|v\rangle$ is given by

$$\langle u|v\rangle = \sum_{j=1}^{n} \sum_{i=1}^{n} u_i^* v_j \langle i|j\rangle$$

If the basis is orthnormal then the $\langle i|j\rangle$ evaluates as follows:

$$\langle i|j\rangle = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

Thus, in the orthonormal basis this gets simplified to

$$\langle u|v\rangle = \sum_{i=1}^{n} u_i^* v_i$$

Thus, defined inner product satisfies the above axioms of inner product. For norm of a vector $|v\rangle$ in an orthonormal basis is given by

$$\| |v\rangle \| = \sqrt{\langle v|v\rangle} = \sqrt{\sum_{i=1}^{n} v_i^* v_i} = \sqrt{\sum_{i=1}^{n} |v_i|^2}$$

$$\langle v|v\rangle = \sum_{i=1}^{n} |v_i|^2 \geq 0$$

thus, it is used to define the length of the vector. Also note that $|v_i|$ is the modulus of a complex number (thus $|v_i| \geq 0$).

Now note that the inner product can be written as matrix multiplication between two vectors. Let $|u\rangle = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$ and $|v\rangle = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ be two vectors in a vector space V over a field $\mathbb{C}$ with basis vectors as $|i\rangle = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ where 1 is in the ith row position. Then the inner product of $|u\rangle$ and $|v\rangle$ is given by

$$\langle u|v\rangle = \begin{pmatrix} u_1^* & u_2^* & \dots & u_n^* \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} u_1^* & u_2^* & \dots & u_n^* \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \sum_{i=1}^{n} u_i^* v_i$$

Thus, the inner product can be written as matrix multiplication between two vectors. Note that here $\langle u|$ is called a bra vector and we can think of it as lying in

some other vector space defined by the basis vecors: $\langle i| = \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{pmatrix}$ where 1 is in the ith column position. Thus we have a vector space of $|v\rangle$ and another vector space of $\langle v|$, this two spaces are called dual spaces. We can convert a vector from one space to other by performing the adjoint operation (transpose conjugate operation). Thus, $\langle v| = ((|v\rangle)^T)^*$ or $|v\rangle = ((\langle v|)^T)^*$.

Properties of Adjoint Operation:

1. $(a\,|v\rangle + b\,|w\rangle)^\dagger = \langle v|\,a^* + \langle w|\,b^*$
2. $(|v\rangle^\dagger)^\dagger = |v\rangle$

A.1.3 Gram-Schmidt Orthogonalization

Given a set of linearly independent vectors $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ in a vector space V over a field $\mathbb{F}$, the Gram-Schmidt orthogonalization process constructs an orthonormal basis $|u_1\rangle, |u_2\rangle, \dots, |u_n\rangle$ of V from the given set of linearly independent vectors. The process is as follows:

1. Let $|u_1\rangle = \frac{|v_1\rangle}{\|v_1\|}$
2. For $i = 2, 3, \dots, n$ do
   (a) Let $|u_i\rangle = |v_i\rangle - \sum_{j=1}^{i-1} \langle u_j|v_i\rangle\,|u_j\rangle$
   (b) Normalize the vector $|u_i\rangle$: $|u_i\rangle = \frac{|u_i\rangle}{\||u_i\rangle\|}$

Proof. Proof that Gram-Scmidt Orthogonalization process constructs an orthonormal basis of V from a given set of linearly independent vectors $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ in a vector space V over a field $\mathbb{F}$. The proof can be shown by using induction.
Base Case
For $i = 1$, $|u_1\rangle = \frac{|v_1\rangle}{\|v_1\|}$ is a unit vector and thus is an orthonormal basis.
Inductive Hypothesis
Assume that for some $k \geq 1$, the vectors $|u_1\rangle, |u_2\rangle, \dots, |u_k\rangle$ are orthonormal.
Inductive Step
We need to show that the vector $|u_{k+1}\rangle$ is orthonormal. We have

$$\langle u_i|u_j\rangle = \delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

for $i, j = 1, 2, \ldots, k$. Thus, we need to show that $\langle u_i | u_{k+1} \rangle = 0$ for $i = 1, 2, \ldots, k$. We have

$$\langle u_i | u_{k+1} \rangle = \langle u_i | v_{k+1} \rangle - \sum_{j=1}^{k} \langle u_j | v_{k+1} \rangle \langle u_i | u_j \rangle$$

Now, $\langle u_i | u_j \rangle = \delta_{ij}$ and thus the above equation becomes,

$$= \langle u_i | v_{k+1} \rangle - \langle u_i | v_{k+1} \rangle \langle u_i | u_i \rangle$$

Uisng the fact that $|u_i\rangle$ is a unit vector (from inductive hypotehsis that $|u_1\rangle, \ldots, |u_k\rangle$ is an orthonormal basis), we have

$$= \langle v_{k+1} | u_i \rangle - \langle u_i | v_{k+1} \rangle$$

$$= 0$$

Thus, $|u_{k+1}\rangle$ is orthogonal to $|u_1\rangle, |u_2\rangle, \ldots, |u_k\rangle$. Now, we need to show that $|u_{k+1}\rangle$ is a unit vector. We have

$$\| |u_{k+1}\rangle \| = \sqrt{\langle u_{k+1} | u_{k+1} \rangle} = \sqrt{\langle v_{k+1} | u_{k+1} \rangle}$$

Thus, $|u_{k+1}\rangle$ is a unit vector. Thus, by induction, the Gram-Schmidt orthogonalization process constructs an orthonormal basis of V from a given set of linearly independent vectors $|v_1\rangle, |v_2\rangle, \ldots, |v_n\rangle$ in a vector space V over a field $\mathbb{F}$.

□

A.1.4 Cauchy-Schwarz and Traingle Inequalities

Theorem A.1.4. *The Schwarz inequality: Dot product of two vectors cannot exceed the product of their lengths.*

$$| \langle V | W \rangle | \leq \sqrt{\langle V | V \rangle \langle W | W \rangle}$$

Proof. Note that this is obviously true for arrows. Now say,

$$|Z\rangle = |V\rangle - \frac{\langle W | V \rangle}{|W|^2} |W\rangle$$

$$\langle Z| = \langle V| - \frac{\langle V | W \rangle}{|W|^2} \langle W|$$

Note here we are required to find $\langle Z | Z \rangle$. Thus we can either write the bra form and then proceed solving with direct inner product or we can use the ket version in

the first factor and then use the antilienearity axiom (for superposition in the first factor) to proceed further. Then, we have

$$\langle Z|Z\rangle = \langle V|V\rangle - \frac{\langle V|W\rangle \langle W|V\rangle}{|W|^2} - \frac{\langle W|V\rangle \langle V|W\rangle}{|W|^2} + \frac{\langle V|W\rangle \langle W|V\rangle \langle W|W\rangle}{|W|^4}$$

Now $\langle Z|Z\rangle \geq 0$ from axiom. ($\langle W\rangle |W = |W|^2$)Thus,

$$|\langle V|V\rangle| \geq \frac{\langle V|W\rangle \langle W|V\rangle}{|W|^2}$$

$$\langle V|V\rangle \langle W|W\rangle \geq |\langle V|W\rangle|^2$$

$$|\langle V|W\rangle|^2 \leq \langle V|V\rangle \langle W|W\rangle$$

$$|\langle V|W\rangle| \leq \sqrt{\langle V|V\rangle \langle W|W\rangle}$$

Hence proved. □

Theorem A.1.5. *The Triangle Inequality: the length of a sum cannot exceed the sum of the lengths.*

$$|V + W| \leq |V| + |W|$$

Proof. Note that this is obviously true for arrows. Now say,

$$|Z\rangle = |V\rangle + |W\rangle$$

$$\langle Z| = \langle V| + \langle W|$$

Thus, we have

$$\langle Z|Z\rangle = |Z|^2 = \langle V|\,|V\rangle + \langle W|\,|W\rangle + \langle V|\,|W\rangle + \langle W|\,|V\rangle$$

$$|V + W|^2 = |V|^2 + |W|^2 + 2Re(\langle V|W\rangle)$$

Now using, $Re(\langle V|W\rangle) \leq |\langle V|W\rangle|$, we get

$$|V + W|^2 \leq |V|^2 + |W|^2 + 2|\langle V|W\rangle|$$

Using Schwarz Inequality, we get

$$|V + W|^2 \leq |V|^2 + |W|^2 + 2|V||W|$$

$$|V + W|^2 \leq (|V| + |W|)^2$$

$$|V + W| \leq |V| + |W|$$

Hence, proved. □

A.1.5 Linear Operators and Matrix Representation

Definition A.1.17. Linear Operator: Let V and W be vector spaces over a field $\mathbb{F}$. A function $A : V \to W$ is called a linear operator if for all $|v\rangle, |w\rangle \in V$ and $a \in \mathbb{F}$ the following properties hold:

1. $A(|v\rangle + |w\rangle) = A(|v\rangle) + A(|w\rangle)$
2. $A(a\,|v\rangle) = aA(|v\rangle)$

In general, when we say a linear operator is defined on a vector space V, we mean that A is a linear operator from V to V.

Once the action of Linear Operators on the basis vectors is known, their action on any vector space is determined. For basis $|1\rangle, |2\rangle, \ldots, |n\rangle$ of a vector space V over a field $\mathbb{F}$, the action of a linear operator A on any vector $|v\rangle = \sum_{i=1}^{n} v_i\,|i\rangle$ say transforms the basis vectors to $|1'\rangle, |2'\rangle, \ldots, |n'\rangle$ and thus, the vector $|v\rangle$ to $|v'\rangle$ as follows:

$$A(|v\rangle) = A\left(\sum_{i=1}^{n} v_i\,|i\rangle\right) = \sum_{i=1}^{n} v_i A(|i\rangle) = \sum_{i=1}^{n} v_i\,|i'\rangle = |v'\rangle$$

Thus, once if we know the action of a linear operator on the basis vectors, we can determine the action of the linear operator on any vector in the vector space. It is the same linear combination of the vector in the new basis as it was in the old basis. Note we haven't talked anything about writing this operators in matrix format till now and so one must not think in terms of matrix and confuse. The above statement simply says that say given a basis vectors of Vector Space V and a linear operator A, then the action of A on any vector in the vector space is determined by the action of A on the basis vectors. In other words, given the basis vectors $|1\rangle, |2\rangle, \ldots, |n\rangle$ of Vector space V and a linear operator A and its transformed basis vectors $|1'\rangle, |2'\rangle, \ldots, |m'\rangle$ of Vector Space W, we can find what the any vector in the Vector Space V will be in the Vector Space W when we apply the Operator A on the vector. The matrix representation and operators are equivalent for the purpose of this notes and is applicable in Quantum Computing.

Definition A.1.18. Matrix Representation of a Linear Operator: Let V be a vector space over a field $\mathbb{F}$ with basis $|1\rangle, |2\rangle, \ldots, |n\rangle$ and let W be a vector space over $\mathbb{F}$ with basis $|1'\rangle, |2'\rangle, \ldots, |m'\rangle$. Let A be a linear operator from V to W. The matrix representation of A with respect to the basis $|1\rangle, |2\rangle, \ldots, |n\rangle$ of V and the basis $|1'\rangle, |2'\rangle, \ldots, |m'\rangle$ of W is the $m \times n$ matrix $[A]$ whose elements are given by

$$[A]_{ij} = \langle i'|A|j\rangle$$

which is the ith row and jth column entry.

To explain this in more depth. We must know what the operator does to the basis vectors. Then to represent this operator in some matrix representation we must first specify the input and ouput basis for the matrix to map. In general, the input and ouput basis are the same and they are the same standard orthonormal basis that we use (i.e. $|1\rangle = 1, 0, 0, \ldots, 0$, $|2\rangle = 0, 1, 0, \ldots, 0$ and so on). Thus, the matrix representation of the operator in this basis will be given as follows:

$$[A] = \begin{pmatrix} \langle 1'|A|1\rangle & \langle 1'|A|2\rangle & \ldots & \langle 1'|A|n\rangle \\ \langle 2'|A|1\rangle & \langle 2'|A|2\rangle & \ldots & \langle 2'|A|n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle m'|A|1\rangle & \langle m'|A|2\rangle & \ldots & \langle m'|A|n\rangle \end{pmatrix}$$

this indicates that the input basis here is chosen as $|1\rangle, |2\rangle, \ldots, |n\rangle$ and the output basis is chosen as $|1'\rangle, |2'\rangle, \ldots, |m'\rangle$. Note that the jth column represents the components of the jth transformed basis vector $|j'\rangle = A|j\rangle$. Thus the value $\langle i'| A |j\rangle$ is a scalar value which is the component of the jth basis vector $|j\rangle$ after being transformed by the action of opertor A to $A|j\rangle$ on the ith transformed basis vector $|i'\rangle$.

Generally for the purpose of this notes and for the most of the applications we assume the input and output basis to be the standard orthonormal basis. Thus, the problems we will face will be of the kind that we will be given operations of an operator on some vectors (sufficient enough to find its matrix representation) and then we will be asked to find its matrix representation in some input and output basis. Consider the following example for more clarity.

Example A.1.14. Suppose V is a vector space with basis vectors $|0\rangle$ and $|1\rangle$, and A is a linear operator from V to V such that $A|0\rangle = |1\rangle$ and $A|1\rangle = |0\rangle$. Find the matrix representation of A in the basis $|0\rangle$ and $|1\rangle$ (since it doesn't say input and ouput basis, we assume that the input and ouput basis are the same and they are $|0\rangle$ and $|1\rangle$). Also find the matrix representation of A in the basis $|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$ and $|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$ represented in the standard orthonormal basis. Note that the basis $|+\rangle$ and $|-\rangle$ are orthonormal.

Solution: The matrix representation of A in the basis $|0\rangle$ and $|1\rangle$ is given by

$$[A] = \begin{pmatrix} \langle 0|A|0\rangle & \langle 0|A|1\rangle \\ \langle 1|A|0\rangle & \langle 1|A|1\rangle \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The matrix representation of A in the basis $|+\rangle$ and $|-\rangle$ is given by

$$[A] = \begin{pmatrix} \langle +|A|+\rangle & \langle +|A|-\rangle \\ \langle -|A|+\rangle & \langle -|A|-\rangle \end{pmatrix}$$

Note that now we are not given the action of operator A on the vector $|+\rangle$ and $|-\rangle$, thus we must find it first.

$$A|+\rangle = A\left(\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}\right) = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle$$

$$A|-\rangle = A\left(\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}\right) = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\end{pmatrix} = -\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = -|-\rangle$$

Thus the matrix representation of A in the basis $|+\rangle$ and $|-\rangle$ using $A|+\rangle = |+\rangle$ and $A|-\rangle = -|-\rangle$ is given by

$$[A] = \begin{pmatrix} \langle +|A|+\rangle & \langle +|A|-\rangle \\ \langle -|A|+\rangle & \langle -|A|-\rangle \end{pmatrix} = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$$

Thus, given action of linear operators on some vectors (assumed sufficient enough to find the matrix representation) we can find its matrix representation in any input and output basis as shown by the above procedure. The matrix may be numerically different in different basis. Note that once the matrix is specified in a given input and output basis it can operate only on the vectors that are specified in the same input basis and the output of the operation will result in a vector specified in the given ouput basis. Note that to make the connection between matrices and linear operators one must specify a set of input and ouput basis states for the input and output vectors spaces of the linear operator. The matrix representation can thus be found for a linear operator using the above mentioned procedure. On a side note, note that it is not at all necessary for the input and output basis to be the same or even orthonormal as that property has not been used anywhere while deriving the formulation, the condition that needs to be satisfied is just that the basis vectors must be specified for the linear operator's input and output vector spaces.

From now on, when we speak of a matrix representation for a linear operator, we mean a matrix representation with respect to orthonormal input and ouput bases. We also use the convention that if the input and output spaces for a linear operator are the same, then the input and ouput bases are the same, unless noted otherwise. Some of the Important linear Operators:

1. Identity Operator: The identity operator I is defined as $I|v\rangle = |v\rangle$ for all $|v\rangle \in V$. The matrix representation of the identity operator in any basis is the identity matrix.
$$\langle i| I |j\rangle = \langle i|j\rangle = \delta_{ij}$$
where δ_{ij} is the Kronecker delta function.

2. Zero Operator: The zero operator 0 is defined as $0|v\rangle = |0\rangle$ for all $|v\rangle \in V$. The matrix representation of the zero operator in any basis is the zero matrix.

3. Projection Operator: Consider a vector $|v\rangle$ which can be written in some basis as:
$$|v\rangle = \sum_{i=1}^{n} v_i |i\rangle$$
This can be further simplified using the inner product definition as v_i is the projection of the $|v\rangle$ onto the ith basis vector $|i\rangle$. Thus, $v_i = \langle i|v\rangle$. On substituting we get,
$$|v\rangle = \sum_{i=1}^{n} \langle i|v\rangle |i\rangle$$
Rearraging we get,
$$|v\rangle = \sum_{i=1}^{n} |i\rangle \langle i|v\rangle = \left(\sum_{i=1}^{n} |i\rangle \langle i|\right) |v\rangle$$
Let $\mathbb{P}_i = |i\rangle \langle i|$ be the projection operator which projects any vector $|v\rangle$ onto the subspace spanned by the basis vector $|i\rangle$. Thus, $\mathbb{P}_i |v\rangle = |i\rangle \langle i|v\rangle$. Thus, $\mathbb{P} = \sum_{i=1}^{n} \mathbb{P}_i = \sum_{i=1}^{n} |i\rangle \langle i| = I$ (Identity Operator) is called the **Completeness Relation**, and $\mathbb{P}|v\rangle = |v\rangle$. Thus, $\mathbb{P}$ is a projection operator which projects any vector $|v\rangle$ onto the subspace spanned by the basis vectors $|i\rangle$. Projection operator acting on bra $\langle v|$ is $\langle v| \mathbb{P}_i = \langle i| v_i^*$. Now,
$$\mathbb{P}_i \mathbb{P}_j = |i\rangle \langle i|j\rangle \langle j| = \delta_{ij} \mathbb{P}_j$$
this equation indicates that once $\mathbb{P}_i$ projects out the part of $|V\rangle$ along $|i\rangle$, further applications of $\mathbb{P}_i$ make no difference; and the subsequent applications of $\mathbb{P}_i (i \neq j)$ will result in zero, since a vector entirely along $|i\rangle$ cannot have a projection along a perpendicular direction $|j\rangle$. Thus, $P^2 = P$ propery must be satisifed by any projection matrix. Note that here we have assumed that the vectors to be projected are on orthonormal basis.

In the standard orthonormal basis the projection operator $\mathbb{P}_i = |i\rangle \langle i|$ will be 1 at the ith position in the diagonal and zero every where else.

$$\mathbb{P}_i = |i\rangle \langle i| = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dots & 0 & 0 \\ \vdots & 1 & \dots & 0 \\ 0 & \dots & 0 & 0 \end{pmatrix}$$

Thus, upon adding all the individual projection operator on i we get the identity operator. The matrix element of the projetion opertors are thus given as:

$$(\mathbb{P}_i)_{kl} = \langle k|i\rangle \langle i|l\rangle = \delta_{ki}\delta_{il} = \delta_{kl}$$

$I - P$ is also a projection operator which projects onto the subspace orthogonal to the subspace spanned by the basis vector $|i\rangle$ and is called orthogonal complement.

4. Rotation operator: $R(\theta)$ on $\mathbb{R}^n$ is defined as rotation by an angle $\theta = |\theta|$ about the axis parallel to the unit vector $\hat{\theta} = \theta/|\theta|$. For example, $R(\frac{\pi}{2}i)$ rotates the vector by $\frac{\pi}{2}$ about the x-axis. The matrix representation of the rotation operator in the standard orthonormal basis is given by

$$[R(\theta)] = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Note that the Unitary operators are generalization of Rotation operator in $\mathbb{V}^n$ which will be seen later.

Matrix Representation for Product of Operators

Given two linear operators A and B acting on a vector space V over a field $\mathbb{F}$, the product of the operators AB is defined as the operator that results from applying A followed by B on any vector $|v\rangle \in V$.

$$[AB]_{ij} = \langle i| AB |j\rangle = \sum_{k=1}^{n} \langle i| A |k\rangle \langle k| B |j\rangle = \sum_{i=1}^{n} A_{ik}B_{kj} = [A][B]$$

where $[A]$ and $[B]$ are the matrix representations of the operators A and B respectively. Note that the order of the product of the operators is important and the product of the matrices is not commutative in general. Thus, the matrix representation of the product of the operators AB is given by the product of the matrix representations of the operators A and B.

Outer Product Representation of Operators

Suppose we are required to represent an operator A action on input vector space V and output vector space W as some linear combination of outer produts. Suppose $A : V \to W$ is a linear operator with $|v_i\rangle$ as orhtonormal basis for V, and $|w_j\rangle$ as orthonormal basis for W. Thus, we obtain

$$A = I_w A I_v = \sum_{i=1}^{n}\sum_{j=1}^{m} |w_j\rangle \langle w_j| A |v_i\rangle \langle v_i| = \sum_{i=1}^{n}\sum_{j=1}^{m} \langle w_j| A |v_i\rangle |w_j\rangle \langle v_i|$$

Thus, the operator A can be represented as a linear combination of outer products of the basis vectors of the input and output vector spaces.

Properties of Adjoint of an Operator

We know, for a given ket

$$A |v\rangle = |Av\rangle$$

and for the corresponding bra

$$\langle Av| = \langle v| A^\dagger$$

Thus, the adjoint of an operator is defined as

$$(A^\dagger)_{ij} = \langle i| A^\dagger |j\rangle = \langle j| A |i\rangle^* = (A_{ji})^*$$

Also,

$$(AB)^\dagger = B^\dagger A^\dagger$$

Example A.1.15.

$$\alpha_1 |V_1\rangle = \alpha_2 |V_2\rangle + \alpha_3 |v_3\rangle \langle V_4|V_5\rangle + \alpha_4 AB |V_6\rangle$$

It's adjoint will be given as:

$$\alpha_1^* \langle V_1| = \alpha_2^* \langle V_2| + \alpha_3^* \langle v_3| \langle V_5|V_4\rangle + \alpha_4^* \langle V_6| B^\dagger A^\dagger$$

In words, when a product of operators, bras, kets and explicit numerical coefficents is encountered, reverse the order of all factors and make the substitutions $A \to A^\dagger$ and $|\rangle \to \langle|$ and $\alpha \to \alpha^*$.

A.1.6 Hermitian and Unitary Operators

Note that both the Hermitian and Unitary operators are defined only for a Square Matrices.

Hermitian Operators

Definition A.1.19. Hermitian Operator is defined as:

$$A^\dagger = A$$

Skew-Hermitian/Anti-Hermitian operator is defined as:

$$A^\dagger = -A$$

Properties of Hermitian operators:

- Since $A^\dagger = A$, thus the princial diagonal elements will be $a_{ii} = a_{ii}^*$, thus the principal diagonal elements of a Hermitian operator are real.
- Principal Diagaonl elements for a skew hermitian matrix will satisfy $a_{ii} = -a_{ii}^*$, thus the principal diagonal elements of a skew hermitian operator are imaginary.
- Any Matrix can be represented as a sum of Hermitian and skew-Hermitian matrices.
$$A = \frac{A + A^\dagger}{2} + \frac{A - A^\dagger}{2}$$
where the $\frac{A+A^\dagger}{2}$ is Hermitian and $\frac{A-A^\dagger}{2}$ is skew-Hermitian.
- Any Complex matrix can be expressed as $P + \iota Q$ where P and Q are Hermitian.
- If A and B are Hermitian matrices and commute (i.e. $AB = BA$), then their product is also Hermitian i.e. AB is Hermitian. Let,
$$(AB)^\dagger = B^\dagger A^\dagger = BA = AB$$
Thus, AB is Hermitian.
- If A and B are Hermitian then $A + B$ and $A - B$ are also Hermitian.

Unitary Operators

Definition A.1.20. Unitary operator is defined as:

$$U^\dagger U = UU^\dagger = I$$

i.e. $U^\dagger = U^{-1}$

Properties of Unitary operators:

- Product of two unitary matrix is unitary. Say U_1 and U_2 are unitary matrices, then U_1U_2 is also unitary.

$$(U_1U_2)^\dagger U_1U_2 = U_2^\dagger U_1^\dagger U_1U_2 = U_2^\dagger U_2 = I$$

- Unitary Operators Preserve inner Products:

$$\langle Uv|Uw\rangle = \langle v|\, U^\dagger U\, |w\rangle = \langle v|w\rangle$$

- Unitary Operators Preserve Norms:

$$\|v\| = \sqrt{\langle v|v\rangle}$$

$$\implies \|Uv\| = \sqrt{\langle Uv|Uv\rangle} = \sqrt{\langle v|\, U^\dagger U\, |v\rangle} = \sqrt{\langle v|v\rangle} = \|v\|$$

Thus, **Unitary operators preserve inner product and norms and hence are generalizations of rotation operators in $\mathbb{V}^n$. Thus action of a Unitary operator on any vector just rotates the vector in the vector space and the vector neither scales up or down in norm.**

- Columns of a $n \times n$ unitary matrix are orthonomal and form a basis for $\mathbb{C}^n$. Similarly rows of a Unitary matrix are orthonormal and form a basis for $\mathbb{C}^n$.

$$\delta_{ij} = \langle i|\, I\, |j\rangle = \langle i|\, U^\dagger U\, |j\rangle = \sum_{k=1}^{n} \langle i|\, U^\dagger\, |k\rangle\, \langle k|\, U\, |j\rangle = \sum_{k=1}^{n} U_{ki}^* U_{kj}$$

Thus, it is 1 only if $i = j$ and 0 otherwise. Thus, the columns of a unitary matrix are orthonormal. Simliary, it can be proved that the rows of a unitary matrix are orthonormal.

- Determinant of a Unitay Matrix is a complex number of unit modulus.

$$UU^\dagger = I$$

Taking determinant on both sides we get,

$$\det(UU^\dagger) = \det(I)$$

$$\det(U)\det(U^\dagger) = 1$$

$$|\det(U)|^2 = 1$$

$$|\det(U) = 1$$

Thus, the determinant of a unitary matrix is a complex number of unit modulus.

Unitarily Diagonziable

Definition A.1.21. Unitarily Diagonizable: A matrix A is said to be unitarily diagonizable if there exists a unitary matrix U such that $U^\dagger AU = D$ where D is a diagonal matrix. Here, U is a unitary matri whose columns are the orthonormal eigen vectors of A. The matrix D is a diagonal matrix whose entries are the eigen values of A corresponding to the eigen vectors of A written in U. **Note that the matrix is diagonal in the basis of the eigen vectors of the matrix A.** Note that any matrix with n (n is the dimension of the square matrix) linearly independent Eigen vectors can be written as:

$$AV = V\Lambda$$

where V is the matrix whose columns are the eigen vectors of A and Λ is the diagonal matrix whose diagonal elements are the eigen values of A. Thus, on post-multiplying by $V^\dagger$ on both sides we get:

$$A = V\Lambda V^\dagger$$

or,

$$V^\dagger AV = \Lambda$$

Thus, the matrix A is diagonal in its Eigen basis (assuming no degeneracy i.e. it has n linearly independent eigen vectors.). In the special case when the eigen vectors are orthonormal then the matrix V is unitary and the matrix A is unitarily diagonizable.

Normal Operators

Definition A.1.22. Normal Operators: A normal operator is an operator that commutes with its adjoint. i.e. $AA^\dagger = A^\dagger A$.

Theorem A.1.6. *An operator is normal if and only if it is unitarily diagonizable or if a matrix is Unitarily diagonizable then it is always a normal operator.*

Proof. Let A be Unitarily diagonalizable. Then,

$$A = V\Lambda V^\dagger$$

$$AA^\dagger - A^\dagger A = V\Lambda V^\dagger V\Lambda^\dagger V^\dagger - V\Lambda^\dagger V^\dagger V\Lambda V^\dagger$$

Now since V is unitary matrix and Λ is a diagonal matrix, thus $V^\dagger V = I$ and $\Lambda^\dagger\Lambda = \Lambda\Lambda^\dagger$. Thus, Substituting we get

$$AA^\dagger - A^\dagger A = V\Lambda\Lambda^\dagger V^\dagger - V\Lambda^\dagger\Lambda V^\dagger = V\Lambda\Lambda^\dagger V^\dagger - V\Lambda\Lambda^\dagger V^\dagger = 0$$

Thus, $AA^\dagger = A^\dagger A$ and thus A is a normal operator.
Hence if A is unitarily diagonalizable then it is a Normal operator. Now let A be a normal operator. Then,

$$AA^\dagger = A^\dagger A$$

Now we are required to show that A is unitarily diagonalizable. We know that any Matrix A can be decomposed onto a Hermitian and Skew Hermitian matrix as:

$$A = \frac{A + A^\dagger}{2} + \frac{A - A^\dagger}{2}$$

where $\frac{A+A^\dagger}{2}$ is Hermitian and $\frac{A-A^\dagger}{2}$ is Skew Hermitian. Now let $A_H = \frac{A+A^\dagger}{2}$ denote the Skew Hermitian matrix and $A_{SH} = \frac{A-A^\dagger}{2}$ denote the skew-Hermitian matrix. Thus, $A = A_H + A_{SH}$. NOw since A is a normal operator we get,

$$AA^\dagger - A^\dagger A = (A_H + A_{SH})(A_H + A_{SH})^\dagger - (A_H + A_{SH})^\dagger(A_H + A_{SH}) = 0$$

$$A_H A_H^\dagger + A_{SH}A_{SH}^\dagger + A_H A_{SH}^\dagger + A_{SH}A_H^\dagger - A_H^\dagger A_H - A_{SH}^\dagger A_{SH} - A_H^\dagger A_{SH} - A_{SH}^\dagger A_H = 0$$

Now substituting the fact that $A_H \implies A_H = A_H^\dagger$ is Hermitian and $A_{SH} \implies A_{SH} = -A_{SH}^\dagger$ is Skew Hermitian we get,

$$A_{SH}A_H = A_H A_{SH}$$

Thus, A_H and A_{SH} commute. Now since A_H and A_{SH} are Hermitian and Skew Hermitian respectively, they are both unitarily diagonizable. **Using the fact that two matrices commute if and only if they have the same eigen vectors. Thus, both A_H and A_{SH} have the same eigen vectors and also the fact that A_H and A_{SH} are both unitarily diagonalizable**, thus we get the same unitary matrix V:

$$A_H = V\Lambda_H V^\dagger$$
$$A_{SH} = V\Lambda_{SH} V^\dagger$$

Now $A = A_H + A_{SH}$, thus

$$A = V\Lambda_H V^\dagger + V\Lambda_{SH} V^\dagger = V(\Lambda_H + \Lambda_{SH})V^\dagger$$

Now, Λ_H and Λ_{SH} are diagonal matrices, thus $\Lambda_H + \Lambda_{SH}$ is also a diagonal matrix.

$$A = V\Lambda V^\dagger$$

Thus, A is unitarily diagonalizable where V are the orthonormal eigen vectors of A and Λ is the diagoanl matrix with entries as the correspoding eigen values. Hence, proved. **Thus the statement that a matrix is normal if and only if it is unitarily diagonalizable.** □

Properties of the Normal operators:

- Hermitian and Unitary operators are both special cases of Normal Operators. Thus, both are unitarily diagonalizable and hence have n linearly independent Eigen vectors.
- A normal matrix is Hermitian if and only if it has real eigen values.

Positive Operators

Definition A.1.23. Positive Operators: A positive operator is an operator such that for any vector $|v\rangle$ in the vector space, the inner product $\langle v|A|v\rangle$ is real, non-negative. i.e. $\langle v|A|v\rangle \geq 0$.

Properties of Positive Operators:

- A positive operator is Hermitian.

- For any operator A, $A^\dagger A$ is a positive operator.

$$\langle v|A^\dagger A|v\rangle \geq 0$$

$$\langle v|A^\dagger A|v\rangle = \langle Av|Av\rangle = \langle u|u\rangle \geq 0$$

Since, the inner product of any vector with itself is non-negative.

A.1.7 Eigen Value Problems

Given a linear operator A acting on a vector space V over a field $\mathbb{F}$, the eigen value problem is to find the eigen values λ and the corresponding eigen vectors $|v\rangle$ such that

$$A\,|v\rangle = \lambda\,|v\rangle$$

$$\implies (A - \lambda I)\,|v\rangle = 0$$

i.e. vectors such that upon the action of the operator A the vector $|v\rangle$ is scaled by a factor λ (could be positive or negative, a negative value indicates change in direction and a value less than $|1|$ indicates scaled down). Thus, the eigen values are the values of λ for which the operator $A - \lambda I$ has a non-trivial null space. The eigen vectors are the vectors in the null space of the operator $A - \lambda I$.

$$det(A - \lambda I) = 0$$

This, results in a characteristic equation of the form:

$$P_A(\lambda) = det(A - \lambda I) = \lambda^n - \beta_1\lambda^{n-1} + \ldots + (-1)^r\beta_r\lambda^{n-r} + \ldots + (-1)^n\beta_n = 0$$

In general, we represent eigen value as ω and the corresponding eigen vector as $|\omega\rangle$.

Example A.1.16. For the identitiy operator I, the eigen value equation is given as:

$$I\,|v\rangle = \lambda\,|v\rangle$$

$$\implies |v\rangle = \lambda\,|v\rangle$$

$$\implies \lambda = 1$$

Thus, the eigen values of the identity operator are $\lambda = 1$ and the corresponding eigen vectors are all the vectors in the vector space.

Example A.1.17. Consider a projection operator $\mathbb{P}_v$ which projects any vector $|v\rangle$ onto the subspace spanned by the basis vector $|v\rangle$. Thus, $\mathbb{P}_v |v\rangle = |v\rangle$. Thus, the eigen value equation is given as:

$$\mathbb{P}_v |v\rangle = \lambda |v\rangle$$

Case 1: Consider any ket $\alpha |v\rangle$, parallel to $|v\rangle$ then $\mathbb{P}_v |v\rangle = |v\rangle = \lambda |v\rangle$. Thus, $\lambda = 1$ is an eigen value of the projection operator $\mathbb{P}_v$.
Case 2: Consider any ket $\alpha |v_\perp\rangle$, perpendicular to $|v\rangle$ then $\mathbb{P}_v |v_\perp\rangle = 0 = \lambda |v_\perp\rangle$. Thus, $\lambda = 0$ is an eigen value of the projection operator $\mathbb{P}_v$.
Case 3: Consider kets that are neither parallel nor perpendicular to $|v\rangle$, then $\mathbb{P}_v(\alpha |v_\perp\rangle + \beta |v\rangle) = \beta |v\rangle \neq \lambda(\alpha |v_\perp\rangle + \beta |v\rangle)$. Thus, λ is not an eigen value of the projection operator $\mathbb{P}_v$. Thus, we have considered every possible ket in the space and have found all the eigen values and eigen vectors. The eigen values of a projection matrix are $\lambda = 0$ and $\lambda = 1$ and their corresponding eigen vectors are the vectors parallel and perpendicular to the subspace spanned by the basis vector $|v\rangle$ resprectively.

Example A.1.18. Consider the Rotation operator $R(\vec{\theta})$ which rotates any vector by an angle $\frac{\vec{\theta}}{|\theta|}$ about the $\vec{\theta}$ axis. The eigen value equation is given as:

$$R(\vec{\theta}) |v\rangle = \lambda |v\rangle$$

Clearly, any ket parallel to the axis of rotation will not change its direction upon rotation and thus will be an eigen vector of the rotation operator with eigen value $\lambda = 1$. Thus, the corresponding eigen vector is $\vec{\theta}$. The other two eigen values are $\lambda = e^{\pm i\theta}$ and the corresponding eigen vectors are the vectors perpendicular to the axis of rotation.

Properties of Eigen Roots:

- The sum of the eigen values is equal to the trace of the matrix.

$$Tr(A) = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i = (-1)^{n-1}\beta_{n-1}$$

- The product of the eigen values is equal to the determinant of the matrix.

$$det(A) = \prod_{i=1}^{n} \lambda_i = (-1)^n \beta_n$$

- if $det(A) = 0$ then at least one of the eigen value is zero.
- For any triangular matrix, the eigen values are the principal diagonal elements.
- Eigen values of Two Equivalent/Similar matrix are not necessarily same.
- Eigen Values of a Unitary matrix is a complex number with unit modulus and the Eigen vecotrs are mutually orthongoal (Assuming no degeneracy). Let,

$$U|u_i\rangle = u_i|u_i\rangle$$

$$U|u_j\rangle = u_j|u_j\rangle$$

Taking adjoint of the second equation we get:

$$\langle u_j|U^\dagger = \langle u_j|u_j^*$$

Taking inner product with the first equation we get:

$$\langle u_j|U^\dagger U|u_i\rangle = \langle u_j|u_j^* u_i|u_i\rangle$$

$$\langle u_j|u_i\rangle = u_j^* u_i \langle u_j|u_i\rangle$$

$$(1 - (u_j^* u_i))\langle u_j|u_i\rangle = 0$$

If i=j then $\langle u_j|u_i\rangle \neq 0$ thus $u_i = u_i^*$ i.e. real eigen values. If i $\neq$ j then $\langle u_j|u_i\rangle = 0$ i.e. the eigen vectors are orthogonal.

In case of no degeneracy, Unitary matrix has complex eigen values of unit modulus and othrongonal eigen vectors.

- If λ is the Eigen root of A then $f(\lambda)$ is the Eigen root of $f(A)$.
- Suppose A is a matrix such that $A^k = 0$ then the all the eigen values of A is zero.
- Complex Eigen Values always exist in pairs.
- Eigen Values of a Hermitian matrix are real. Given A is a Hermitian matrix,

$$A^\dagger = A$$

$$A|v\rangle = \lambda|v\rangle$$

Taking inner product with $\langle v|$, we get:

$$\langle v| A |v\rangle = \lambda \langle v|v\rangle$$

Now adjoint of the equation $A|v\rangle = \lambda|v\rangle$ is

$$\langle v| A^\dagger = \lambda^* \langle v|$$

Taking inner prodcut with $|v\rangle$, we get:

$$\langle v| A^\dagger |v\rangle = \lambda^* \langle v|v\rangle$$

$$\langle v| A |v\rangle = \lambda^* \langle v|v\rangle \quad \text{(since A is Hermitian, } A = A^\dagger)$$

Thus, equating the two equations we get:

$$(\lambda - \lambda^*) \langle v|v\rangle = 0$$

Thus,

$$\lambda = \lambda^*$$

Hence, the **Eigen Values of a Hermitian matrix are always real**.

- The Eigen vectors of the Hermitian matrix with respect to distinct eigen values are orthogonal.

  Consider the matrix A with distinct eigen values λ_i and λ_j and the corresponding eigen vectors $|v_i\rangle$ and $|v_j\rangle$. Thus,

$$A|v_i\rangle = \lambda_i |v_i\rangle$$

$$A|v_j\rangle = \lambda_j |v_j\rangle$$

  Taking inner product of the above two equations with $\langle v_j|$ and $\langle v_i|$ respectively, we get:

$$\langle v_j| A |v_i\rangle = \lambda_i \langle v_j|v_i\rangle$$
$$\langle v_i| A |v_j\rangle = \lambda_j \langle v_i|v_j\rangle$$

  Taking adjoint of the above second equations, we get:

$$\langle v_j| A^\dagger |v_i\rangle = \lambda_j^* \langle v_j|v_i\rangle$$

  Since A is Hermitian, $A = A^\dagger$, thus

$$\langle v_j| A |v_i\rangle = \lambda_j^* \langle v_j|v_i\rangle$$

Thus, equating the two equations, we get:

$$\lambda_i \langle v_j | v_i \rangle = \lambda_j^* \langle v_j | v_i \rangle$$

$$(\lambda_i - \lambda_j^*) \langle v_j | v_i \rangle = 0$$

Thus if $\lambda_i \neq \lambda_j$ then $\langle v_j | v_i \rangle = 0$ i.e. the eigen vectors of a Hermitian matrix with respect to distinct eigen values are orthogonal. If $i = j$ then $\langle v_j | v_i \rangle \neq 0$ thus, $\lambda_i = \lambda_j^* \implies \lambda_i = \lambda_i^*$ i.e. real eigen values.

The Hermitian matrices have all real eigen values and the eigen vectors corresponding to distinct eigen values are orthogonal.

Definition A.1.24. Geometric Multiplicity: The number of linearly indepndent Eigen vectors associated with a particular λ is called the Geometric multiplicity. Suppose λ_i has geometric multiplicity k $\implies$ $GM(\lambda_i) = k$ means that k linearly independent eigen vectors exist corresponding to eigen value λ. These vectors form a vector space which is called Eigen Space. In the eigen space all the vectors are only scaled by λ_i and not rotated.

Definition A.1.25. Algebraic Multiplicity: The mupltiplicity of λ as a root of the characteristic equation $P_A(x) = 0$ is called the algebraic multiplicity. Suppose λ_i has algebraic multiplicity k $\implies$ $AM(\lambda_i) = k$ means that λ_i is a root of the characteristic equation of multiplicity k.
where n is the dimension of the matrix.

For any matrix A, the algebraic multiplicity of an eigen value is always greater than or equal to the geometric multiplicity.

$$1 \leq AM(\lambda_i) \leq GM(\lambda_i) \leq n$$

Note that any matrix A will have n eigen values (may be repeated i.e. $AM(\lambda_i) \geq 1$), but it is not necessary for it to have n linearly independent eigen vectors. Corresponding to each unique eigen value λ there will be at least one eigen vector. If the $AM(\lambda_i) > 1$ then there could be more than one linearly independent eigen vector corresponding to the eigen value λ_i but still the number of linearly independent eigen vectors will be less than or equal to the $AM(\lambda_i)$ i.e. $GM(\lambda_i) \leq AM(\lambda_i)$.

Theorem A.1.7. *A Hermitian Matrix is always Unitarily Diagonalizable i.e. it has n linearly independent Eigen vectors. ($AM(\lambda_i)$=$GM(\lambda_i)$) for all λ_i.*

Proof. Let A be a Hermitian matrix. Let the gemetric multiplicity for some eigen value of A be r ($< n$) where n is the dimension of the matrix. Thus, $GM(\lambda) = r$. Thus, we can find r linearly independent eigen vectors (say $\{|v_1\rangle, |v_2\rangle, \ldots, |v_r\rangle\}$) correspoding to λ. Let,

$$V^E = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ |v_1\rangle & |v_2\rangle & \ldots & |v_r\rangle \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

which is the set of Linearly Independent Orthogonal Eigen vectors. Let,

$$V^{\perp} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ |v_{r+1}\rangle & |v_{r+2}\rangle & \ldots & |v_n\rangle \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

be the set of orthogonal vectors spanning space orthogonal to V^E. (Note that V^E and $V^{\perp}$ are orthogonal subspaces to each other). Let,

$$V = \begin{bmatrix} V^E & V^{\perp} \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ |v_1\rangle & |v_2\rangle & \ldots & |v_n\rangle \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Now $A|v_i\rangle = \lambda|v_i\rangle$, for $i = 1, 2, \ldots, r$. Thus,

$$AV = \begin{bmatrix} \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \lambda|v_1\rangle & \lambda|v_2\rangle & \ldots & \lambda|v_r\rangle & A|v_{r+1}\rangle & \ldots & A|v_n\rangle \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix}$$

Now, a vectors in $\mathbb{C}^n$ can be written as a linear combination of the columns of V. This, $Av_i = \sum_{j=1}^{m} c_{ij} v_j$ for $i = r+1, r+2, \ldots, n$. Thus, in matrix form we can write this as:

$$A \begin{bmatrix} \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ |v_1\rangle & |v_2\rangle & \ldots & |v_r\rangle & |v_{r+1}\rangle & \ldots & |v_n\rangle \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix} = V \begin{bmatrix} \lambda_1 & & & & & c_{r+1,1} & \ldots & c_{n,1} \\ & \lambda_2 & & & & c_{r+1,2} & \ldots & c_{n,2} \\ & & \lambda_3 & & & c_{r+1,3} & \ldots & c_{n,3} \\ & & & \ddots & & \vdots & \vdots & \vdots \\ & & & & \lambda_r & c_{r+1,r} & \ldots & c_{n,r} \\ 0 & 0 & 0 & \ldots & 0 & c_{r+1,r+1} & \ldots & c_{n,r+1} \\ 0 & 0 & 0 & \ldots & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & c_{r+1,n} & \ldots & c_{n,n} \end{bmatrix}$$

$$AV = VM$$

Thus, pre-multiplying by $V^\dagger$ on both sides we get:

$$V^\dagger AV = V^\dagger VM$$

Now, $V^\dagger V = I$ since the columns of V are orthonormal (V^E and $V^\perp$ are orthogonal subspaces, we have chosen the vectors in V^E and $V^\perp$ such that they are orthonormal). Thus,

$$V^\dagger AV = M$$

Now taking adjoint on both sides we get:

$$V^\dagger A^\dagger V = M^\dagger$$

Using the fact that A is Hermitian, we get:

$$V^\dagger AV = M^\dagger$$

Thus, $M = M^\dagger$ i.e. M is Hermitian. Thus, M can be written as:

$$M = \begin{bmatrix} \lambda & & & & & 0 & \dots & 0 \\ & \lambda & & & & 0 & \dots & 0 \\ & & \lambda & & & 0 & \dots & 0 \\ & & & \ddots & & \vdots & \vdots & \vdots \\ & & & & \lambda & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,r+1} & \dots & c_{n,r+1} \\ 0 & 0 & 0 & \dots & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,n} & \dots & c_{n,n} \end{bmatrix}$$

Thus, let the matrix C be:

$$C = \begin{bmatrix} c_{r+1,r+1} & \dots & c_{n,r+1} \\ \vdots & \vdots & \vdots \\ c_{r+1,n} & \dots & c_{n,n} \end{bmatrix}$$

Since, M is a Hermitian Matrix thus C must also be a Hermitian Matrix. Now, since $M = V^\dagger AV = V^{-1}AV$, thus M and A are similar matrices. Thus, M and A must have the same Eigen values and Characteristic Polynomial. Thus,

$$P_M(z) = P_A(z)$$

$$(\lambda - z)^r det(C - zI) = det(A - zI)$$

Thus, now we need to show that λ is not an eigen value of C thus proving that $AM(\lambda) = GM(\lambda) = r$ for matrix A. This is done using proof by contradiction.
Let λ be a root of $det(C - zI)$. Thus,

$$Cu = \lambda u$$

where $u \in \mathbb{C}^{m-r}$ is an eignen vector of C. Thus, we can write $\hat{u} \in \mathbb{C}^m$ as:

$$\hat{u} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ u_{r+1} \\ \vdots \\ u_n \end{bmatrix}$$

Now,

$$V^\dagger AV\hat{u} = M\hat{u} = \begin{bmatrix} \lambda & & & & & 0 & \dots & 0 \\ & \lambda & & & & 0 & \dots & 0 \\ & & \lambda & & & 0 & \dots & 0 \\ & & & \ddots & & \vdots & \vdots & \vdots \\ & & & & \lambda & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,r+1} & \dots & c_{n,r+1} \\ 0 & 0 & 0 & \dots & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,n} & \dots & c_{n,n} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ u_{r+1} \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \lambda u_{r+1} \\ \vdots \\ \lambda u_n \end{bmatrix}$$

$$\implies \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \lambda u_{r+1} \\ \vdots \\ \lambda u_n \end{bmatrix} = \lambda \begin{bmatrix} 0 \\ \vdots \\ 0 \\ u_{r+1} \\ \vdots \\ u_n \end{bmatrix} = \lambda \hat{u}$$

Thus,

$$V^\dagger AV\hat{u} = \lambda \hat{u}$$

$\hat{u}$ is an eigen vector of $V^\dagger AV$ with eigen value λ. Premultiplying by V on both sides we get:

$$AV\hat{u} = \lambda V\hat{u}$$

Hence, $V\hat{u}$ is an eigen vector of A corresponding to λ. But, $V\hat{u}$ lies in the vector space of $\{|v_{r+1}\rangle, \ldots, |v_n\rangle\}$ which contradicts the assumption that $\{|v_1\rangle, \ldots, |v_r\rangle\}$ are the only linearly independent eigen vectors of A corresponding to λ. Thus, λ is not an eigen value of C. **Thus, a Hermitian Matrix is always Unitarily diagonalizable. A similar proof can be used to show that a Skew Hermitian Matrix is also always Unitarily diagonalizable.** □

Spectral Decomposition

Theorem A.1.8. *Any normal operator M on a vector space V is diaongal with repspect to some orthonormal basis for V. Conversely, any diagonizable operator is normal.*

This means that M can be written as a sum of orthogonal projection operators onto the eigenspaces of M where $|\lambda_i\rangle$ are eigen vectors of M.

$$M = \sum_{i=1}^{n} \lambda_i \mathbb{P}_i$$

where λ_i are the eigen values of M and $\mathbb{P}_i = |\lambda_i\rangle \langle\lambda_i|$ are the orthogonal projection operators onto the eigenspaces of M. This can also be realised as the fact that any normal operator is always unitarily diagonalizable and hence can be written as an outer product representation. Thus,

$$M = V\Lambda V^\dagger = \sum_{i=1}^{n} \lambda_i |\lambda_i\rangle \langle\lambda_i| = \sum_{i=1}^{n} \lambda_i \mathbb{P}_i$$

where V is the matrix of orthogonal eigen vectors, Λ is a diagonal marix of corresponding eigen values.

Commutator and Anti-Commutator

Definition A.1.26. The commutator of two operators A and B is defined as:

$$[A, B] = AB - BA$$

The anti-commutator of two operators A and B is defined as:

$$\{A, B\} = AB + BA$$

Properites of the Commutator

- $AB = \dfrac{[A,B] + \{A,B\}}{2}$
- if $[A,B] = 0, \{A,B\} = 0$ and A is invertible then $B = 0$.
- $[A,B]^\dagger = [B^\dagger, A^\dagger]$
- $[A,B] = -[B,A]$
- If A and B are Hermitian then $\iota[A,B]$ is Hermitian.

Simultaneous Diagonalization of two Hermitian Operators

Definition A.1.27. Suppose A and B are Hermitian operators, Then $[A,B] = 0$ if and only if there exists an orthonormal bassis such that both A and B are diagonal with respect to that basis. We can say that A and B are simulatneously diagonizable.

Operator Functions

We restrict ourselves to duntions that can be written as a power series. Consider a series

$$f(x) = \sum_{n=0}^{\infty} a_n x^n$$

where a_n are complex numbers. Then we can define the same function for an operator to be:

$$f(A) = \sum_{n=0}^{\infty} a_n A^n$$

where A^n is the product of the operator A with itself n times.

Example A.1.19. Consider the exponential series:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{x^n}{n!}$$

Then the exponential of an operator A is defined as:

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{A^n}{n!}$$

Now we restrict ourselves to Hermitian operator A, then in the eigen basis of A we can write:

$$A = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

Thus, the A^n in the exponential series will be given as:

$$A^n = \begin{pmatrix} \lambda_1^n & 0 & \dots & 0 \\ 0 & \lambda_2^n & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^n \end{pmatrix}$$

Thus, the exponential of the Hermitian operator A will be given as:

$$e^A = \begin{pmatrix} \sum_{i=1}^{\infty} \frac{\lambda^n}{n!} & 0 & \dots & 0 \\ 0 & \sum_{i=1}^{\infty} \frac{\lambda^n}{n!} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sum_{i=1}^{\infty} \frac{\lambda^n}{n!} \end{pmatrix}$$

$$e^A = \begin{pmatrix} e^{\lambda_1} & 0 & \dots & 0 \\ 0 & e^{\lambda_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{\lambda_n} \end{pmatrix}$$

Some results:

- $e^{A^\dagger} = (e^A)^\dagger$
- $e^{A+B} = e^A e^B$ if $[A, B] = 0$
- $e^{A^\dagger} e^A = I$
- $e^{\iota A}$ is unitary if A is Hermitian and $det(e^{\iota A}) = e^{\iota Tr(H)}$.

Trace of a Matrix

Definition A.1.28. The trace of a matrix is the sum of the diagonal elements of the matrix. It is denoted by $Tr(A)$.

$$Tr(A) = \sum_{i=1}^{n} a_{ii}$$

Properties of Trace

- $Tr(AB) = Tr(BA)$
- $Tr(A + B) = Tr(A) + Tr(B)$
- $Tr(A^\dagger) = Tr(A)$
- If $tr(AA^\dagger) = 0$ then $A = 0$ (Null matrix)
- $\|v\|^2 = Tr(vv^\dagger)$
- $Tr(ABC) = Tr(BCA) = Tr(CAB)$

A.1.8 Norms

Definition A.1.29. A real valued function defined on a vector space ($\mathbb{C}^m \to \mathbb{R}$) is called a norm (denoted by $\|x\|$) if it satisfies the following properties:

- $\|x\| \geq 0$ and $\|x\| = 0$ if and only if $x = 0$.
- $\|\alpha x\| = |\alpha|\|x\|$ for any scalar α.
- $\|x + y\| \leq \|x\| + \|y\|$ (Triangle Inequality).

Thus,

$$\|\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n\| \leq |\alpha_1|\|x_1\| + |\alpha_2|\|x_2\| + \cdots + |\alpha_n|\|x_n\|$$

where α_i are scalars and x_i are vectors.
In general, the norm of a vector is the measure of the length of the vector. The p-norm of a vector is defined as:

$$\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

where $x = (x_1, x_2, \ldots, x_n)$ is a vector in $\mathbb{C}^n$. Here, $|x_i|$ is the modulus of the complex number x_i.
The 1-norm of a vector is defined as:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

The 2-norm of a vector is called the Euclidean norm and is defined as:

$$\|x\|_2 = (\sum_{i=1}^{n} |x_i|^2)^{\frac{1}{2}}$$

The ∞-norm of a vector is defined as:

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

The ∞-norm is the maximum of the absolute values of the elements of the vector.

Properties of Norms

- $\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1$
- $\|x\|_2 \leq \sqrt{n}\|x\|_\infty$

where $x \in \mathbb{C}^n$.

A.1.9 Induce Matrix Norm

Definition A.1.30. The induced matrix norm of a matrix A is defined as:

$$\|A\|_p = \max_{\|v\|_p = 1} \|Av\|_p$$

where $\|v\|_p$ is p-the norm of the vector v.

In other words, it is the factor by which a matrix can scale a vector.It is also called Spectral Norm. For the purpose of this notes, we will consider the induced 2-norm i.e. the Euclidean norm for the vectors.

Properties of Induced Matrix Norm

- $\|A\| = \max_{\|v\|=1} \|Av\| = \max_{\|v\|=1} \sqrt{\langle Av|Av\rangle} = \max_{\|v\|=1} \sqrt{\langle v|A^\dagger A|v\rangle}$
- For a diagonal matrix D, $\|D\|_1 = \|D\|_2 = \max_{1 \leq i \leq n} |d_i|$ where d_i are the diagonal elements of the matrix D.
- For any matrix A, $\|A\|_1$ = maximum absolute column sum of the matrix A.
- For any matrix A, $\|A\|_\infty$ = maximum absolute row sum of the matrix A.

- For any matrix A, $\|A\|_2$ = maximum singular value of the matrix A.
- For any matrix A, $\|A\|_2 = \sqrt{\rho(A^\dagger A)}$ where $\rho(A^\dagger A)$ is the maximum eigen value of the matrix $A^\dagger A$.
- For any matrix A, $\|Ax\| \leq \|A\|\|x\|$. This is because $\|A\| = sup_{x\neq 0}\frac{\|Ax\|}{\|x\|}$.
- Sub-Multiplicative Property: $\|AB\| \leq \|A\|\|B\|$ Let $\|x\| = 1$ then,

  $$\|AB\| = sup_{\|x\|=1}\|ABx\|$$

  Now, using the property that $\|Ax\| \leq \|A\|\|x\|$ we get:

  $$\|ABx\| \leq \|A\|\|Bx\| \leq \|A\|\|B\|$$

  Now, for some $x = \hat{x}$, $\|AB\hat{x}\| = \|AB\|$. Thus, $\|AB\| \leq \|A\|\|B\|$.
- $\|A + B\| \leq \|A\| + \|B\|$
- For a Unitary matrix U, $\|U\| = 1$. Thus, $\|UA\|_2 = \|A\|_2$. This is because Unitary Matrix preserves norm of a vector (it only rotates the vectors).

A.1.10 Conditioning and Condition Number

In this section we find the condition number of a matrix times vector multiplication.

Let $y = Ax$ where A is matrix and x, b are vectors. The input given is a vector and the output is Ax. Thus,

$$\begin{aligned}
\kappa(x) &= \max_{\delta x} \frac{\frac{\|A(x+\delta x)-Ax\|}{\|Ax\|}}{\frac{\|\delta x\|}{\|x\|}} \\
&= \max_{\delta x} \frac{\|A\delta x\|}{\|Ax\|}\frac{\|x\|}{\|\delta x\|} \\
&= \max_{\delta x} \frac{\|A\delta x\|}{\|\delta x\|}\frac{\|x\|}{\|Ax\|} \\
&= \frac{\|x\|}{\|Ax\|}\left(\max_{\delta x} \frac{\|A\delta x\|}{\|\delta x\|}\right) \\
&= \frac{\|x\|}{\|Ax\|}\|A\| \\
&= \frac{\|A\|\|x\|}{\|Ax\|}
\end{aligned}$$

Let A be a square matrix and non-singular. Then A^{-1} exists, thus, $x = A^{-1}Ax \implies \|x\| = \|A^{-1}Ax\| \leq \|A^{-1}\|\|Ax\|$. Thus, $\frac{\|x\|}{\|Ax\|} \leq \|A^{-1}\|$. Substituting this, in e previous result, we get,

$$\kappa(x) \leq \|A\|\|A^{-1}\|$$

Thus, we define the condition number for a matrix vector multiplication Ax with $\kappa(x) = \|A\|\|A^{-1}\|$. Building upon this, consider the problem of solving the system of linear equations $Ax = b$ by using $x = A^{-1}b$. Then the condition number will be given as $\kappa(b) = \|A^{-1}\|\|A\|$. Thus, we define the condition number of a matrix A as $\kappa(A) = \|A\|\|A^{-1}\|$. In the 2-norm sense,

$$\kappa(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_{max}}{\sigma_{min}} = \sqrt{\frac{\lambda_{max}(A^\dagger A)}{\lambda_{min}(A^\dagger A)}}$$

where σ_{max} and σ_{min} are the maximum and minimum singular values respectively. If A is Hermitian then the condition number is given as:

$$\kappa(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|}$$

Note that here λ_{max} and λ_{min} are in the absolute sense.

A.1.11 Conclusions

- **A Normal operators ($AA^\dagger = A^\dagger A$) are always unitarily diagonizable (i.e. $A = U^\dagger DU$ where U is a Unitary matrix whose columns are orthonormal eigen vectors and D is a diagonal matrix made up of eigen values of the matrix A.) Thus, any normal matrix has orthogonal (linearly independent) eigen vectors (n is the dimension of the matrix).**
- **Hermitian matrix and Unitary matrix are special cases of Normal operators. Thus, the Hermitian matrix is unitarily diagonizable and has real eigen values and n orthogonal linearly independent eigen vectors. The Unitary matrix is also unitarily diagonizable and has complex eigen values of unit modulus and n orthogonal linearly independent eigen vectors. Since bith the Hermitian and Unitary matrices are unitarily diagonizable since they are normal, thus both of them have spectral decomposition. Thus both Unitary and Hermitian matrix**

can be written as a sum of orthogonal projection operators onto the eigenspaces of the matrix as shown:

$$M = \sum_{i=1}^{n} \lambda_i |\lambda_i\rangle \langle\lambda_i| = \sum_{i=1}^{n} \lambda_i \mathbb{P}_i$$

where λ_i are the eigen values of M and $|\lambda_i\rangle$ are the eigen vectors of M. Here, $\mathbb{P}_i = |\lambda_i\rangle \langle\lambda_i|$ are the orthogonal projection operators onto the eigenspaces of M.

- **If two matrices commute then they have the same eigen vectors (note that the eigen values could be same but still the matrix would have the n linearly independent eigen vectors). Let, $AB = BA$ then,**

$$A|v\rangle = \lambda|v\rangle$$

$$\implies BA|v\rangle = \lambda B|v\rangle$$

$$\implies AB|v\rangle = \lambda B|v\rangle \quad \textbf{(Since AB=BA)}$$

Let $B|v\rangle = |u\rangle$ then,

$$A|u\rangle = \lambda|u\rangle$$

Thus $|u\rangle$ is an eigen vector of A. Thus, $|u\rangle = B|v\rangle$ is an eigen vector of A corresponding to λ (assuming no degeneracy). Thus, $B|v\rangle = p|v\rangle$. Thus, $|v\rangle$ is an eigen vector of B with the eigen value correspoding to p.

A.2 Tensor Products

It is a method for combining two vector spaces to obtain a new larger vector space.

Definition A.2.1. Suppose V and W are vector spaces of dimensions m and n respectively. For convinence we also suppose that V and W are Hilbert Spaces. Then, $V \otimes W$ (read as V tensor W) is a vector space of dimension mn such that any vector in $V \otimes W$ can be written as a linear combination of the tensor products of the vectors in V and W. i.e The elements of $V \otimes W$ are linear combination of 'tensor products' $|v\rangle \otimes |w\rangle$ of elements $|v\rangle$ of V and $|w\rangle$ of W. For example, if $|i\rangle$ and $|j\rangle$ are orthonormal bases for the spaces V and W then $|i\rangle \otimes |j\rangle$ is a basis fr $V \otimes W$. Note that we often use abbreviated notations $|v\rangle|w\rangle, |v, w\rangle$ or $|vw\rangle$ for the tensor product $|v\rangle \otimes |w\rangle$.

- For an arbitrary scalar z and elements of $|v\rangle$ of V and $|w\rangle$ of W.

$$z(|v\rangle \otimes |w\rangle) = (z\,|v\rangle) \otimes |w\rangle = |v\rangle \otimes (z\,|w\rangle)$$

- For arbitrary $|v_1\rangle$ and $|v_2\rangle$ in V and $|w\rangle$ in W,

$$(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$$

- For arbitrary $|v\rangle$ in V and $|w_1\rangle$ and $|w_2\rangle$ in W,

$$|v\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle$$

Example A.2.1. if V is a two-dimensional vector space with basis vectors $|0\rangle$ and $|1\rangle$ then $|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle$ is an element of $V \otimes V$.

Linear operators acting on the space $V \otimes W$. Suppose $|v\rangle$ and $|w\rangle$ are vectors in V and W, and A and B are linear operators on V and W respectively. Then we can deined a linear operator $A \otimes B$ on $V \otimes W$ by the equation

$$(A \otimes B)(|v\rangle \otimes |w\rangle) = A\,|v\rangle \otimes B\,|w\rangle$$

It can be shown that $A \otimes B$ is a well-defined linear operator on V *otimes*W. The definition of $A \otimes B$ is then exteneded to all the elements of $V \otimes W$ in the natural way to ensure linearity of $A \otimes B$, that is,

$$(A \otimes B)(\sum_i a_i\,|v_i\rangle \otimes |w_i\rangle) = \sum_i a_i A\,|v_i\rangle \otimes B\,|w_i\rangle$$

Consider the case where $A : V \to V'$ and $B : W \to W'$ are linear operators on the vector spaces V and W. Then, an arbitrary linear operator C mapping $V \otimes W$ to $V' \otimes W'$ can be represented as a linear combination of tensor products of operators mapping V to V' and W to W'. That is,

$$C = \sum_i c_i A_i \otimes B_i$$

where by definition,

$$\left(\sum_i c_i A_i \otimes B_i\right) |v\rangle \otimes |w\rangle = \sum_i c_i A_i\,|v\rangle \otimes B_i\,|w\rangle$$

The inner product on the spaces V and W can be used to define a natural inner product on $V \otimes W$. Define

$$\left(\sum_i a_i \left| v_i \right\rangle \otimes \left| w_i \right\rangle , \sum_j b_j \left| v'_j \right\rangle \otimes \left| w'_j \right\rangle \right) \equiv \sum_{ij} a_i^* b_j \left\langle v_i | v'_j \right\rangle \left\langle w_i | w'_j \right\rangle$$

It can be shown that the function so defined is a well-defined inner product. The inner product space $V \otimes W$ inherits the other structure we are familiar with such as notions of an adjoint , unitary, normality and Hermiticity.

Kronecker Product
Suppose A is a m by n matrix, and B is a p by q matrix. Then we have the matrix representation:

$$A \otimes B \equiv \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix}$$

In this representation terms like $a_1 1B$ denote p by q matrices whose entries are proportional to B, with overall proportionality constant a_{11}.

Useful notation: $|\psi\rangle^{\otimes n}$ denotes ψ tensored with itself k times. For example, $|\psi\rangle^{\otimes 2} = \psi \otimes \psi$. Similar notation can be used for operators/matrices. Properties of Tensor Products:

- Transpose, Complex conjugation and adjoint operations distributed over the tensor product,
$$(A \otimes B)^T = A^T \otimes B^T$$
$$(A \otimes B)^* = A^* \otimes B^*$$
$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$$
- Tensor product of two unitary operators is unitary.
- Tensor product of two positive operators is positive.
- Tensor product of two projectors is a projector.

Example A.2.2. Consider one qubit $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$. Then the tensor product of $|\psi\rangle$ with itself is:

$$|\psi\rangle \otimes |\psi\rangle = (\alpha |0\rangle + \beta |1\rangle) \otimes (\alpha |0\rangle + \beta |1\rangle) = \alpha^2 |00\rangle + \alpha\beta |01\rangle + \alpha\beta |10\rangle + \beta^2 |11\rangle$$

Thus, in matrix form:

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\otimes\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\begin{pmatrix}\alpha\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\\ \beta\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\end{pmatrix}=\begin{pmatrix}\alpha^2\\ \alpha\beta\\ \alpha\beta\\ \beta^2\end{pmatrix}$$

Example A.2.3. Consider the matrix $A=\begin{pmatrix}1&2\\3&4\end{pmatrix}$ and $B=\begin{pmatrix}5&6\\7&8\end{pmatrix}$. Then the Kronecker product of A and B is:

$$A\otimes B=\begin{pmatrix}1\begin{pmatrix}5&6\\7&8\end{pmatrix}&2\begin{pmatrix}5&6\\7&8\end{pmatrix}\\ 3\begin{pmatrix}5&6\\7&8\end{pmatrix}&4\begin{pmatrix}5&6\\7&8\end{pmatrix}\end{pmatrix}=\begin{pmatrix}5&6&10&12\\7&8&14&16\\15&18&20&24\\21&24&28&32\end{pmatrix}$$

Example A.2.4. Consider $|\psi\rangle=\iota|0\rangle+7|1\rangle$ and $|\phi\rangle=|00\rangle+3|10\rangle+7|11\rangle$. Then, the tensor product $|\psi\rangle\otimes|\phi\rangle=|\psi\rangle|\phi\rangle=|\psi\phi\rangle=(\iota|0\rangle+7|1\rangle)(|00\rangle+3|10\rangle+7|11\rangle)=\iota|0\rangle|00\rangle+3\iota|0\rangle|10\rangle+7\iota|0\rangle|11\rangle+7|1\rangle|00\rangle+21|1\rangle|10\rangle+49|1\rangle|11\rangle=\iota|000\rangle+3\iota|010\rangle+7\iota|011\rangle+7|100\rangle+21|110\rangle+49|111\rangle$. Thus, in matrix form:

$$\begin{pmatrix}\iota\\7\end{pmatrix}\otimes\begin{pmatrix}1\\0\\3\\7\end{pmatrix}=\begin{pmatrix}\iota\begin{pmatrix}1\\0\\3\\7\end{pmatrix}\\ 7\begin{pmatrix}1\\0\\3\\7\end{pmatrix}\end{pmatrix}=\begin{pmatrix}\iota\\0\\3\iota\\7\iota\\7\\0\\21\\49\end{pmatrix}$$

Example A.2.5. Say $|\alpha\rangle=3|0\rangle+\iota|1\rangle$ and $|\beta\rangle=|00\rangle+2|10\rangle+7|11\rangle$ the the outer product $|\alpha\rangle\langle\beta|$ is given as:

$$|\alpha\rangle\langle\beta|=(3|0\rangle+\iota|1\rangle)(\langle 00|+2\langle 10|+7\langle 11|)$$

$$=3|0\rangle\langle 00|+6|0\rangle\langle 10|+21|0\rangle\langle 11|+\iota|1\rangle\langle 00|+2\iota|1\rangle\langle 10|+7\iota|1\rangle\langle 11|$$

In matrix form, it can be written as:

$$\begin{pmatrix}3&0&6&21\\ \iota&0&2\iota&7\iota\end{pmatrix}$$

Example A.2.6. Say a matrix $A = \begin{pmatrix} 0 & 1 \\ 3 & \iota \\ 7 & 0 \\ 0 & 13 \end{pmatrix}$. Then in the outer product form it can be written as:

$$A = |00\rangle\langle 1| + 3\,|01\rangle\langle 0| + \iota\,|01\rangle\langle 1| + 7\,|10\rangle\langle 0| + 13\,|11\rangle\langle 1|$$

Factorizing Tensor Products

Consider a $|\psi\rangle = \frac{1}{2}|00\rangle - \frac{\iota}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle$, then we can factorize the tensor product by following the following procedure. Assume that the $|\psi\rangle = |\phi\rangle|\chi\rangle$ where $|\phi\rangle = a|0\rangle + b|1\rangle$ and $|\chi\rangle = c|0\rangle + d|1\rangle$ are vectors in the spaces V and W respectively. Then we can write:

$$|\psi\rangle = |\phi\rangle\otimes|\chi\rangle = |\phi\rangle|\chi\rangle = |\phi\chi\rangle = (a|0\rangle+b|1\rangle)(c|0\rangle+d|1\rangle) = ac|00\rangle+ad|01\rangle+bc|10\rangle+bd|11\rangle$$

Now equating it with $|\psi\rangle$ we get:

$$|\psi\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle = \frac{1}{2}|00\rangle - \frac{\iota}{2}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Thus,

$$ac = \frac{1}{2}, ad = 0, bc = -\frac{\iota}{2}, bd = \frac{1}{\sqrt{2}}$$

Thus, $a = \frac{1}{\sqrt{2}}, b = 0, c = 1, d = \frac{1}{\sqrt{2}}$. Thus, the tensor product can be factorized as $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle|1\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle$.

Consider a $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$, then we can factorize the tensor product by foolowing the following procedure. Assume that the $|\psi\rangle = |\phi\rangle|\chi\rangle$ where $|\phi\rangle = a|0\rangle + b|1\rangle$ and $|\chi\rangle = c|0\rangle + d|1\rangle$ are vectors in the spaces V and W respectively. Then we can write:

$$|\psi\rangle = |\phi\rangle\otimes|\chi\rangle = |\phi\rangle|\chi\rangle = |\phi\chi\rangle = (a|0\rangle+b|1\rangle)(c|0\rangle+d|1\rangle) = ac|00\rangle+ad|01\rangle+bc|10\rangle+bd|11\rangle$$

Now equating it with $|\psi\rangle$ we get:

$$|\psi\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$$

Thus,

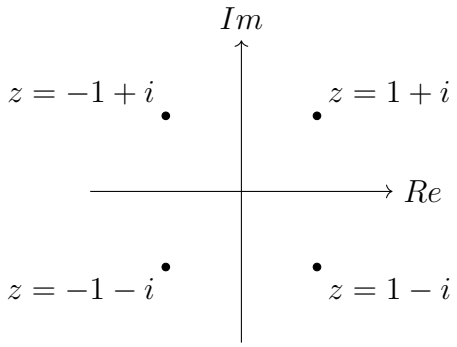
$$ac = \frac{1}{\sqrt{2}}, ad = 0, bc = 0, bd = \frac{1}{\sqrt{2}}$$

Thus, no such values of a,b, c and d exist. Hence, it cannot be factorized.

A.3 Complex Numbers

A.3.1 Complex Plane

The complex plane is as denoted in the figure below:



With y axis as imaginary axis, and x axis as real axis. A complex number in cartesian coordinate form is denoted as $z = a + \iota b$ where $\iota = \sqrt{-1}$ and $a, b \in \mathbb{R}$ which can be written as a column vector $z = \begin{pmatrix} a \\ b \end{pmatrix}$.

A.3.2 Addition of Complex Numbers

The addition of complex numbers is as follows:

$$z_1 + z_2 = (a_1 + \iota b_1) + (a_2 + \iota b_2) = (a_1 + a_2) + \iota(b_1 + b_2)$$

Thus, the addition of complex numbers is the same as the addition of vectors in the complex plane.
Properties:

- Commutative: $z_1 + z_2 = z_2 + z_1$
- Associative: $(z_1 + z_2) + z_3 = z_1 + (z_2 + z_3)$
- Identity: $z + 0 = z$ where $0 = 0 + 0\iota$
- Inverse: $z + (-z) = 0$
- Distributive: $z_1(z_2 + z_3) = z_1 z_2 + z_1 z_3$

A.3.3 Multiplication of Complex Numbers

The multiplication of complex numbers is as follows:

$$z_1 z_2 = (a_1 + \iota b_1)(a_2 + \iota b_2) = a_1 a_2 - a_1 b_2 + \iota(a_1 b_2 + a_2 b_1)$$

Thus, the multiplication of complex numbers is the same as the multiplication of vectors in the complex plane.
Properties:

- Commutative: $z_1 z_2 = z_2 z_1$
- Associative: $(z_1 z_2) z_3 = z_1 (z_2 z_3)$
- Identity: $z1 = 1z = z$
- Distributive: $z_1(z_2 + z_3) = z_1 z_2 + z_1 z_3$
- Scalar Multiplication: $\alpha(z_1 z_2) = \alpha z_1 z_2$
- Scalar Multiplication: $(\alpha + \beta)z = \alpha z + \beta z$
- Scalar Multiplication: $\alpha(\beta z) = (\alpha\beta)z$
- Existence of multiplicative idenetity: $1z = z$
- Existence of Multiplicative inverse: $z \neq 0 \implies \frac{1}{z} = \frac{z*}{z\bar{z}}$

A.3.4 Complex Conjugate

The complex conjugate of a complex number $z = a + \iota b$ is denoted by $\bar{z} = a - \iota b$. The complex conjugate of a complex number is the reflection of the complex number about the real axis.
Properties:

- $(z_1 + z_2)^* = z_1^* + z_2^*$
- $(z_1 z_2)^* = z_1^* z_2^*$
- $(z^*)^* = z$
- $z + z^* = 2Re(z)$

- $z - z^* = 2iIm(z)$
- $zz^* = |z|^2$
- $\frac{1}{z} = \frac{z^*}{zz^*}$
- $z^{-1} = \frac{z^*}{|z|^2}$

A.3.5 Modulus of a Complex Number

The modulus of a complex number $z = a + \iota b$ is denoted by $|z| = \sqrt{a^2 + b^2}$. The modulus of a complex number is the distance of the complex number from the origin in the complex plane.
Properties:

- $|z|^2 = zz^*$
- $|z_1 z_2| = |z_1||z_2|$
- $|z_1 + z_2| \leq |z_1| + |z_2|$
- $|z_1 - z_2| \geq ||z_1| - |z_2||$
- $|z_1 - z_2|^2 = |z_1|^2 + |z_2|^2 - 2Re(z_1 z_2^*)$
- $|z_1 + z_2|^2 = |z_1|^2 + |z_2|^2 + 2Re(z_1 z_2^*)$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 2(|z_1|^2 + |z_2|^2)$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 4|z_1|^2|z_2|^2$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 4|z_1 z_2|^2$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 4|z_1|^2|z_2|^2$

A.3.6 Polar form

The polar form of a complex number $z = a + \iota b$ is denoted by $z = r(\cos(\theta) + \iota \sin(\theta))$ where $r = |z|$ and $\theta = \tan^{-1}(\frac{b}{a})$. The polar form of a complex number is the representation of the complex number in terms of its modulus and argument.
Properties:

- $z = r(\cos(\theta) + \iota \sin(\theta)) = r(\cos(\theta + 2\pi n) + \iota \sin(\theta + 2\pi n))$ where n is an integer.
- $z_1 z_2 = r_1 r_2(\cos(\theta_1 + \theta_2) + \iota \sin(\theta_1 + \theta_2))$
- $\frac{1}{z} = \frac{1}{r}(\cos(-\theta) + \iota \sin(-\theta))$
- $z^n = r^n(\cos(n\theta) + \iota \sin(n\theta))$

A.3.7 Euler's Formula

$z = a + \iota b = re^{\iota\theta}$. Recall that the expansion of e^x using taylor series can be written as:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{x^n}{n!}$$

Now substituting $\iota\theta$ in place of x, we get:

$$e^{\iota\theta} = 1 + \iota\theta + \frac{(\iota\theta)^2}{2!} + \frac{(\iota\theta)^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{(\iota\theta)^n}{n!}$$

$$e^{\iota\theta} = 1 + \iota\theta - \frac{\theta^2}{2!} - \iota\frac{\theta^3}{3!} + \ldots = \cos(\theta) + \iota \sin(\theta)$$

On rearranging the terms we get:

$$e^{\iota\theta} = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \ldots + \iota(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \ldots)$$

Now recall that the expansion of sin and cos using taylor series expansion are:

$$\sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \ldots = \sum_{i=0}^{\infty} \frac{(-1)^i \theta^{2i+1}}{(2i+1)!}$$

$$\cos(\theta) = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \ldots = \sum_{i=0}^{\infty} \frac{(-1)^i \theta^{2i}}{(2i)!}$$

Thus, Euler's formula states that $e^{\iota\theta} = \cos(\theta) + \iota\sin(\theta)$. Hence, on subtituting this in the polar form of a complex number we get:

$$z = a + \iota b = \sqrt{a^2 + b^2}(\cos\left(\tan^{-1}\frac{b}{a}\right) + \iota\sin\left(\tan^{-1}\frac{b}{a}\right)) = r(\cos(\theta) + \iota\sin(\theta)) = re^{\iota\theta}$$

where $r = |z|$ and $\theta = \tan^{-1}(\frac{b}{a})$. Using, this we can also write:

$$\cos\theta = \frac{e^{\iota\theta} + e^{-\iota\theta}}{2}$$

$$\sin\theta = \frac{e^{\iota\theta} - e^{-\iota\theta}}{2\iota}$$

A.3.8 Roots of Unity

The nth roots of unity are the solutions to the equation $z^n = 1$ (recall that in the complex numbers, htere exist n solutions to the equation, for example, if n=2, z could be 1 or -1. If n=4, z could be $1, \iota, -1, -\iota$). These roots can be written as powers of $\omega = e^{2\pi\iota/n}$, which is called the primitive nth root of unity. The nth roots of unity are given by:

$$\omega_n^k = e^{\frac{2\pi k\iota}{n}}$$

where k is an integer such that $0 \leq k \leq n-1$. The nth roots of unity are equally spaced around the unit circle in the complex plane (it can be seen that ω lies on the units circle so $|\omega| = 1$, and the angle made by the root ω is the angle $\phi = 2\pi/n$, and squaring ω, we double the angle. thus, raising to the jth power, ω^j has a phase angle $\phi = 2j\pi/n$ and is still an nth root of unity.).
Properties:

- $\omega_n^k = \omega_n^{(k \bmod n)}$ where k mod n is the remainder when k is divided by n. For example, the 4th roots of unity are $\omega_4^0 = e^{\frac{2\pi\iota 0}{4}} = 1, \omega_4^1 = e^{\frac{2\pi\iota 1}{4}} = i, \omega_4^2 = e^{\frac{2\pi\iota 2}{4}} = -1, \omega_4^3 = e^{\frac{2\pi\iota 3}{4}} = -i$. Note: The roots are periodic with a period of n. Here, $n = 4$.

A.3.9 Discrete Fourier Transform Matrix

The Discrete Fourier Transform (DFT) matrix is a square matrix whose entries are the nth roots of unity. The i,jth elemenet of the matrix is given by ω_n^{ij}. The DFT

matrix of size n is given by:

$$F_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix}$$

The DFT matrix is a unitary matrix, that is, $F_n^\dagger F_n = I$. The DFT matrix is used to perform the Discrete Fourier Transform which is used in signal processing, image processing, data compression, etc.

Example A.3.1. Transform $|\psi\rangle = \frac{|0\rangle+|3\rangle}{\sqrt{2}}$ using F_4.

$$F_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \iota & -1 & \iota \\ 1 & -1 & 1 & -1 \\ 1 & \iota & -1 & \iota \end{pmatrix}$$

$$|\psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$F_4 |\psi\rangle = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \iota & -1 & \iota \\ 1 & -1 & 1 & -1 \\ 1 & \iota & -1 & \iota \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{2\sqrt{2}} \begin{bmatrix} 2 \\ 1-\iota \\ 0 \\ 1+\iota \end{bmatrix}$$

$$= \frac{2|0\rangle + (1-\iota)|1\rangle + (1+\iota)|3\rangle}{2\sqrt{2}}$$

Example A.3.2. Write QFT_2. Because $n = 2, \omega_2 = e^{\frac{2\pi\iota}{2}} = -1$. Thus, the QFT matrix of size 2 is given by:

$$QFT_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & \omega \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

which is clearly equal to H, the hadamard matrix. Write QFT_4. Because $n = 4, \omega_4 = e^{\frac{2\pi\iota}{4}} = i$. Thus, the QFT matrix of size 4 is given by:

$$QFT_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

which is clearly equal to $H^{\otimes 2}$.

Example A.3.3. Find the quantum fourier transform for $n = 4$ of the functions $|f\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle) = \frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$, $|g\rangle = |0\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ and $|h\rangle = |1\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$.
Their correspondiong QFT are given by:

$$QFT_4\,|f\rangle = \frac{1}{4}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{pmatrix}\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \frac{1}{4}\begin{pmatrix}4\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$QFT_4\,|g\rangle = \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}0.5\\0.5\\0.5\\0.5\end{pmatrix}$$

$$QFT_4\,|h\rangle = \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\i\\-1\\-i\end{pmatrix} = \begin{pmatrix}0.5\\0.5i\\-0.5\\-0.5i\end{pmatrix}$$

In examples, note that the columns of QFT_4 are orthogonal i.e. their inner produt is zero and along with that the 2-norm of the columns is 1. Thus, the QFT matrix is unitary.

Note that the vectors like $|f\rangle$ that had fewer zeros (narrow spread) had fourier transforms with a lot of zeros (larger spread), and vice-versa.

Finally, in examples, notice how the only difference between the foueir transforms of $|g\rangle$ and $|h\rangle$ is a difference of realtive phase shifts.

Thus, the fourier transform takes the vector $\begin{pmatrix}\alpha_0\\ \alpha_1\\ \vdots\\ \alpha_{n-1}\end{pmatrix}$ to the vector $\begin{pmatrix}\beta_0\\ \beta_1\\ \vdots\\ \beta_{n-1}\end{pmatrix}$ where $\beta_k = \frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}\omega_n^{kj}\alpha_j$. Quantum Fourier transform on measurement we would get one of the values of this superposition with probability $|\alpha|^2$ where α is the coefficient of the state in the superposition. The quantum Fourier transform is used in quantum

algorithms such as Shor's algorithm for factoring large numbers and the quantum phase estimation algorithm for estimating the eigenvalues of unitary operators.

Properties of QFT

- **QFT is unitary**

  Proof. To prove that QFT is a unitary matrix we are required to prove that any two columns of QFT matrix are orthonormal i.e.

  $$\langle C_i|C_j\rangle = \delta_{ij}$$

  where C_i and C_j are the columns of the QFT matrix.
  Let C_i and C_j be the ith and jth columns of the QFT matrix. Then, the inner product of the two columns is:

  $$\langle C_i|C_j\rangle = \frac{1}{n}\sum_{k=0}^{n-1}\omega_n^{-ik}(\omega_n^j)^k$$

  $$= \frac{1}{n}\sum_{k=0}^{n-1}\omega_n^{(j-i)k}$$

  If $i = j$, then the sum is equal to n i.e. $\langle C_i|C_j\rangle = 1$. If $i \neq j$, then the sum is equal to zero. Here, for the case $i \neq j$ it forms a geometric series. Recall, that the sum of a geometric series is given by $S_n = \frac{a(r^n-1)}{r-1}$ where a is the first term, r is the common ratio and n is the number of terms. Here $a = 1, r = \omega^{i-j}, n = n$. Thus, the sum of the geometric series is:

  $$S_n = \frac{\omega_n^{(j-i)n} - 1}{\omega_n^{j-i} - 1}$$

  Using the property that $\omega_n^{ln} = 1$, for some integer l. Thus, the sum of the geometric series is:

  $$S_n = \frac{1-1}{\omega_n^{j-i} - 1} = 0$$

  Thus, the inner product of the two columns is zero. Thus, the QFT matrix is a unitary matrix.

  because, the Foruier transform is a unitary opeator, we can implement it in a quantum circuit. Thus is $N = 2^n$, we can apply the Fourier transform $QFT_N = QFT_{2^n}$ to a n-qubit system. □

- **Linear Shift:** QFT Matrix when acts on a linearly shifted state vector causes a phase shift in its Fourier transform. Mathematically $|f(x)\rangle$, $x \in \mathbb{Z}_{2^n}$ has Fourier transform $|\hat{f}(x)\rangle$, then $|f(x+j)\rangle$ has Fourier transform $|\hat{f}(x)\rangle\, e^{\frac{2\pi xj}{2^n}}$. Also since QFT_{2^n} is unitary and $QFT_{2^n} QFT_{2^n}^{\dagger} = I$, the converse is true. A linear phase shift on $|f\rangle$ produces a linear shift in $ket\hat{f}(x)$. Consider the following QFT Matrix of size n:

$$F_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix}$$

Let $|\psi\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{n-1} \end{pmatrix}$ be a vector of size n. Then, the QFT of $|\psi\rangle$ is given by:
Say, when F_n acts on $|\psi\rangle$ we get,

$$F_n |\psi\rangle = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{n-1} \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \end{pmatrix}$$

So now both input and output vectors are in superposition state. Thus, on measuring the output we get, $|k\rangle$ with probability $|\beta_k|^2$. Now, we linearly shft the vector $|\psi\rangle$ by say 1 position. Thus, the new vector will be as shown:

$$|\psi'\rangle = \begin{pmatrix} \psi_{n-1} \\ \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-3} \\ \psi_{n-2} \end{pmatrix}$$

Now, when F_n acts on $|\psi'\rangle$ we get,

$$F_n|\psi'\rangle = \frac{1}{\sqrt{n}}\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \psi_{n-1} \\ \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-3} \\ \psi_{n-2} \end{pmatrix} = \begin{pmatrix} \omega_n^0\beta_0 \\ \omega_n^1\beta_1 \\ \omega_n^2\beta_2 \\ \vdots \\ \omega_n^{n-2}\beta_{n-2} \\ \omega_n^{n-1}\beta_{n-1} \end{pmatrix}$$

Thus, upon measurement we get $|k\rangle$ with probability $|\omega^k\beta_k|^2 = |\omega^k|^2|\beta_k|^2 = |\beta_k|^2$, since the modulus of $|\omega_k| = 1$. Thus, the QFT matrix when acts on a linearly shifted vector introduces a phase shift in the output.

Proof. This can be shown as follows: Consider the β_k before the linear shift is:

$$\beta_k = \frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}\omega_n^{jk}\psi_j$$

Now, after linear shifting input by m we get:

$$\beta'_k = \frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}\omega_n^{jk}\psi_{j-m} = \frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}\omega_n^{k(j+m)}\psi_j$$

Now we are required to show that this output is the same as the previous output but with a phase shift to it. Thus, upon multiplying the β_k by ω_n^{mk} we get:

$$\omega_n^{mk}\beta_k = \frac{1}{\sqrt{n}}\sum_{j=0}^{n-1}\omega_n^{(j+m)k}\psi_j = \beta'_k$$

Thus, we proved that the QFT matrix when acts on a linearly shifted vector introduces a phase shift in the output. □

Example A.3.4. For $N = 2^2 = 4$, let $|\Theta\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}$ and $|\Phi\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_0 \end{pmatrix}$.

$$QFT_4|\Theta\rangle = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}$$

$$QFT_4 |\Phi\rangle = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_0 \end{pmatrix} = \begin{pmatrix} \beta_0 \\ -\iota\beta_1 \\ -\iota\beta_2 \\ \iota\beta_3 \end{pmatrix}$$

The only difference between $|\Theta\rangle$ and $|\Phi\rangle$ is a relative phase shift. Note that it doesn't mattter if we are going to measure a state, then the phases don't matter at all, because if the phase is $e^{\iota\theta}$, then upon measurement $|e^{\iota\theta}| = 1$. Therefore the phase of a given state does not effect the probability of measuring that state. In order to gather information about the phases, consider the following example. Consider the states $|\Theta\rangle = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ and $|\Phi\rangle = \frac{1}{2}\begin{pmatrix} 1 \\ \iota \\ -1 \\ -\iota \end{pmatrix}$. We can't tell the difference by measuring the states. Upon applying QFT_4. We get:

$$QFT_4 |\Theta\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$QFT_4 |\Phi\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Thus, measuring the Fourier Transform of the states will reveal the relative phases.

- **Period and Wavelength Relationship:**
Consider a periofic function $f : \{0,1\}^n \to \{0,1\}^n$ with a period r which takes n bits as input and n bits as output. Now we apply a Quantum Fourier Transformation on it which is Discrete Fourier Transform.

$$QFT_{2^n} f(x) = \hat{f}(x)$$

where $\hat{f}(x)$ is the Fourier Transform of the function f. Now, the period of the function f is the smallest positive integer r such that $f(x) = f(x + r)$ for all x. This is called the period of the function f and is denoted by r. The wavelength of the Fourier Transform of the function f is the smallest positive integer s such that $\hat{f}(x) = \hat{f}(x + s)$ for all x. The wavelength of the Fourier Transform of

the function f is denoted by s. The period and wavelength of the function f are related by the equation:

$$s = \frac{2^n}{r}$$

where n is the number of bits in the input/output of the function f. Thus, in matrix form we can write it as:

$$\frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_r \\ \psi_0 \\ \psi_1 \\ \vdots \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{\frac{2^n}{r}-1} \\ \beta_0 \\ \beta_1 \\ \vdots \end{pmatrix}$$

This, can be as shown in the figure A.1.

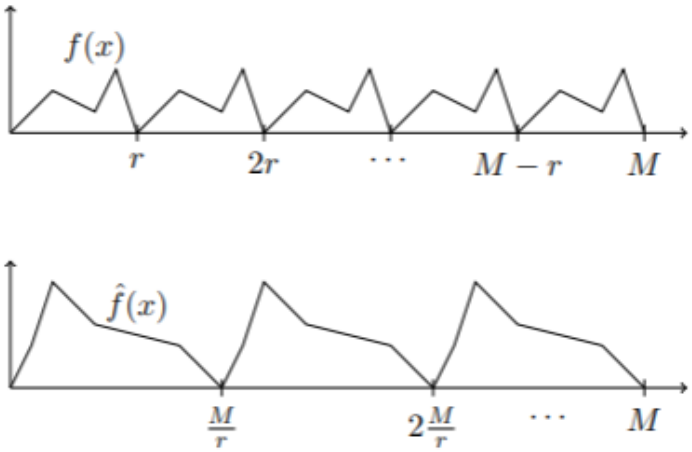


Figure A.1: Period and Wavelength Relationship

In general, the wider the range of a function, the sharper the range in the Fourier domain; and vice versa. In example, the fourier transform of a delta

funcition is an even spread, while the transform of an even spread is a delta function. Consider the case where,

$$f(x) = \begin{cases} \sqrt{\frac{r}{2^n}} & \text{if } x = 0 \text{ (mod r)} \\ 0 & \text{otherwise} \end{cases}$$

This is what is used in Shor's algorithm.

Proof. Conisder the following figure A.2 and A.3 ($M = 2^n$).

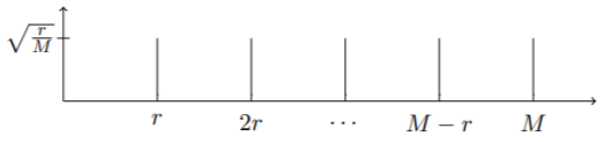


Figure A.2: f(x)

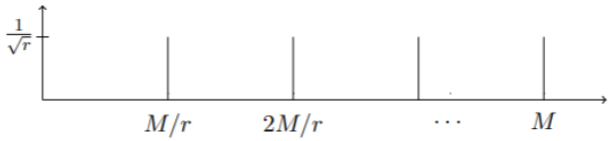


Figure A.3: QFT of f(x)

Suppose $|\alpha\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2^n-1} \end{pmatrix}$ has the Fourier transform $|\beta\rangle = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{2^n-1} \end{pmatrix}$, then the jth component of its Fourier trasnform is given by (recall that n is the number of Qubits):

$$\beta_j = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \omega_{2^n}^{jk} \alpha_k$$

Now in our case,

$$f(x) = \begin{cases} \sqrt{\frac{r}{2^n}} & \text{if } x = 0 \text{ (mod r)} \\ 0 & \text{otherwise} \end{cases}$$

Thus, we calculate the Fourier Transform as:

$$\hat{f}(x) = \frac{\sqrt{r}}{2^n} \sum_{i=0}^{\frac{2^n}{r}-1} \omega^{rix}$$

where we have written $\hat{f}(x)$ instead of $\hat{f}(j)$, because the $\hat{f}(j)$ is equal to the Fourier transform evaluated at $x = j$. Using the formula for summation of geometric series. We get:

$$\hat{f}(x) = \frac{\sqrt{r}}{2^n} \sum_{i=0}^{\frac{2^n}{r}-1} \omega^{rix} = \frac{\sqrt{r}}{2^n} \frac{1-\omega^{x2^n}}{1-\omega^{rx}}$$

Recall that $\omega^{2^n j} = 1$ (because $\omega^{2^n} = 1$), so that the numerator is always equal to 0. So the only time the denominator is equal to zero is when $rx = k2^n$, or $x = \frac{k2^n}{r} \equiv 0(mod \frac{M2^n}{r})$. In this case, the numerator and denominator are both 0, so we must compute the limit using the l'Hospitals rule.

$$\lim_{x \to \frac{k2^n}{r}} \frac{1-\omega^{x2^n}}{1-\omega^{rx}} = \lim_{x \to \frac{k2^n}{r}} \frac{-2^n \omega^{x2^n-1}}{-r\omega^{rx-1}} = \lim_{x \to \frac{k2^n}{r}} \frac{2^n \omega^{x2^n}}{r\omega^{rx}} = \frac{2^n}{r}$$

Pluggin this result back, the outcome is

$$\hat{f}(x) = \begin{cases} \frac{1}{\sqrt{r}} & \text{if } x = 0 \pmod{\frac{2^n}{r}} \\ 0 & \text{otherwise} \end{cases}$$

This property can also be proced in general, imagin a periodic function in r of this tupe s a basis for any periodic function. Allow the possibility of relative phase shifts and we can thus prove this property in general. □

A.3.10 Classical Fast Fourier Transform

The Fourier transform is an $N \times N$ matrix, straightforward multiplication by F_N takes $\mathcal{O}(N^2)$ stpes to carry out, because multiplication of f on each row takes N multiplications. The FFT reduced this to $\mathcal{O}(N \log N)$ steps.The FFT required only that $N = 2^n$ for some integer n, but this is a relatively easy requirement becuase the computers can simply choose their domain.

The fast Fourier transform uses the symmetry of the Fourier transform to reduce the computation time. We rewrite the Fourier transform of size n as two fourier

transforms of size $N/2$ - the odd and the even terms. We then repeat this over and over agina to exponentially reduce the time. To see this work in detail, the matrix of the Fourier transform consider the QFT_8:

$$QFT_8 = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix}$$

Note that how row j and j+4 are similar and how column j and j+4 are similar. Thus, we split the fourier transform up nito even and odd columns. Consider the following figure A.4:

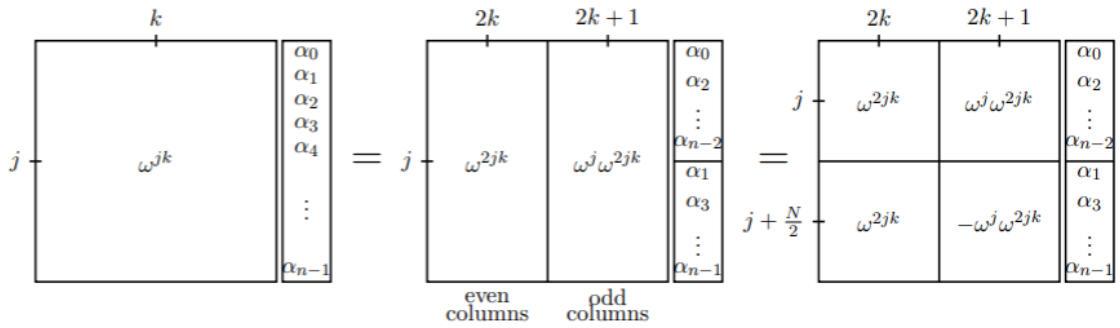


Figure A.4: FFT of size 8

In the first frame, we have represented the whole Fourier transform matrix, by describing the jth row and kth column: ω^{jk}. In the next frame, we separate the odd an even columns, and similarly spearate the vector that is to be transformed. In the third frame, we add a little symmetry by noticing that $\omega^{j+N/2} = -\omega^j$ (since $\omega^{N/2} = -1$).

Notice that both the odd side an even side contain the term ω^{2jk}. But if ω is the primitive Nth root of unity, then ω^2 is the primitive $N/2nd$ root of unity. Therefore, the matrices whose entires are j.kth entry is ω^{2jk} are really just $QFT_{N/2}$. Now we can write QFT_N is a new way:

Now suppose we are calculating the Fourier transform of the function $f(x)$. We can write the above manipulations as an equation taht computs the jth term $\hat{f}(j)$.

$$\hat{f}(j) = (F_{2^n/2}\vec{f_{even}})(j) + \omega^j(F_{2^n/2}\vec{f_{odd}})(j)$$

This turns our calculation of QFT_N into two applications of $QFT_{N/2}$. We can turn this into four applications of $QFT_{N/4}$, and so forth. As long as $N = 2^n$ for some n, we can break down our calculation of QFT_N into log N steps, each of which takes $\mathcal{O}(N)$ steps. Thus, the FFT takes $\mathcal{O}(N \log N)$ steps to compute the Fourier transform of a function of size N.

A.4 Probability Theory

A.5 Calculus

Appendix B

Classical Computations

The Turing Machine is an abstract model of computation that can perform any computation that can be performed by a digital computer. It is a mathematical model of computation that defines an abstract machine which manipulates symbols on a strip of tape according to a table of rules. The machine operates on an infinite memory tape divided into discrete cells. Each cell contains a symbol from a finite alphabet. The machine has a read/write head that can read the symbol on the tape and write a new symbol. The machine can move the tape left or right one cell at a time. The machine has a finite set of states and a table of rules that determine the next state of the machine based on the current state and the symbol read from the tape. The machine halts when it reaches a halting state. The Turing Machine can perform any computation that can be performed by a digital computer. It is a universal model of computation that can simulate any other computational model.

But in real world, the computers are finite in size. An alternative model of computation is the circuit model which is equivalent to the Turing Machine and is suitable for studying the complexity of algorithms. The circuit model consists of gates that perform logical operations on bits. The gates are connected by wires that carry the bits from one gate to another. The circuit model can be used to represent any computation that can be performed by a digital computer. The circuit model is a useful abstraction that allows us to reason about the efficiency of algorithms and design better algorithms.

Logic Gate: It is a function defined as $f : \{0,1\}^k \to \{0,1\}^l$ where k is the number of input bits and l is the number of output bits. Note that no loops are allowed in such circuits to aviod instabilitiyes. Thus, they are Acylcic circuits. This is the Circuit Model of Computation. Some of the useful Logic Gates are:

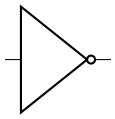
B.1 Logic Gates

B.1.1 NOT Gate

The NOT gate is a unary gate that takes a single input bit and produces a single output bit. The output bit is the complement of the input bit i.e. if the input bit is 0, the output bit is 1 and if the input bit is 1, the output bit is 0. The truth table for the NOT gate is:

Input	Output
0	1
1	0

The NOT gate is denoted by the symbol $\neg$. It's symbol is as shown:



Mathematically, the NOT gate is defined as:

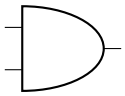
$$f(x) = \neg x$$

B.1.2 AND Gate

The AND gate is a binary gate that takes two input bits and produces a single output bit. The output bit is 1 if both input bits are 1 and 0 otherwise. The truth table for the AND gate is:

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

The AND gate is denoted by the symbol $\wedge$. It's symbol is as shown:



Mathematically, the AND gate is defined as:

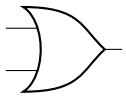
$$f(x, y) = x \wedge y$$

B.1.3 OR Gate

The OR gate is a binary gate that takes two input bits and produces a single output bit. The output bit is 1 if at least one of the input bits is 1 and 0 otherwise. The truth table for the OR gate is:

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

The OR gate is denoted by the symbol $\vee$. It's symbol is as shown:



Mathematically, the OR gate is defined as:

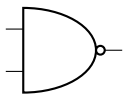
$$f(x, y) = x \vee y$$

B.1.4 NAND Gate

The NAND gate is a binary gate that takes two input bits and produces a single output bit. The output bit is the complement of the AND gate i.e. the output bit is 0 if both input bits are 1 and 1 otherwise. The truth table for the NAND gate is:

Input 1	Input 2	Output
0	0	1
0	1	1
1	0	1
1	1	0

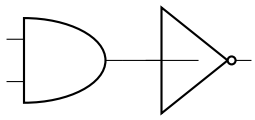
The NAND gate is denoted by the symbol $\uparrow$. It's symbol is as shown:



Mathematically, the NAND gate is defined as:

$$f(x, y) = \neg(x \wedge y)$$

NOte that the NAND Gate can also be written by adding a NOT gate at the output of the AND gate as shown:

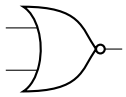


B.1.5 NOR Gate

The NOR gate is a binary gate that takes two input bits and produces a single output bit. The output bit is the complement of the OR gate i.e. the output bit is 0 if at least one of the input bits is 1 and 1 otherwise. The truth table for the NOR gate is:

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	0

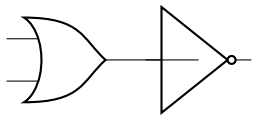
The NOR gate is denoted by the symbol $\downarrow$. It's symbol is as shown:



Mathematically, the NOR gate is defined as:

$$f(x, y) = \neg(x \vee y)$$

NOR Gate can also be written by adding a NOT gate at the output of the OR gate as shown:

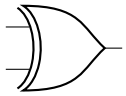


B.1.6 XOR Gate

The XOR gate is a binary gate that takes two input bits and produces a single output bit. The output bit is 1 if the input bits are different and 0 otherwise. The truth table for the XOR gate is:

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

The XOR gate is denoted by the symbol $\oplus$. It's symbol is as shown:



Mathematically, the XOR gate is defined as:

$$f(x, y) = x \oplus y$$

B.1.7 FANOUT Gate

The FANOUT gate is a unary gate that takes a single input bit and produces two output bits. The output bits are the same as the input bit. The truth table for the FANOUT gate is:

Input	Output 1	Output 2
0	0	0
1	1	1

Mathematically, the FANOUT gate is defined as:

$$f(x) = (x, x)$$

B.1.8 CROSSOVER Gate

The CROSSOVER gate is a binary gate that takes two input bits and produces two output bits. The output bits are the complement of the input bits i.e. if the input bits are (0,1), the output bits are (1,0) and if the input bits are (1,0), the output bits are (0,1). The truth table for the CROSSOVER gate is:

Input 1	Input 2	Output 1	Output 2
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

This is same as saying that the value of two bits are interchanged.

B.2 Classical Circuits

The Ancilla or work bits are the extra bits that are used in the computation to store the intermediate results. They are temporary storage devices that is used to store the intermediate results are are not a part of the inputs or the outuput bits of the computation. **These elements can be used to compute any function**. For example, a half adder circuit as shown in figure

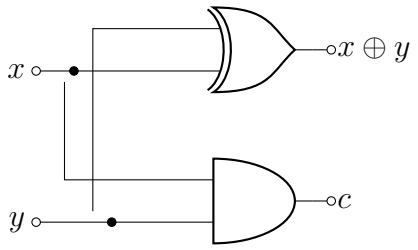


Figure B.1: Half Adder

We can use half-adders to make a full-adder as shown in figure: B.2 As shown in

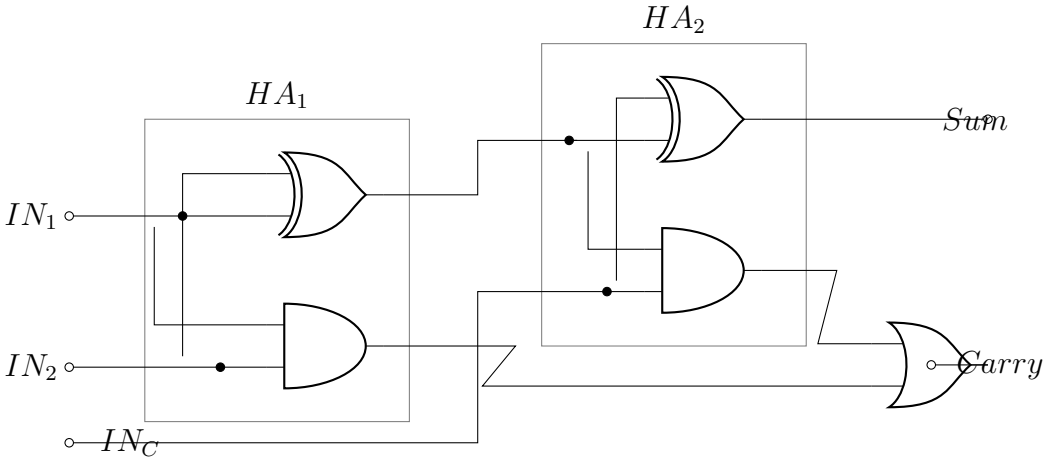


Figure B.2: Full Adder

the figure, the full adder is made up of two half adders and an OR gate. The carry bit from the first half adder is used as the input carry bit for the second half adder. The sum bits from both the half adders are fed to the OR gate to get the final sum bit. The carry bit from the second half adder is the final carry bit. By cascading, many of these full-adders together we obtain a circuit to add two n-bit integers.

Proof. Any fixed gates can be used to compute any function $f : \{0,1\}^n \to \{0,1\}^m$. Consider the proof for the simplified function $f : \{0,1\}^n \to \{0,1\}$ with n input bits and a single output bit. This is a Boolean function. Using mathematical indcution, For $n = 1$, there are four possible functions: the identity, which is a single wire; the bit flip, implemented using a NOT Gate; the function which replaces the input bit with a 0, which can be obtained using ANDing the input with a work bit initially in the 0 state; and the function which replaces the input bit with a 1, which can be obtained using ORing the input with a work bit initially in the 1 state. Thus, we have proved for n=1.

Now for some n, suppose that any function on n bits may be computed by a circuit and let f be a function on $n + 1$ bits. Define f_0 and f_1 to be the functions on n bits obtained by fixing the last bit of the input to 0 and 1 respectively. So $f_0 = f(x_1, x_2, \ldots, x_n, 0)$ and $f_1 = f(x_1, x_2, \ldots, x_n, 1)$. By the induction hypothesis, there are circuits C_0 and C_1 that compute f_0 and f_1 respectively. Thus, now we are required to design a circuit with computes f. Thus, we design a circuit that computes both f_0 and f_1 on the first n bits and then dependening on whether the first bit of the input was 0 or a 1 it outputs the appropriate answer. This can be done using a circuit as shown in figure B.3.

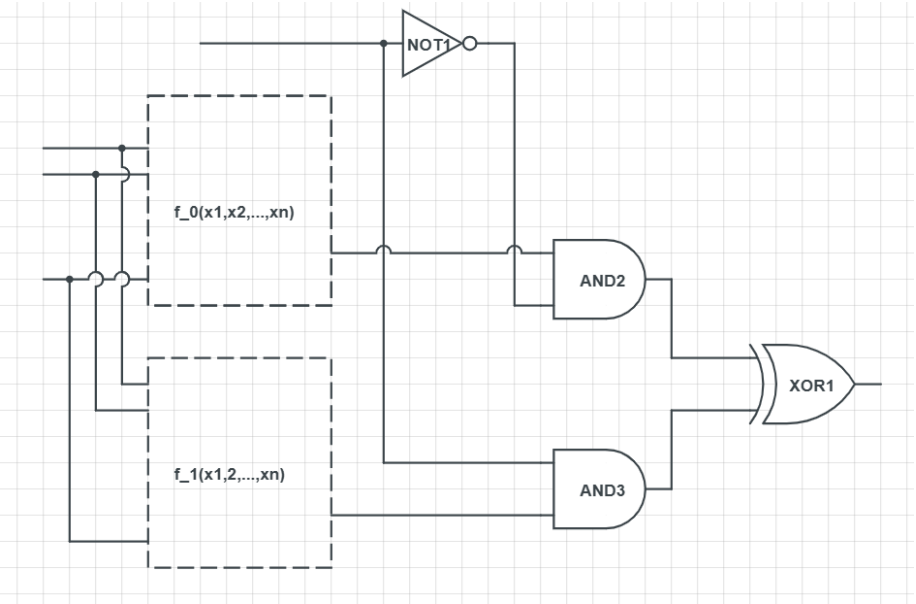


Figure B.3: Circuit to compute f

□

For the Universal Circuit Construction:

- wires, which preserve the states of the bits;
- ancilla bits prepared in the stnadard states, used in the n=1 case of the proof;
- FANOUT operation, which takes a single bits as input and outputs two copies of that bit;
- The CROSSOVER opertaion, which interchanges the values of the two bits;
- the AND, OR, and NOT operations, which are the basic logic gates.

In the case of quantum wirers for the preservation of qubits qubit state, it is not necessarily obvious that good quantum wires for the preservation of qubits can be constructed. The FANOUT operation cannot be performed because of the **No - cloning Theorem**. The AND and XOR Gates are not invertible thus irreversible and hence can't be implemented in a straight forward manner as unitary quantum gates. Thus, implementing a universal quantum circuit is not as straight forward as in the classical case.

Example B.2.1. NAND gate can be used to simulate AND, XOR and NOT gates provided wires, ancilla bits and FANOUT are available as shown. For implementing AND gate using NAND gates is shown in figure B.4

Figure B.4: NAND gate used to simulate AND, XOR and NOT gates

Similarly, we can implement XOR as shown in figure B.5 and NOT as shown in figure B.6.

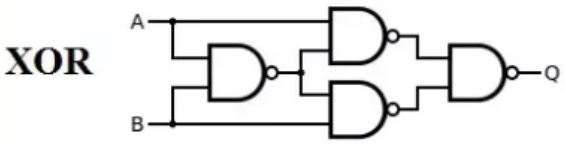


Figure B.5: NAND gate used to simulate XOR gate

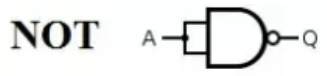


Figure B.6: NAND gate used to simulate NOT gate

Appendix C

Qiskit

At the time of writing this we are using Qiskit 1.x to develop/ The following content has been copied directly from Qiskit IBM documentation.

C.1 Introduction to Qiskit

The name "Qiskit" is a general term referring to a collection of software for executing programs on quantum computers. Most notable among these software tools is the **open-source Qiskit SDK**, and the **runtime environment** (accessed using Qiskit Runtime) through which you can execute workloads on IBM quantum computers. As quantum technology evolves, so does Qiskit, with new capabilities released every year that expands this core collection of quantum software.

In addition, many open-source projects are part of the broader Qiskit ecosystem. These software tools are not part of Qiskit itself, but rather interface with Qiskit and can provide valuable additional functionality. Parts of Qiskit environment are as shown in the figure C.1.

C.1.1 The Qiskit SDK

The Qisit SDK (package name qiskit) is an open-source SDK for working with quantum computers at the level of extended (static, dynamic, and scheduled) quantum circuits, operators and primitives. This library is the core component of Qiskit; it is the largest package under the Qiskit name with the broadest suit of tools for quantum computation, and many other components interface with it. Some of the most useful features of the Qiskit SDK include:

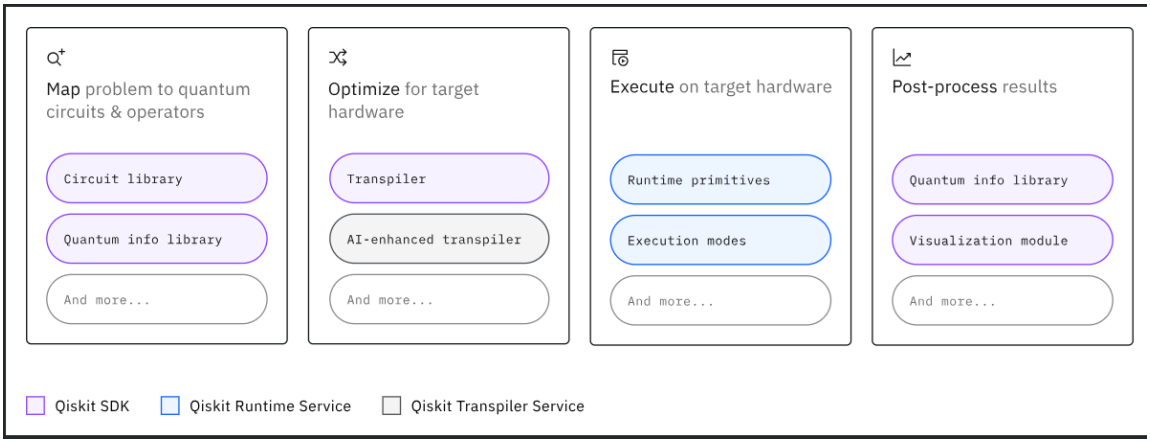


Figure C.1: Qiskit Steps

- Circuit-building tools (qiskit.circuit) - For initializing and manipulating registers, circuits, instructions, gates, parameters, and control flow objects.
- Circuit library (qiskit.circuit.library) - A vast range of circuits, instructions, and gates - key building blocks for circuit-based quantum computations.
- Quantum info library - (qiskit.quantum_info) - A toolkit for working with quantum states, operators and channels, using exact calculations (no sampling noise). Use this module to specify input observables and analyse fidelity of outputs from primitive queries.
- Transpiler (qiskit.transpiler) - For transforming and adapting quantum circuits to suit specific device topology, and optimizing for execution on real quantum systems.
- Primitives (qiskit.primtives) - The module that contains the base definitions and reference implementations of the **Sampler** and **Estimator** primitives, from which different quantum hardware providers can derive their own implmentations. See more information about the Qiskit runtime primitives in the documentation.

C.1.2 Qiskit Runtime

Qiskit Runtime is a cloud-based service for executing quantum computations on IBM Quantum hardware. The qiskit-ibm-runtime package is a client for that service,s an

is the successor of the Qiskit IBM provider. The Qiskit Runtime service streamlines quantum computations and provides optimal implementation of the Qiskit primitives for IBM Quantum hardware. To get started with Qiskit runtime primitives, visit the documentation.

With Qiskit Runtime you can choose to run your quantum programs on IBM Quantum hardware through the IBM Quantum Platform or IBM Cloud. See more information on selecting an IBM Quantum channel in the documentation.

Qiskit Runtime is designed to use additional classical and quantum computer resources, including techniques such as error suppression and error mitigation, to return a higher-quality result from executing quantum circuits on quantum processors. Examples include dynamical decoupling for error suppression, and readout mitigation and zero-noise extrapolation (ZNE) for error mitigation. Learn how to configure these options on the Configure error mitigation page.

Qiskit Runtime also includes three types of execution modes for running you quantum program on IBM hardware: Job, Session, and Batch, each of which have different use cases and implications for the quantum job queue. A Job is single query to a primitive that can be run over a specified number of shots. Sessions allow you to efficiently run multiple jobs in iterative workloads on quantum computers. Batch mode allows you to submit all your jobs at once for parallel processing.

Is Qiskit Runtime open-source?

The short answer is, not all of it. The Qiskit Runtime service software that handles the technicalities of running your quantum program on IBM Quantum device (includeing any error mitigation and suppression) is not open-source. However, the Qiskit Runtime client (the interface for users to access the Qiskit Runtime service), the Qiskit SDK running on the server side, and some of the software used for error mitigation, are open-source. To get involved with Qiskit open-source efforts, visit the GitHub organization at github.com/Qiskit and github.com/Qiskit-Extensions.

C.1.3 Qiskit Serverless

Creating utility-scale quantum applications generally requires a variety of computer resource requirements. Premium users can use Qiskit Serverless (package name qiskit-serverless) to easily submit quantum wrokflows for remote, managed execution. See more information here about how to use this collection of tools.

C.1.4 Qiskti Transpiler as a Service

The Qiskit transpiler service (package name qiskit-transpiler service) is a new experimental service that provides remote transpilation capabilities on the cloud to IBM Quantum Premium Plan users. In addition to the local Qiskit SDK transpiler capabilities, your transpilation tasks can benefit from both IBM Quantum cloud resources and AI-powered transpiler passes using this service. To learn more about how to integrate cloud-based transpilation into your Qiskit workflow you can check out the documentation.

C.1.5 The Qiskit ecosystem

Beyond Qiskit there are many open-source projects that use the "Qiskit" name but are not part of Qiskit itself; rather, theuy interface with Qiskit and can provide valuable additional functionality to supplement the core Qiskit workflow. Some of these projects are mintained by IBM Quantum teams, whereas others are supported by the broader open-source community. The Qiskit SDK is designed in a modular, extensible way to make it easy for developers to create projects like these that extend its capabilities. Some popular projects in the Qiskit ecosystem include:

- Qiskit (qiskit-aer) - a package for quantum computing simulators with realistic noise models. It provides interfaces to run quantum circuits with or without noise using multiple different simulation methods. Maintained by IBM Quantum.

- qBraid SDK (qbraid) - a platform-agnostic quantum runtime framework for both quantum software and hardware providers, designed to streamline the full lifecycle management of quantum jobs- from defining program specifications to job submission through to the post-prcoessing and visualization of results. Maintained by qBraid.

- mthree (mthree) - a package for implementing M3(Matrix-free Measurement Mitigation), a measurement mitigation technique that solves for corrected measurement probabilities using a dimensionality reduction step followed by either direct LU factorization or a preconditioned iterative method that nominally converges $\mathcal{O}(1)$ steps, and can be computed in parallel. Maintained by IBM Quantum.

You can find a catalog of projects in the Qiskit ecosystem page, as well as information about how to nominate your own project.

Please refer to the YouTube lectures by Derek Wang, research scientist at IBM for Coding with Qiskit 1.x. Qiskit version 1.x for utility scale for Quantum Computing. From installing Qiskit version1.x from scratch to learning how to run Quantum circuits both Unitary and dynamic based on some of the recent research papers by IBM Quantum. Using devices over 100 qubits using latest air suppression and mitigation techniques. Then learning how to contribute to the Qiskit ecosystem through open-source extra ordinate Abby Mitchell.

This is to get you up to speed with the latest developments on quantum computers for your own work. The foundation and focused on Qiskit 1 - an open source and freely available software development kit that allows you to program useful Quantum computational workflows. From designing Quantum Circuits and designing Quantum algorithms to submitting them to real Quantum Computers and orchestrating large scale complex workloads. Qiskit has now been for almost six years with previous iterations with Coding with Qiskit series. In the past few years with major developments in Quantum Computing most notably with the onset of the Utility era where Quantum Computers have now reached the level of scale and reliability such that researchers can now use them as legitimate tool for scientific exploration as highlighted by Research Nature paper by IBM Quantum.

In utility era, the Quantum computational workflows will grow only in scale and complexity. There are high immpact scientific papers that already use more than 100 qubtis on IBM's 127 qubits chips, with thousands of gates in the circuit and tens and thousands of circuits in a workflow. With the proliferation of methods like circuit cutting and error mitigation that require fluid orchestration between multiple classical and quantum computers. The technology has evolved dramatically. Qiskit 1.0 is highly performant - faster and takes up less memory making it convenient for faster transpiler circuits with 100 plus qubits and thousands of gates. It is learner, removed meta package architecture, spun off packages like qiskit.algorithms into a seperate one and deprecated modules like qiskit.opflow to make it more focused and maintainable. It is more stable. More features like Primitives in the form of sampler and estimator are now the access model for qunautm hardware. Improved upon backend.run method from pre 1.0 versions. Primitives ensure that access to quantum hardware is consistent and reliable. Qiskit 1.0 is ready for dynamic circuits - enabling complex logic within a Quantum circuit more efficiently generating entangled states, teleport long range gates and path toward error correction. It is now easier to build on Qiskit. Like build your own transpiler plugin. extend qiskit capabilities for your own needs and share your code throughout the Qiskit ecosystem.

C.2 Install Qiskit

Whether you will work locally or in a cloud environment, the first step for all users is to install Qiskit. For those wanting to run on a real system, your next step is to choose one of two channels in order to access IBM Quantum systems: IBM Quantum Platform or IBM Cloud.

Checkout the IBM Quantum Qiskit guide on how to install Qiskit and Qiskit Runtime and setup an IBM Quantum channel.

Bibliography

[1] Ronald de Wolf. Quantum computing: Lecture notes, 2023.

[2] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[3] Lin Lin. Lecture notes on quantum algorithms for scientific computation, 2022.

[4] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*, volume 2. Cambridge university press Cambridge, 2001.

[5] Ramamurti Shankar. *Principles of quantum mechanics*. Springer Science & Business Media, 2012.

[6] J Shewchuk. An introduction to the conjugate gradient method without the agonizing pain source. *Technical Report, Carnegie Mellon Univ.*, 1994.