

Quantum Computing

Nihar Shah

June 20, 2024

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 Appendix to get a good grasp on the maths used in the entire 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!

Contents

List of Figures	v
List of Tables	vii
Notations	viii
1 Introduction	1
1.1 Axioms of Quantum Mechanics	1
1.1.1 Postulate 1: The State Space Postulate	1
1.1.2 Postulate 2: Evolution Postulate	3
1.1.3 Postulate 3: Measurement Postulate	4
1.1.4 Postulate 4: Composite Systems	12
1.2 Bloch Sphere Representation of a Qubit	15
1.3 Single Qubit Gates	20
1.3.1 Pauli Matrices	20
1.3.2 Hadamard Gate	28
1.3.3 S and S^\dagger Gate	31
1.3.4 T and T^\dagger Gate	34
1.3.5 Quantum Rotation Gates	37
1.3.6 Universal Quantum Gates	40
1.4 Multi-Qubit Gates	41
1.4.1 Two-qubit Gates	41
1.4.2 Three-qubit Gates	51
1.5 Quantum Circuits	59
1.6 Entanglement	63
1.6.1 Bell States/EPR Pairs	64
1.6.2 Teleportation	67
1.6.3 Superdense Coding	70
1.7 Quantum Algorithms	75

1.7.1	Oracles	75
1.7.2	Elitzur-Vaidman Bomb Detection Algorithm	81
1.7.3	Deutsch's Algorithm	86
1.7.4	Deutsch-Josza Algorithm	90
1.7.5	Simon's Algorithm	97
1.7.6	Bernstein-Vazirani Algorithm	103
1.7.7	Shor's Algorithm	103
1.7.8	Grover's Algorithm	104
1.7.9	Quantum Fourier Transform	109
2	Advanced Quantum Computing	110
2.1	Linear Growth and Duhamel's Principle	110
2.2	Universal gate sets and Reversible Computation	112
2.3	Fixed Point Number Representation and Classical Arithmetic operations	115
2.4	Fault tolerant Computation	116
2.5	Complexity of Quantum Algorithms	116
A	Linear Algebra and Calculus Prerequisites	119
A.1	Linear Algebra	120
A.1.1	Basics of Linear Algebra	120
A.1.2	Inner Product	126
A.1.3	Gram-Schmidt Orthogonalization	129
A.1.4	Cauchy-Schwarz and Traingle Inequalities	130
A.1.5	Linear Operators and Matrix Representation	132
A.1.6	Hermitian and Unitary Operators	138
A.1.7	Eigen Value Problems	143
A.1.8	Norms	154
A.1.9	Induce Matrix Norm	155
A.1.10	Conclusions	156
A.2	Tensor Products	157
A.3	Complex Numbers	162
A.3.1	Complex Plane	162
A.3.2	Addition of Complex Numbers	162
A.3.3	Multiplication of Complex Numbers	163
A.3.4	Complex Conjugate	163
A.3.5	Modulus of a Complex Number	164
A.3.6	Polar form	165
A.3.7	Euler's Formula	165

A.3.8	Roots of Unity	166
A.3.9	Discrete Fourier Transform Matrix	166
A.4	Probability Theory	170
A.5	Calculus	170
B	Classical Computations	171
B.1	Logic Gates	172
B.1.1	NOT Gate	172
B.1.2	AND Gate	172
B.1.3	OR Gate	173
B.1.4	NAND Gate	173
B.1.5	NOR Gate	174
B.1.6	XOR Gate	174
B.1.7	FANOUT Gate	175
B.1.8	CROSSOVER Gate	175
B.2	Classical Circuits	176
	Bibliography	180

List of Figures

1.1	Bloch-Poincare Sphere	16
1.2	Bloch Sphere	18
1.3	Pauli-I Gate	21
1.4	Pauli-X Gate	22
1.5	Pauli-Y Gate	24
1.6	Pauli-Z Gate	26
1.7	Hadamard Gate	28
1.8	S Gate	32
1.9	S^\dagger Gate	33
1.10	T Gate	35
1.11	T^\dagger Gate	36
1.12	R_X Gate	38
1.13	R_Y Gate	38
1.14	R_Z Gate	39
1.15	U_3 Gate	40
1.16	CNOT Gate	42
1.17	CY Gate	44
1.18	CZ Gate	46
1.19	CH Gate	47
1.20	SWAP Gate	50
1.21	SWAP Gate using CNOT Gates	51
1.22	SWAP Gate using CNOT Gates	51
1.23	Toffoli Gate	53
1.24	Fredkin Gate	56
1.25	Fredkin Gate using CNOT and Toffoli Gates	57
1.26	Quantum Circuit	59
1.27	Reverse Quantum Circuit	59
1.28	Bell States	65

1.29	Reverse Bell States	66
1.30	Creating Bell States	67
1.31	Alice's Operations	68
1.32	Teleportation Circuit	70
1.33	Creating Bell States	71
1.34	Alice's Encoding	71
1.35	Bob's Decoding	72
1.36	Superdense Coding Circuit	74
1.37	Classical Circuit	76
1.38	Control Swap Gate	78
1.39	Reversible AND Gate	78
1.40	Quantum Circuit	79
1.41	Quantum Oracle	80
1.42	Oracle	81
1.43	Elitzur-Vaidman Bomb Detection Algorithm	82
1.44	Elitzur-Vaidman Bomb Detection Algorithm	84
1.45	Deutsch's Algorithm	87
1.46	Deutsch-Josza Algorithm	91
1.47	Deutsch-Josza Algorithm for $n=2$	94
1.48	Simon's Algorithm	99
1.49	Grover's Algorithm	105
1.50	Grover's Algorithm	108
2.1	Uncomputation of a classical gate	113
B.1	Half Adder	176
B.2	Full Adder	176
B.3	Circuit to compute f	177
B.4	NAND gate used to simulate AND, XOR and NOT gates	178
B.5	NAND gate used to simulate XOR gate	178
B.6	NAND gate used to simulate NOT gate	179

List of Tables

1.1	State of a Qubit for different values of θ and ϕ	18
1.2	Truth Table for Pauli-I Gate	20
1.3	Truth Table for Pauli-X Gate	22
1.4	Truth Table for Pauli-Y Gate	23
1.5	Truth Table for Pauli-Z Gate	25
1.6	Truth Table for Hadamard Gate	28
1.7	Truth Table for S Gate	31
1.8	Truth Table for S^\dagger Gate	33
1.9	Truth Table for T Gate	34
1.10	Truth Table for T^\dagger Gate	35
1.11	Truth Table for CNOT Gate	41
1.12	Truth Table for CY Gate	43
1.13	Truth Table for CZ Gate	45
1.14	Truth Table for CH Gate	47
1.15	Truth Table for SWAP Gate	50
1.16	Truth Table for Toffoli Gate	52
1.17	Truth Table for Fredkin Gate	55
1.18	Reversible AND Gate	79
1.19	Probability of Bomb Exploding after N iterations	86
1.20	Possible Functions	86
1.21	Possible Functions	90
1.22	Example of Simon's Algorithm	98
1.23	Function	100
1.24	Example of Grover's Algorithm	104
1.25	Example of Grover's Algorithm	106

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) $|0\rangle, |1\rangle, \dots, |N-1\rangle$. Then, a *pure quantum state* (or *state*) $|\psi\rangle$ is a superposition of classical states, written as:

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

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 $|0\rangle, |1\rangle, \dots, |N-1\rangle$ with amplitudes $\alpha_0, \alpha_1, \dots, \alpha_{N-1}$ respectively. Mathematically, the states $|0\rangle, |1\rangle, \dots, |N-1\rangle$ 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}_A with orthonormal basis $|0\rangle, |1\rangle, \dots, |N-1\rangle$ and another Hilbert space \mathcal{H}_B with an orthonormal basis $|0'\rangle, |1'\rangle, \dots, |M-1\rangle$. Then we can combine the two Hilbert spaces using tensor products to form a new Hilbert space $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$ 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}_A ($|i\rangle \in \{0, \dots, N-1\}$) and $|j\rangle$ is a basis vector of \mathcal{H}_B ($|j\rangle \in \{0', \dots, M-1\}$). The basis vectors of \mathcal{H} are $|i\rangle \otimes |j\rangle$ where $i \in \{0, \dots, N-1\}$ and $j \in \{0', \dots, M-1\}$. 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 $|\psi\rangle$. Such a 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 it's 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 states 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 $|\psi\rangle = 7|0\rangle + (3 + 5i)|1\rangle$. In the matrix form, it can be written as:

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

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 Hermitian and as proved in the appendix we know that e^A is unitary for any Hermitian 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 description 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 exponential 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 hamiltonian. 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 often apply a unitary operator to a closed quantum state. This is a contradictory statement. The act of applying a unitary operator is in itself an external interaction 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 Schrodinger'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 probability 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 $e^{i\theta}|\psi\rangle$. The term $e^{i\theta}$ is called the **global phase** and thus has no physical significance.

Note that when the experimentalist 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 described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space $|\psi\rangle$ being measured. 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 measurement 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:

$$\begin{aligned} \frac{M_0 |\psi\rangle}{\sqrt{p(0)}} &= \frac{\alpha |0\rangle}{|\alpha|} \\ \frac{M_1 |\psi\rangle}{\sqrt{p(1)}} &= \frac{\beta |1\rangle}{|\beta|} \end{aligned}$$

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 $|0\rangle, |1\rangle, \dots, |N-1\rangle$.
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:

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

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

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

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

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

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

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

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 certainty. 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 distinguishing 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 measurement 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 $|\psi_2\rangle$ because $|\psi_2\rangle$ can be decomposed into a (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, \dots, P_m that all act on the 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 system immediately after the measurement is $\frac{P_j |\psi\rangle}{\sqrt{p(j)}}$. Note that the probabilities sum to 1.

We cannot choose 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 certainty. 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 $|0\rangle, |1\rangle, \dots, |N-1\rangle$. We can choose any other orthonormal basis B of states say $|\phi_0\rangle, |\phi_1\rangle, \dots, |\phi_{N-1}\rangle$ and consider the projective measurement defined by the projectors $P_j = |\phi_j\rangle\langle\phi_j|$. This is called measuring 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, \dots, P_m and associated distinct outcomes $\lambda_1, \lambda_2, \dots, \lambda_m \in \mathbb{R}$, 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, it's 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 occurred, 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 definition, the average value of the measurement is:

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

Positive 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 measurement called *positive-operator-valued-measure (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 n positive semi-definite (Psd) matrices E_1, \dots, E_m such that $\sum_m E_m = I$. When measuring a state $|\psi\rangle$, the probability of the outcome i is given by $\text{Tr}(E_i |\psi\rangle \langle \psi|) = \langle \psi | E_i | \psi \rangle$. 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 measurement on a slightly larger space that yields the exact probability distribution over measurement outcomes (Neumark's theorem)*

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, these 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 measurement with three possible outcomes: 0, 1 and 2, such that:

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

This cannot be achieved with projective measurement on the qubit, but can be achieved with following 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 $|0\rangle$ state)
- $E_2 = I - E_0 - E_1$

Here E_0, E_1, E_2 are all psd and add up to identity, so they form a 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 different 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 copmonent 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 $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle$. 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:

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

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-qubit 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. Recall 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 probability 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 probability 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 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 probability 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.

1.2 Bloch Sphere Representation of a Qubit

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.

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 $\alpha = |\alpha|e^{i\phi_\alpha}$ and $\beta = |\beta|e^{i\phi_\beta}$. 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:

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

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

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

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

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

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:

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

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 1.1.

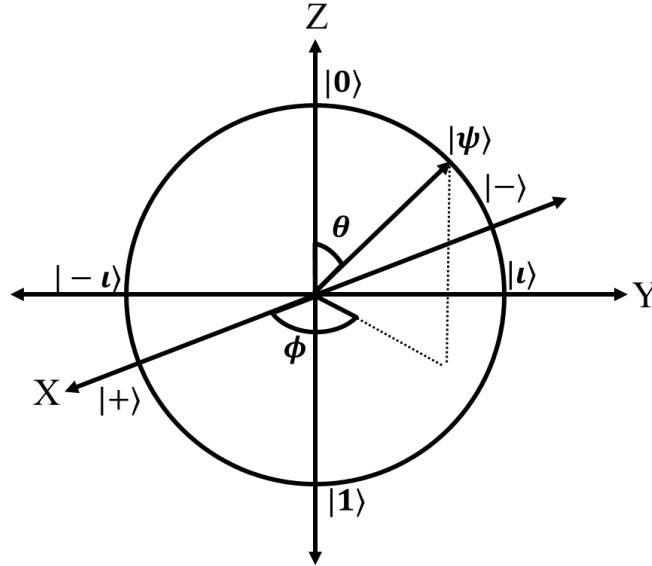


Figure 1.1: Bloch-Poincare Sphere

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:

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

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

```

1 import matplotlib.pyplot as plt
2 from qiskit.visualization import plot_bloch_vector # Import the
   function to plot the Bloch vector
3 from qiskit.visualization import plot_bloch_multivector # Import the
   function to plot the Bloch vector
4
5 '''
6 Function to plot the Bloch Sphere
7
8 plot_bloch_vector(bloch, title='', ax=None, figsize=None, coord_type='
   spherical', font_size=None)
9
10 Plots a Bloch Sphere
11
12 Parameters
13 - bloch (list[double]) - array of three elements where [<x>, <y>, <z>]
   (Cartesian) or
14 [<r>, <theta>, <phi>] (spherical in radians) <theta> is inclination
   angle from +z direction
15 <phi> is azimuth from +x direction
16 - title (str) - a string that represents the title of the plot
17 - ax (matplotlib.axes.Axes) - an axes of the current figure to plot the
   Bloch sphere into
18 - figsize (tuple) - a tuple (width, height) in inches that represents
   the size of the figure
19 - coord_type (str) - the coordinate system to use. 'spherical' or '
   cartesian', default is cartesian
20 - font_size (int) - the font size of the tex
21
22 Returns
23 - matplotlib.figure.Figure - a matplotlib figure object
24
25 Raises
26 - ImportError - if matplotlib is not installed
27 '''
28
29 plot_bloch_vector([0,1,0], title='Bloch Sphere')
```

The output of the above code is shown in the figure 1.2.

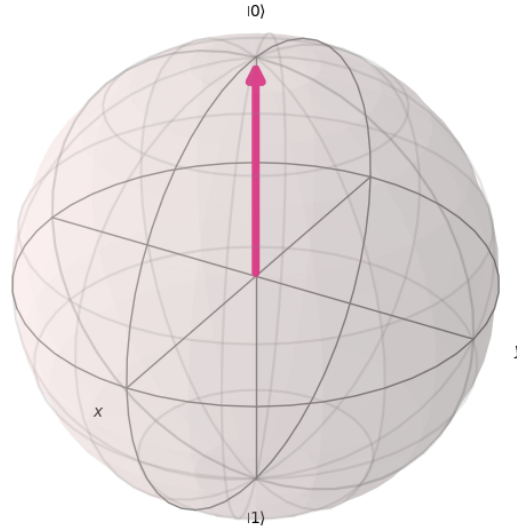


Figure 1.2: Bloch Sphere

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 1.2.1. The state of a qubit for different values of θ and ϕ is shown in the table 1.1.

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 1.1: 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:

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

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

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

Thus, upon simplifying we get:

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

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

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

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

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 1.1 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 cartesian 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 $\vec{n} = (\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta)$. Points on a Bloch sphere is always a pure state.

1.3 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 1.3.1. 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.

1.3.1 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 1.2. It does no rotation of the bloch vector on the Bloch Sphere.

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

Table 1.2: Truth Table for Pauli-I Gate

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

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5
6 circuit = QuantumCircuit(qreg-q)
7
8 circuit.id(qreg-q[0]) #For Pauli-Identity Gate
9 circuit.draw(output='mpl')#.savefig(' ../images/pauli-i.png') #Draw the
   circuit

```

The circuit symbol is as shown in figure 1.3



Figure 1.3: Pauli-I Gate

In operator Notation it is:

$$\begin{aligned}
 I|0\rangle &= |0\rangle \\
 I|1\rangle &= |1\rangle
 \end{aligned}$$

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 diagonalizable. 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 1.3. 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 1.3: Truth Table for Pauli-X Gate

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

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='Not Gate')
6
7 circuit.x(qreg-q[0])
8
9 circuit.draw(output='mpl')#.savefig(' ../ images / pauli-x.png ') #Draw the
  circuit

```

The circuit symbol is as shown in figure 1.4.



Figure 1.4: 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 $\sigma_X = H\sigma_ZH$ 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 anticlockwise π rotation about Y axis of Bloch Sphere. It flips the Qubit and multiplies a complex amplitude. The truth table is as in table 1.4.

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

Table 1.4: Truth Table for Pauli-Y Gate

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


```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='Not Gate')
6
7 circuit.y(qreg-q[0]) #Apply the Y gate
8
9 circuit.draw(output='mpl')#.savefig(' ../images/pauli-x.png') #Draw the
   circuit

```

The circuit symbol is as shown in figure 1.5.



Figure 1.5: Pauli-Y Gate

In the operator Notation it is:

$$\begin{aligned}
 Y|0\rangle &= i|1\rangle \\
 Y|1\rangle &= -i|0\rangle
 \end{aligned}$$

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:

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

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

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

Note that the Pauli-Y Gate can be written as $Y = iXZ$ or $\sigma_Y = i\sigma_X\sigma_Z$ 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 anticlockwise 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 $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is:

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

Pauli-Z Gate

It is a anticlockwise π rotation about Z axis of Bloch Sphere. It flips the sign of the $|1\rangle$ state. It is also called Phase Flip/Phase shift/Sign flip gate. The truth table is as in table 1.5.

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

Table 1.5: Truth Table for Pauli-Z Gate

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

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='Not Gate')
6
7 circuit.z(qreg-q[0]) #Apply the Z gate
8
9 circuit.draw(output='mpl')#.savefig(' ../ images/pauli-z.png') #Draw the
   circuit

```

The circuit symbol is as shown in figure 1.6.

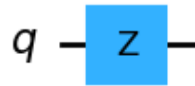


Figure 1.6: Pauli-Z Gate

In the operator Notation it is:

$$\begin{aligned} Z |0\rangle &= |0\rangle \\ Z |1\rangle &= -|1\rangle \end{aligned}$$

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, $U^\dagger U = U U^\dagger = I$. Thus combining, the two we get $U^2 = I$. This, can also be verified as shown below:

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

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.

1.3.2 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 1.6.

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 1.6: Truth Table for Hadamard Gate

The code for the Hadamard Gate is as follows:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='Hadamard Gate')
6
7 circuit.h(qreg-q[0]) #Apply the H gate
8
9 circuit.draw(output='mpl')#.savefig(' ../images/hadamard.png') #Draw the
  circuit

```

The circuit symbol is as shown in figure 1.7.

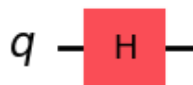


Figure 1.7: Hadamard Gate

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 initialise 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:

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

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.

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

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 1.3.2. 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,

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

1.3.3 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 1.7.

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

Table 1.7: Truth Table for S Gate

The code for the S Gate is as follows:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='S Gate')
6
7 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 1.8.

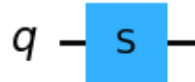


Figure 1.8: S Gate

In the operator Notation it is:

$$\begin{aligned}
 S|0\rangle &= |0\rangle \\
 S|1\rangle &= i|1\rangle
 \end{aligned}$$

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 1.8.

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

Table 1.8: Truth Table for S^\dagger Gate

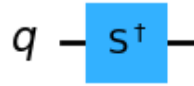
The code for the S^\dagger Gate is as follows:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='T Gate')
6
7 circuit.sdg(qreg-q[0]) #Apply the sdg gate
8
9 circuit.draw(output='mpl')#.savefig(' ../images/sdag-gate.png') #Draw
  the circuit

```

The circuit symbol is as shown in figure 1.9.

Figure 1.9: S^\dagger Gate

In the operator Notation it is:

$$\begin{aligned}
 S^\dagger |0\rangle &= |0\rangle \\
 S^\dagger |1\rangle &= -i|1\rangle
 \end{aligned}$$

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. Recall, that the only criteria for a gate is to be a Unitary (from Postulate 2: Unitary evolution) gate and not necessarily Hermitian.

1.3.4 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 1.9.

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

Table 1.9: Truth Table for T Gate

The code for the T Gate is as follows:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='T Gate')
6
7 circuit.sdg(qreg-q[0]) #Apply the T gate
8
9 circuit.draw(output='mpl')#.savefig(' ../images/sdag-gate.png') #Draw
   the circuit

```

The circuit symbol is as shown in figure 1.10.

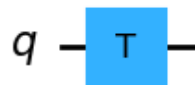


Figure 1.10: T Gate

In the operator Notation it is:

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

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 1.10.

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

Table 1.10: Truth Table for T^\dagger Gate

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

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg_q, name='T Gate')
6
7 circuit.tdg(qreg_q[0]) #Apply the T gate
8
9 circuit.draw(output='mpl')#.savefig(' ../images/tdag-gate.png') #Draw
   the circuit
```

The circuit symbol is as shown in figure 1.11.

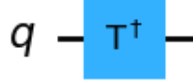


Figure 1.11: T^\dagger Gate

In the operator Notation it is:

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

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 relations.

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

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

1.3.5 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

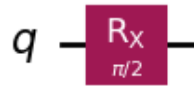
It is a rotation about X axis of Bloch Sphere by an angle θ . This is denoted as R_θ^X . The matrix representation of R_θ^X is:

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

The code for the R_X Gate is as follows:

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg_q, name='U1 Gate')
6
7 circuit.rx(pi/2, qreg_q[0]) #Apply the RX gate
8
9 circuit.draw(output='mpl')#.savefig('../images/rx-gate.png') #Draw the
   circuit
```

The circuit symbol is as shown in figure 1.12.

Figure 1.12: R_X Gate

Note that the value $\pi/2$ given in the figure 1.12 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:

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

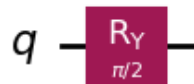
The code for the R_Y Gate is as follows:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg_q, name='RY Gate')
6
7 circuit.ry(pi/2, qreg_q[0]) #Apply the RY gate
8
9 circuit.draw(output='mpl')#.savefig('../images/ry-gate.png') #Draw the
   circuit

```

The circuit symbol is as shown in figure 1.13.

Figure 1.13: 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:

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

The code for the R_Z Gate is as follows:

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='RZ Gate')
6
7 circuit.rz(pi/2, qreg-q[0]) #Apply the RZ gate
8
9 circuit.draw(output='mpl')#.savefig('../images/rz-gate.png') #Draw the
   circuit
```

The circuit symbol is as shown in figure 1.14.

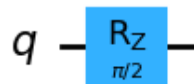


Figure 1.14: R_Z Gate

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.

1.3.6 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:

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(1, 'q')
5 circuit = QuantumCircuit(qreg-q, name='U3 Gate')
6
7 circuit.u(pi/2, pi/2, pi/2, qreg-q[0]) #Apply the U3 gate
8
9 circuit.draw(output='mpl')#.savefig('../images/u3-gate.png') #Draw the
   circuit
```

The circuit symbol is as shown in figure 1.15.



Figure 1.15: 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.

1.4 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.

1.4.1 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 1.11.

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

Table 1.11: Truth Table for CNOT Gate

For the table in 1.11, 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:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 # Add 2 qubits for CNOT gate demo
5 qreg-q = QuantumRegister(2, 'q')
6 circuit = QuantumCircuit(qreg-q, name='CNOT Gate')
7
8 circuit.cx(qreg-q[0], qreg-q[1]) #Apply the CNOT gate
9
10 circuit.draw(output='mpl')#.savefig(' ../ images / cnot - gate . png ') #Draw
    the circuit

```

The circuit symbol is as shown in figure 1.16.

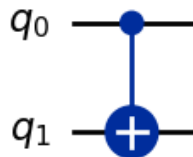


Figure 1.16: CNOT Gate

Here it can be seen that the 1st Qubit is control and the 2nd Qubit is target. In the figure 1.16, the dot denotes the control qubit and the cross denotes the target qubit.

In the operator Notation it is:

$$CX |00\rangle = |00\rangle$$

$$CX |01\rangle = |01\rangle$$

$$CX |10\rangle = |11\rangle$$

$$CX |11\rangle = |10\rangle$$

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

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

In Compact form we can write the CNOT gate as:

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

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

Teleportation 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 1.12.

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

Table 1.12: Truth Table for CY Gate

For the table in 1.12, 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:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 # Add 2 qubits for CNOT gate demo
5 qreg-q = QuantumRegister(2, 'q')
6 circuit = QuantumCircuit(qreg-q, name='CY Gate')
7
8 circuit.cy(qreg-q[0], qreg-q[1]) #Apply the CY gate
9
10 circuit.draw(output='mpl')#.savefig(' ../ images/cy-gate.png ') #Draw the
    circuit

```

The circuit symbol is as shown in figure 1.17.

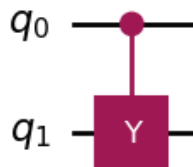


Figure 1.17: CY Gate

In this figure 1.17, 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:

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

In Outer product representation, it can be written as:

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

In Compact form we can write the CY gate as:

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

Flipping amplitude of 1 state. Marking of this state can be utilised 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 1.13.

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

Table 1.13: Truth Table for CZ Gate

For the table in 1.13, 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:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(2, 'q')
5 circuit = QuantumCircuit(qreg-q, name='CZ Gate')
6
7 circuit.cz(qreg-q[0], qreg-q[1]) #Apply the CZ gate
8
9 circuit.draw(output='mpl')#.savefig('../images/CZ-gate.png') #Draw the
  circuit

```

The circuit symbol is as shown in figure 1.18.

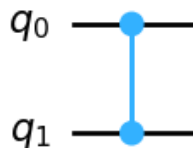


Figure 1.18: CZ Gate

In this figure 1.18, 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 1.14.

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 1.14: Truth Table for CH Gate

For the table in 1.14, 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:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 # Add 2 qubits for CNOT gate demo
5 qreg-q = QuantumRegister(2, 'q')
6 circuit = QuantumCircuit(qreg-q, name='CY Gate')
7
8 circuit.ch(qreg-q[0], qreg-q[1]) #Apply the CH gate
9
10 circuit.draw(output='mpl')#.savefig('../images/ch-gate.png') #Draw the
    circuit

```

The circuit symbol is as shown in figure 1.19.

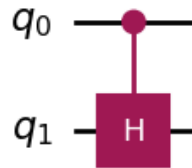


Figure 1.19: CH Gate

In this figure 1.19, 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 which 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 1.15.

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

Table 1.15: Truth Table for SWAP Gate

The code for the SWAP Gate is as follows:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 # Add 2 qubits for CNOT gate demo
5 qreg-q = QuantumRegister(2, 'q')
6 circuit = QuantumCircuit(qreg-q, name='SWAP Gate')
7
8 circuit.swap(qreg-q[0], qreg-q[1]) #Apply the SWAP gate
9
10 circuit.draw(output='mpl')#.savefig(' ../ images / swap - gate . png ') #Draw
    the circuit

```

The circuit symbol is as shown in figure 1.20.

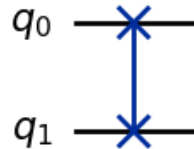


Figure 1.20: SWAP Gate

In the operator Notation it is:

$$\begin{aligned}
 SWAP |00\rangle &= |00\rangle \\
 SWAP |01\rangle &= |10\rangle \\
 SWAP |10\rangle &= |01\rangle \\
 SWAP |11\rangle &= |11\rangle
 \end{aligned}$$

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

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

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:

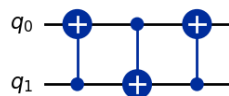


Figure 1.21: SWAP Gate using CNOT Gates

Or using CNOT gates as follows:

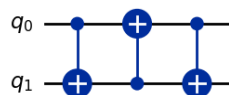


Figure 1.22: SWAP Gate using CNOT Gates

1.4.2 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 1.16.

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 1.16: Truth Table for Toffoli Gate

For the table in 1.16, 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:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(3, 'q')
5 circuit = QuantumCircuit(qreg-q)
6
7 circuit.ccx(qreg-q[0], qreg-q[1], qreg-q[2])
8
9 circuit.draw(output='mpl')#.savefig('../images/ccx-gate.png')
```

The circuit symbol is as shown in figure 1.23.

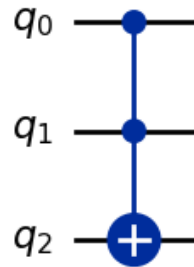


Figure 1.23: Toffoli Gate

In this figure 1.23, 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:

$$CCX |000\rangle = |000\rangle$$

$$CCX |001\rangle = |001\rangle$$

$$CCX |010\rangle = |010\rangle$$

$$CCX |011\rangle = |011\rangle$$

$$CCX |100\rangle = |100\rangle$$

$$CCX |101\rangle = |101\rangle$$

$$CCX |110\rangle = |111\rangle$$

$$CCX |111\rangle = |110\rangle$$

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

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:

$$CCX = |000\rangle\langle 000| + |001\rangle\langle 001| + |010\rangle\langle 010| + |011\rangle\langle 011| \\ + |100\rangle\langle 100| + |101\rangle\langle 101| + |110\rangle\langle 111| + |111\rangle\langle 110|$$

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:

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

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 1.17.

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 1.17: Truth Table for Fredkin Gate

For the table in 1.17, 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:

```

1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg-q = QuantumRegister(3, 'q')
5 circuit = QuantumCircuit(qreg-q)
6
7 circuit.cswap(qreg-q[0], qreg-q[2], qreg-q[1])
8
9 circuit.draw(output='mpl')#.savefig('../images/cswap-gate.png')
```

The circuit symbol is as shown in figure 1.24.

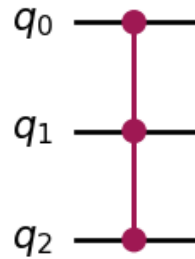


Figure 1.24: Fredkin Gate

In this figure 1.24, 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:

$$CSWAP |000\rangle = |000\rangle$$

$$CSWAP |001\rangle = |001\rangle$$

$$CSWAP |010\rangle = |010\rangle$$

$$CSWAP |011\rangle = |011\rangle$$

$$CSWAP |100\rangle = |100\rangle$$

$$CSWAP |101\rangle = |110\rangle$$

$$CSWAP |110\rangle = |101\rangle$$

$$CSWAP |111\rangle = |111\rangle$$

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

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:

$$CSWAP = |000\rangle\langle 000| + |001\rangle\langle 001| + |010\rangle\langle 010| + |011\rangle\langle 011| \\ + |100\rangle\langle 100| + |101\rangle\langle 110| + |110\rangle\langle 101| + |111\rangle\langle 111|$$

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:

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

We can construct Fredkin gate using CNOT and Toffoli gate as shown in the figure 1.25.

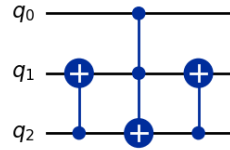


Figure 1.25: Fredkin Gate using CNOT and Toffoli Gates

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 $U(\alpha|0\rangle + \beta|1\rangle)$ 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 $CNOT|\psi\rangle = CNOT\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \frac{CNOT|00\rangle + CNOT|11\rangle}{\sqrt{2}}$. which will thus be $\frac{|00\rangle + |10\rangle}{\sqrt{2}}$. In the matrix form,

$$\begin{aligned}
 CNOT|\psi\rangle &= CNOT\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \\
 &= \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \\
 &= (CNOT \frac{1}{\sqrt{2}}|00\rangle + CNOT \frac{1}{\sqrt{2}}|11\rangle) = \frac{1}{\sqrt{2}}CNOT|00\rangle + \frac{1}{\sqrt{2}}CNOT|11\rangle \\
 &= \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)
 \end{aligned}$$

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.

1.5 Quantum Circuits

Example 1.5.1. Consider the quantum circuit as shown in figure 1.26.

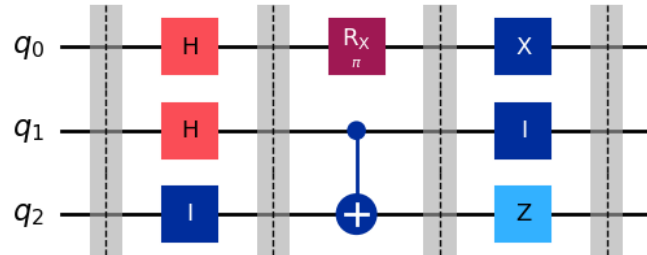


Figure 1.26: Quantum Circuit

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 except for the rotation gate all the other gates are their own inverses. Thus, the reverse circuit is as shown in figure 1.27.

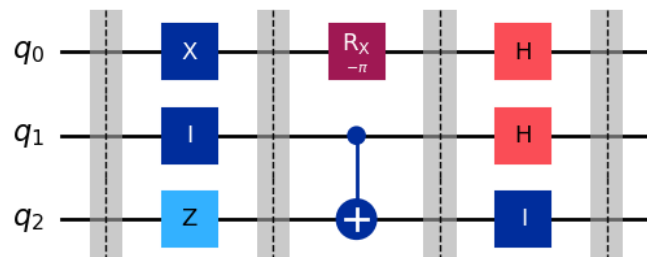


Figure 1.27: Reverse Quantum Circuit

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 1.26 the circuit has been divided 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:

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

Here the action of RX 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 complex 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.

1.6 Entanglement

Given a state of multiple entangled qubit, one cannot express individual qubit separately. 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 separable 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.

1.6.1 Bell States/EPR Pairs

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

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

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 1.28.

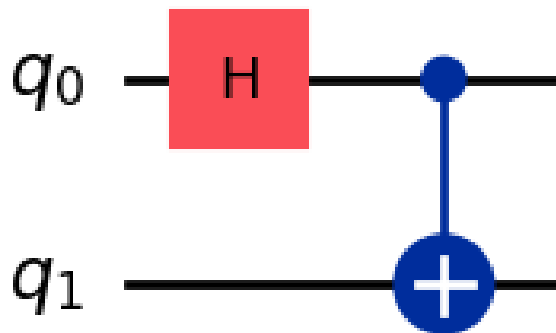


Figure 1.28: Bell States

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 1.6.1. Given the state $|\psi\rangle = \frac{|01\rangle-|10\rangle}{\sqrt{2}}$, find whether the state is entangled or not. We prove this by contradiction, Assume that $|\psi\rangle$ can be written separately 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.

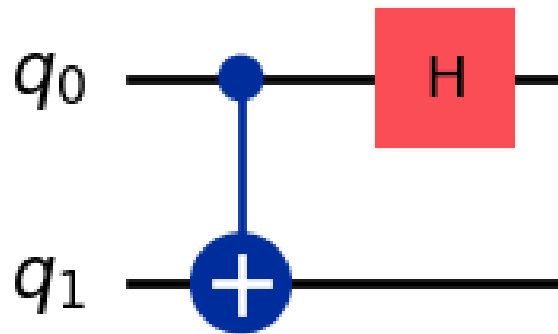


Figure 1.29: Reverse Bell States

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. (Hadamard gate is also self-inverse).

1.6.2 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 Alice 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 1.30 which has both the inputs $q_0 = |0\rangle$ and $q_1 = |0\rangle$.

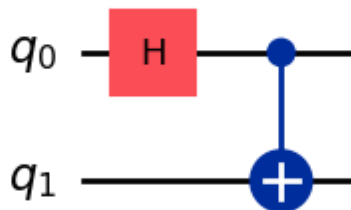


Figure 1.30: Creating Bell States

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 1.31 with $q_0 = |\psi\rangle$ and $q_1 = |\beta_A\rangle$.

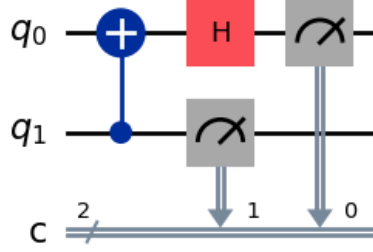


Figure 1.31: Alice's Operations

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 target 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 unknown 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 1.32.

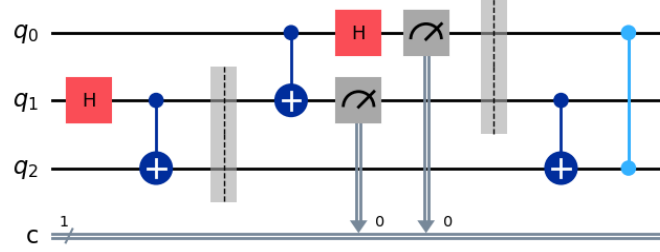


Figure 1.32: Teleportation Circuit

We can see from the figure that the measurement output from the Alice's qubits act as control bits for applying 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 1.32, show that after Alice's measurement the bits act as control bits for the operations to be applied by Bob.

1.6.3 Superdense 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.

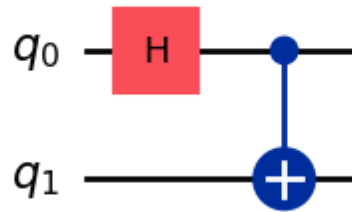


Figure 1.33: Creating Bell States

This is how the Bell state is created shown in the figure 1.33.

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 1.34.

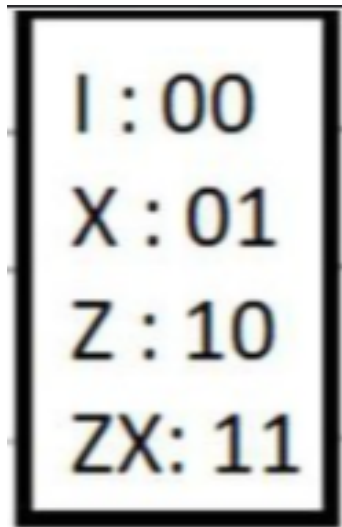


Figure 1.34: Alice's Encoding

- 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 1.35.

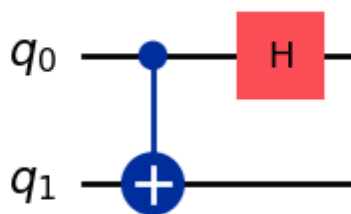


Figure 1.35: Bob's Decoding

- 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 1.36.

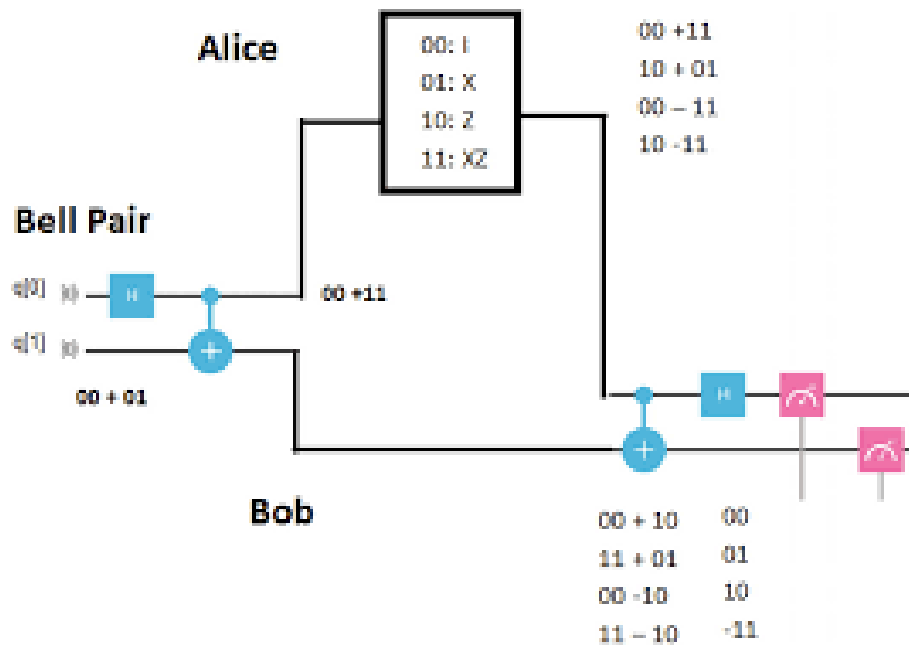


Figure 1.36: Superdense Coding Circuit

1.7 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.

1.7.1 Oracles

Consider a classical circuit representing a classical function f . Consider the classical function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ 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 Appendix B.2) Thus, any classical boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ can be represented using a classical circuit as shown in the figure 1.37.

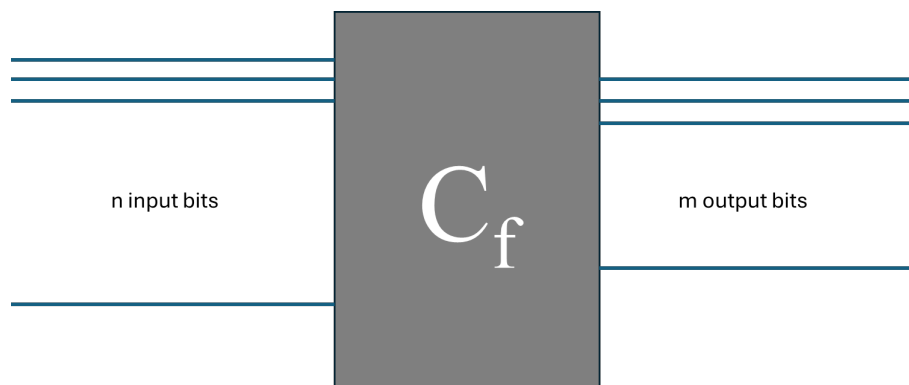


Figure 1.37: Classical Circuit

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 unitary operations (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 because 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. Its 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 1.4.2). 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 1.38

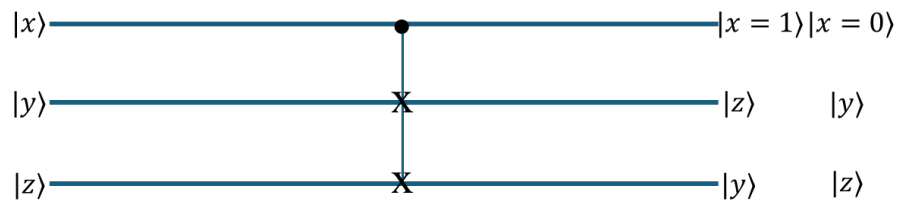


Figure 1.38: Control Swap Gate

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 referred to as junk which is a function of x i.e. $junk(x)$. This is as shown in the figure 1.39.

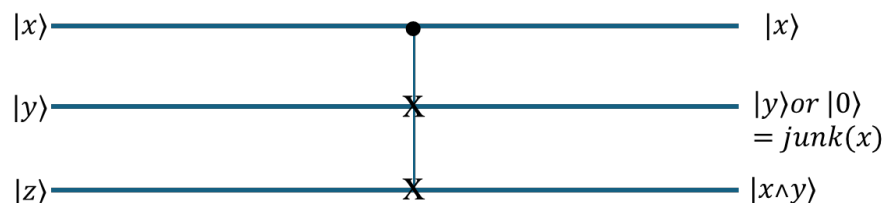


Figure 1.39: Reversible AND Gate

The truth table for the above circuit is as shown in the table 1.18.

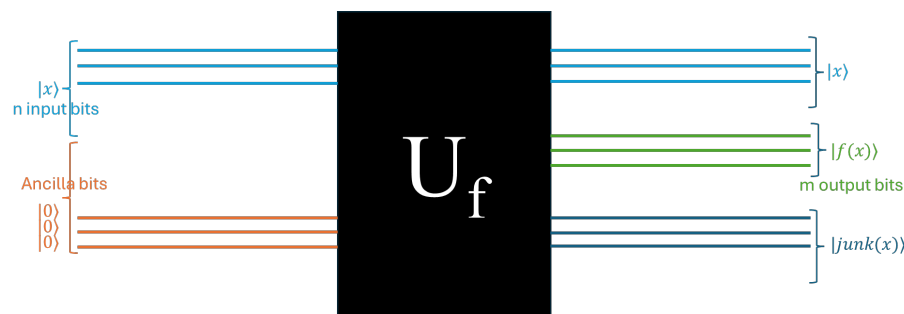


Figure 1.40: Quantum Circuit

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 1.18: 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 1.40. **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 1.40, 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 1.41

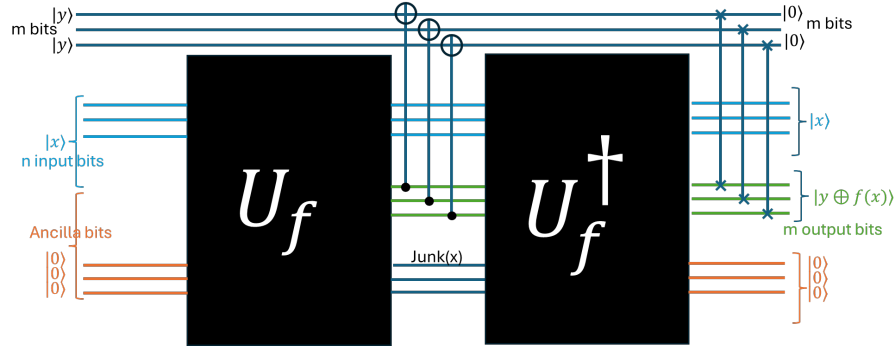


Figure 1.41: Quantum Oracle

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 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 function 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 simplicity we generally ignore the $|0\rangle$ in inputs and outputs.

Thus, whenever given a classical function f , we can make it as a reversible quantum function and thus, implement it as a quantum circuit using reversible quantum gates. This can be visualized as shown in the figure 1.42.

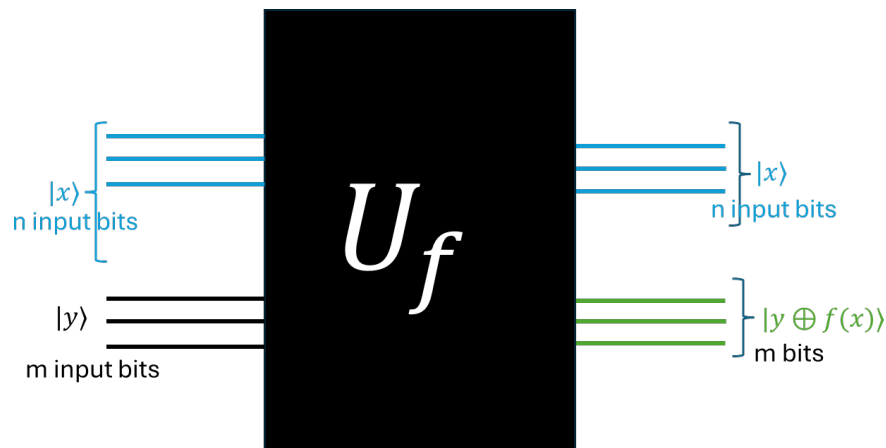


Figure 1.42: Oracle

1.7.2 Elitzur-Vaidman Bomb Detection Algorithm

We are given a device which can be a working Bomb or dud. The device 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: Naïve 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, because the outcome of the measurement will be $|0\rangle$ so the bomb doesn't explode and returns $|0\rangle$. If we give device input of $|1\rangle$ 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 $|0\rangle$ is of no help as it provides no information about the device since in both the cases if it's 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 detect 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 1.43.

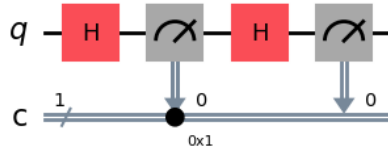


Figure 1.43: Elitzur-Vaidman Bomb Detection Algorithm

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 through 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 thorough 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 output 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 without 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 without 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 orthonormal 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 1.44.

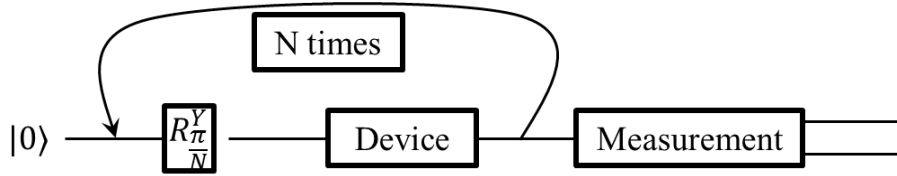


Figure 1.44: Elitzur-Vaidman Bomb Detection Algorithm

Thus, 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 k th 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 $\lim_{N \rightarrow \infty} \cos^2 \frac{\pi}{2N} \approx \cos 0 \approx 1$ and the probability of getting $|1\rangle$ is $\lim_{N \rightarrow \infty} \sin^2 \frac{\pi}{2N} \approx \sin 0 \approx 0$. 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 probability 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. 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 $P(A \cap B \cap \dots \cap Z) = P(A) \cdot P(B|A) \dots P(Z|A, B, \dots, Z) = P(A) \cdot P(B) \dots P(Z)$, 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}
\vdots	\vdots
500	1.409×2^{-2503}

Table 1.19: Probability of Bomb Exploding after N iterations

As can be seen in the table 1.19, 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.

1.7.3 Deutsch's Algorithm

It is a simple and good to learn basic algorithm.

Definition 1.7.1. Given a function $f : \{0, 1\} \rightarrow \{0, 1\}$, 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 1.20. 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 1.20: 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 function call. This speedup is because of the superposition.

The circuit for Deutsch's Algorithm is as shown in the figure 1.45.

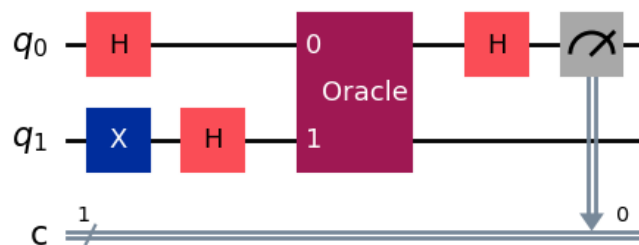


Figure 1.45: Deutsch's Algorithm

Consider the oracle as shown in the figure 1.45. 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 through 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:

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

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

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:

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

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

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

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

Hence, as the qubit q_0 passes through the Hadamard gate, the action of the Hadamard gate on the first qubit will be $H|+\rangle = |0\rangle$. Thus, upon measurement the output will be $|0\rangle$ with probability 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:

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

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

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

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

Hence, as the qubit q_0 passes through the Hadamard gate, the action of the Hadamard gate on the first qubit will be $H|+\rangle = |0\rangle$. Thus, upon measurement the output will be $|0\rangle$ with probability 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:

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

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

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

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

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 probability 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:

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

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

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

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

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 probability 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.

1.7.4 Deutsch-Josza Algorithm

This is a generalization of the Deutsch Algorithm.

Definition 1.7.2. 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 1.21.

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 1.21: 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 1.46.

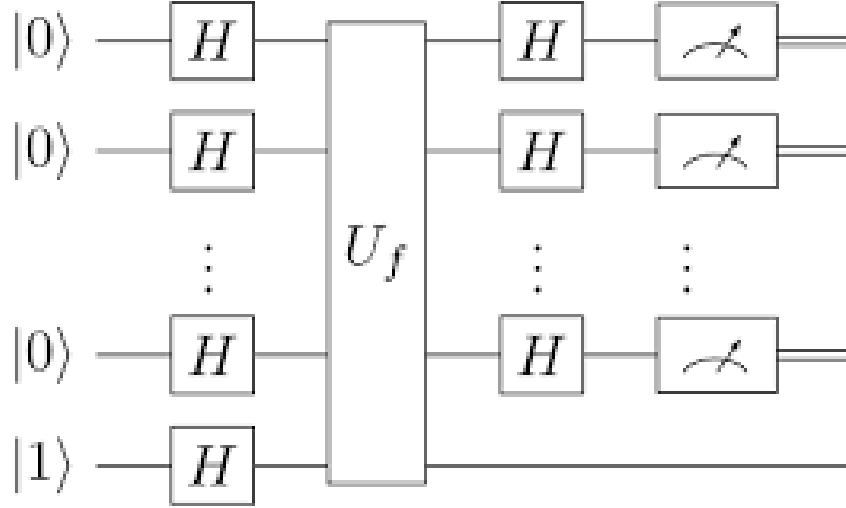


Figure 1.46: Deutsch-Josza Algorithm

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:

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

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 $\frac{|f(x) - \overline{f(x)}\rangle}{\sqrt{2}} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$ and when $f(x) = 1$ then $\frac{|f(x) - \overline{f(x)}\rangle}{\sqrt{2}} = \frac{|1\rangle - |0\rangle}{\sqrt{2}} = -|-\rangle$. 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:

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

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 1.3.2) 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:

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

Upon simplifying we get,

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

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:

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

Upon simplifying we get,

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

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:

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

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 $|11\rangle$.

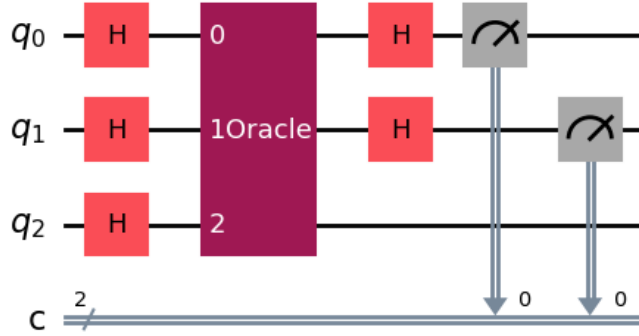


Figure 1.47: Deutsch-Josza Algorithm for n=2

Example 1.7.1. Consider the Deutsch-Josza algorithm for $n=2$. The possible functions are as shown in the table 1.21 and the circuit for the Deutsch-Josza algorithm is as shown in the figure 1.46. The input to the circuit is $|00\rangle |1\rangle$. The action of the Hadamard gate on the input will be:

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

The action of the oracle will be:

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

$$\begin{aligned}
& \frac{1}{2\sqrt{2}}(U_f |000\rangle + U_f |010\rangle + U_f |100\rangle + U_f |110\rangle - U_f |001\rangle - U_f |011\rangle - U_f |101\rangle - U_f |111\rangle) \\
&= \frac{1}{2\sqrt{2}}(|00\rangle |f(00)\rangle + |01\rangle |f(01)\rangle + |10\rangle |f(10)\rangle + |11\rangle |f(11)\rangle - |00\rangle |\overline{f(00)}\rangle \\
&\quad - |01\rangle |\overline{f(01)}\rangle - |10\rangle |\overline{f(10)}\rangle - |11\rangle |\overline{f(11)}\rangle)
\end{aligned}$$

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:

$$\begin{aligned}
& \frac{1}{2\sqrt{2}}(|00\rangle |0\rangle + |01\rangle |0\rangle + |10\rangle |0\rangle + |11\rangle |0\rangle - |00\rangle |1\rangle - |01\rangle |1\rangle - |10\rangle |1\rangle - |11\rangle |1\rangle) \\
&= \frac{1}{2\sqrt{2}}(|000\rangle + |010\rangle + |100\rangle + |110\rangle - |001\rangle - |011\rangle - |101\rangle - |111\rangle) \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}
\end{aligned}$$

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

$$\begin{aligned}
& (H^{\otimes 2} \otimes I) \frac{1}{2}((|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes |-\rangle) \\
&= \frac{1}{2}(H^{\otimes 2} |00\rangle + H^{\otimes 2} |01\rangle + H^{\otimes 2} |10\rangle + H^{\otimes 2} |11\rangle) \otimes |-\rangle \\
&= \frac{1}{2}[\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\
&\quad \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)] \otimes |-\rangle
\end{aligned}$$

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:

$$\begin{aligned}
& \frac{1}{2\sqrt{2}}(|00\rangle|1\rangle + |01\rangle|1\rangle + |10\rangle|1\rangle + |11\rangle|1\rangle - |00\rangle|0\rangle - |01\rangle|0\rangle - |10\rangle|0\rangle - |11\rangle|0\rangle) \\
&= \frac{-1}{2\sqrt{2}}(|000\rangle + |010\rangle + |100\rangle + |110\rangle - |001\rangle - |011\rangle - |101\rangle - |111\rangle) \\
&= \frac{-1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}
\end{aligned}$$

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

$$\begin{aligned}
& (H^{\otimes 2} \otimes I) \frac{-1}{2}((|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes |-\rangle) \\
&= \frac{-1}{2}(H^{\otimes 2}|00\rangle + H^{\otimes 2}|01\rangle + H^{\otimes 2}|10\rangle + H^{\otimes 2}|11\rangle) \otimes |-\rangle \\
&= \frac{-1}{2}[\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\
&\quad \frac{-1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) + \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)] \otimes |-\rangle
\end{aligned}$$

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 $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 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.

1.7.5 Simon's Algorithm

Definition 1.7.3. Given a function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ 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 1.7.2. Consider the function $f : \{0,1\}^3 \rightarrow \{0,1\}^3$ 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 following 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 1.22: 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 1.22 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 classically, 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 1.48.

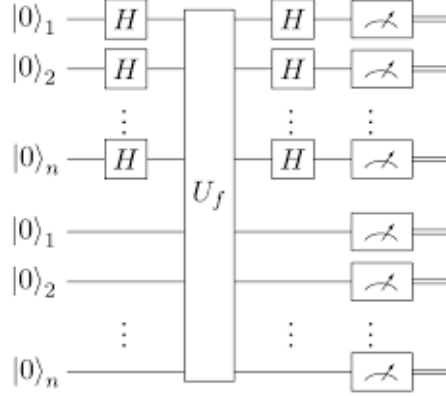


Figure 1.48: Simon's Algorithm

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 $f(z)$. Because 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:

$$\begin{aligned}
(H^{\otimes n} \otimes I) \left(\frac{|z\rangle + |z \oplus s\rangle}{\sqrt{2}} |f(z)\rangle \right) &= \frac{1}{\sqrt{2}} [(H^{\otimes n} |z\rangle + H^{\otimes n} |z \oplus s\rangle) |f(z)\rangle] \\
&= \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} |y\rangle + \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{(z \oplus s) \cdot y} |y\rangle \right] |f(z)\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} ((-1)^{z \cdot y} + (-1)^{(z \oplus s) \cdot y}) |y\rangle |f(z)\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} (1 + (-1)^{s \cdot y}) |y\rangle |f(z)\rangle
\end{aligned}$$

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 1.7.3. Consider the function $f : \{0,1\}^4 \rightarrow \{0,1\}^4$ as shown in the table 1.23.

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 1.23: 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:

$$\begin{aligned} (H^{\otimes 4} \otimes I^{\otimes 4})(|0000\rangle \otimes |0000\rangle) &= \frac{1}{\sqrt{2^4}} \sum_{x \in \{0,1\}^4} |x\rangle |0000\rangle \\ &= \left(\frac{1}{4} \sum_{x \in \{0,1\}^4} |x\rangle \right) \otimes |0000\rangle \\ &= \left(\frac{1}{4} \sum_{x \in \{0,1\}^4} |x\rangle |0000\rangle \right) \end{aligned}$$

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:

$$\begin{aligned} U_f \left(\frac{1}{4} \sum_{x \in \{0,1\}^4} |x\rangle |0000\rangle \right) &= \frac{1}{4} \sum_{x \in \{0,1\}^4} U_f |x\rangle |0000\rangle \\ &= \frac{1}{4} \sum_{x \in \{0,1\}^4} |x\rangle |0000 \oplus f(x)\rangle = \frac{1}{4} \sum_{x \in \{0,1\}^4} |x\rangle |f(x)\rangle \\ &= \frac{(|0000\rangle |f(0000)\rangle + |0001\rangle |f(0001)\rangle + \dots + |1110\rangle |f(1110)\rangle + |1111\rangle |f(1111)\rangle)}{4} \end{aligned}$$

Thus, upon substituting the corresponding values from the table 1.23 we get, since $n=4$ we will get 16 terms:

$$\begin{aligned} \frac{1}{4} [&|0000\rangle |1111\rangle + |0001\rangle |0001\rangle + |0010\rangle |1110\rangle + |0011\rangle |1101\rangle + |0100\rangle |0000\rangle + |0101\rangle |0101\rangle \\ &|0110\rangle |1010\rangle + |0111\rangle |1001\rangle + |1000\rangle |0001\rangle + |1001\rangle |1111\rangle + |1010\rangle |1101\rangle + |1011\rangle |1110\rangle \\ &+ |1100\rangle |0101\rangle + |1101\rangle |0000\rangle + |1110\rangle |1001\rangle + |1111\rangle |1010\rangle] \end{aligned}$$

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:

$$(H^{\otimes 4} \otimes I^{\otimes 4})\left(\frac{|0110\rangle + |1111\rangle}{\sqrt{2}} \otimes |1010\rangle\right)$$

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 (recall 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 independent 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 linear 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 as $s = 0000$. Let $s_4 = 1$ then we get, $s_1 = -1 \implies (-1) \bmod 2 = 1$. Thus, we get $s = 1001$ which is the correct solution.

1.7.6 Bernstein-Vazirani Algorithm

1.7.7 Shor's Algorithm

Definition 1.7.4. Given a number N which is a composite number. We are required to find the prime factors of the number N .

1.7.8 Grover's Algorithm

Grover's algorithm provides polynomial speedup over the classical algorithm for searching an unsorted database. It is optimal i.e. we can never find a quantum algorithm which is faster than Grover's algorithm for searching an unsorted database. No classical algorithm can outclass grover's algorithm. It has widespread applications in machine learning.

Definition 1.7.5. Given a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ which takes n bit inputs and produces 1 bit output. The output 0,1, indicates true or false i.e. 0 represents false, and 1 represents true. Our goal is to find a specific input x_0 such that $f(x_0) = 1$.

Recall, that we are not given the function but just an oracle U_f i.e. a black box which implements the classical function f on a quantum computer. The oracle is implemented as $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$. We are required to find the input x_0 such that $f(x_0) = 1$ using the oracle U_f by querying the oracle as minimum number of times as possible. We supply the oracle with an input and it returns the output of that function on that input. Its implemented as $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$.

Example 1.7.4. Say for $n=3$, we are given a function $f : \{0, 1\}^3 \rightarrow \{0, 1\}$ as shown in the table 1.24.

Input	Output
000	0
001	1
010	0
011	0
100	0
101	0
110	0
111	0

Table 1.24: Example of Grover's Algorithm

From all the possible input values the function outputs 1 for only one of the input and for all the other inputs it outputs 0. Thus, the input x_0 for which $f(x_0) = 1$ is $x_0 = 001$. We are required to find this input using Grover's algorithm.

On a Classical Computer: In the worst case, we are required to check all the possible inputs to the function to find the input x_0 for which $f(x_0) = 1$. Thus, the

time complexity of the classical algorithm is $\mathcal{O}(2^n)$ i.e. exponential time complexity. Even a randomized algorithm will not do better than $\mathcal{O}(2^{n/2})$.

On a Quantum Computer: Grover's algorithm provides a quadratic speedup over the classical algorithm. Thus, the time complexity of the Grover's algorithm is $\mathcal{O}(\sqrt{2^n})$. This is a polynomial speed up which any quantum or classical algorithm can't outclass. Grover's algorithm is optimal i.e. no quantum or classical algorithm can outclass Grover's algorithm.

Analysis of Grover's Algorithm:

The idea here is to make a superposition of all the inputs which yield output 1 (say $|A\rangle$) and make another superposition of all the inputs which yield output 0 (say $|B\rangle$). Then, upon measurement of $|A\rangle$ it will collapse to the inputs for which output is 1. Similarly, upon measurement of $|B\rangle$ it will collapse to the inputs for which the output was 0. Clearly, the vector $|A\rangle$ is orthogonal to the vector $|B\rangle$ because if a given entry is 1 in one of the vector then the corresponding entry in the other vector must be 0 (an input to the function cannot output both 0 and 1, it can be either 0 or 1). Now consider another vector $X = |h\rangle$ which is an equal superposition of all of the inputs irrespective of the output of the function. Thus, $|h\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$ which can be created with the help of Hadamard gate with inputs $|0\rangle^{\otimes n}$.

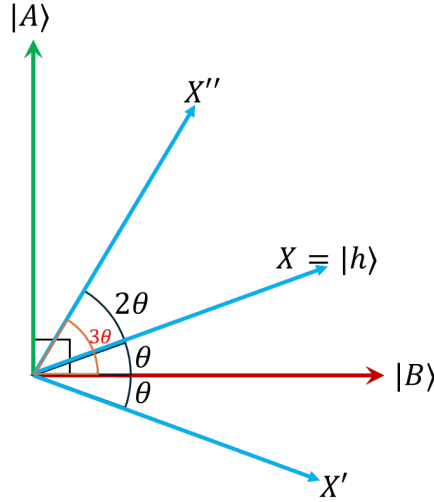


Figure 1.49: Grover's Algorithm

Say $|h\rangle$ makes an angle θ with $|B\rangle$. Now we wish to bring $|h\rangle$ close to $|A\rangle$ and farther from $|B\rangle$. So we reflect X about the vector $|B\rangle$ and obtain another vector (say X'). Now we reflect this vector X' about X and obtain another vector X'' which will

be close to $|A\rangle$. These rotations are called Grover's rotations. We repeat this process $\mathcal{O}(\sqrt{2^n})$ times to get the vector $|A\rangle$. This is as shown in the figure 1.49. Thus, we are now 3θ away from $|B\rangle$.

Grover's Algorithm

1. Create equal superposition of all possible inputs $X = |h\rangle$ using hadamard gate.
2. for $i=1$ to $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ do:
 - (a) $X = \text{Reflect } X \text{ over } |B\rangle$.
 - (b) $X = \text{reflect } X \text{ over } |h\rangle$.
3. Measure X .

This algorithm results in the correct answer with a very high probability. Thus, from the above algorithm we can see that the running time is $\mathcal{O}(\frac{\pi}{4}\sqrt{N} + 1) = \mathcal{O}(\sqrt{N})$ where $N = 2^n$. Note that 2^n is an exponentially large number and thus finding an input x_0 for which $f(x_0) = 1$ is also referred to as finding needle in a haystack.

Let set $A = x \in \{0, 1\}^n | f(x) = 1$ and $B = x \in \{0, 1\}^n | f(x) = 0$. Then, the vector $|A\rangle$ is orthogonal to the vector $|B\rangle$. For example, consider the function $f : \{0, 1\}^3 \rightarrow \{0, 1\}$ as shown in the table 1.25.

Input	Output
000	0
001	0
010	1
011	0
100	1
101	0
110	0
111	0

Table 1.25: Example of Grover's Algorithm

Thus, the set $A = \{010, 100\}$ and $B = \{000, 001, 011, 101, 110, 111\}$. Let the Cardinality i.e. the number of elements in the set represented as $|A| = a$ and $|B| = b$. Here, in this example, $a = 2$ and $b = 6$. Now, we define the vector $|A\rangle$ as:

$$|A\rangle = \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle = \frac{1}{\sqrt{2}}(|010\rangle + |100\rangle)$$

Similarly, we define the vector $|B\rangle$ as:

$$|B\rangle = \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle = \frac{1}{\sqrt{6}}(|000\rangle + |001\rangle + |011\rangle + |101\rangle + |110\rangle + |111\rangle)$$

Now, we define the vector $|h\rangle$ as an equal superposition of all the inputs:

$$\begin{aligned} |h\rangle &= H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^3}} \sum_{x \in \{0,1\}^3} |x\rangle = \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle) \\ &= \sqrt{\frac{a}{N}} |A\rangle + \sqrt{\frac{b}{N}} |B\rangle \end{aligned}$$

We can check this by substituting the definitions of $|A\rangle$ and $|B\rangle$ in the above equation as shown:

$$\begin{aligned} \sqrt{\frac{a}{N}} |A\rangle + \sqrt{\frac{b}{N}} |B\rangle &= \sqrt{\frac{2}{8}} \frac{1}{\sqrt{2}}(|010\rangle + |100\rangle) + \sqrt{\frac{6}{8}} \frac{1}{\sqrt{6}}(|000\rangle + |001\rangle + |011\rangle + |101\rangle + |110\rangle + |111\rangle) \\ &= \frac{1}{\sqrt{8}}(|010\rangle + |100\rangle) + \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |011\rangle + |101\rangle + |110\rangle + |111\rangle) \\ &= \frac{1}{\sqrt{8}}(|010\rangle + |100\rangle + |000\rangle + |001\rangle + |011\rangle + |101\rangle + |110\rangle + |111\rangle) \\ &= \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle) \\ &= |h\rangle \end{aligned}$$

Now, let R_B be the matrix that reflects X about B , and let R_h be the matrix that reflects X about h . Then, from Grover's algorithm we are required to first reflect X about B , thus $X = R_B |X\rangle$ and then reflect X about h i.e. $X = R_h |X\rangle$. Thus, we get $X = R_h R_B |X\rangle$ after the 1st iteration of the Grover's algorithm. Now from the figure 1.49 we can see that after first iteration the angle of rotation from B has incremented by 2θ thus, the final angle with $|B\rangle$ is $\theta + 2\theta = 3\theta$. After two iterations the angle with $|B\rangle$ will be $\theta + 2 \times 2\theta = \theta + 4\theta = 5\theta$. Thus, after k iterations the angle with $|B\rangle$ will be $\theta + 2k\theta = (2k + 1)\theta$.

Now, since the norm of $|h\rangle = 1$, and from the figure 1.49 the angle between $|h\rangle$ and $|B\rangle$ is θ thus, we can write $|h\rangle = \sin(\theta) |A\rangle + \cos(\theta) |B\rangle$. Now, recall that $|h\rangle = \sqrt{\frac{a}{N}} |A\rangle + \sqrt{\frac{b}{N}} |B\rangle$. Thus, upon comparing we get,

$$\sin(\theta) = \sqrt{\frac{a}{N}} \quad \text{and} \quad \cos(\theta) = \sqrt{\frac{b}{N}}$$

Now, assuming that $a \ll N$, hence using small angle approximation we get, $\sin \theta \approx \theta = \sqrt{\frac{a}{N}}$. Now, after k iterations the angle with $|B\rangle$ will be $(2k+1)\theta$. hence, we can write $|X\rangle = \sin((2k+1)\theta) |A\rangle + \cos((2k+1)\theta) |B\rangle$. To get the value of $|A\rangle$ with probability 1 after k iterations, we need $\sin^2(2k+1)\theta = 1 \implies (2k+1)\sqrt{\frac{a}{N}} = \frac{\pi}{2}$. On solving for the value of k we get, $k = \frac{\pi}{4}\sqrt{\frac{N}{a}} - \frac{1}{2}$. Thus, the number of iterations required to get the value of $|A\rangle$ with probability 1 is $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$.

Now, lets create a quantum circuit for reflection of X about $|B\rangle$. At any intermediate iteration we can write the state of $|X\rangle = \delta |A\rangle + \gamma |B\rangle$. To reflect the vector X about B we will get $|X\rangle = -\delta |A\rangle + \gamma |B\rangle$. (In general, we can reflect $|\mu\rangle$ about $|\pi\rangle$ by changing signs of components orthogonal to $|\pi\rangle$.) Now, consider the matrix

$$R_B = \begin{bmatrix} (-1)^{f(0)} & 0 & \dots & 0 & 0 \\ 0 & (-1)^{f(1)} & \vdots & 0 & 0 \\ 0 & 0 & (-1)^{f(2)} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & (-1)^{f(2^n-1)} \end{bmatrix}$$

This matrix flips only the components when the function $f(x) = 1$. Thus, the matrix R_B flips the sign of the components of $|X\rangle$ for which $f(x) = 1$. Thus, it reflects the vector $|X\rangle$ about $|B\rangle$. NOW writin the unitary tranformation for this matrix R_B we get:

$$R_B |x\rangle = |x\rangle (-1)^{f(x)}$$

This is implemented in an oracle as $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$. Thus, the matrix R_B is implemented as $U_{R_B} |x\rangle |-\rangle = (-1)^{f(x)} |x\rangle |-\rangle$. This implementation is as shown in the circuit diagram 1.50.

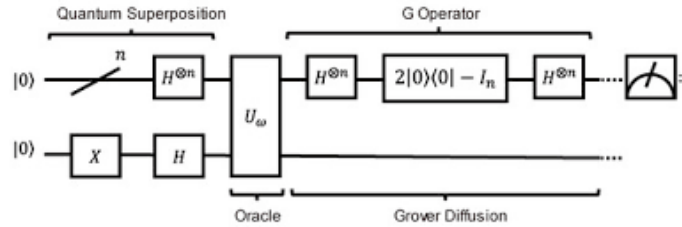


Figure 1.50: Grover's Algorithm

To verify that the Unitary matrix does that Operation

1.7.9 Quantum Fourier Transform

Quantum fourier transform is a quantum implementation of Discrete Fourier Transform.

Chapter 2

Advanced Quantum Computing

2.1 Linear Growth and Duhamel's Principle

Consider a Quantum Algorithm denoted 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 Unitary 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 2.1.1. *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:

$$\begin{aligned} U_K \dots U_2 U_1 - \tilde{U}_K \dots \tilde{U}_1 &= U_K \dots U_2 U_1 - U_K \dots U_2 \tilde{U}_1 + U_K \dots U_2 \tilde{U}_1 \\ &\quad - U_K \dots \tilde{U}_2 \tilde{U}_1 + \dots + U_K U_{K-1} \dots \tilde{U}_1 - \tilde{U}_K \dots \tilde{U}_1 \\ &= U_K \dots U_2 (U_1 - \tilde{U}_1) + U_K \dots U_3 (U_2 - \tilde{U}_2) \tilde{U}_1 + \dots \\ &\quad + U_K (U_{K-1} - \tilde{U}_{K-1}) \dots \tilde{U}_1 + (U_K - \tilde{U}_K) \dots \tilde{U}_1 \end{aligned}$$

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:

$$\begin{aligned} \|U_K \dots U_2 U_1 - \tilde{U}_K \dots \tilde{U}_2 \tilde{U}_1\| &\leq \|U_K \dots U_2 (U_1 - \tilde{U}_1)\| \\ &\quad + \|U_K \dots U_3 (U_2 - \tilde{U}_2) \tilde{U}_1\| + \dots + \|U_K (U_{K-1} - \tilde{U}_{K-1}) \dots \tilde{U}_1\| \\ &\quad + \|(U_K - \tilde{U}_K) \dots \tilde{U}_1\| \end{aligned}$$

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:

$$\begin{aligned} \|U_K \dots U_2 U_1 - \tilde{U}_K \dots \tilde{U}_2 \tilde{U}_1\| &\leq \|U_K \dots U_2\| \|U_1 - \tilde{U}_1\| \\ &\quad + \|U_K \dots U_3\| \|U_2 - \tilde{U}_2\| \|\tilde{U}_1\| + \dots + \|U_K\| \|U_{K-1} - \tilde{U}_{K-1}\| \dots \|\tilde{U}_1\| \\ &\quad + \|U_K - \tilde{U}_K\| \dots \|\tilde{U}_1\| \\ &\leq \|U_1 - \tilde{U}_1\| + \|U_2 - \tilde{U}_2\| \dots \|U_K - \tilde{U}_K\| \\ &\leq \sum_{i=1}^K \|U_i - \tilde{U}_i\| \\ &\leq K\epsilon \end{aligned}$$

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). \square

Proposition 2.1.2. (Duhamel's principle for Hamiltonian Simulation) Let $U(t), \tilde{U}(t) \in \mathbb{C}^{N \times N}$ satisfy

$$i\partial_t U(t) = HU(t), \quad i\partial_t \tilde{U}(t) = H\tilde{U}(t) + B(t), \quad U(0) = \tilde{U}(0) = I$$

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

$$\tilde{U}(t) = U(t) - i \int_0^t U(t-s)B(s)ds$$

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. \square

2.2 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 $U_1, \dots, U_m \in S$ such that

$$\|U - U_m \dots U_1\| \leq \epsilon$$

Some of the possible choices of the universal gate sets are $\{H, T, \text{CNOT}\}$, $\{H, \text{Toffoli}\}$

Theorem 2.2.1. (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 $\mathcal{O}(m \cdot \text{polylog}(m/\epsilon))$ gates from the gate set T , and there is a classical alogirhtm for finding thsi circuit in time $\mathcal{O}(m \cdot \text{polylog}(m\epsilon))$. (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 $x \rightarrow f(x)$ 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 $x, y \rightarrow x, f(x) \oplus y$. Thus, in particular we have $(x, 0) \rightarrow (x, f(x))$. 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
- Intermediate bits stored in some extra qubits are still entangled to the quantum state of interest. (If environment interferes with the intermediate result, then the quantum state of interest also gets affected.)

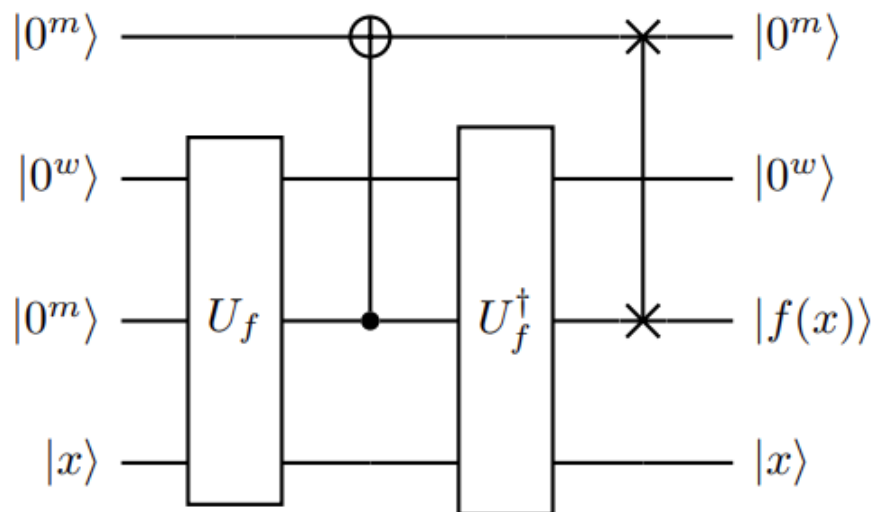


Figure 2.1: Uncomputation of a classical gate

Step 2: **Uncomputation:** To implement a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ assume that there is an oracle.

$$|0^m\rangle |x\rangle \rightarrow |f(x)\rangle |x\rangle$$

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

$$|0^m\rangle |g(x)\rangle |f(x)\rangle |x\rangle \rightarrow |f(x)\rangle |g(x)\rangle |f(x)\rangle |x\rangle$$

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, $U_f^\dagger = U_f^{-1}$,

$$(I_m \otimes U_f^\dagger) |f(x)\rangle |g(x)\rangle |f(x)\rangle |x\rangle = |f(x)\rangle |g(x)\rangle |f(x)\rangle |x\rangle = |f(x)\rangle |0^w\rangle |0^m\rangle |x\rangle$$

Then, 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 unchanged before and after the application of the circuit, though they are changed during the intermediate steps. Therefore 2.1 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 that 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 operation as

$$U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$$

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.

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

Note: This does not mean that we can efficiently implement the map $|x\rangle \rightarrow |f(x)\rangle$. If f is a bijection, and we have access to the inverse of the reversible circuit for computing f^{-1} then we may use the technique of uncomputation to implement such a map.

Example 2.2.1. Suppose there are three bits and we wish to apply AND gate twice and get the result in the classical setting as:

2.3 Fixed Point Number Representation and Classical Arithmetic operations

Any integer $k \in [N] = \{0, 1, \dots, N-1\}$ where $N = 2^n$ can be expressed as an n -bit string as $k = k_{n-1} \dots k_0$ 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 ($0 \leq m \leq n$) (note that the decimal is shifted to be after k_m).

$$a = \frac{k}{2^m} = \sum_{i \in [n]} k_i 2^{i-m} = (k_{n-1} \dots k_m . k_{m-1} \dots k_0)$$

when $m=n$,

$$a = \frac{k}{2^n} = \sum_{i \in [n]} k_i 2^{i-n} = 0.k_{n-1} \dots k_0$$

$a = 0.k_{n-1} \dots k_0$ so that k_i is the i th decimal of a in the binary representation. Note that here $0 \leq a \leq 1$. The number $(0.k_{n-1} \dots k_i)$ 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 $O(\text{poly}(n))$.** Thus, quantum computer is theoretically as powerful as classical computers, there is a very significant overhead in implementing reversible classical circuits on quantum devices, both in terms of the number of ancilla qubits and the circuit depth.

2.4 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 difficulties quantum gates, and measurements involve significant errors on near-term quantum devices. **Threshold theorem: If the noise in individual quantum gates is below a certain constant threshold (around 10^{-4} or above) it is possible to perform an arbitrarily large computation with any desired precision. This requires quantum error correction protocols.**

2.5 Complexity of Quantum Algorithms

Let n be the number of input bits. **Efficiency of Quantum Algorithm:** A Quantum Algorithm is efficient if the number of gates in the quantum circuit is $O(\text{poly}(n))$. 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 $p > 1/2 + 1/\text{poly}(n)$. Consider 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 repetitions 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 $X_i = 1$):

$$P\left(\sum X_i \leq n/2\right) \leq e^{-2n\epsilon^2}$$

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 2.5.1. (Query access to boolean function) $f : \{0,1\}^n \rightarrow \{0,1\}$ 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 2.5.2. (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 registers - storing quantum states of interest and ancilla registers - auxillary registers needed to implement the

unitary operations acting on system registers) starting from an initial 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 asymptotic 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 contains all the pre-requisites required for understanding the contents of the given Book. The following references [Nielsen and Chuang(2001)], [de Wolf(2023)], [Shankar(2012)] and [Lin(2022)] have been extensively used for the preparation of 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 Uncertainty Principle) but still the important concepts of Quantum Mechanics are explained up to the required depth whenever it is found to be necessary for the explanation 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 $T : V \rightarrow W$ 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 $T : \mathbb{F}^n \rightarrow \mathbb{F}^m$ defined by $T(|v\rangle) = A|v\rangle$ is a linear transformation.

Example A.1.2. Differentiation, Integration, Laplace transform are examples of linear transformations.

Definition A.1.2. Composition Transformation: Let $T : V \rightarrow W$ and $U : W \rightarrow X$ be linear transformations. The composition transformation $UT : V \rightarrow X$ 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 $T : V \rightarrow W$ be a linear transformation. A linear transformation $T^{-1} : W \rightarrow V$ 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 field 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 Mechanics 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 \mathbf{v} called as ket- \mathbf{v} and the symbol $\langle v|$ represents a row vector called as bra- \mathbf{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, picture of arrow comes to our mind but arrows are just one of the manifestation 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 $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ be vectors in V . A vector of the form

$$a_1 |v_1\rangle + a_2 |v_2\rangle + \dots + a_n |v_n\rangle \quad (\text{A.1})$$

where $a_1, a_2, \dots, a_n \in \mathbb{F}$ is called a linear combination of $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$

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 $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ be vectors in V . The set of all linear combinations of $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ is called the span of $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ and is denoted by $\text{Span}(|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle)$

Definition A.1.9. Linear Independence: Let V be a vector space over a field \mathbb{F} and let $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ be vectors in V . The vectors $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ are said to be linearly independent if the only solution to the equation

$$a_1 |v_1\rangle + a_2 |v_2\rangle + \dots + a_n |v_n\rangle = |0\rangle \quad (\text{A.2})$$

is trivial solution $a_1 = a_2 = \dots = a_n = 0$

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 $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ in V is called a basis of V if

1. $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ are linearly independent
2. $\text{Span}(|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle) = V$

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 indepede 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:

$$a|v\rangle = a \sum_{i=1}^n v_i |i\rangle = \sum_{i=1}^n (av_i) |i\rangle$$

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 occurs 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 $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ of V is called an orthonormal basis if

1. $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ are orthogonal
2. $\| |v_i\rangle \| = \sqrt{\langle v_i|v_i \rangle} = 1$ for all $i = 1, 2, \dots, n$

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 orthonormal 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 i th row position.

Then the inner product of $|u\rangle$ and $|v\rangle$ is given by

$$\langle u|v\rangle = (u_1^* \ u_2^* \ \dots \ u_n^*) \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = (u_1^* \ u_2^* \ \dots \ u_n^*) \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \sum_{i=1}^n u_i^* v_i$$

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 vectors: $\langle i| = (0 \ \dots \ 0 \ 1 \ 0 \ \dots \ 0)$ where 1 is in the i th column position. Thus we have a vector space of $|v\rangle$ and another vector space of $\langle v|$, these 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 $|u_i\rangle$: $|u_i\rangle = \frac{|u_i\rangle}{\|u_i\|}$

Proof. Proof that Gram-Schmidt 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 $i, j = 1, 2, \dots, k$. Thus, we need to show that $\langle u_i | u_{k+1} \rangle = 0$ for $i = 1, 2, \dots, k$. 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,

$$= \langle u_i | v_{k+1} \rangle - \langle u_i | v_{k+1} \rangle \langle u_i | u_i \rangle$$

Using the fact that $|u_i\rangle$ is a unit vector (from inductive hypothesis that $|u_1\rangle, \dots, |u_k\rangle$ is an orthonormal basis), we have

$$\begin{aligned} &= \langle v_{k+1} | u_i \rangle - \langle u_i | v_{k+1} \rangle \\ &= 0 \end{aligned}$$

Thus, $|u_{k+1}\rangle$ is orthogonal to $|u_1\rangle, |u_2\rangle, \dots, |u_k\rangle$. 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 $|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle$ 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,

$$\begin{aligned} |Z\rangle &= |V\rangle - \frac{\langle W | V \rangle}{|W|^2} |W\rangle \\ \langle Z| &= \langle V| - \frac{\langle V | W \rangle}{|W|^2} \langle W| \end{aligned}$$

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 antilinearity 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. ($\langle W|W \rangle = |W|^2$) 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. \square

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

$$\langle Z|Z \rangle = |Z|^2 = \langle V|V \rangle + \langle W|W \rangle + \langle V|W \rangle + \langle W|V \rangle$$

$$|V + W|^2 = |V|^2 + |W|^2 + 2\text{Re}(\langle V|W \rangle)$$

Now using, $\text{Re}(\langle V|W \rangle) \leq |\langle V|W \rangle|$, 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. \square

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 $A : V \rightarrow W$ 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 $|1\rangle, |2\rangle, \dots, |n\rangle$ 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 $|1'\rangle, |2'\rangle, \dots, |n'\rangle$ 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 $|1\rangle, |2\rangle, \dots, |n\rangle$ of Vector space V and a linear operator A and its transformed basis vectors $|1'\rangle, |2'\rangle, \dots, |m'\rangle$ 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 $|1\rangle, |2\rangle, \dots, |n\rangle$ and let W be a vector space over \mathbb{F} with basis $|1'\rangle, |2'\rangle, \dots, |m'\rangle$. Let A be a linear operator from V to W . The matrix representation of A with respect to the basis $|1\rangle, |2\rangle, \dots, |n\rangle$ of V and the basis $|1'\rangle, |2'\rangle, \dots, |m'\rangle$ of W is the $m \times n$ matrix $[A]$ whose elements are given by

$$[A]_{ij} = \langle i' | A | j \rangle$$

which is the i th row and j th 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 output basis for the matrix to map. In general, the input and output basis are the same and they are the same standard orthonormal basis that we use (i.e. $|1\rangle = 1, 0, 0, \dots, 0$, $|2\rangle = 0, 1, 0, \dots, 0$ and so on). Thus, the matrix representation of the operator in this basis will be given as follows:

$$[A] = \begin{pmatrix} \langle 1'|A|1\rangle & \langle 1'|A|2\rangle & \dots & \langle 1'|A|n\rangle \\ \langle 2'|A|1\rangle & \langle 2'|A|2\rangle & \dots & \langle 2'|A|n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle m'|A|1\rangle & \langle m'|A|2\rangle & \dots & \langle m'|A|n\rangle \end{pmatrix}$$

this indicates that the input basis here is chosen as $|1\rangle, |2\rangle, \dots, |n\rangle$ and the output basis is chosen as $|1'\rangle, |2'\rangle, \dots, |m'\rangle$. Note that the j th column represents the components of the j th 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 j th basis vector $|j\rangle$ after being transformed by the action of operator A to $A|j\rangle$ on the i th 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 output basis, we assume that the input and output 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 output basis. Note that to make the connection between matrices and linear operators one must specify a set of input and output 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 output bases. We also use the convention that if the input and output spaces for a linear operator are the same, then the input and output 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 i th basis vector $|i\rangle$. Thus, $v_i = \langle i|v\rangle$. On substituting we get,

$$|v\rangle = \sum_{i=1}^n \langle i|v\rangle |i\rangle$$

Rearranging 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, $P^2 = P$ property must be satisfied 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 i th position in the diagonal and zero everywhere 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 projection operators 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 products. Suppose $A : V \rightarrow W$ is a linear operator with $|v_i\rangle$ as orthonormal 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 coefficients is encountered, reverse the order of all factors and make the substitutions $A \rightarrow A^\dagger$ and $| \rangle \rightarrow \langle |$ and $\alpha \rightarrow \alpha^*$.

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 principal diagonal elements will be $a_{ii} = a_{ii}^*$, thus the principal diagonal elements of a Hermitian operator are real.
- Principal Diagonal 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 $P + iQ$ 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:

$$U^\dagger U = U U^\dagger = I$$

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 $U_1 U_2$ is also unitary.

$$(U_1 U_2)^\dagger U_1 U_2 = U_2^\dagger U_1^\dagger U_1 U_2 = U_2^\dagger U_2 = I$$

- 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 orthonormal 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 Unitary 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 Diagonalizable

Definition A.1.21. Unitarily Diagonalizable: A matrix A is said to be unitarily diagonalizable if there exists a unitary matrix U such that $U^\dagger AU = D$ where D is a diagonal matrix. Here, U is a unitary matrix 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 diagonalizable.

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 diagonalizable or if a matrix is Unitarily diagonalizable 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 diagonalizable. **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 :

$$\begin{aligned} A_H &= V\Lambda_H V^\dagger \\ A_{SH} &= V\Lambda_{SH} V^\dagger \end{aligned}$$

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 diagonal matrix with entries as the corresponding eigen values. Hence, proved. **Thus the statement that a matrix is normal if and only if it is unitarily diagonalizable.** \square

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

$$\begin{aligned} A |v\rangle &= \lambda |v\rangle \\ \implies (A - \lambda I) |v\rangle &= 0 \end{aligned}$$

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:

$$P_A(\lambda) = \det(A - \lambda I) = \lambda^n - \beta_1 \lambda^{n-1} + \dots + (-1)^r \beta_r \lambda^{n-r} + \dots + (-1)^n \beta_n = 0$$

In general, we represent eigen value as ω and the corresponding eigen vector as $|\omega\rangle$.

Example A.1.16. For the identity operator I , the eigen value equation is given as:

$$\begin{aligned} I |v\rangle &= \lambda |v\rangle \\ \implies |v\rangle &= \lambda |v\rangle \\ \implies \lambda &= 1 \end{aligned}$$

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 $|v\rangle$ respectively.

Example A.1.18. Consider the Rotation operator $R(\vec{\theta})$ which rotates any vector by an angle $\frac{\vec{\theta}}{|\vec{\theta}|}$ 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 orthongol (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 product with $|v\rangle$, we get:

$$\langle v| A^\dagger |v\rangle = \lambda^* \langle v|v\rangle$$

$$\langle v| A |v\rangle = \lambda^* \langle v|v\rangle \quad (\text{since } A \text{ is Hermitian, } A = A^\dagger)$$

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 independent 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 multiplicity 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 geometric 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 $\{|v_1\rangle, |v_2\rangle, \dots, |v_r\rangle\}$) corresponding to λ . Let,

$$V^E = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ |v_1\rangle & |v_2\rangle & \dots & |v_r\rangle \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

which is the set of Linearly Independent Orthogonal Eigen vectors. Let,

$$V^\perp = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ |v_{r+1}\rangle & |v_{r+2}\rangle & \dots & |v_n\rangle \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

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,

$$V = [V^E \quad V^\perp] = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ |v_1\rangle & |v_2\rangle & \dots & |v_n\rangle \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Now $A|v_i\rangle = \lambda|v_i\rangle$, for $i = 1, 2, \dots, r$. Thus,

$$AV = \begin{bmatrix} \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \lambda|v_1\rangle & \lambda|v_2\rangle & \dots & \lambda|v_r\rangle & A|v_{r+1}\rangle & \dots & A|v_n\rangle \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

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 $i = r+1, r+2, \dots, n$. Thus, in matrix form we can write this as:

$$A \begin{bmatrix} \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ |v_1\rangle & |v_2\rangle & \dots & |v_r\rangle & |v_{r+1}\rangle & \dots & |v_n\rangle \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} = V \begin{bmatrix} \lambda_1 & & & & & & c_{r+1,1} & \dots & c_{n,1} \\ & \lambda_2 & & & & & c_{r+1,2} & \dots & c_{n,2} \\ & & \lambda_3 & & & & c_{r+1,3} & \dots & c_{n,3} \\ & & & \ddots & & & \vdots & \vdots & \vdots \\ & & & & \lambda_r & & c_{r+1,r} & \dots & c_{n,r} \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,r+1} & \dots & c_{n,r+1} \\ 0 & 0 & 0 & \dots & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,n} & \dots & c_{n,n} \end{bmatrix}$$

$$AV = VM$$

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 eigen 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,

$$\begin{aligned} V^\dagger AV \hat{u} = M \hat{u} &= \begin{bmatrix} \lambda & & & & 0 & \dots & 0 \\ & \lambda & & & 0 & \dots & 0 \\ & & \lambda & & 0 & \dots & 0 \\ & & & \ddots & \vdots & \vdots & \vdots \\ & & & & \lambda & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,r+1} & \dots & c_{n,r+1} \\ 0 & 0 & 0 & \dots & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & c_{r+1,n} & \dots & c_{n,n} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ u_{r+1} \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \lambda u_{r+1} \\ \vdots \\ \lambda u_n \end{bmatrix} \\ &\Rightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \lambda u_{r+1} \\ \vdots \\ \lambda u_n \end{bmatrix} = \lambda \begin{bmatrix} 0 \\ \vdots \\ 0 \\ u_{r+1} \\ \vdots \\ u_n \end{bmatrix} = \lambda \hat{u} \end{aligned}$$

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 $\{|v_{r+1}\rangle, \dots, |v_n\rangle\}$ which contradicts the assumption that $\{|v_1\rangle, \dots, |v_r\rangle\}$ 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.** \square

Spectral Decomposition

Theorem A.1.8. *Any normal operator M on a vector space V is diagonal with respect to some orthonormal basis for V . Conversely, any diagonalizable 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 matrix 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:

$$\{A, B\} = AB + BA$$

Properties of the Commutator

- $AB = \frac{[A, B] + \{A, B\}}{2}$
- 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 basis such that both A and B are diagonal with respect to that basis. We can say that A and B are simultaneously diagonalizable.

Operator Functions

We restrict ourselves to functions 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:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Then the exponential of an operator A is defined as:

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{A^i}{i!}$$

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:

$$e^A = \begin{pmatrix} \sum_{i=1}^{\infty} \frac{\lambda_1^i}{i!} & 0 & \dots & 0 \\ 0 & \sum_{i=1}^{\infty} \frac{\lambda_2^i}{i!} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sum_{i=1}^{\infty} \frac{\lambda_n^i}{i!} \end{pmatrix}$$

$$e^A = \begin{pmatrix} e^{\lambda_1} & 0 & \dots & 0 \\ 0 & e^{\lambda_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{\lambda_n} \end{pmatrix}$$

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 $\det(e^{\iota A}) = e^{\iota \text{Tr}(A)}$.

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 $\text{Tr}(A)$.

$$\text{Tr}(A) = \sum_{i=1}^n a_{ii}$$

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 ($\mathbb{C}^m \rightarrow \mathbb{R}$) 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:

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

where $x = (x_1, x_2, \dots, x_n)$ 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:

$$\|x\|_2 = \left(\sum_{i=1}^n |x_i|^2 \right)^{\frac{1}{2}}$$

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 Conclusions

- **A Normal operators ($AA^\dagger = A^\dagger A$) are always unitarily diagonalizable (i.e. $A = U^\dagger D U$ 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 diagonalizable and has real eigen values and n orthogonal linearly independent eigen vectors. The Unitary matrix is also unitarily diagonalizable and has complex eigen values of unit modulus and n orthogonal linearly independent eigen vectors. Since both the Hermitian and Unitary matrices are unitarily diagonalizable 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,

$$\begin{aligned} A|v\rangle &= \lambda|v\rangle \\ \implies BA|v\rangle &= \lambda B|v\rangle \\ \implies AB|v\rangle &= \lambda B|v\rangle \quad (\text{Since } AB=BA) \end{aligned}$$

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 corresponding 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 for $V \otimes W$. 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 $V \otimes W$.

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 define 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 extended to all the elements of $V \otimes W$ in the natural way to ensure linearity of $A \otimes B$, that is,

$$(A \otimes B)\left(\sum_i a_i |v_i\rangle \otimes |w_i\rangle\right) = \sum_i a_i A|v_i\rangle \otimes B|w_i\rangle$$

Consider the case where $A : V \rightarrow V'$ and $B : W \rightarrow W'$ 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

$$\left(\sum_i a_i |v_i\rangle \otimes |w_i\rangle, \sum_j b_j |v'_j\rangle \otimes |w'_j\rangle \right) \equiv \sum_i a_i b_i \langle v_i | v'_i \rangle \langle w_i | w'_i \rangle$$

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 $a_{11}B$ 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 n 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:

$$\begin{aligned} |\alpha\rangle \langle\beta| &= (3|0\rangle + \iota|1\rangle)(\langle 00| + 2\langle 10| + 7\langle 11|) \\ &= 3|0\rangle \langle 00| + 6|0\rangle \langle 10| + 21|0\rangle \langle 11| + \iota|1\rangle \langle 00| + 2\iota|1\rangle \langle 10| + 7\iota|1\rangle \langle 11| \end{aligned}$$

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 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:

$$|\psi\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$$

Thus,

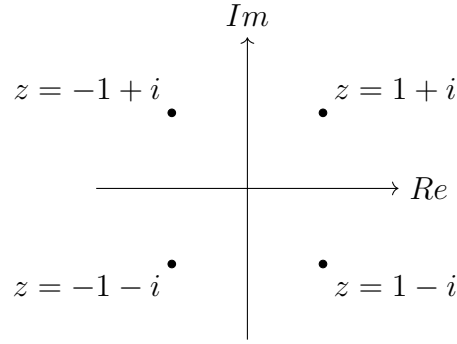
$$ac = \frac{1}{\sqrt{2}}, ad = 0, bc = 0, bd = \frac{1}{\sqrt{2}}$$

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: $z_1(z_2 + z_3) = z_1z_2 + z_1z_3$

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: $z 1 = 1 z = z$
- 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: $1 z = z$
- 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:

- $(z_1 + z_2)^* = z_1^* + z_2^*$
- $(z_1 z_2)^* = z_1^* z_2^*$
- $(z^*)^* = z$
- $z + z^* = 2 \operatorname{Re}(z)$

- $z - z^* = 2i\text{Im}(z)$
- $zz^* = |z|^2$
- $\frac{1}{z} = \frac{z^*}{zz^*}$
- $z^{-1} = \frac{z^*}{|z|^2}$

A.3.5 Modulus of a Complex Number

The modulus of a complex number $z = a + ib$ 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:

- $|z|^2 = zz^*$
- $|z_1 z_2| = |z_1| |z_2|$
- $|z_1 + z_2| \leq |z_1| + |z_2|$
- $|z_1 - z_2| \geq ||z_1| - |z_2||$
- $|z_1 - z_2|^2 = |z_1|^2 + |z_2|^2 - 2\text{Re}(z_1 z_2^*)$
- $|z_1 + z_2|^2 = |z_1|^2 + |z_2|^2 + 2\text{Re}(z_1 z_2^*)$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 2(|z_1|^2 + |z_2|^2)$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 4|z_1|^2 |z_2|^2$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 4|z_1 z_2|^2$
- $|z_1 + z_2|^2 + |z_1 - z_2|^2 = 4|z_1|^2 |z_2|^2$

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

$z = a + \iota b = r e^{\iota \theta}$. Recall that the expansion of e^x using Taylor series can be written as:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Now substituting $\iota \theta$ in place of x , we get:

$$e^{\iota \theta} = 1 + \iota \theta + \frac{(\iota \theta)^2}{2!} + \frac{(\iota \theta)^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{(\iota \theta)^i}{i!}$$

$$e^{\iota \theta} = 1 + \iota \theta - \frac{\theta^2}{2!} - \iota \frac{\theta^3}{3!} + \dots = \cos(\theta) + \iota \sin(\theta)$$

On rearranging the terms we get:

$$e^{\iota \theta} = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \dots + \iota \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots \right)$$

Now recall that the expansion of \sin and \cos using Taylor series expansion are:

$$\sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots = \sum_{i=0}^{\infty} \frac{(-1)^i \theta^{2i+1}}{(2i+1)!}$$

$$\cos(\theta) = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \dots = \sum_{i=0}^{\infty} \frac{(-1)^i \theta^{2i}}{(2i)!}$$

Thus, Euler's formula states that $e^{i\theta} = \cos(\theta) + i \sin(\theta)$. Hence, on substituting this in the polar form of a complex number we get:

$$z = a + ib = \sqrt{a^2 + b^2} \left(\cos \left(\tan^{-1} \frac{b}{a} \right) + i \sin \left(\tan^{-1} \frac{b}{a} \right) \right) = r(\cos(\theta) + i \sin(\theta)) = re^{i\theta}$$

where $r = |z|$ and $\theta = \tan^{-1}(\frac{b}{a})$. Using, this we can also write:

$$\begin{aligned} \cos \theta &= \frac{e^{i\theta} + e^{-i\theta}}{2} \\ \sin \theta &= \frac{e^{i\theta} - e^{-i\theta}}{2i} \end{aligned}$$

A.3.8 Roots of Unity

The n th roots of unity are the solutions to the equation $z^n = 1$. The n th roots of unity are given by:

$$\omega_n^k = e^{\frac{2\pi i k}{n}}$$

where k is an integer such that $0 \leq k \leq n-1$. The n th roots of unity are equally spaced around the unit circle in the complex plane (can be shown since their modulus is one).

Properties:

- $\omega_n^k = \omega_n^{(k \bmod n)}$ where $k \bmod n$ is the remainder when k is divided by n . For example, the 4th roots of unity are $\omega_4^0 = e^{\frac{2\pi i 0}{4}} = 1$, $\omega_4^1 = e^{\frac{2\pi i 1}{4}} = i$, $\omega_4^2 = e^{\frac{2\pi i 2}{4}} = -1$, $\omega_4^3 = e^{\frac{2\pi i 3}{4}} = -i$. 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 n th roots of unity. The i, j th element 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 .

$$F_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \iota & -1 & \iota \\ 1 & -1 & 1 & -1 \\ 1 & \iota & -1 & \iota \end{pmatrix}$$

$$|\psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$F_4 |\psi\rangle = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \iota & -1 & \iota \\ 1 & -1 & 1 & -1 \\ 1 & \iota & -1 & \iota \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{2\sqrt{2}} \begin{bmatrix} 2 \\ 1 - \iota \\ 0 \\ 1 + \iota \end{bmatrix}$$

$$= \frac{2|0\rangle + (1 - \iota)|1\rangle + (1 + \iota)|3\rangle}{2\sqrt{2}}$$

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 DFT

- DFT matrix is a unitary matrix.

Proof. To prove that DFT is a unitary matrix we are required to prove that any two columns of DFT matrix are orthonormal i.e.

$$\langle C_i | C_j \rangle = \delta_{ij}$$

where C_i and C_j are the columns of the DFT matrix.

Let C_i and C_j be the i th and j th columns of the DFT 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 . 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 $\omega_n^{kn} = 1$, we get: Thus, the DFT matrix is a unitary matrix. \square

- DFT Matrix when acts on a linearly shifted vector introduces a phase shift in the frequency domain. Consider the following DFT 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 DFT 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 shift

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 DFT 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 DFT matrix when acts on a linearly shifted vector introduces a phase shift in the output. \square

- **Period and Wavelength Relationship:**

Consider a periodic function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ 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.

$$QFT f(x) = \hat{f}(x)$$

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 N . 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:

$$\frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \vdots \\ \psi_r \\ \psi_0 \\ \psi_1 \\ \vdots \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{\frac{2^n}{r}-1} \\ \beta_0 \\ \beta_1 \\ \vdots \end{pmatrix}$$

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 $f : \{0, 1\}^k \rightarrow \{0, 1\}^l$ 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 avoid instabilities. Thus, they are Acyclic circuits. This is the Circuit Model of Computation. Some of the useful Logic Gates are:

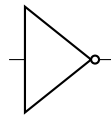
B.1 Logic Gates

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:

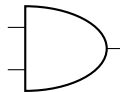
$$f(x) = \neg x$$

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:

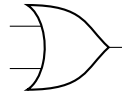
$$f(x, y) = x \wedge y$$

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 . Its symbol is as shown:



Mathematically, the OR gate is defined as:

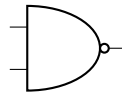
$$f(x, y) = x \vee y$$

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

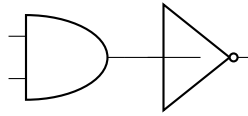
The NAND gate is denoted by the symbol \uparrow . Its symbol is as shown:



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:

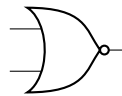


B.1.5 NOR Gate

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

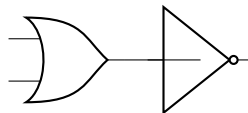
The NOR gate is denoted by the symbol \downarrow . It's symbol is as shown:



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:

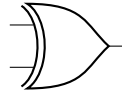


B.1.6 XOR Gate

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 . Its 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

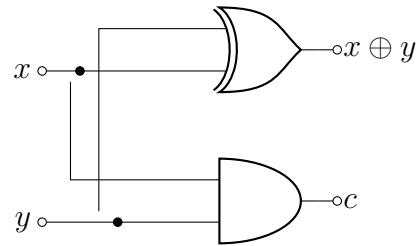


Figure B.1: Half Adder

We can use half-adders to make a full-adder as shown in figure: B.2 As shown in

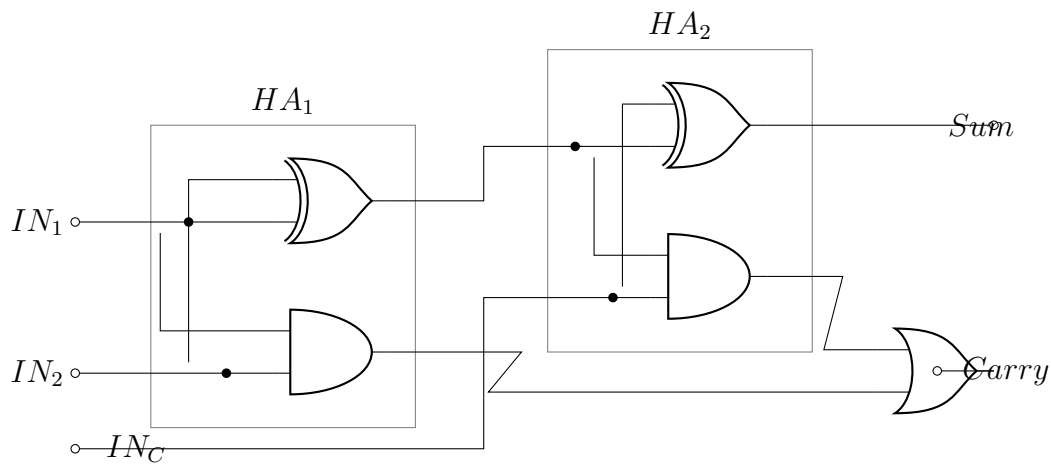


Figure B.2: Full Adder

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 $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$. Consider the proof for the simplified function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ with n input bits and a single output bit. This is a Boolean function. Using mathematical induction, 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 $f_0 = f(x_1, x_2, \dots, x_n, 0)$ and $f_1 = f(x_1, x_2, \dots, x_n, 1)$. 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 dependending 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.

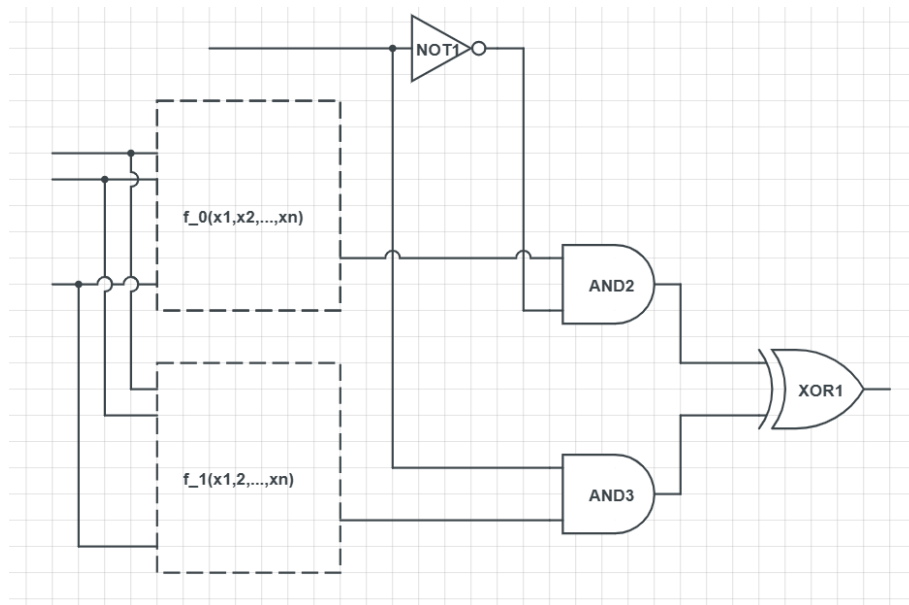


Figure B.3: Circuit to compute f

□

For the Universal Circuit Construction:

- wires, which preserve the states of the bits;
- ancilla bits prepared in the standard states, used in the $n=1$ case of the proof;
- FANOUT operation, which takes a single bit as input and outputs two copies of that bit;
- The CROSSOVER operation, 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 wires 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.

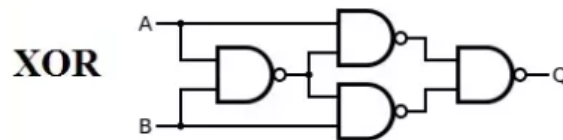


Figure B.5: NAND gate used to simulate XOR gate

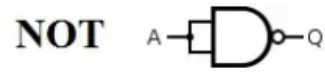


Figure B.6: NAND gate used to simulate NOT gate

Bibliography

- [de Wolf(2023)] Ronald de Wolf. Quantum computing: Lecture notes, 2023.
- [Lin(2022)] Lin Lin. Lecture notes on quantum algorithms for scientific computation, 2022.
- [Nielsen and Chuang(2001)] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*, volume 2. Cambridge university press Cambridge, 2001.
- [Shankar(2012)] Ramamurti Shankar. *Principles of quantum mechanics*. Springer Science & Business Media, 2012.