Quantum Computing

An Unofficial Guide to the Georgia Institute of Technology's CS7400

Christian Salas
csalas9@gatech.edu

Last Updated: April 30, 2025

These notes were created with personal effort and dedication. They may contain typos, errors, or incomplete sections. If you find them helpful or wish to provide feedback, feel free to reach out.

Have fun!

Disclaimer: I am a student, not an expert. While I aim for accuracy, there may be mistakes. If you spot any issues, please contribute or contact me.

Contents

1 Introduction to Quantum Computing

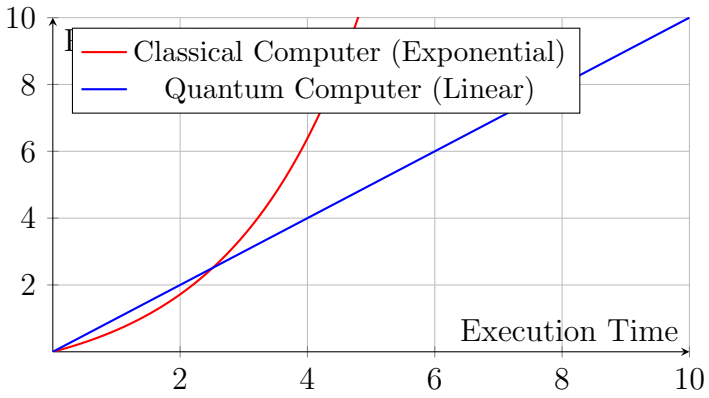
Factoring a 2048-bit number is a computational challenge. A classical computer the size of Germany, using all the Earth's energy, would take hundreds of years to solve this problem.

A quantum computer can factor a 2048-bit number in just 8 hours.

1.1 Potential of Quantum Computers

Quantum computers have transformative potential in solving complex problems across various domains:

1. **Logistics:** Optimizing supply chains and transportation networks.
2. **Machine Learning:** Accelerating data analysis and AI training.
3. **Drug Discovery:** Simulating molecular structures for faster breakthroughs.
4. **Optimization Problems:** Tackling NP-hard problems efficiently.
5. **Algebraic Problems:** Solving systems of equations and factoring.
6. **Quantum Simulation:** Modeling quantum systems in physics and chemistry.



1.2 Classical vs. Quantum Bits

1. **Classical Bit:** A classical bit represents two possible states: 0 or 1, like points at the North and South poles of a sphere.
2. **Quantum Bit (Qubit):** A qubit can represent **any point** on a sphere, allowing superposition and vastly increasing computational possibilities.

1.3 Computational Models

1. **Classical Machines:** Operate as state machines, transitioning from state S_1 to S_2.
2. **Quantum Machines:** Use probabilistic states, transitioning from probability P_1 to P_2.

1.4 How Are Qubits Made?

Qubits are created using advanced physical systems:

- **Ion-Trap Qubits:** Isolated ions manipulated by electromagnetic fields.
- **Quantum-Dot-Based Qubits:** Electrons confined in nanoscale structures.
- **Superconducting Qubits:** Circuits cooled to near absolute zero to exploit superconductivity.

1.5 Quantum Computing Model

Classical computing consists of multiple interconnected layers:

- **Algorithms**
- **Programming Languages (PL) and Compilers**
- **Architecture**
- **Circuits**
- **Devices**

In this course, we will focus on the **Programming Languages (PL)** and **Architecture** of quantum states, which lie between:

1. The mathematical foundations behind quantum algorithms.
2. The physics driving the interfaces and quantum devices.

1.6 Quantum Control and Computer

A quantum computer consists of two main components:

1. **Qubits:** The fundamental units of quantum information.
2. **Control Computer:** This system manipulates the qubit states and coordinates their operations.

1.7 The Biggest Challenge: Errors

- **Qubit Errors:** Qubits are highly sensitive and prone to errors, often leaking information.
- **Quantum Gate Errors:** Quantum gates have a non-negligible error rate, typically between 0.1% and 1%.

Quantum Error Correction

Quantum error correction is crucial, but extremely resource-intensive.
It requires:

- Between 10 and 500 physical qubits to construct a single fault-tolerant logical qubit.

Current and Future Approaches

- In the near term, we rely on **Noisy Intermediate-Scale Quantum (NISQ)** devices to perform computations.
- In the future, **Fault-Tolerant Quantum Computing (FTQC)** will pave the way for scalable and reliable quantum systems.

1.8 NISQ (Noisy Intermediate-Scale Quantum)

The process for running a quantum program on a NISQ device involves the following steps:

1. Write the **Quantum Program** (e.g., `qft.qc`).
2. Use a **Compiler** to generate a **Quantum Executable (QEXE)**.
3. Execute the **QEXE** on the **Quantum Computer**.
4. Receive the output (e.g., `1010`).

Since NISQ devices are prone to errors, the quantum executable is repeated for N trials. The final answer is derived by:

- Identifying the most frequently occurring outcome, which is assumed to be error-free.

Key Insight: Despite qubit errors, NISQ devices can produce correct outcomes through repeated execution and statistical analysis.

Compiler Policies

Existing compiler policies for NISQ devices prioritize:

- Minimizing the use of **SWAP gates**, which introduce additional errors due to qubit movement constraints.

2 Superposition and Single Qubit ψ

2.1 Understanding the Elements of a Single Qubit

A single qubit can be mathematically represented as:

$$\psi = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \underbrace{\alpha|0\rangle + \beta|1\rangle}_{\text{Dirac notation}},$$

where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

- $|\psi\rangle$**: The Qubit State**
  Read as *"ket-psi"* represents the quantum state of the qubit. It is a linear combination (or superposition) of the basis states $|0\rangle$ and $|1\rangle$, where:
  - α: The probability (**likelihood**) amplitude associated with the $|0\rangle$ state.
  - β: The probability (**likelihood**) amplitude associated with the $|1\rangle$ state.

  Both α and β are **complex numbers** that satisfy the **normalization condition**:

$$|\alpha|^2 + |\beta|^2 = 1.$$

2.1.1 Hilbert Space

A **Hilbert space** is a mathematical framework at the heart of quantum mechanics. It is a vector space equipped with an inner product, which **allows us to measure lengths and angles between vectors**. This structure provides the foundation for representing and analyzing quantum states.

In quantum computing, the state of a single qubit is represented as a vector in a **two-dimensional Hilbert space**. The basis states $|0\rangle$ and $|1\rangle$ form the computational basis of this space, meaning any qubit state can be expressed as a combination of these two vectors.

Example:
Consider a single qubit state:

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

where α and β are complex numbers satisfying the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1.$$

Using the **computational basis**, the basis vectors are defined as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

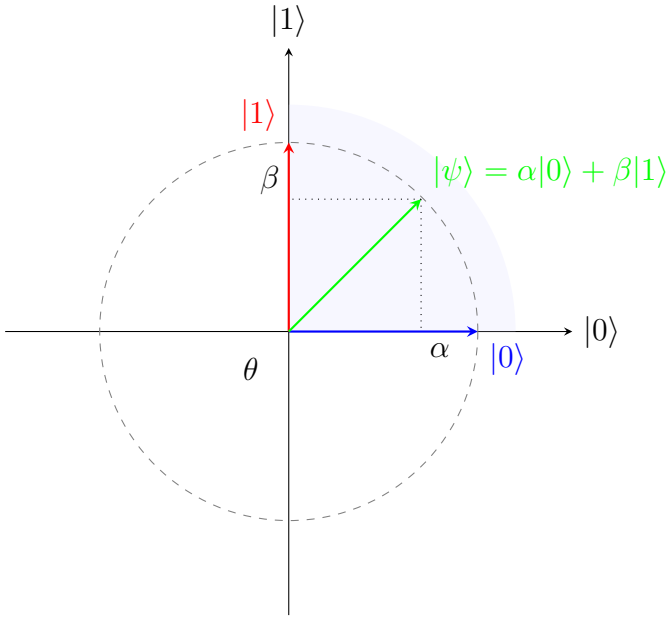
For instance, if $\alpha = \frac{\sqrt{3}}{2}$ and $\beta = \frac{1}{2}$, the qubit state is:

$$|\psi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle = \begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}.$$

This example demonstrates how a qubit state exists as a vector in the Hilbert space.

Visualizing a Hilbert Space for a Single Qubit

Below is a geometric visualization of a two-dimensional Hilbert space for a single qubit:



Explanation:

- The **shaded area** shows the space spanned by all possible linear combinations of $|0\rangle$ and $|1\rangle$, which includes all normalized and non-normalized states.
- A general quantum state $|\psi\rangle$ is a **linear combination** of these two basis states, given by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$.
- The unit circle indicates the subset of **normalized quantum states**, ensuring total probability is 1 ($|\alpha|^2 + |\beta|^2 = 1$).

2.1.2 Superposition and Quantum Behavior

The state $|\psi\rangle$ represents a **superposition** of the basis states $|0\rangle$ and $|1\rangle$. **This means the qubit simultaneously exists in a combination of both $|0\rangle$ and $|1\rangle$ until it is measured.** Upon measurement, the qubit **collapses** to one of the basis states, with the probabilities determined by the magnitudes of the coefficients $|\alpha|^2$ and $|\beta|^2$.

Standard Basis ($|0\rangle, |1\rangle$):

- The standard basis consists of the states:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

- A qubit is in superposition if it is a non-trivial linear combination of $|0\rangle$ and $|1\rangle$, such as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{where } |\alpha|^2 + |\beta|^2 = 1.$$

Hadamard Basis ($|+\rangle, |-\rangle$):

- The Hadamard basis consists of the states:

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

- The Hadamard gate creates an equal superposition of the computational basis states. Its matrix representation is:

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

- Applying H to $|0\rangle$ or $|1\rangle$ gives:

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

Current Example (Non-Hadamard): Consider a qubit in the state:

$$|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle.$$

1. **Coefficients:**
   The coefficients of the state are:

$$\alpha = \frac{1}{\sqrt{3}}, \quad \beta = \sqrt{\frac{2}{3}}.$$

2. **Probabilities of Measurement:**
   The probabilities of collapsing to each basis state upon measurement are:

$$P(0) = |\alpha|^2 = \left(\frac{1}{\sqrt{3}}\right)^2 = \frac{1}{3},$$

$$P(1) = |\beta|^2 = \left(\sqrt{\frac{2}{3}}\right)^2 = \frac{2}{3}.$$

3. **Interpretation:**
   (a) There is a $\frac{1}{3}$ probability of measuring the state $|0\rangle$.
   (b) There is a $\frac{2}{3}$ probability of measuring the state $|1\rangle$.

Hadamard Gate Example: Consider applying the Hadamard gate to the state $|0\rangle$:

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

1. **Coefficients:**
   The coefficients of the state are:

$$\alpha = \frac{1}{\sqrt{2}}, \quad \beta = \frac{1}{\sqrt{2}}.$$

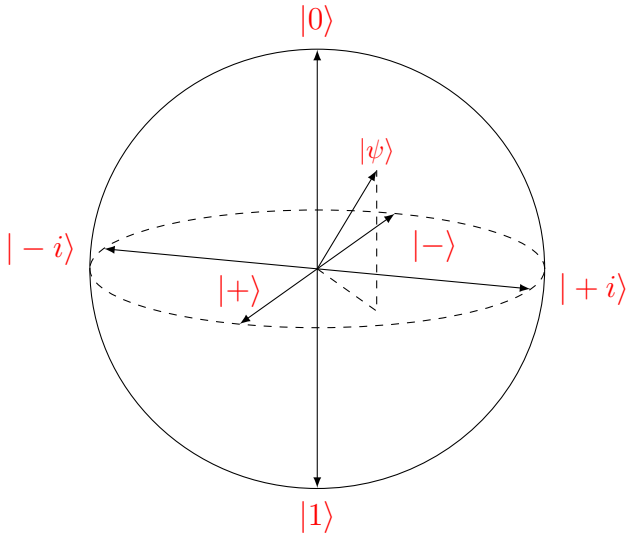
2. **Probabilities of Measurement:**
   The probabilities of collapsing to each basis state upon measurement are:

$$P(0) = |\alpha|^2 = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2},$$

$$P(1) = |\beta|^2 = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}.$$

3. **Interpretation:**
   (a) There is a $\frac{1}{2}$ probability of measuring the state $|0\rangle$.
   (b) There is a $\frac{1}{2}$ probability of measuring the state $|1\rangle$.

2.2 Bloch Sphere Representation



The Bloch Sphere is a geometrical **representation of a single qubit state**. It provides an intuitive way to visualize qubit states as points on or inside a unit sphere in a 3D space.

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

Here, θ and ϕ are **spherical coordinates** that describe the position of the state on the Bloch Sphere:

- θ: The polar angle from the z-axis (between 0 and π).
- ϕ: The azimuthal angle in the x-y plane (between 0 and 2π).

Features of the Bloch Sphere:

- A qubit $|\psi\rangle$ can be represented as a point on the surface of the Bloch sphere.
- The z-axis represents the computational basis states:
  1. $|0\rangle$ (north pole)

2. $|1\rangle$ (south pole)

- The x-axis represents the superposition states:
  1. $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ (positive x-axis)
  2. $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ (negative x-axis)
- The y-axis represents the phase-superposition **(spin around sphere)** states:
  1. $|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ (positive y-axis)
  2. $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$ (negative y-axis)
- The equator represents superposition states:
  1. $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ along the x-axis
  2. $|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ along the y-axis

2.3 Measurement

Given a qubit is measured, its state **collapses** to one of the basis states, $|0\rangle$ or $|1\rangle$.

Before Measurement: A qubit in superposition is represented as:

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

where:

- $|\alpha|^2$: Probability of measuring $|0\rangle$,
- $|\beta|^2$: Probability of measuring $|1\rangle$.

After Measurement

- The qubit collapses to $|0\rangle$ with probability $|\alpha|^2$,
- Or collapses to $|1\rangle$ with probability $|\beta|^2$.

Example: For the state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

the qubit **collapses** to $|0\rangle$ or $|1\rangle$ with equal probability (50%) upon measurement.

Key Point: Measurement **destroys** the superposition, leaving the qubit in a definite classical state (0 or 1).

2.4 Valid States For Qubits

Qubit State	Valid?	Prob-0	Prob-1	Reason (if not valid)
$\|1\rangle$	Yes	0%	100%	–
$\|0\rangle$	Yes	100%	0%	–
$\|0\rangle + \|1\rangle$	No	–	–	Not normalized: the sum of probabilities does not equal 1.
$\frac{\sqrt{3}}{2}\|0\rangle - \frac{1}{2}\|1\rangle$	Yes	75%	25%	–
$\frac{i}{2}\|0\rangle - \frac{i}{2}\|1\rangle$	No	–	–	Not normalized: the magnitudes of coefficients do not satisfy $\|\alpha\|^2 + \|\beta\|^2 = 1$.
$\frac{(1-i)}{2}\|0\rangle - \frac{(1+i)}{2}\|1\rangle$	Yes	50%	50%	–

Table 1: Validation and probabilities of qubit states, with explanations for invalid states.

2.5 Multi-Qubit Notation

When working with multiple qubits, their combined quantum state is represented as a tensor product of individual qubit states. This allows for the description of more complex quantum systems, including entanglement.

2.5.1 Two-Qubit States

For two qubits, the basis states are formed by combining the individual states $|0\rangle$ and $|1\rangle$:

$$\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}.$$

Here:

- $|00\rangle = |0\rangle \otimes |0\rangle$,
- $|01\rangle = |0\rangle \otimes |1\rangle$,
- $|10\rangle = |1\rangle \otimes |0\rangle$,
- $|11\rangle = |1\rangle \otimes |1\rangle$.

A general two-qubit state is written as:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

where $|\alpha_{ij}|^2$ represents the probability of measuring the state $|ij\rangle$.

2.5.2 Three-Qubit States

For three qubits, the basis states expand to:

$$\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}.$$

A general three-qubit state is expressed as:

$$|\psi\rangle = \sum_{i,j,k\in\{0,1\}} \alpha_{ijk}|ijk\rangle,$$

where $|\alpha_{ijk}|^2$ gives the probability of measuring $|ijk\rangle$.

2.5.3 Entanglement in Multi-Qubit Systems

In multi-qubit systems, states can exhibit **entanglement**, meaning the state of one qubit cannot be described independently of the others. For example:

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

is an entangled state (Bell state) where measurement of one qubit determines the state of the other.

Key Point: Multi-qubit systems **scale exponentially** with the number of qubits, as the size of the state space is 2^n, where n is the number of qubits.

2.6 BB84 Protocol: Quantum Key Distribution

The BB84 protocol is a foundational quantum key distribution (QKD) scheme introduced by Bennett and Brassard in 1984. It uses the principles of quantum mechanics to securely generate a shared secret key between two parties, typically called **Alice** (sender) and **Bob** (receiver), while detecting any eavesdropping by a third party (**Eve**).

Steps of the Protocol:

1. **Key Encoding:**
   - Alice generates a random sequence of bits (0 or 1).
   - She encodes each bit onto a qubit using one of two possible bases:
     - **Rectilinear Basis** ($|0\rangle, |1\rangle$): Encodes 0 as $|0\rangle$ and 1 as $|1\rangle$.
     - **Diagonal Basis** ($|+\rangle, |-\rangle$): Encodes 0 as $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and 1 as $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
2. **Key Transmission:**
   - Alice sends the qubits to Bob over a quantum channel.
3. **Basis Selection and Measurement:**
   - Bob randomly selects one of the two bases (rectilinear or diagonal) to measure each qubit.

- Due to the laws of quantum mechanics, if Bob's chosen basis matches Alice's encoding basis, he will measure the correct bit value. Otherwise, the result will be random.

4. **Basis Reconciliation:**
   - Alice and Bob communicate over a public channel to compare their chosen bases for each qubit (without revealing the bit values).
   - They discard the bits where their bases do not match.
5. **Key Verification and Finalization:**
   - Alice and Bob verify a subset of the remaining bits to detect potential eavesdropping by Eve.
   - If no eavesdropping is detected, the remaining bits form a secure shared key.

Security of the Protocol:

The BB84 protocol is secure because:

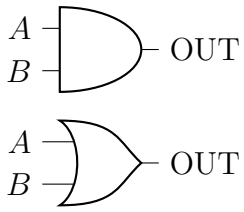
- Measurement disturbs the quantum state, so any eavesdropping by Eve introduces detectable errors in the key.
- The no-cloning theorem prevents Eve from duplicating the qubits without altering them.

3 Quantum Gates and Circuits

Quantum gates are the building blocks of quantum circuits, which operate on qubits. **Unlike classical logic gates, quantum gates are reversible and utilize superposition and entanglement properties**. Below is an example of a simple quantum circuit.

3.1 Logic Gates

- **Operation:** Operates on Boolean values.
- **Representation:** Truth tables.
- **Directionality:** Most of them only run forward.

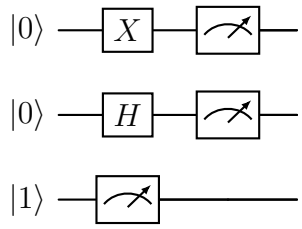


Truth table for logic gates:

A	B	Out (AND)
0	0	0
0	1	0
1	0	0
1	1	1

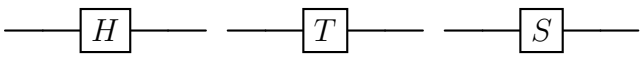
A	B	Out (OR)
0	0	0
0	1	1
1	0	1
1	1	1

3.2 Quantum Gates



$|\psi_0\rangle \quad |\psi_1\rangle \quad |\psi_2\rangle \quad |\psi_3\rangle \quad |\psi_4\rangle$

- **Operation:** Operates on complex states.
- **Representation:** Unitary matrices.
- **Reversibility:** They are reversible.



Matrix representation of common quantum gates:

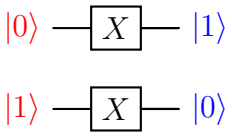
$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

3.3 X Gate (Pauli-X Gate)

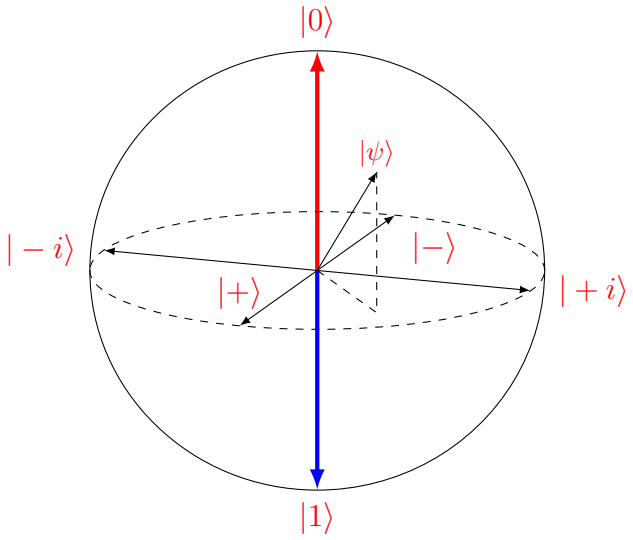
The Pauli-X gate, also known as the quantum **NOT** gate, is a fundamental single-qubit quantum gate. It flips the state of a qubit around the X-axis of the Bloch sphere. Mathematically, it is represented by the matrix:

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

When applied to the computational basis states:



In the Bloch sphere representation, the Pauli-X gate swaps the positions of $|0\rangle$ and $|1\rangle$, effectively mirroring the qubit state across the X-axis. The following diagram highlights the X-axis on the Bloch sphere:



Vector Representation

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

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

$$|\psi\rangle = \alpha_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \alpha_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$$

The Unitary Matrix for X-gate is:

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

$$X \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_0 \end{pmatrix}$$

3.4 Z Gate (Pauli-Z Gate)

The Pauli-Z gate, also known as the quantum phase-flip gate, is a fundamental single-qubit quantum gate. It flips the phase of a qubit around the Z-axis of the Bloch sphere. Mathematically, it is represented by the matrix:

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

When applied to the computational basis states:

$$|0\rangle \text{ —[Z]— } |0\rangle$$

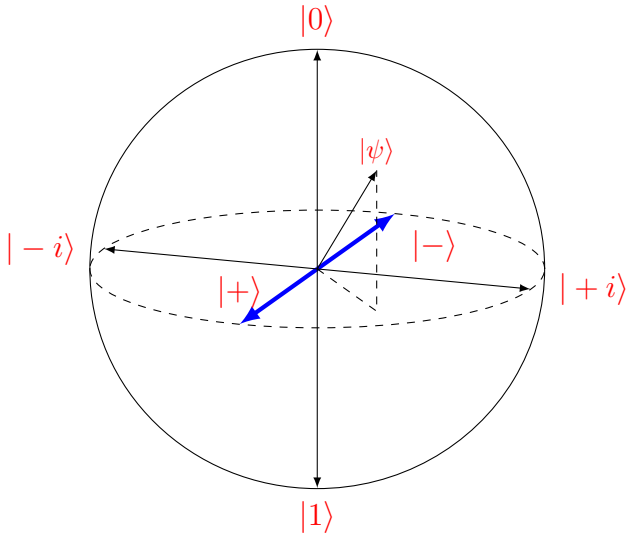
$$|1\rangle \text{ —[Z]— } -|1\rangle$$

$$\alpha_0|0\rangle + \alpha_1|1\rangle \text{ —[Z]— } \alpha_0|0\rangle - \alpha_1|1\rangle$$

More formally:

$$\begin{aligned} Z\left(\alpha|0\rangle + \beta|1\rangle\right) &= \alpha Z|0\rangle + \beta Z|1\rangle \\ &= \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\ &= \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta(-1) \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \alpha|0\rangle - \beta|1\rangle \end{aligned}$$

In the Bloch sphere representation, the Pauli-Z gate rotates the qubit state around the Z-axis by π, flipping the phase of $|1\rangle$ while leaving $|0\rangle$ unchanged. The following diagram highlights the Z-axis on the Bloch sphere:



Why X-gates and Z-gates Cannot Create Superposition

The X-gate (Pauli-X) and Z-gate (Pauli-Z) are essential single-qubit quantum gates, but neither can create superposition states from classical basis states ($|0\rangle$ or $|1\rangle$).

- X**-gate**: This gate flips the state of the qubit, transforming $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$. It acts similarly to a classical NOT gate and cannot mix states.
- Z**-gate**: This gate applies a phase flip to $|1\rangle$, leaving $|0\rangle$ unchanged. While it changes the relative phase of the qubit's components, it does not introduce a superposition.

Both gates operate on the existing state vector and do not combine or redistribute amplitudes in a way that creates a superposition. For creating superposition states, we require the Hadamard gate.

3.5 H-Gate (Hadamard Gate)

The Hadamard (H) gate is a crucial single-qubit gate in quantum computing that **creates and manipulates superposition states**. It is represented by the unitary matrix:

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

When applied to the computational basis states:

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

This operation creates equal superposition states with well-defined relative phases. The Hadamard gate is often the first gate applied in quantum algorithms, setting up qubits in a superposition to explore multiple states simultaneously.
The Hadamard gate has the following transformation properties:

$$|0\rangle \xrightarrow{H} |+\rangle \quad |+\rangle \xrightarrow{H} |0\rangle$$

$$|1\rangle \xrightarrow{H} |-\rangle \quad |-\rangle \xrightarrow{H} |1\rangle$$

Here, $|+\rangle$ and $|-\rangle$ represent the superposition states:

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

These transformations highlight the gate's ability to switch between the computational basis and the superposition basis, making it a fundamental tool in quantum computation.

Quantum Circuit Representation

Below is a quantum circuit that demonstrates the action of the Hadamard gate:

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

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

3.6 Gates as Generalized Rotations

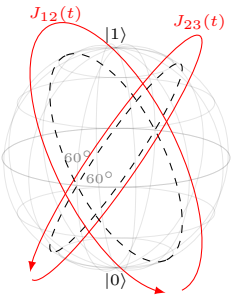
Single-qubit gates can be visualized as rotations on the Bloch sphere. These rotations are represented by $R_{\hat{n}}(\theta)$, where:

- $\hat{n}$ is the axis of rotation.
- θ is the rotation angle.

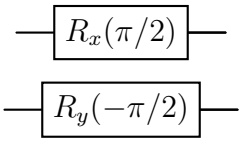
The Pauli gates (X, Y, and Z) are special cases of these rotations about their respective axes:

$$R_x(\pi) = X, \quad R_y(\pi) = Y, \quad R_z(\pi) = Z.$$

The following Bloch sphere illustrates a rotation along an arbitrary axis $\hat{n}$, emphasizing the geometric interpretation of quantum gates:



The following quantum circuits illustrate specific rotations around the x- and y-axes:



These gates perform rotations by $\frac{\pi}{2}$ or $-\frac{\pi}{2}$ around the respective axes, modifying the qubit's state on the Bloch sphere.

3.7 CNOT Gate

The Controlled-NOT (CNOT) gate, also referred to as the CX gate, can be described as follows:

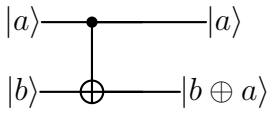
- It acts on two qubits:
  - **Control qubit:** Determines whether the operation is applied.
  - **Target qubit:** The qubit on which the operation (Pauli-X gate) is performed.
- The target qubit's state is flipped (from $|0\rangle$ to $|1\rangle$ or vice versa) if and only if the control qubit is in the state $|1\rangle$.
- **Matrix Representation:**

– The CNOT gate can be expressed as a 4×4 matrix:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

– Each row and column corresponds to the states of the control and target qubits ($|00\rangle, |01\rangle, |10\rangle, |11\rangle$).

Example:



Input States:

1. The control qubit starts in state $|a\rangle$ (top line).
2. The target qubit starts in state $|b\rangle$ (bottom line).

Output States:

1. The control qubit remains unchanged, outputting $|a\rangle$.
2. The target qubit's state is updated based on the value of the control qubit:
   - If $|a\rangle = 0$, the target qubit remains $|b\rangle$.
   - If $|a\rangle = 1$, the target qubit flips, becoming $|b \oplus 1\rangle$ (where $\oplus$ denotes addition modulo 2, also known as XOR).

3.8 SWAP Gate

The SWAP gate is a fundamental two-qubit gate in quantum computing that exchanges the states of two qubits. It can be described as follows:

- **Operation:**
  - The SWAP gate exchanges the states of the two qubits.
  - If the two qubits are in states $|a\rangle$ and $|b\rangle$, their states are swapped:

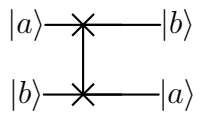
$$\text{Input: } |a\rangle |b\rangle \quad \rightarrow \quad \text{Output: } |b\rangle |a\rangle.$$

- **Matrix Representation:**
  - The SWAP gate can be expressed as a 4×4 matrix:

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

– Each row and column corresponds to the states of the two qubits ($|00\rangle, |01\rangle, |10\rangle, |11\rangle$).

Example:



Input States:

1. The first qubit starts in state $|a\rangle$ (top line).
2. The second qubit starts in state $|b\rangle$ (bottom line).

Output States:

1. The first qubit outputs $|b\rangle$.
2. The second qubit outputs $|a\rangle$.

3.9 Toffoli Gate (CCX)

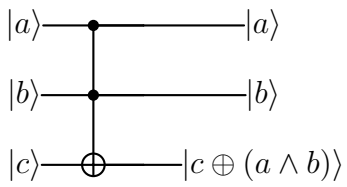
The Toffoli gate, also referred to as the Controlled-Controlled-NOT (CCX) gate, is a three-qubit gate in quantum computing. It can be described as follows:

- **Operation:**
  - The Toffoli gate operates on three qubits:
    - **Two control qubits:** Determine whether the operation is applied.
    - **One target qubit:** The qubit on which the operation (Pauli-X gate) is performed.
  - The target qubit's state is flipped (from $|0\rangle$ to $|1\rangle$ or vice versa) if and only if both control qubits are in the state $|1\rangle$.
- **Matrix Representation:**
  - The Toffoli gate can be expressed as an 8×8 matrix:

$$\text{Toffoli} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

  - Each row and column corresponds to the states of the three qubits ($|000\rangle$, $|001\rangle$, …, $|111\rangle$).

Example:



Input States:

1. The first qubit starts in state $|a\rangle$ (top line, first control qubit).
2. The second qubit starts in state $|b\rangle$ (middle line, second control qubit).
3. The third qubit starts in state $|c\rangle$ (bottom line, target qubit).

Output States:

1. The first qubit remains unchanged, outputting $|a\rangle$.
2. The second qubit remains unchanged, outputting $|b\rangle$.
3. The third qubit's state is updated based on the values of the control qubits:
   - If $|a\rangle = 1$ and $|b\rangle = 1$, the target qubit flips: $|c\rangle \to |c \oplus 1\rangle$.
   - Otherwise, the target qubit remains unchanged: $|c\rangle \to |c\rangle$.

3.10 Fredkin Gate

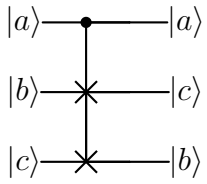
The Fredkin gate, also referred to as the Controlled-SWAP (CSWAP) gate, is a three-qubit gate in quantum computing. It can be described as follows:

- **Operation:**
  - The Fredkin gate operates on three qubits:
    - **One control qubit:** Determines whether the operation is applied.
    - **Two target qubits:** The qubits whose states are swapped.
  - The two target qubits are swapped only if the control qubit is in the state $|1\rangle$.
  - If the control qubit is $|0\rangle$, the target qubits remain unchanged.
- **Matrix Representation:**
  - The Fredkin gate can be expressed as an 8×8 matrix:

$$\text{Fredkin} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

– Each row and column corresponds to the states of the three qubits ($|000\rangle$, $|001\rangle$, ..., $|111\rangle$).

Example:



Input States:

1. The first qubit starts in state $|a\rangle$ (top line, control qubit).
2. The second qubit starts in state $|b\rangle$ (middle line, target qubit 1).
3. The third qubit starts in state $|c\rangle$ (bottom line, target qubit 2).

Output States:

1. The control qubit remains unchanged, outputting $|a\rangle$.
2. The target qubits' states depend on the value of the control qubit:
   - If $|a\rangle = 1$, the states of the second and third qubits are swapped:
   $$|b\rangle \to |c\rangle\,, \quad |c\rangle \to |b\rangle\,.$$
   - If $|a\rangle = 0$, the target qubits remain unchanged:
   $$|b\rangle \to |b\rangle\,, \quad |c\rangle \to |c\rangle\,.$$

3.11 Reversibility Requirement

The **reversibility requirement** ensures that a system can reverse any operation to restore its original state without losing information.

3.12 Gate Effect Table

Gate	Effect	Angular Movement	Matrix Representation	Common Use
X Gate	180° rotation about the X-axis	180° (π)	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Flips $\|0\rangle \leftrightarrow \|1\rangle$
Z Gate	180° rotation about the Z-axis	180° (π)	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	Introduces a phase flip to $\|1\rangle$
S Gate	90° rotation about the Z-axis	90° ($\pi/2$)	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	Introduces a phase of i to $\|1\rangle$
H Gate	180° rotation about the $X + Z$-axis	180° (π)	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Creates superposition
T Gate	45° rotation about the Z-axis	45° ($\pi/4$)	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	Adds a $\pi/4$ phase shift

Table 2: Summary of quantum gates, their effects, angular movement, matrix representations, and common use cases.

4 Linear Algebra Basics

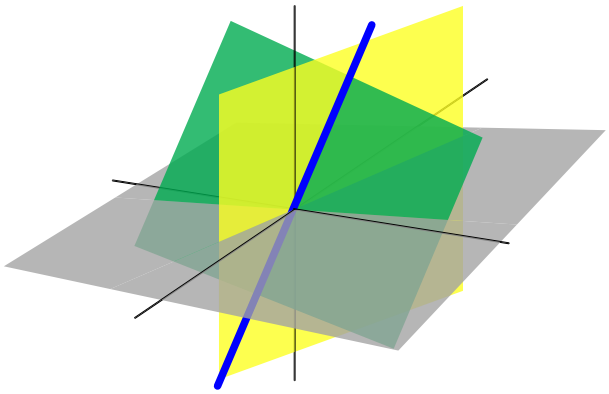


Figure 1: In three-dimensional Euclidean space, these three planes represent solutions to linear equations, and their intersection represents the set of common solutions: in this case, a unique point. The blue line is the common solution to two of these equations.

4.1 Representing Classical States

Vector representation of probabilistic states:

Flipping a biased coin X:

$$\Pr[\textbf{Heads}] = 25\%, \quad \Pr[\textbf{Tails}] = 75\%$$

Notation:

$$\textbf{Heads} = 0, \quad \textbf{Tails} = 1$$

We can represent this as a **probability vector**:

$$v = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix} \quad \Rightarrow \quad \begin{aligned} \Pr[\textbf{state of } X \textbf{ is } 0] &= \frac{1}{4} \\ \Pr[\textbf{state of } X \textbf{ is } 1] &= \frac{3}{4} \end{aligned}$$

Requirement:

The sum of all entries in the **column** vector (L1-norm) must equal 1:

$$\sum v = 1$$

Key Insight: The probabilistic state only exists until the coin lands. At this point, the probability state *collapses* into a deterministic vector representing the outcome.

Collapse upon flip:

If the outcome is **Heads** (0):

$$v = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

If the outcome is **Tails** (1):

$$v = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

4.2 2 Qubit States

A qubit state represents a **superposition** of the basis states $|0\rangle$ and $|1\rangle$. This superposition can be described by a two-dimensional complex vector:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

where:

$$\alpha, \beta \in \mathbb{C}, \quad \text{and } |\alpha|^2 + |\beta|^2 = 1$$

Key Points:

- Each entry in the vector corresponds to the **probability amplitude** for that state.
- The **probability** of measuring state $|0\rangle$ is $|\alpha|^2$, and the probability of measuring state $|1\rangle$ is $|\beta|^2$.
- **Note: α and β may not be positive or even real numbers. They can be complex.**

Examples of Valid Qubit States:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} \frac{3}{5} \\ \frac{4i}{5} \end{pmatrix}$$

Requirement:

The **L2-norm** (Euclidean norm) of the **column** vector must be 1:

$$\sqrt{|\alpha|^2 + |\beta|^2} = 1$$

Interpretation:

This normalization ensures that the total probability of the qubit collapsing to $|0\rangle$ or $|1\rangle$ sums to 1 when measured. Superposition enables quantum systems to exist in a combination of states, with probabilities determined by $|\alpha|^2$ and $|\beta|^2$.

4.3 N Qubits States

The vector representation of **quantum probabilistic states** can be described as follows:

Notation:

$$\textbf{Heads} = 0, \quad \textbf{Tails} = 1$$

A state of 2 qubits (X, Y) can be represented as:

$$v = \begin{pmatrix} \frac{1}{8} \\ \frac{1}{2} \\ 0 \\ \frac{3}{8} \end{pmatrix} \quad \Rightarrow \quad \begin{aligned} \Pr[X = 0, Y = 0] &= \frac{1}{8} \\ \Pr[X = 0, Y = 1] &= \frac{1}{2} \\ \Pr[X = 1, Y = 0] &= 0 \\ \Pr[X = 1, Y = 1] &= \frac{3}{8} \end{aligned}$$

Restriction: Normalization Condition

In a valid quantum system, the state vector must satisfy the **normalization condition**, ensuring that the total probability is 1:

$$\|v\|^2 = \sum_i |v_i|^2 = 1$$

For the given state:

$$\|v\|^2 = \left(\frac{1}{8}\right)^2 + \left(\frac{1}{2}\right)^2 + 0^2 + \left(\frac{3}{8}\right)^2 = \frac{1}{64} + \frac{4}{16} + \frac{9}{64} = \frac{16}{16} = 1$$

Thus, the state is valid as it satisfies the normalization condition.

Operation: X Control, Y Random if $X = 1$

- The sum of all entries in each column (L1-norm) must equal 1:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad \text{where} \quad \sum_i |a_{ij}| = 1 \quad \forall j$$

- The matrix must be **unitary**, meaning:

$$U^\dagger U = I$$

  This ensures that the operation preserves the normalization condition and represents a valid quantum transformation.

Impact of Measurement:

$$v = \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix} \quad \Rightarrow \quad \begin{matrix} X = 0 \\ X = 1 \end{matrix}$$

$$\Pr[X = 0] = \left|\tfrac{3}{5}\right|^2 \qquad \Pr[X = 1] = \left|\tfrac{4}{5}\right|^2$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ Resolves to 0} \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ Resolves to 1}$$

Note: $\Pr[X = 0] \neq 36\%$ **Note:** $\Pr[X = 1] \neq 64\%$

4.4 Unitary Matrices and Qubit Operations

Operations on qubits can be represented by a **Unitary Matrix**. A matrix U is unitary if it satisfies:

$$\boxed{U^\dagger U = I}$$

where $U^\dagger$ is the **conjugate transpose** of U.

Conjugate Transpose:

- **Take the transpose of U:** Flip the matrix over its diagonal. In other words, **swap the rows and columns**. For example:

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \text{becomes} \quad U^T = \begin{pmatrix} a & c \\ b & d \end{pmatrix}.$$

- **Take the complex conjugate of each entry in U:** If a number has an imaginary part (e.g., $3+4i$), just change the sign of the imaginary part (e.g., $3-4i$). For real numbers, nothing changes. For example:

$$U = \begin{pmatrix} 1 & i \\ -i & 2 \end{pmatrix} \quad \text{becomes} \quad U^* = \begin{pmatrix} 1 & -i \\ i & 2 \end{pmatrix}.$$

- **Both steps together (Conjugate Transpose):** When you do both steps together, that's the **conjugate transpose** (denoted $U^\dagger$):

$$U = \begin{pmatrix} 1 & i \\ -i & 2 \end{pmatrix} \quad \text{becomes} \quad U^\dagger = \begin{pmatrix} 1 & i \\ -i & 2 \end{pmatrix}^T = \begin{pmatrix} 1 & -i \\ i & 2 \end{pmatrix}.$$

Inverse of Unitary Matrices:

A key property of unitary matrices is that their inverse is simply their conjugate transpose. That is, for a unitary matrix U, the inverse U^{-1} is given by:

$$U^{-1} = U^\dagger$$

This follows directly from the definition of a unitary matrix, where $U^\dagger U = I$ and $UU^\dagger = I$. This means that applying a unitary operation followed by its inverse (or vice versa) returns the original quantum state, preserving the normalization and reversibility required for qubit operations. For example, if $U = H$ (the Hadamard matrix), then $U^{-1} = H^\dagger = H$, since H is its own inverse in this case.

Examples of Unitary Matrices:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad R_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Application:

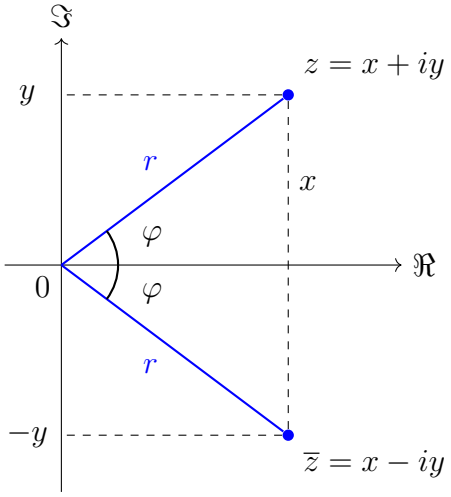
Applying the Hadamard matrix (H) to the qubit $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ results in:

$$Hv = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Requirement:

To be valid for qubit operations, a matrix must be **unitary**.

4.5 Complex Numbers and Conjugate Transpose



Complex Numbers:

Complex numbers are of the form $z = x + iy$, where:

- x is the **real part**.
- y is the **imaginary part**.

Complex Conjugate:

The complex conjugate of a number z is obtained by flipping the sign of the imaginary part:

$$z = 3 - i4 \quad \Rightarrow \quad \overline{z} = 3 + i4$$

Amplitude:

The amplitude (or magnitude) of z, which represents the distance of z from the origin in the complex plane, is calculated as:

$$r = \sqrt{z\overline{z}} = \sqrt{(3 - i4)(3 + i4)} = 5$$

Geometric Representation:

In the complex plane:

- The horizontal axis ($\Re$) represents the real part.
- The vertical axis ($\Im$) represents the imaginary part.
- The amplitude r is the distance from the origin to the point $z = x + iy$.

Conjugate Transpose:

For a matrix A, the conjugate transpose (A^*) involves:

- Taking the **transpose** of A.
- Taking the **complex conjugate** of each entry.

Example:

$$A = \begin{pmatrix} 3+7i & 0 \\ 2i & 4-i \end{pmatrix} \quad \xrightarrow{\text{Conjugate Transpose}} \quad A^* = A^T = \begin{pmatrix} 3-7i & -2i \\ 0 & 4+i \end{pmatrix}$$

Summary:

- The complex conjugate flips the imaginary part.
- The amplitude measures the distance from the origin.
- The conjugate transpose combines transposing and conjugating matrix entries.

4.6 Tensor Products for 2 Qubits

Tensor products are used to combine the states of two qubits or systems. Below is an example with two contexts: flipping coins and qubits in superposition.

Flipping 2 Coins, X and Y:

Given the state vectors for X and Y:

$$v = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}, \quad w = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix}$$

The tensor product $v \otimes w$ results in:

$$v \otimes w = \begin{pmatrix} \frac{2}{3} \cdot \frac{1}{4} \\ \frac{2}{3} \cdot \frac{3}{4} \\ \frac{1}{3} \cdot \frac{1}{4} \\ \frac{1}{3} \cdot \frac{3}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{6} \\ \frac{1}{2} \\ \frac{1}{12} \\ \frac{1}{4} \end{pmatrix}$$

Two Qubits in Superposition:

For two qubits in superposition, let the state vectors be:

$$v = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad w = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}$$

The tensor product $v \otimes w$ results in:

$$v \otimes w = \begin{pmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{pmatrix}$$

Example of Tensor Product

Let v and w be two vectors:

$$v = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad w = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

The tensor product $v \otimes w$ is calculated as:

$$v \otimes w = \begin{pmatrix} v_1 \cdot w \\ v_2 \cdot w \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} \\ 2 \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

Thus, the resulting tensor product $v \otimes w$ is:

$$\begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

Example of Tensor Product with Qubit Operations

In quantum computing, the tensor product is used to combine the operations of multiple gates. Let us consider the U_{NOT} (X gate) and the Hadamard (H) gate.

Definitions

The U_{NOT} (X gate) and the Hadamard (H) gate are represented as:

$$U_{\text{NOT}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

The tensor product $U_{\text{NOT}} \otimes H$ is calculated as:

$$U_{\text{NOT}} \otimes H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Breaking this down, the result is:

$$U_{\text{NOT}} \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & 1 \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & 0 \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}$$

Interpretation

The resulting tensor product represents the combined operation of U_{NOT} acting on one qubit and H acting on another qubit. This forms a 4x4 matrix, which describes the combined state transformation in the 2-qubit Hilbert space.

4.7 Dirac Notation

Dirac notation, also known as **bra-ket notation**, is widely used in quantum mechanics and quantum computing to describe quantum states. Below are examples demonstrating its use.

Basic States

The computational **basis states** are defined as:

$$|0\rangle \stackrel{\text{def}}{=} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle \stackrel{\text{def}}{=} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Superposition of States

A qubit in **superposition** can be expressed as:

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix}$$

For a **two-qubit state** in superposition:

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{i}{\sqrt{2}}|11\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{i}{\sqrt{2}} \end{pmatrix}$$

Sparse Representation

For a **five-qubit state**:

$$\frac{1}{\sqrt{2}}|00000\rangle + \frac{i}{\sqrt{2}}|11111\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \vdots \\ 0 \\ \frac{i}{\sqrt{2}} \end{pmatrix}$$

Interpretation: This representation contains many zeros in between, which highlights the **advantage of sparse representation**, enabling efficient storage and computation. For instance, this vector contains 30 zeros.

4.8 Eigenvalues and Eigenvectors

In quantum mechanics, **eigenvalues** and **eigenvectors** are critical concepts for understanding quantum operators and their measurable properties. For an operator A, an eigenvector $|v\rangle$ satisfies:

$$A|v\rangle = \lambda|v\rangle,$$

where: λ is the **eigenvalue** corresponding to the **eigenvector** $|v\rangle$.

Example: Understanding the Pauli-Z Matrix

The **Pauli-Z matrix** is a tool in quantum computing, and it looks like this:

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

Think of this matrix as a set of rules that tells us how a quantum bit (or qubit) behaves when we measure or transform it.

Step 1: What Are Eigenvalues? (The Special Numbers)

Eigenvalues are like special numbers that tell us what "values" we get when the Pauli-Z matrix acts on certain quantum states. To find them, we solve a simple math problem called the "characteristic equation":

$$\det(Z - \lambda I) = 0$$

Here, λ is the eigenvalue we're looking for, and I is the identity matrix (like a "do nothing" matrix):

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

So, we subtract λI from Z:

$$Z - \lambda I = \begin{pmatrix} 1-\lambda & 0 \\ 0 & -1-\lambda \end{pmatrix}$$

Now, we find the determinant (a way to "combine" the numbers in the matrix):

$$\det(Z - \lambda I) = (1-\lambda)(-1-\lambda) - (0)(0) = \lambda^2 - 1$$

Solving $\lambda^2 - 1 = 0$ is like solving $\lambda^2 = 1$:

$$\lambda = +1 \quad \text{or} \quad \lambda = -1$$

So, the eigenvalues are $+1$ and -1. These numbers tell us the possible outcomes when we measure a qubit using the Pauli-Z matrix.

Step 2: What Are Eigenvectors? (The Special States)

Eigenvectors are the specific quantum states (or vectors) that, when the Pauli-Z matrix is applied, just get scaled by their eigenvalue (not changed in direction). Let's find them for each eigenvalue.

1. **For $\lambda = +1$:** We solve $(Z - I)|v\rangle = 0$, where $|v\rangle$ is the eigenvector. First, compute $Z - I$:

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

   This equation $\begin{pmatrix} 0 & 0 \\ 0 & -2 \end{pmatrix} |v\rangle = 0$ means the second number in $|v\rangle$ must be 0 (since $-2 \times$ second number $= 0$). So, let's try:

   $$|v\rangle = \begin{pmatrix} x \\ 0 \end{pmatrix}$$

   where x can be any number (we'll pick $x = 1$ for simplicity):

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

2. **For $\lambda = -1$:** We solve $(Z + I)|v\rangle = 0$. First, compute $Z + I$:

$$Z + I = \begin{pmatrix} 1+1 & 0 \\ 0 & -1+1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$$

This equation $\begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} |v\rangle = 0$ means the first number in $|v\rangle$ must be 0 (since $2 \times$ first number $= 0$). So, let's try:

$$|v\rangle = \begin{pmatrix} 0 \\ y \end{pmatrix}$$

where y can be any number (we'll pick $y = 1$ for simplicity):

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

Result:

The **eigenvalues** and **eigenvectors** of the Pauli-Z matrix are:

$$\lambda = 1, \quad |v\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad \lambda = -1, \quad |v\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

In quantum terms, $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ is often written as $|0\rangle$, and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ as $|1\rangle$.

What Does This Mean?

1. The Pauli-Z matrix is like a measurement tool that checks if a qubit is in the $|0\rangle$ state or the $|1\rangle$ state.
2. If you measure a qubit and get $+1$, it means the qubit was in the $|0\rangle$ state ($\begin{pmatrix} 1 \\ 0 \end{pmatrix}$).
3. If you measure a qubit and get -1, it means the qubit was in the $|1\rangle$ state ($\begin{pmatrix} 0 \\ 1 \end{pmatrix}$).
4. These states and values help us understand how qubits behave in quantum computing.

5 Entanglement

Entanglement is a quantum phenomenon where the quantum states of two or more particles become correlated in such a way that the state of one particle cannot be described independently of the others, even when separated by large distances.

5.1 Bell States

Bell states are a set of four **maximally entangled** two-qubit quantum states. They serve as fundamental resources in quantum information processing, including quantum teleportation and superdense coding. The four Bell states are defined as:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$
$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$
$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

These states exhibit perfect quantum correlations and violate Bell inequalities, demonstrating nonlocality. They are commonly generated using a Hadamard gate (H) and a Controlled-NOT (CNOT) gate as follows:

1. Apply a Hadamard gate to the first qubit:

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

2. Apply a CNOT gate with the first qubit as control and the second as target:

$$\text{CNOT}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

   which creates the $|\Phi^+\rangle$ Bell state.

Bell states form a basis for two-qubit quantum systems and play a crucial role in quantum algorithms and protocols.

5.2 Creating and Destroying Entanglement

Entanglement can be both created and destroyed through various quantum operations.

5.2.1 Creating Entanglement

To create entanglement, quantum gates such as the Hadamard (H) and Controlled-NOT (CNOT) gates are commonly used. For example, applying a Hadamard gate to a qubit in the $|0\rangle$ state, followed by a CNOT gate with this qubit as control and another qubit as target, results in the maximally entangled Bell state:

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

which is one of the four Bell states.

5.2.2 Destroying Entanglement

Entanglement can be destroyed through decoherence, measurement, or specific quantum operations:

- **Measurement:** Measuring an entangled qubit collapses the entire entangled system into a definite state, breaking the entanglement.
- **Decoherence:** Interaction with the environment causes loss of quantum coherence, effectively destroying entanglement.
- **Local Operations:** Certain local quantum operations can disentangle qubits, transforming them into separable states.

5.3 Testing for Entanglement

Determining whether a quantum state is entangled is essential for quantum information processing. There are several criteria and methods for testing entanglement:

5.3.1 Bell Inequality Violation

One of the most common methods for testing entanglement is through Bell's theorem, which states that entangled states can exhibit correlations that violate classical Bell inequalities. By measuring the correlations between qubits and checking for a violation of these inequalities, one can confirm entanglement.

5.4 Testing for Entanglement

Determining whether a quantum state is entangled is essential for quantum information processing. There are several criteria and methods for testing entanglement:

5.4.1 Bell Inequality Violation

One of the most common methods for testing entanglement is through Bell's theorem, which states that entangled states can exhibit correlations that violate classical Bell inequalities. By measuring the correlations between qubits and checking for a violation of these inequalities, one can confirm entanglement.

5.5 Classically Controlled Gates in Entanglement

Entanglement generation and manipulation often involve **controlled quantum gates**, where the operation on one qubit depends on the state of another. Some key gates used in entanglement include:

- **Controlled-NOT (CNOT) Gate:** The CNOT gate flips the state of the target qubit if and only if the control qubit is in the $|1\rangle$ state. It is crucial for creating Bell states and entanglement-based quantum circuits.

$$\begin{aligned} \text{CNOT}\,|00\rangle &= |00\rangle\,, \quad \text{CNOT}\,|01\rangle = |01\rangle \\ \text{CNOT}\,|10\rangle &= |11\rangle\,, \quad \text{CNOT}\,|11\rangle = |10\rangle \end{aligned}$$

- **Controlled-Z (CZ) Gate:** The CZ gate applies a Pauli-Z operation to the target qubit if the control qubit is in state $|1\rangle$. It is another common gate for generating entanglement.

$$CZ\,|00\rangle = |00\rangle\,, \quad CZ\,|01\rangle = |01\rangle\,, \quad CZ\,|10\rangle = |10\rangle\,, \quad CZ\,|11\rangle = -\,|11\rangle$$

- **Toffoli Gate (CCNOT):** A three-qubit gate where the third qubit (target) flips if both control qubits are in the $|1\rangle$ state. It extends entanglement control in multi-qubit systems.

- **Controlled-Hadamard (CH) Gate:** A gate that applies the Hadamard transformation only when the control qubit is in state $|1\rangle$, enabling superposition-based entanglement operations.

5.6 Quantum Teleportation

Quantum teleportation is a protocol that allows the transfer of an unknown quantum state from one qubit to another, without physically transmitting the qubit itself. This process relies on **entanglement** and **classical communication**.

5.6.1 Steps of Quantum Teleportation

The teleportation protocol consists of the following steps:

1. **Preparation of an Entangled State:**
   Alice and Bob share an entangled Bell pair. Suppose they each have one qubit of the Bell state:
   $$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$
   Alice also has a third qubit, $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, which she wants to teleport to Bob.

2. **Bell Measurement and Classical Communication:**
   Alice performs a **Bell measurement** on her qubit $|\psi\rangle$ and her part of the entangled pair. This measurement collapses the system into one of four possible Bell states:
   $$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),$$
   $$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle),$$
   $$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$
   $$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$
   Alice then **sends** the result of her measurement to Bob via a classical communication channel.

3. **Bob's Conditional Operations:**
   Based on Alice's measurement result, Bob applies a **corresponding quantum operation** to his qubit:

- If Alice measures $|\Phi^+\rangle$, no operation is needed.
- If Alice measures $|\Phi^-\rangle$, Bob applies a Pauli-Z gate.
- If Alice measures $|\Psi^+\rangle$, Bob applies a Pauli-X gate.
- If Alice measures $|\Psi^-\rangle$, Bob applies both X and Z gates.

After this correction, Bob's qubit is in the original state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, completing the teleportation process.

5.7 Key Features of Quantum Teleportation

- **No Faster-Than-Light Communication:** Since classical communication is required, quantum teleportation does not allow faster-than-light information transfer.
- **Entanglement as a Resource:** Entanglement must be established before teleportation can occur.
- **Application in Quantum Networks:** Quantum teleportation is essential for quantum communication, quantum repeaters, and distributed quantum computing.

5.8 Superdense Coding

Superdense coding is a quantum communication protocol that allows the transmission of **two classical bits of information** using only **one qubit**, with the help of prior entanglement. This protocol demonstrates how quantum entanglement can enhance classical communication efficiency.

5.8.1 Steps of Superdense Coding

The superdense coding protocol consists of the following steps:

1. **Preparation of an Entangled Pair:**
   Alice and Bob share a maximally entangled Bell state:
   $$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$
   Alice wants to send **two classical bits** of information (**00, 01, 10, or 11**) to Bob.
2. **Alice Encodes Information:**
   Depending on the two-bit message she wants to send, Alice applies a quantum operation to her qubit as follows:

Classical Bits (Message)	Quantum Operation Applied by Alice
00	**Identity** (I) (No change)
01	**Pauli-X** (X) gate
10	**Pauli-Z** (Z) gate
11	**Pauli-X and Z** (XZ or Y) gate

   After applying the corresponding quantum gate, Alice sends her qubit to Bob.

3. **Bob Decodes the Message:**
   Bob now possesses both qubits and performs a Bell measurement on the two-qubit system. This measurement collapses the state into one of the four Bell states:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad \textbf{(Message: 00)}$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \quad \textbf{(Message: 01)}$$
$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \quad \textbf{(Message: 10)}$$
$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \quad \textbf{(Message: 11)}$$

   Based on the measurement result, Bob retrieves the original two-bit message.

5.8.2 Key Features of Superdense Coding

- **Increased Classical Capacity:** Superdense coding enables the transmission of **two classical bits** using **one qubit**.
- **Requires Prior Entanglement:** The protocol relies on a pre-shared entangled pair between Alice and Bob.
- **Quantum Advantage:** This method demonstrates the power of quantum communication by doubling the classical capacity of a single qubit.

6 Quantum Algorithms

The Key Properties of Quantum Algorithms:

- **Superposition:** The ability to explore multiple states simultaneously by representing a quantum state as a linear combination of basis states.
- **Entanglement:** The correlation between quantum states that enables shared information, even across large separations.
- **Interference:** The constructive or destructive combination of quantum amplitudes to amplify the "correct" answers and cancel out incorrect ones.

Quantum algorithms utilize these properties, particularly interference, to magnify the amplitude of the correct answers while suppressing incorrect ones. This makes quantum algorithms particularly powerful in scenarios where classical counterparts struggle.
Quantum algorithms are often designed to learn properties of a function, such as:

- Is the function constant (always 0 or always 1)?
- Is the function balanced (outputting 0 and 1 equally)?

Query Complexity: The efficiency of a quantum algorithm is often measured by the number of calls it makes to a black-box function, also known as an **oracle**. An oracle

takes an input x and produces an output $f(x)$. In quantum mechanics, this could be interpreted as mapping the input state $|x\rangle$ to an output state, such as:

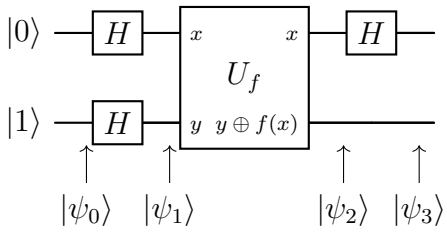
$$|x\rangle \to |x\rangle\,|f(x)\rangle\,,$$

or more commonly:

$$|x\rangle\,|y\rangle \to |x\rangle\,|y \oplus f(x)\rangle\,,$$

where $\oplus$ represents addition modulo 2.

6.1 Deutsch's Algorithm



Deutsch's Algorithm is a foundational quantum algorithm that demonstrates the power of quantum computation in determining a global property of a function using **fewer queries** than any classical algorithm.

Problem Statement: Consider a function $f : \{0, 1\} \to \{0, 1\}$. The task is to determine whether f is:

- **Case 1: Constant** ($f(0) = f(1)$), or
- **Case 2: Balanced** ($f(0) \neq f(1)$).

Classically, solving this problem requires evaluating $f(0)$ and $f(1)$, requiring two queries. Deutsch's Algorithm achieves this in just one query by leveraging quantum superposition and interference.

Algorithm:

1. **Prepare the Initial State (ψ_0):** Start with two qubits initialized to:

$$|\psi_0\rangle = |0\rangle\,|1\rangle\,.$$

2. **Apply Hadamard Gates (ψ_1):** Apply a Hadamard gate (H) to each qubit, creating a superposition state:

$$|\psi_1\rangle = H\,|0\rangle \otimes H\,|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

which simplifies to:

$$|\psi_1\rangle = \frac{1}{2}(|0\rangle\,(|0\rangle - |1\rangle) + |1\rangle\,(|0\rangle - |1\rangle)).$$

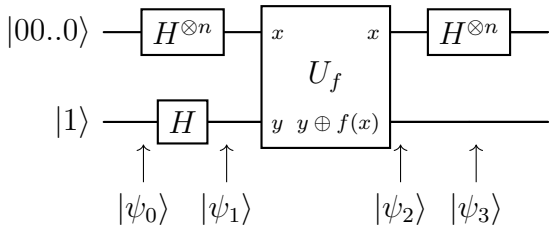
3. **Oracle Query (ψ_2):** Pass the state through the oracle U_f, which performs the transformation:
$$|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle .$$
The state after applying the oracle is:
$$|\psi_2\rangle = \frac{1}{\sqrt{2}} \left[|0\rangle |f(0) \oplus 1\rangle + |1\rangle |f(1) \oplus 1\rangle \right].$$

4. **Undo the Hadamard Transformation on the First Qubit (ψ_3):** Apply another Hadamard gate to the first qubit, creating constructive and destructive interference:
$$|\psi_3\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} (-1)^{x \cdot f(0) \oplus f(1)} |x\rangle |f(x) \oplus 1\rangle .$$

5. **Final State (ψ_3) Before Measurement:** After simplification, the state encodes the function property in the first qubit:
$$|\psi_3\rangle = \begin{cases} |0\rangle |-\rangle & \textbf{(Case 1: Constant, if } f(0) = f(1)\textbf{)}, \\ |1\rangle |-\rangle & \textbf{(Case 2: Balanced, if } f(0) \neq f(1)\textbf{)}. \end{cases}$$

6. **Measure the First Qubit:** Measure the first qubit:
   - $|0\rangle$: **The function is constant.**
   - $|1\rangle$: **The function is balanced.**

6.2 Deutsch-Jozsa Algorithm



The Deutsch-Jozsa Algorithm generalizes Deutsch's Algorithm to determine a property of a function $f : \{0,1\}^n \to \{0,1\}$. It was the first quantum algorithm to demonstrate an exponential speedup over classical algorithms in a specific problem.

Problem Statement: Given a function f, determine whether it is:

- **Case 1: Constant** ($f(x) = 0$ for all x or $f(x) = 1$ for all x), or
- **Case 2: Balanced** ($f(x) = 0$ for half of all inputs and $f(x) = 1$ for the other half).

Classically, solving this problem requires evaluating $f(x)$ for at least $2^{n-1} + 1$ inputs in the worst case. The Deutsch-Jozsa Algorithm solves this in a single query to the oracle using quantum computation.

Algorithm:

1. **Prepare the Initial State (ψ_0):** Start with $n+1$ qubits initialized to:

$$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle .$$

2. **Apply Hadamard Gates (ψ_1):** Apply Hadamard gates (H) to all $n+1$ qubits to create a superposition state:

$$|\psi_1\rangle = H^{\otimes n+1} |\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

3. **Oracle Query (ψ_2):** Pass the state through the oracle U_f, which performs the transformation:

$$|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle .$$

   After applying the oracle, the state becomes:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

4. **Apply Hadamard Gates to the First n Qubits (ψ_3):** Apply Hadamard gates to the first n qubits. The transformation simplifies the state:

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{z=0}^{2^n-1} \left[\sum_{x=0}^{2^n-1} (-1)^{x \cdot z + f(x)} \right] |z\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

   where $x \cdot z$ is the bitwise dot product.

5. **Measure the First n Qubits:** The result depends on whether $f(x)$ is constant or balanced:

   - **Case 1: Constant** — The measurement yields $|0\rangle^{\otimes n}$, as the interference terms constructively add for $z = 0$.
   - **Case 2: Balanced** — The measurement yields a non-zero state, as the interference cancels out for $z = 0$.

6.3 Bernstein-Vazirani Algorithm

The Bernstein-Vazirani Algorithm is a quantum algorithm that determines a hidden string $s \in \{0,1\}^n$ encoded in a function $f : \{0,1\}^n \to \{0,1\}$. It provides a significant speedup compared to classical methods.

Problem Statement: The function $f(x)$ is defined as:

$$f(x) = s \cdot x \pmod 2,$$

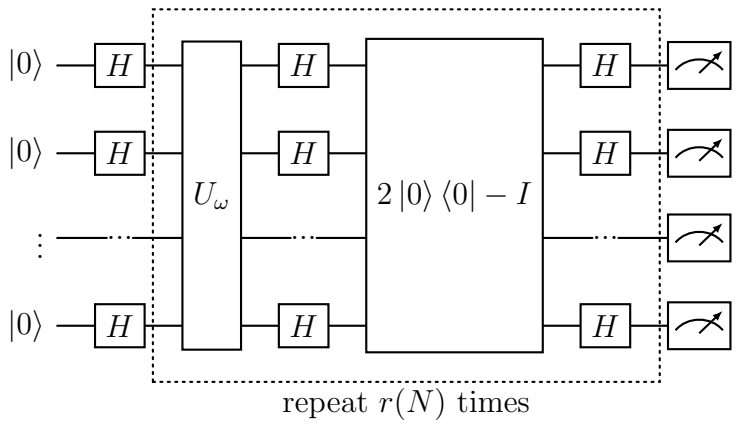
where $s \cdot x = \bigoplus_{i=1}^{n} s_i x_i$ is the bitwise dot product of the hidden string s and the input x. The task is to determine the hidden string s.

- **Classical Approach:** To determine s, a classical algorithm would require n queries to the function f, where each query reveals one bit of s.
- **Quantum Approach:** The Bernstein-Vazirani Algorithm solves this in a single query to the oracle using quantum computation.

Algorithm:

1. **Prepare the Initial State (ψ_0):** Start with $n+1$ qubits initialized to:
$$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle .$$
2. **Apply Hadamard Gates (ψ_1):** Apply Hadamard gates (H) to all $n+1$ qubits to create a superposition state:
$$|\psi_1\rangle = H^{\otimes n+1} |\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$
3. **Oracle Query (ψ_2):** Pass the state through the oracle U_f, which implements the transformation:
$$|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle .$$
Using the definition of $f(x) = s \cdot x \pmod 2$, the oracle flips the phase of the target qubit based on the value of $f(x)$. The resulting state is:
$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{s \cdot x} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$
4. **Apply Hadamard Gates to the First n Qubits (ψ_3):** Apply Hadamard gates to the first n qubits. The Hadamard transformation simplifies the state to:
$$|\psi_3\rangle = |s\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$
where s is the hidden string encoded in $f(x)$.
5. **Measure the First n Qubits:** Measuring the first n qubits yields the hidden string s. The last qubit is discarded as it does not contain any additional information.

6.4 Grover's Algorithm



Grover's Algorithm is a quantum algorithm designed to search an unsorted database or solve the unstructured search problem. It provides a quadratic speedup compared to classical algorithms.

Problem Statement: Given an unstructured database of $N = 2^n$ items and an oracle $f(x)$ such that:

$$f(x) = \begin{cases} 1 & \text{if } x = x_{\text{target}}, \\ 0 & \text{otherwise,} \end{cases}$$

the task is to find the unique target item x_{target}.

Classically, searching an unsorted database requires $O(N)$ queries in the worst case. Grover's Algorithm achieves this in $O(\sqrt{N})$ queries using quantum computation.

Algorithm:

1. **Prepare the Initial State (ψ_0):** Start with n qubits initialized to:
$$|\psi_0\rangle = |0\rangle^{\otimes n}.$$

2. **Apply Hadamard Gates (ψ_1):** Apply Hadamard gates ($H^{\otimes n}$) to all n qubits to create an equal superposition of all possible states:
$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

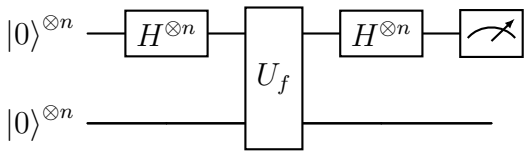
3. **Oracle Query (O_f):** Pass the state through the oracle O_f, which flips the amplitude of the target state $|x_{\text{target}}\rangle$:
$$O_f : |x\rangle \to (-1)^{f(x)} |x\rangle.$$
After applying the oracle, the state becomes:
$$|\psi_2\rangle = \frac{1}{\sqrt{N}} \left[\sum_{x \neq x_{\text{target}}} |x\rangle - |x_{\text{target}}\rangle \right].$$

4. **Apply the Diffusion Operator (Amplification Step):** The diffusion operator (also called the Grover diffusion operator) amplifies the amplitude of the target state by reflecting the quantum state around the mean amplitude. Mathematically, the operator is given by:
$$D = 2 |\psi_1\rangle \langle\psi_1| - I,$$
where I is the identity operator. After applying the diffusion operator, the state becomes:
$$|\psi_3\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} a_x |x\rangle,$$
where the amplitude $a_{x_{\text{target}}}$ of the target state is increased.

5. **Repeat the Oracle and Diffusion Steps:** Repeat the oracle and diffusion steps $O(\sqrt{N})$ times to maximize the probability of measuring x_{target}.

6. **Measure the State:** Measure the quantum state. The result is x_{target} with high probability.

6.5 Simon's Algorithm



Simon's Algorithm is a quantum algorithm that solves Simon's problem, which involves finding a hidden binary string s for a given function $f(x)$ with the promise $f(x) = f(y) \iff x \oplus y = s$. The algorithm achieves exponential speedup over classical methods and can be broken down into three simple steps:

1. **Setup Random Superposition:** Start with $2n$ qubits in the initial state $|0\rangle^{\otimes 2n}$. Apply Hadamard gates to the first n qubits to create an equal superposition of all possible inputs:
$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0\rangle^{\otimes n}.$$
Then, apply the oracle U_f, which maps $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$. Ignoring the second register, the first register collapses to a superposition of states $|x\rangle$ and $|x \oplus s\rangle$:
$$|\psi\rangle = \frac{1}{\sqrt{2}} (|x\rangle + |x \oplus s\rangle).$$

2. **Fourier Sample to Get a Random y:** Apply Hadamard gates to the first n qubits to transform the state into the Fourier basis:
$$H^{\otimes n} |\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} (|z\rangle + (-1)^{s \cdot z} |z\rangle).$$
Measurement of the first n qubits yields a random string y such that:
$$y \cdot s = 0 \mod 2.$$
This gives a single linear equation in the unknown string s. (substrings)

3. **Repeat $n-1$ Times to Solve for s:** Repeat the algorithm $n-1$ additional times to collect a total of $n-1$ independent linear equations in s. Using classical linear algebra, solve the system of equations modulo 2 to determine the hidden string s.

Simon's Algorithm provides a solution to the problem in $O(n)$ queries to the oracle, showcasing an exponential advantage over classical approaches that require $O(2^{n/2})$ queries.

6.5.1 Measurement of $f(x)$ Collapses $|x\rangle$

In quantum computing, measurement causes a quantum state to collapse to a specific basis state. This concept is essential in Simon's Algorithm because it helps distinguish between the superposition of states that encode information about the function $f(x)$.

When an oracle U_f acts on a quantum state, it maps $|x\rangle |0\rangle$ to $|x\rangle |f(x)\rangle$. After this operation, measuring the second register collapses it into a definite state $|f(x)\rangle$. This measurement also impacts the first register. Since the function $f(x)$ is defined such that

$f(x) = f(y) \iff x \oplus y = s$, the first register collapses into a superposition of states $|x\rangle$ and $|x \oplus s\rangle$, where s is the hidden string:

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|x\rangle + |x \oplus s\rangle \right).$$

This collapse ensures that the quantum system retains only the states that are consistent with the promise of the problem, effectively encoding information about the hidden string s.

6.5.2 Fourier Sampling: Hadamard Transform on a Binary String

The concept of partial measurement plays a key role in quantum algorithms, including Simon's Algorithm and Period Finding. To understand this, let us consider the example of a 3-bit function $f(x)$ with inputs and outputs as shown in the table below.

x	$f(x)$
000	001
001	111
010	111
011	010
100	101
101	111
110	000
111	001

The circuit uses a Hadamard transform applied to the input register, followed by the oracle U_f, and then measures the output of the second register. Below is the step-by-step explanation for this example:

1. **Initial State Preparation:** The input qubits are initialized to $|0\rangle^{\otimes 3} |0\rangle^{\otimes 3}$. After applying the Hadamard gates to the first three qubits, the system enters a superposition:
$$|\psi_1\rangle = \frac{1}{\sqrt{8}} \sum_{x \in \{0,1\}^3} |x\rangle |0\rangle .$$

2. **Oracle Application:** The oracle U_f maps $|x\rangle |0\rangle \rightarrow |x\rangle |f(x)\rangle$, entangling the input x with the function value $f(x)$. The state becomes:
$$|\psi_2\rangle = \frac{1}{\sqrt{8}} \sum_{x \in \{0,1\}^3} |x\rangle |f(x)\rangle .$$

3. **Partial Measurement:** The second register (representing $f(x)$) is measured. For this example, let us assume the measurement outcome is $|000\rangle$. This collapses the system to include only the values of x such that $f(x) = 000$. From the table, we observe that this occurs for:
$$x = 110.$$
Therefore, after measurement, the first register collapses to:
$$|\psi_3\rangle = |110\rangle .$$

4. **Hadamard Transform (Fourier Sampling):** To extract further information about the periodicity or hidden structure, a Hadamard transform is applied to the first register. For a 3-bit input, the Hadamard transform is applied individually to each qubit, resulting in:

$$H^{\otimes 3} |110\rangle = \frac{1}{\sqrt{8}} \sum_{z \in \{0,1\}^3} (-1)^{110 \cdot z} |z\rangle .$$

This produces a superposition where the amplitudes encode information about the binary dot product $110 \cdot z \mod 2$. Measurement of this state in the computational basis yields a random string z such that:

$$110 \cdot z = 0 \mod 2.$$

For example, possible outcomes for z include $000, 001, 011, 100$, which satisfy the equation.

Conclusion: Partial measurement of the output register collapses the input register into a subspace consistent with the observed function value. Subsequent operations, like the Hadamard transform, allow extraction of hidden properties, such as periodicity or a hidden string, based on the problem being solved.

6.5.3 Example: Simon's Algorithm with $s = 101$

To better understand Simon's Algorithm, let's go through an example where the hidden string is $s = 101$ (a binary string of length 3). The goal is to find s using the algorithm.

1. **Setup Random Superposition:** Start with $2n = 6$ qubits in the initial state:

$$|\psi_0\rangle = |0\rangle^{\otimes 6} .$$

Apply Hadamard gates to the first $n = 3$ qubits to create a superposition:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^3}} \sum_{x \in \{0,1\}^3} |x\rangle |0\rangle^{\otimes 3} .$$

Pass the state through the oracle U_f, which maps $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$. Since $f(x) = f(y)$ if and only if $x \oplus y = s$, the first register collapses into a superposition:

$$|\psi_2\rangle = \frac{1}{\sqrt{2}} (|x\rangle + |x \oplus 101\rangle) .$$

2. **Fourier Sample to Get a Random y:** Apply Hadamard gates to the first 3 qubits:

$$H^{\otimes 3} |\psi_2\rangle = \frac{1}{\sqrt{2^3}} \sum_{z \in \{0,1\}^3} (-1)^{x \cdot z} \left(|z\rangle + (-1)^{(x \oplus 101) \cdot z} |z\rangle \right) .$$

This simplifies to:

$$|\psi_3\rangle = \frac{1}{\sqrt{2^3}} \sum_{z \in \{0,1\}^3} [1 + (-1)^{s \cdot z}] |z\rangle .$$

Nonzero amplitude exists only for z such that $s \cdot z = 0 \mod 2$. Measuring the first 3 qubits gives a random string y satisfying this equation. For example:

$$y = 110 \quad (\text{since } 101 \cdot 110 = 0 \mod 2).$$

3. **Repeat to Solve for s:** Repeat the process to obtain additional random strings y satisfying $s \cdot y = 0 \mod 2$. For example:

$$y_1 = 110, \quad y_2 = 011, \quad y_3 = 111.$$

These provide the following linear equations in s:

$$1s_1{+}1s_2{+}0s_3 = 0 \mod 2, \quad 0s_1{+}1s_2{+}1s_3 = 0 \mod 2, \quad 1s_1{+}1s_2{+}1s_3 = 0 \mod 2.$$

Solving this system of equations (modulo 2) reveals $s = 101$.

6.6 Period Finding

Period finding is a fundamental problem in quantum computation that underpins important algorithms like Shor's Algorithm for factoring large integers. The task involves finding the period r of a function $f(x)$, which is defined as the smallest positive integer such that:

$$f(x + r) = f(x) \quad \text{for all } x.$$

This problem demonstrates the power of quantum algorithms, as the period can be found exponentially faster using quantum techniques compared to classical methods.

6.6.1 Classical Period Finding

Classically, period finding relies on evaluating the function $f(x)$ for different values of x until a repeating pattern is detected. For example, consider the function:

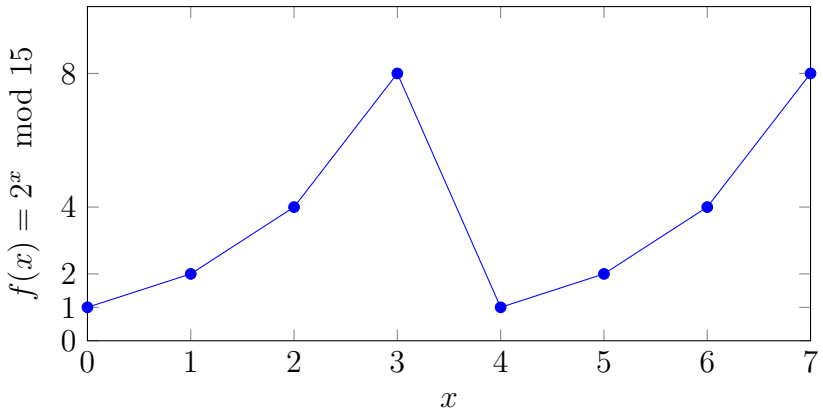
$$f(x) = \text{mod}(2^x, 15),$$

where $f(x)$ is periodic. The table below shows the values of $f(x)$ for $x = 0, 1, 2, \ldots$:

x	$f(x) = 2^x \mod 15$
0	1
1	2
2	4
3	8
4	1
5	2
6	4
7	8

From this table, we can observe that the function has a period of $r = 4$ since $f(x{+}4) = f(x)$ for all x.

Graphical Representation: Classically, we would evaluate $f(x)$ at discrete points and look for patterns. The graph below illustrates how $f(x)$ repeats periodically.



Classical methods require evaluating $f(x)$ for many values to detect the period. In general, this requires $O(r)$ evaluations in the worst case.

6.6.2 Quantum Period Finding

Quantum computation allows us to find the period r exponentially faster by leveraging superposition and interference. The process involves the following steps:

1. **Create a Superposition:** Start with two quantum registers:

$$|\psi_0\rangle = |0\rangle^{\otimes n} |0\rangle^{\otimes m} .$$

   Apply Hadamard gates to the first register to create a superposition of all possible inputs:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle .$$

2. **Apply the Oracle:** Pass the state through a quantum oracle U_f, which maps $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$. The state becomes:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle .$$

3. **Fourier Transform:** Discard the second register and apply the Quantum Fourier Transform (QFT) to the first register. The QFT maps the input state into the frequency domain, encoding information about the period r in the amplitudes of the resulting state:

$$|\psi_3\rangle = \frac{1}{r} \sum_{k=0}^{r-1} \sum_{j=0}^{r-1} e^{2\pi i k j / r} |k\rangle .$$

   Measurement of the state yields a value k, which is related to the period r.

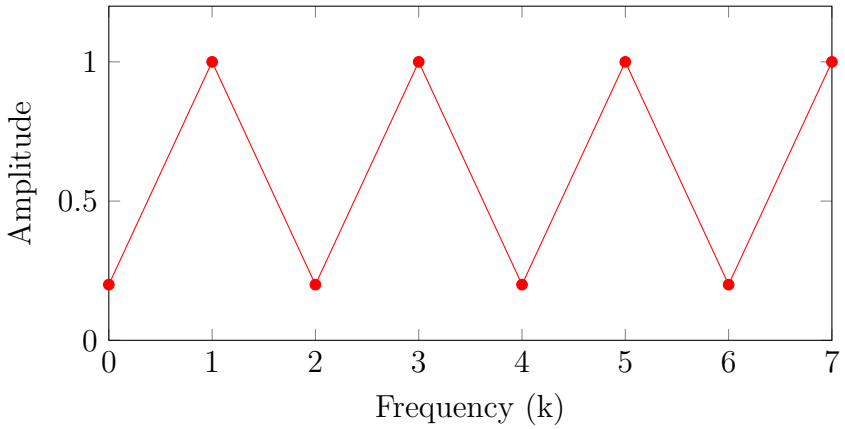
4. **Classical Post-Processing:** Using the measured value k, compute the period r using continued fractions or other classical techniques.

6.6.3 Requirements for Period Finding

For the quantum period-finding algorithm to work correctly, the following requirements must be met:

- **All the values within the period must be unique:** This ensures that the periodic structure of the function $f(x)$ can be clearly identified and leveraged during the algorithm.
- **The number of data points (M) must be much larger than the period (N) or it must be an integer multiple of the period (N):** This condition ensures that the periodic structure is adequately represented in the domain of the function $f(x)$.

Graphical Representation: In the quantum approach, the Fourier Transform amplifies the frequencies associated with the period r, as shown in the conceptual graph below.



Quantum period finding requires $O(\log N)$ oracle calls, showcasing exponential speedup over the classical approach.

6.7 Quantum/Classical Runtimes

The table below compares the time complexities of classical and quantum algorithm solutions for various problems. The quantum algorithms offer significant improvements in query complexity over their classical counterparts.

Problem	Classical Time Complexity	Quantum Query Complexity
Deutsch-Jozsa Problem	$O(2^N)$	$O(1)$
Bernstein-Vazirani Problem	$O(N)$	$O(1)$
Simon's Problem	$O(2^{N/2})$	$O(N)$
Grover's Problem	$O(N)$	$O(\sqrt{N})$

7 Errors and Benchmarking

7.1 Overview of Errors

Errors in quantum computing refer to undesired changes in the state of a qubit. Qubits are inherently low-energy devices, making their states susceptible to interactions with

the environment. Additionally, qubit control is imprecise, leading to operations that may produce unintended effects.

- **Idle Errors**: Occur when no operation is being performed on the qubit.
- **Active Errors**: Occur during the execution of an operation.

7.2 Coherence Errors

Coherence errors quantify how long a qubit can maintain its quantum state:

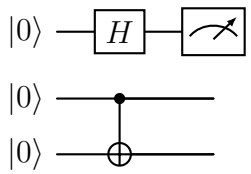
- **T1 (Relaxation Time)**: Time for a qubit to decay from the excited state ($|1\rangle$) to the ground state ($|0\rangle$).
- **T2 (Dephasing Time)**: Time for a qubit to lose phase coherence between superposition states.

7.3 Gate Errors

Quantum gates are highly error-prone due to imperfect control and environmental noise. Gate errors are categorized as follows:

- **Single-Qubit Gate Errors**: Affect gates like X, Y, Z, and Hadamard (H). Typical error rate: $\sim 0.1\%$.
- **Multi-Qubit Gate Errors**: Affect gates like CNOT. Typical error rate: $\sim 1\%$.

Example of a single-qubit gate (H) and a multi-qubit gate (CNOT):



How to Measure Gate Errors? Gate fidelity is often measured using randomized benchmarking or quantum process tomography to quantify the difference between the ideal and actual operation.

7.4 Measurement Errors

Even if a quantum computation is error-free, measurement introduces its own errors. These errors are not symmetrical (e.g., misreading $|0\rangle$ as $|1\rangle$ may differ in probability from the reverse). Typical measurement error rates depend on the hardware but can range from 1% to 10%.

7.5 Crosstalk and Idle Errors

- **Crosstalk**: CNOT error rates increase significantly when nearby qubits undergo operations. Mitigation involves regulating operation concurrency.
- **Idle Errors**: An idle qubit becomes more error-prone if nearby qubits are active. This can be reduced by keeping idle qubits "busy" with dummy operations (e.g., identity gates).

Example of a dummy operation to keep an idle qubit busy: $|\psi\rangle$ —[I]—[I]—

7.6 Benchmarking

Benchmarking aims to quantify and compare the effectiveness of different quantum hardware and to track technological milestones, assessing progress in the field. It consists of two key elements:

1. **Metric**: What to measure.
2. **Workload**: On what to measure (e.g., specific circuits or applications).

7.6.1 What Metrics to Compare?

Metrics for benchmarking quantum systems are categorized into three levels:

1. **Device-Level Metrics**:
   - Gate errors (e.g., single-qubit: $\sim 0.1\%$, CNOT: $\sim 1\%$).
   - Coherence times (T1, T2).
   - SPAM (State Preparation and Measurement) errors.
2. **Circuit-Level Metrics**:
   - Probability of error-free output.
   - Distance-based metrics (see below).
3. **Application-Level Metrics**:
   - Inference ability or quality of the answer for specific tasks.

7.6.2 Probability of Successful Trial (PST)

The Probability of Successful Trial (PST) is defined as:

$$\text{PST} = \frac{\text{Number of error-free trials}}{\text{Total number of trials}}$$

PST depends not only on device-level metrics but also on circuit design and workload complexity. For example, a simple circuit to test PST might look like: $|0\rangle$ —[H]—[X]—[↗]

Here, success is measured by correctly obtaining the expected output after applying H and X gates.

7.6.3 Distance-Based Metrics

Distance-based metrics compare the ideal output distribution (P_{ideal}) to the actual device output (P_{device}). Common examples include:

- **Hellinger Distance**: Measures the "overlap" between two probability distributions.
- **KL Divergence**: Quantifies how much one distribution diverges from another.
- **Total Variation Distance**: Measures the maximum difference between probabilities.
- **Cross Entropy**: Evaluates the difference between predicted and true distributions.

Fidelity is often defined as:

$$F = D(P_{\text{ideal}}, P_{\text{device}})$$

where D is a distance function (e.g., 1 minus the Hellinger distance). Note: All these metrics require knowledge of the ideal output.

7.6.4 Inference Strength (IST)

Inference Strength (IST) measures a quantum device's ability to solve application-specific problems (e.g., machine learning inference). Unlike device-level metrics, IST focuses on the quality of the output rather than raw error rates. Details depend on the specific application.

7.7 Applications

7.7.1 IBM Quantum Volume

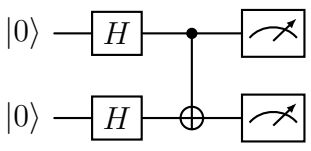
Quantum Volume (QV) is a metric developed by IBM to assess the capability of quantum hardware. It is based on the largest square circuit (equal number of qubits and depth) that can be run reliably.

- **Circuit Characteristics**: A square circuit produces "peaky" outputs, meaning the probability distribution of outcomes is concentrated on a subset of "heavy" outputs.
- **Reliability Definition**: A circuit is considered reliable if the probability of measuring "heavy" outputs exceeds 2/3.
- **Quantum Volume Formula**: For a reliable square circuit on d qubits, QV is defined as:

$$\text{QV} = 2^d$$

- **Reporting**: Quantum Volume is reported as the largest 2^d achieved reliably on the hardware.

Example of a simple square circuit (2 qubits, depth 2):



7.7.2 Pitfalls of Quantum Volume

While Quantum Volume is a useful metric, it has limitations:

- **Qubit Mapping Dependency**: Not all qubits have the same error rates, and QV depends on how the circuit is mapped to the physical hardware.
- **Classical Computation Overhead**: Determining "heavy" outputs requires classical simulation, which limits the testable circuit size.
- **Near-Term Implications**: In the near term, achieving a higher QV often requires qubits with lower error rates, which may not reflect overall system improvements.

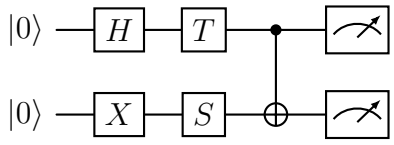
7.8 Quantum Supremacy

Quantum supremacy refers to demonstrating that a programmable quantum device can solve a problem that no classical computer can solve in any feasible amount of time, regardless of the problem's practical utility.

7.8.1 Forms of Quantum Supremacy

- **Weak Form**:
  - The problem need not be useful (e.g., sampling from random quantum circuits).
  - Can potentially be demonstrated with near-term, noisy intermediate-scale quantum (NISQ) machines.

  Example of a random circuit for weak supremacy:

  This circuit applies a mix of single-qubit gates (H, T, X, S) and a CNOT, producing a hard-to-simulate output distribution.

- **Strong Form**:
  - Involves problems of commercial interest (e.g., integer factorization using Shor's algorithm).
  - Likely requires larger machines that support fault-tolerant quantum computation.

8 NISQ Model of Computing

Noisy Intermediate-Scale Quantum (NISQ) computers represent the current era of quantum computing, characterized by devices with a moderate number of qubits (typically 50–100) that are prone to noise and lack full error correction. In the NISQ model, a quantum program is executed for N trials. The outputs from these trials are statistically analyzed to determine the most likely error-free result, often by identifying the outcome with the highest probability of being correct. This probabilistic approach compensates

for the inherent noise in NISQ hardware, which introduces errors in gate operations and qubit measurements.

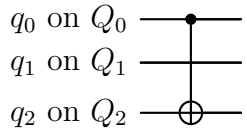
8.1 Limited Connectivity in NISQ Systems

One of the primary challenges in NISQ computing is the limited connectivity between qubits. In quantum circuits, operations like the Controlled-NOT (CNOT) gate require direct interaction between two qubits (e.g., qubits A and B). However, in many NISQ architectures, such as those from IBM, qubits are arranged in a specific topology (e.g., a 2D grid or a linear chain) where only physically adjacent qubits can interact directly. If A and B are not neighbors in the hardware's connectivity graph, the CNOT gate cannot be applied without additional steps, leading to the need for qubit mapping and routing.

8.2 Qubit Mapping Problem

The qubit mapping problem involves assigning logical qubits (from the quantum algorithm) to physical qubits (on the hardware) in a way that respects the hardware's connectivity constraints. The efficiency of this mapping directly impacts the number of SWAP operations required to execute the circuit. A SWAP operation exchanges the states of two adjacent qubits, allowing non-adjacent qubits to be brought together for a two-qubit gate. However, each SWAP operation introduces additional gates (typically three CNOTs), increasing circuit depth, execution time, and error rates due to noise.

For example, consider a simple circuit with three logical qubits q_0, q_1, and q_2, where q_0 needs a CNOT with q_2, but the hardware topology is a linear chain: $Q_0 - Q_1 - Q_2$. If q_0 is mapped to Q_0 and q_2 to Q_2, a SWAP between Q_1 and Q_2 might be needed to make q_0 and q_2 adjacent.



After SWAP: q_0 on Q_0, q_2 on Q_1, q_1 on Q_2, followed by CNOT.

Efficient mapping algorithms aim to minimize the number of SWAPs by optimizing the initial placement of qubits based on the circuit's gate sequence and the hardware's topology.

8.3 Qubit Routing Problem

The qubit routing problem is closely related to mapping and is handled by the quantum compiler. It involves determining how to move qubit states across the hardware topology to enable multi-qubit gates between non-adjacent qubits. The compiler allocates physical qubits and inserts SWAP operations as needed, striving to minimize the total number of SWAPs to reduce overhead. This process is computationally challenging, often formulated as a g**raph-based optimization problem where the hardware topology is a graph, and edges represent possible interactions.**

For instance, on a 2D grid topology, routing a qubit from one corner to the opposite corner may require multiple SWAPs along a path of adjacent qubits. Advanced routing algorithms, such as those based on A* search or dynamic programming, are employed to find near-optimal solutions.

8.4 NISQ Compilation Pipeline

The compilation process transforms a high-level quantum program into an executable form tailored to the target NISQ hardware. The pipeline can be summarized as follows:

1. **Quantum Program**: The user writes a high-level algorithm (e.g., in Qiskit).
2. **Circuit Generator**: Converts the program into a virtual quantum circuit with logical qubits and gates.
3. **Quantum Mapper**: Maps logical qubits to physical qubits, producing a mapped circuit.
4. **Assembler**: Translates the mapped circuit into a machine-executable sequence of instructions.

8.5 Optimization and Decomposition Passes

Quantum compilers perform several optimization and rewriting steps to adapt the circuit to NISQ constraints:

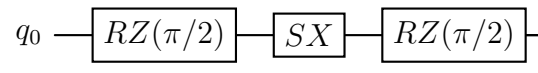
1. **Virtual Circuit Optimization**: Simplifies the circuit by removing redundant gates (e.g., consecutive identical gates) or commuting operations to reduce depth.
2. **3+ Qubit Gate Decomposition**: Breaks down multi-qubit gates (e.g., Toffoli) into sequences of one- and two-qubit gates, as NISQ hardware typically supports only these.
    - Example: A Toffoli gate can be decomposed into 6 CNOTs and several single-qubit gates.
3. **Placement of Physical Qubits**: Assigns logical qubits to physical qubits based on connectivity and error rates.
4. **Routing on Restricted Topology**: Inserts SWAP operations to align qubits for two-qubit gates.
5. **Translation to Basis Gates**: Converts all gates into the hardware's native basis gate set.
6. **Physical Circuit Optimization**: Applies final optimizations, such as gate cancellation or rescheduling, to minimize errors and depth.

8.6 Basis Gates and Decomposition

Basis gates are the fundamental operations directly implementable on the hardware. For example, on IBM quantum machines, the basis gates are:

- **SX**: Square-root of X gate (a $\pi/2$ rotation around X).
- **RZ(θ)**: Rotation around the Z-axis by angle θ.
- **X**: Pauli X gate (bit flip).
- **CX**: Controlled-X (CNOT) gate.

The compiler translates all gates in the circuit (e.g., Hadamard H, T, or CZ) into sequences of these basis gates. For instance:

- $H = RZ(\pi/2) \cdot SX \cdot RZ(\pi/2)$.
- A T gate might be approximated using $RZ(\pi/4)$ and other gates, depending on the hardware.
- 

Illustrates a Hadamard gate decomposed into basis gates.

8.7 Qubit Mapping and Routing Algorithms

Qubit mapping and routing are critical tasks in compiling quantum circuits for NISQ devices, where limited connectivity between physical qubits necessitates careful allocation and movement of qubit states. Two approaches are outlined below: the SABRE algorithm, which leverages the reversibility of quantum circuits, and the typical layer-based mapping and routing strategy.

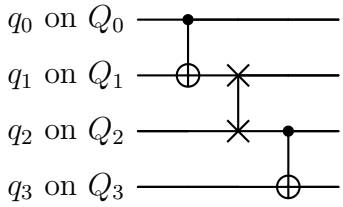
8.7.1 SABRE Algorithm

The SABRE (Swap-based BidiRectional heuristic search) algorithm is an advanced method for qubit mapping and routing that exploits the reversibility of quantum circuits to optimize performance on NISQ hardware. Since quantum circuits (excluding measurements) are reversible, SABRE uses both the forward circuit and its reverse counterpart to iteratively refine qubit mappings. The key steps of SABRE are:

1. **Initial Random Allocation**: Start with a random mapping of logical qubits to physical qubits on the hardware topology.
2. **Bidirectional Execution**: Run the forward circuit (original sequence of gates) and the reverse circuit (gates applied in reverse order) together. The reverse circuit effectively "undoes" the forward circuit, providing insight into how qubit states evolve across the topology.
3. **Destination as Starting Point**: Use the final qubit mapping of the forward circuit as the initial mapping for the next iteration of the forward circuit. This bidirectional approach helps identify mappings that minimize SWAP operations.

4. **Iterative Refinement**: Repeat the process, adjusting the mapping heuristically based on a cost function (e.g., number of SWAPs or circuit depth), until an optimized mapping is achieved.

SABRE's strength lies in its ability to explore the solution space efficiently, balancing the trade-off between computational overhead and circuit fidelity. By iterating with both forward and reverse circuits, it reduces the number of SWAP operations needed to align qubits for two-qubit gates, such as CNOTs, on restricted topologies.



Illustrates a forward circuit with a SWAP inserted to enable a subsequent CNOT, followed by reverse circuit considerations.

8.7.2 Typical Qubit Mapping and Routing

The more conventional approach to qubit mapping and routing involves a layer-based design that systematically transforms the circuit to fit the hardware's connectivity graph. This process can be broken down into the following steps:

1. **Split Program into Layers**: Divide the quantum circuit into sequential layers, where each layer consists of gates that can be executed simultaneously (i.e., gates acting on disjoint sets of qubits). For example, in a circuit with gates $\text{CNOT}(q_0, q_1)$ and $\text{CNOT}(q_2, q_3)$ followed by $\text{CNOT}(q_1, q_2)$, the first two gates might form one layer, and the third gate a subsequent layer.

2. **Find Variable-to-Device Map for Each Layer**: For each layer, determine a mapping of logical qubits to physical qubits that satisfies the hardware's connectivity constraints for the gates in that layer. This mapping may differ between layers due to changing qubit interaction requirements.

3. **Find SWAP Sequence to Transform i-th Layer into $(i+1)$-th Layer**: Between consecutive layers, compute a sequence of SWAP operations to transition the qubit mapping from the i-th layer to the $(i+1)$-th layer. The goal is to minimize the number of SWAPs while ensuring that qubits involved in two-qubit gates in the next layer are adjacent.

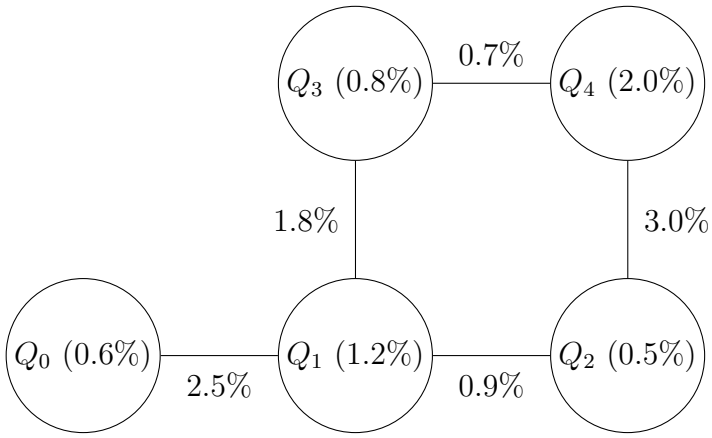
This layer-based approach is systematic but can be computationally expensive, as finding optimal SWAP sequences is an NP-hard problem. Heuristic or greedy algorithms are typically employed to approximate solutions.

9 Error Mitigation Techniques

9.1 Variability-Aware Mapping

Not all qubits are created equal. Variability exists due to differences in qubit quality and connectivity, where some qubits and links fail with higher probability than others. Avoiding unreliable links can significantly improve the overall reliability of quantum computations. By exploiting the variation in error rates, we can enhance reliability by assigning more operations to dependable qubits and links.

Graph Example: Consider a 5-qubit quantum processor with the connectivity shown below. Each node represents a qubit, labeled with its single-qubit gate error rate, and each edge represents a link, labeled with its two-qubit gate error rate. This layout can guide VQA and VQM decisions.



In this graph, Q_2 (0.5%) is the most reliable qubit for single-qubit operations, making it a prime candidate for VQA. For two-qubit gates, the Q_3–Q_4 link (0.7%) is the most reliable, so VQM would favor routing operations through it over, say, the Q_2–Q_4 link (3.0%).

9.1.1 Characterization Methodology

Quantum computers require frequent calibration to optimize gate pulses for the best performance. For instance, IBM calibrates its machines multiple times a day to maintain accuracy and minimize errors. This process ensures that the error characteristics of qubits and links are well-understood and up-to-date, enabling effective mitigation strategies.

9.1.2 Variation-Aware Qubit Allocation (VQA)

This technique involves mapping quantum circuits to physical qubits by prioritizing the allocation of operations to qubits with lower error rates. VQA aims to minimize the impact of variability by leveraging the most reliable qubits available on the hardware.
Example: In the graph above, a circuit requiring a series of single-qubit gates would prioritize Q_2 (0.5% error rate) over Q_4 (2.0% error rate) to reduce cumulative errors.

9.1.3 Variation-Aware Qubit Movement (VQM)

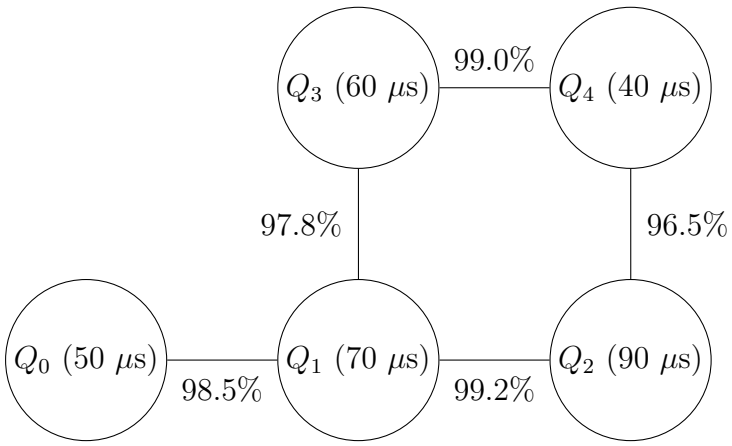
VQM focuses on optimizing the movement of qubits during computation. By strategically routing quantum states through reliable links and avoiding high-error connections, VQM

reduces the likelihood of errors introduced during qubit interactions or swaps.
Example: Using the graph, a CNOT between Q_1 and another qubit would prefer the Q_1–Q_2 link (0.9% error) over the Q_0–Q_1 link (2.5% error) to minimize two-qubit gate errors.

9.2 Diversity-Aware Mapping

Quantum hardware exhibits diversity not only in error rates but also in qubit properties such as coherence times, gate fidelities, and susceptibility to specific noise types (e.g., dephasing or crosstalk). Diversity-Aware Mapping exploits these differences to match circuit requirements with qubit characteristics, optimizing performance for specific workloads. Unlike variability-aware mapping, which focuses on avoiding high-error components, this approach actively selects qubits and links based on their unique strengths.

Graph Example: Below is a 5-qubit processor where qubits are labeled with coherence times (in microseconds, μs) instead of error rates, and edges show two-qubit gate fidelities. This highlights diversity in qubit properties.



In this graph, Q_2 (90 μs) has the longest coherence time, ideal for operations requiring sustained quantum states, while the Q_1–Q_2 link (99.2% fidelity) is best for high-precision two-qubit gates. Diversity-Aware Mapping would assign tasks based on these strengths.

Example: For a quantum circuit with a long sequence of single-qubit gates followed by a CNOT, Diversity-Aware Mapping might allocate the single-qubit gates to Q_2 (90 μs coherence) to minimize decoherence, and route the CNOT through the Q_1–Q_2 link (99.2% fidelity) for maximum accuracy.

9.3 Mitigating Measurement Errors

- **Static Invert and Measure (SIM):** This technique involves applying a fixed inversion operation to the system before measurement to counteract known errors. By calibrating the inversion based on a static model of noise or distortion, SIM aims to recover the true signal or state with minimal additional computation. It is particularly effective in systems where error characteristics are well-understood and stable over time.

- **Adaptive Invert and Measure (AIM):** Unlike SIM, this method dynamically adjusts the inversion operation based on real-time feedback or evolving error profiles. AIM employs adaptive algorithms to continuously refine the measurement process, making it suitable for environments with fluctuating noise or unpredictable disturbances. This approach often requires more computational resources but offers greater robustness in complex scenarios.

10 QAOA

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm **designed to tackle combinatorial optimization problems**. It employs a parameterized quantum circuit, with parameters optimized classically to approximate the optimal solution. QAOA is particularly effective for problems like Max-Cut, utilizing quantum superposition and entanglement to efficiently explore the solution space.

10.1 MaxCut Problem

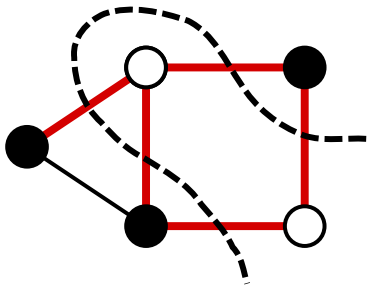


Figure 2: Example cut where the white and black nodes are in their own sets

The MaxCut problem involves partitioning all nodes of an undirected graph $G = (V, E)$ into two sets, Set 0 and Set 1, to maximize the number of edges connecting nodes between the two sets. This is a classic NP-hard optimization problem with applications in physics, circuit design, and machine learning.

10.1.1 Cost Function

The cost function for MaxCut quantifies the **quality** of a partition. For a graph $G = (V, E)$, assign each node $v_i \in V$ a binary variable x_i, where $x_i = 0$ if v_i is in Set 0, and $x_i = 1$ if v_i is in Set 1. The cost function, or the number of edges cut, is given by:

$$C(x) = \sum_{(i,j)\in E} \frac{1 - (-1)^{x_i + x_j}}{2}$$

For an edge (i, j):

- If $x_i = x_j$ (both nodes in the same set), the term $\frac{1-(-1)^{x_i+x_j}}{2} = 0$, meaning the edge is not cut.
- If $x_i \neq x_j$ (nodes in different sets), the term equals 1, meaning the edge is cut.

The goal is to maximize $C(x)$, which counts the total number of cut edges.

10.2 Solving Maxcut with QAOA

QAOA solves the MaxCut problem by encoding it into a quantum circuit and iteratively optimizing parameters. The process is as follows:

1. **Create a Problem Hamiltonian U_C:** Define the cost Hamiltonian H_C based on the MaxCut cost function:

$$H_C = \sum_{(i,j)\in E} \frac{1 - Z_i Z_j}{2}$$

   where Z_i is the Pauli-Z operator on qubit i. The unitary operator $U_C(\gamma) = e^{-i\gamma H_C}$ is applied with parameter γ, encoding the problem's objective into the quantum state. Eigenstates of H_C correspond to possible partitions, with eigenvalues equal to the number of cut edges.

2. **Apply Mixing Hamiltonian U_B:** Introduce a mixing Hamiltonian $H_B = \sum_{i\in V} X_i$, where X_i is the Pauli-X operator on qubit i. The unitary $U_B(\beta) = e^{-i\beta H_B}$ with parameter β drives transitions between states, helping the system avoid local minima by exploring the solution space via quantum superposition.

3. **Measure the Output and Calculate Expected Cost:** Prepare an initial state (typically $|+\rangle^{\otimes n}$, a uniform superposition), apply $U_C(\gamma)$ and $U_B(\beta)$ for p layers (where p is the depth), and measure in the computational basis. Compute the expectation value $\langle H_C \rangle$ of the cost Hamiltonian, which represents the expected number of cut edges for the current parameters γ and β.

4. **Adjust Parameters to Maximize Expected Cost:** Use a classical optimizer (e.g., gradient descent) to adjust γ and β such that $\langle H_C \rangle$ is maximized. This step tunes the quantum circuit to favor states with higher cut values.

5. **Repeat Until Convergence:** Iterate steps 3 and 4, measuring and optimizing, until $\langle H_C \rangle$ converges to a stable maximum or a predefined number of iterations is reached.

6. **Compute MaxCut from Solution String:** After optimization, sample the final quantum state multiple times. The most frequent bitstring (e.g., 0101 for 4 nodes) encodes the partition (e.g., Set 0: nodes with 0, Set 1: nodes with 1). Verify the cut size using the cost function $C(x)$.

11 Quantum Error Correction

Quantum error correction (QEC) is essential for building reliable quantum computers, as quantum states are highly susceptible to noise and decoherence. Unlike classical error correction, QEC faces unique challenges due to the principles of quantum mechanics. Below, we outline these challenges and describe how quantum errors are modeled.

11.1 Challenges in Quantum Error Correction

1. **Encoding and the No-Cloning Theorem:** The no-cloning theorem states that an arbitrary quantum state $|\psi\rangle$ cannot be perfectly copied. This prevents straightforward redundancy (e.g., duplicating a qubit), a common classical error correction technique. Instead, QEC encodes logical qubits into entangled states across multiple physical qubits, such as in the $|0_L\rangle = |000\rangle$ and $|1_L\rangle = |111\rangle$ code.

2. **Detection: Measurement is Destructive:** Measuring a quantum state collapses it to a basis state (e.g., $|0\rangle$ or $|1\rangle$), destroying superposition or entanglement. Direct measurement of a qubit to detect errors is thus impractical. QEC uses syndrome measurements, which indirectly infer errors by measuring parity without collapsing the logical state.

3. **Inability to Correct Phase Errors:** Early codes, like the bit-flip code, correct amplitude errors (e.g., $|0\rangle \to |1\rangle$) but fail to address phase errors (e.g., $|+\rangle \to |-\rangle$). Phase errors arise from operators like the Pauli-Z (Z), requiring codes like the Shor code to handle both types.

4. **Inability to Correct Small Errors:** Quantum errors are continuous (e.g., small rotations), not just discrete flips. A small error $|\psi\rangle \to \cos\theta|\psi\rangle + i\sin\theta|\phi\rangle$ (where θ is small) cannot be perfectly corrected by discrete codes. However, if errors are below a threshold, QEC can approximate correction by projecting onto a discrete error set.

11.2 Modeling Quantum Errors

Quantum errors are typically modeled using Pauli operators (X, Y, Z) acting on a single qubit.

1. **Bit Flip Error:** A bit flip error, represented by the Pauli-X operator, flips $|0\rangle$ to $|1\rangle$ and vice versa

2. **Phase Flip Error:** A phase flip error, represented by the Pauli-Z operator, introduces a relative phase of -1 to $|1\rangle$

3. **Bit and Phase Flip Error:** A combined bit and phase flip is represented by the Pauli-Y operator (up to a global phase, $Y = iXZ$)

4. **Measurement Errors:** Measurement errors occur post-measurement and are modeled as classical bit flips (e.g., recording 0 as 1).

11.3 Classical Replication Codes

Classical replication codes add redundancy to protect data in computing systems from errors.

- **Triple Modular Redundancy (TMR):** TMR makes three copies of data (e.g., bit 1 becomes $(1, 1, 1)$). A majority vote picks the output. If one copy fails (e.g., $(1, 1, 0)$), the vote still gives 1. It fixes one error but not two.

- **Role of Distance (d):** Distance d is how many spots two codewords differ (e.g., $(0,0,0)$ vs. $(1,1,1)$ has $d=3$). In TMR, $d=3$ means it can fix 1 error and detect up to 2. Bigger d (like 5 copies) means more errors can be handled, but it uses more resources.

  - A code can **detect** up to $d-1$ errors.
  - A code can **correct** up to $\lfloor (d-1)/2 \rfloor$ errors, where $\lfloor x \rfloor$ denotes the floor function (the largest integer less than or equal to x).

11.4 Shor's Code

- **Purpose**: Protects quantum information from errors due to decoherence and other noise in quantum systems.
- **Type**: A quantum error-correcting code, specifically a 9-qubit code.
- **Error Correction Capability**: Corrects arbitrary single-qubit errors (bit-flip, phase-flip, or both).
- **Structure**: Encodes **1 logical qubit** into **9 physical qubits**.
- **Error Detection**: Uses syndrome measurements to identify the type and location of errors. The code employs 8 stabilizer operators to detect bit-flip and phase-flip errors across the 9 qubits.
- **Error Correction**: Applies recovery operations based on the syndrome. Bit-flip errors are corrected by majority voting within three-qubit blocks, while phase-flip errors are corrected across the three blocks.

11.5 Steane Code and Concatenated Code

Steane Code

- **Purpose**: A quantum error-correcting code designed to protect quantum information from noise and errors.
- **Type**: A 7-qubit code, based on the classical [7,4,3] Hamming code extended to quantum systems.
- **Error Correction Capability**: Corrects arbitrary single-qubit errors (bit-flip, phase-flip, or both).
- **Structure**: Encodes **1 logical qubit** into **7 physical qubits**.
- **Error Detection**: Uses 6 stabilizer operators (3 for bit-flips, 3 for phase-flips) to detect errors via syndrome measurements.
- **Error Correction**: Applies recovery operations based on the syndrome, leveraging the classical Hamming code structure for both bit and phase corrections.
- **Advantage**: More efficient than Shor's 9-qubit code in terms of qubit usage.

Concatenated Code

- **Purpose**: Enhances error correction by repeatedly encoding quantum information to achieve fault tolerance.
- **Type**: A hierarchical quantum code where a base code (e.g., Steane or Shor) is recursively applied.
- **Error Correction Capability**: Reduces error rates exponentially with each level of concatenation, provided the physical error rate is below a threshold.
- **Structure**: Encodes a logical qubit into multiple layers, e.g., 1 logical qubit into 7 physical qubits (Steane), then each of those into 7 more, and so on.
- **Error Detection**: Errors are detected at each concatenation level using the stabilizers of the base code.
- **Error Correction**: Corrects errors layer by layer, with each level refining the protection of the logical qubit.
- **Trade-off**: Increases the number of physical qubits and complexity but allows for arbitrarily low logical error rates.

11.6 Correcting Errors in Ancilla and Measurement

- **Purpose**: Addresses errors introduced during quantum error correction processes involving ancilla qubits and syndrome measurements.
- **Ancilla Role**: Ancilla qubits are auxiliary qubits used to extract error syndromes without directly measuring the data qubits, preserving quantum coherence.
- **Error Sources**:
  - Errors in ancilla preparation (e.g., faulty initialization).
  - Errors during ancilla-data qubit interactions (e.g., faulty gates).
  - Measurement errors when reading out the ancilla state.
- **Error Detection in Ancilla**:
  - Repeated measurements of ancilla qubits to verify consistency of syndromes.
  - Use of multiple ancilla qubits per syndrome to detect discrepancies (e.g., majority voting).
- **Error Correction in Ancilla**:
  - Reset and re-prepare ancilla qubits if errors are detected.
  - Apply fault-tolerant gate sequences (e.g., using transversal gates) to minimize error propagation from ancilla to data qubits.
- **Measurement Error Mitigation**:
  - Perform multiple syndrome measurements over time and use error-correcting codes (e.g., repetition codes) to filter out measurement noise.

  - Implement fault-tolerant measurement protocols, such as Shor-style or Steane-style ancilla coupling, to reduce the impact of single measurement failures.

- **Challenges**:
  - Balancing resource overhead (more ancilla qubits, more measurements) with error suppression.
  - Ensuring fault tolerance, where a single ancilla or measurement error does not corrupt the logical qubit.

- **Connection to Codes**: Techniques are often integrated with codes like Shor's or Steane's, enhancing their reliability in realistic quantum systems.