Quantum Technology

Justin H. Wilson

2024-12-21

Table of contents

Preface **5**

1 Introduction **6**
1.1 The Quantum-Classical Arms Race . . . 6
1.2 Early Computing: The Lesson of Transistors . . . 7
1.2.1 The Dream: Field Effect Transistors . . . 7
1.2.2 The Point Contact Transistor . . . 9
1.2.3 Surface physics and transistors . . . 10
1.2.4 Where are we with quantum computing? . . . 10
1.3 Review of the Postulates of Quantum Mechanics . . . 11
1.3.1 Postulate I: The Hilbert space . . . 11
1.3.2 Postulate II: Physical Observables and Measurements . . . 12
1.3.3 Postulate III: Time-evolution of a system . . . 13
1.4 A Brief History of Computing: From Classical to Quantum . . . 13
1.4.1 Classical Computing Foundations . . . 13
1.4.2 The Emergence of Quantum Information . . . 14
1.4.3 Analog vs. Digital Quantum Simulation . . . 14
1.5 Quantum Technologies . . . 15

2 The Qubit **17**
2.1 The Qubit Hilbert space . . . 17
2.1.1 Qubit states . . . 17
2.1.2 Qubit operators . . . 19
2.1.3 The Pauli operators . . . 20
2.2 The Bloch sphere . . . 22
2.2.1 General Unitary Rotations . . . 24
2.2.2 The Phase Gate . . . 25
2.3 Quantum Circuits . . . 26
2.4 Noise and decoherence . . . 27
2.4.1 The Density Matrix . . . 28
2.4.2 Modeling Noise . . . 30
2.4.3 Physical Qubit Implementations . . . 34

3 Multiple Qubits **36**
3.1 Tensor Products and the multi-qubit Hilbert space . . . 36
3.2 Entanglement . . . 40
3.2.1 Bell States and Non-local Correlations . . . 41
3.2.2 Mathematical definition and separability . . . 42
3.2.3 Reduced density matrices . . . 43
3.2.4 Quantifying entanglement . . . 45

3.3 Multi-qubit operations . . . 48
3.3.1 Single-qubit gates . . . 48
3.3.2 Two-qubit gates . . . 49
3.3.3 Measurement . . . 54
3.4 Decoherence in multi-qubit systems . . . 59
3.4.1 Gate Errors . . . 60

4 Quantum Computing Algorithms 61
4.1 Deutsch's Algorithm . . . 61
4.1.1 The Problem Statement . . . 61
4.1.2 Quantum Implementation . . . 62
4.1.3 Key Insights . . . 65
4.1.4 Mathematical Details . . . 66
4.2 Deutsch–Jozsa Algorithm . . . 67
4.2.1 The Problem Statement . . . 67
4.2.2 Quantum Implementation . . . 67
4.2.3 Mathematical Analysis . . . 67
4.2.4 Key Insights . . . 71
4.3 Bernstein–Vazirani Algorithm . . . 71
4.3.1 Problem Statement . . . 72
4.3.2 Quantum Implementation . . . 72
4.3.3 Step-by-step state evolution with Hadamard gates . . . 72
4.3.4 Connection to Deutsch-Jozsa Algorithm . . . 75
4.4 Simon's Algorithm . . . 75
4.4.1 Problem Statement . . . 75
4.4.2 Quantum Implementation . . . 76
4.4.3 Quantum Speedup . . . 78
4.5 Quantum Phase Estimation . . . 78
4.5.1 Problem Statement . . . 79
4.5.2 Quantum Implementation . . . 79
4.5.3 Accuracy . . . 82
4.5.4 Example . . . 82
4.5.5 Applications . . . 82
4.6 Grover's Algorithm . . . 82
4.6.1 The Unstructured Search Problem . . . 83
4.6.2 Quantum Oracle . . . 83
4.6.3 The Diffusion Operator . . . 83
4.6.4 Grover Iteration as a Rotation . . . 84
4.6.5 Measurement . . . 85
4.6.6 Quantum Speedup . . . 85
4.7 Shor's Algorithm (High-Level Overview) . . . 85
4.7.1 Quantum Period Finding in Shor's Algorithm . . . 86
4.8 HHL Algorithm . . . 87
4.8.1 The Linear Systems Problem . . . 87
4.8.2 Quantum Solution Approach . . . 88
4.8.3 Mathematical Details . . . 88
4.8.4 Assumptions and Limitations . . . 89
4.8.5 Applications . . . 90
4.8.6 Conclusion . . . 90

4.9 Hybrid Quantum-Classical Algorithms 90
4.9.1 Variational Quantum Eigensolver (VQE) 90
4.9.2 Quantum Approximate Optimization Algorithm (QAOA) 91
4.10 Overview and comparison with classical 91

Interlude: Hardwire requirements **93**
DiVincenzo's Criteria for Quantum Computation 93
Measurement-Based Quantum Computation 94
Magic State Distillation 94
The Need for Magic States 94
Resource Overhead and Error Correction 94

References **96**

Preface

The goal for this text is to be a snapshot in time of quantum technologies: the good, the bad, and the ugly. As of 2024, there has been a large amount of industry interest and progress to develop quantum technologies, and a student approaching this industry should have the basic knowledge to understand *what* is trying to be achieved and *how* they are trying to achieve it. To cut through the PR of industry, this text offers the author's personal perspective on what has been achieved, where things need to go, and the challenges to get there. This is not meant to be an authoritative guide on any one of the technologies presented here, but a jumping off point for the interested student.

Finally, this is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

1 Introduction

In 2019 [1], Google reached a significant milestone in quantum computing when they achieved "Quantum Advantage"–demonstrating for the first time that a quantum computer could surpass the capabilities of classical computers, albeit for a specific, specialized task. Using their Sycamore processor with 53 qubits, they showed that sampling from a quantum circuit a million times took only 200 seconds, while the equivalent task would take approximately 10,000 years on a classical supercomputer. This breakthrough marked a turning point in quantum computing, igniting increased interest and investment in quantum technologies. Building on this success, in December 2024, Google announced their new Willow chip [2], representing their latest advancement in quantum hardware. Today, numerous companies and research institutions worldwide are racing to develop quantum hardware, each pursuing different technological approaches to challenge the computational limits of classical computers.

> **What is Quantum Advantage?**
>
> Quantum advantage (or quantum supremacy) refers to the demonstration that a quantum computer can solve a specific problem significantly faster than the best known classical algorithm. However, the problem doesn't necessarily need to be useful–it just needs to be well-defined and verifiable.

This text aims to give undergraduate students a comprehensive introduction to quantum technology and computation. We will begin by exploring the fundamental mathematical framework that underlies quantum computing, building from basic principles to more advanced concepts. Crucially, we will see what specific things a quantum computer can achieve that a classical computer would struggle with. With this foundation, we will examine three of the most promising current quantum computing technologies: superconducting qubits, photonic quantum computing, and ion traps. Each of these approaches offers unique advantages and faces distinct challenges, which we will analyze in detail.

To bridge theory with practice, we will utilize IBM's Qiskit software platform to implement basic quantum computations, providing hands-on experience with quantum programming. As we progress, we will explore critical practical considerations in quantum computing, including error mitigation and correction strategies. Time permitting, we will venture into the cutting-edge field of topological quantum computing, which offers a potentially more robust approach to quantum computation.

Throughout this text, we will maintain a balanced perspective, examining both the tremendous potential and significant challenges facing quantum computing technology. Our goal is to equip students with both theoretical understanding and practical insights into this rapidly evolving field.

1.1 The Quantum-Classical Arms Race

The story of Google's quantum supremacy claim illustrates a fascinating dynamic in the field of quantum computing–an ongoing arms race between quantum and classical algorithms. When Google first announced their achievement with the Sycamore processor [1], they estimated that their quantum

sampling task would take a classical supercomputer approximately 10,000 years. However, within months, IBM researchers developed improved classical algorithms that could potentially perform the same calculation in just 2.5 days [3]. Further work even demonstrated that using tensor networks, the problem could be solved *faster* on a modern superconductor with ExaFLOPS performance [4].

This back-and-forth highlights several important lessons. First, it demonstrates the remarkable adaptability of classical computing. As quantum computers advance, classical algorithm developers find increasingly clever ways to simulate quantum systems or solve specific problems more efficiently. This competition drives innovation in both fields–quantum hardware must continually improve to maintain its advantage, while classical algorithms become more sophisticated in response.

Second, it serves as a cautionary tale about interpreting quantum computing announcements, particularly those aimed at the general public. While the achievement of quantum advantage represents a genuine milestone, the initial 10,000-year estimate proved overly optimistic. This pattern has repeated with various quantum computing companies, where marketing claims sometimes outpace peer-reviewed scientific validation. For students and researchers in the field, it's crucial to maintain a balanced perspective–acknowledging genuine breakthroughs while critically evaluating bold claims.

The recent announcement of Google's Willow chip [2] represents another step forward, but should be viewed within this context of ongoing competition and careful validation. This healthy tension between quantum and classical approaches ultimately benefits both fields, pushing the boundaries of what's computationally possible while maintaining rigorous scientific standards.

1.2 Early Computing: The Lesson of Transistors

In Ref. [5] there are a few quotes about early computing

> *"Computers in the future may weigh no more than 1.5 tons."* –Popular Mechanics, forecasting the relentless march of science, 1949

> *"I think there is a world market for maybe five computers."* –Thomas Watson, chairman of IBM, 1943

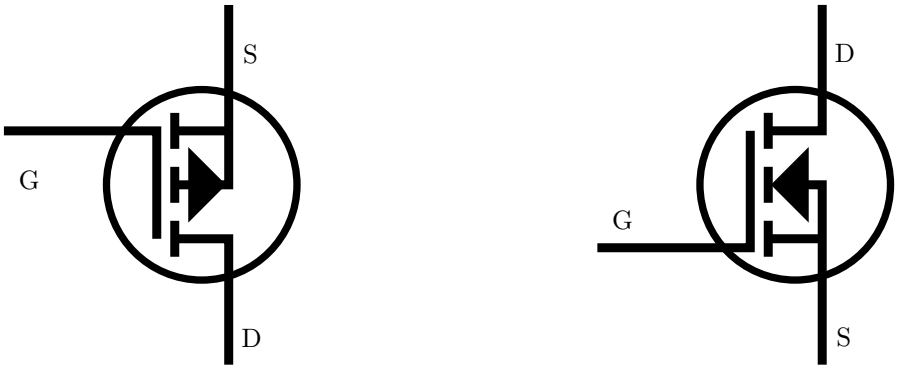
These quotes raise important points about early technology. While the theory of modern computing really took off with Turing in 1937 [6], the technological advancement necessary for modern computurs would not occur until later: when the transistor came about.

1.2.1 The Dream: Field Effect Transistors

To enable classical computing, we need something like a "switch" that can be on and off, keeping track of whether or not something like, current is flowing. This is hard to do with traditional circuit elements like resistors, capacitors, and inductors. It requires something more nonlinear: A switch that only allows current flow when a voltage is applied, a **transistor**.

Fig. 1.1 shows two of the types of circuit diagrams for a bipolar junction transistor (G = gate, S = source, and D = drain)[1]. A voltage applied at the gate enables a larger current to flow between collector and emitter. There are also *bipolar junction transistors* that use a smaller current to enable a larger current, in this case there is "base", "collector", and "emitter".

[1]This is also known as the CX-gate (the controlled-X gate).



(a) P-channel (b) N-channel

Fig. 1.1: Example Circuit diagrams for MOSFETs (*enh*)

With these building blocks, logical gates can be created, but *how* to build these? Which device can be made small and in abundance? And what challenges were encountered on the way.

> **Scale of Modern Computing in your pocket**
>
> The iPhone A17 Pro chip's 19 billion transistors would cover an area of about 1 square centimeter. If each transistor were the size of a grain of rice, they would cover an area larger than 75 football fields! This incredible miniaturization is what enables modern computing.

It was recognized early on that semiconductors provided an ideal platform for these kinds of circuits, and much work was done to try and create the above "field effect transistors." Schematically, these take the form:

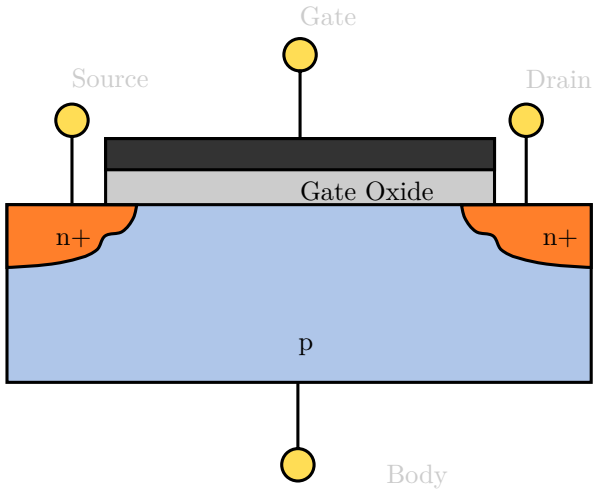
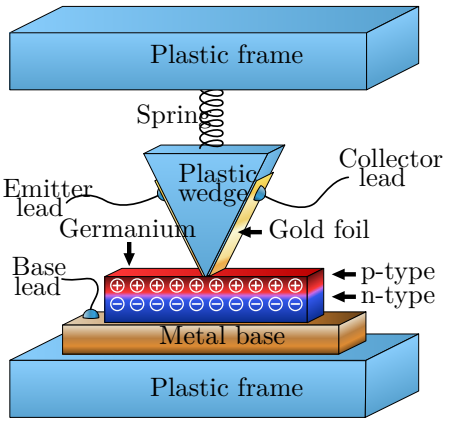


Fig. 1.2: Schematic of Field Effect Transistor[2]

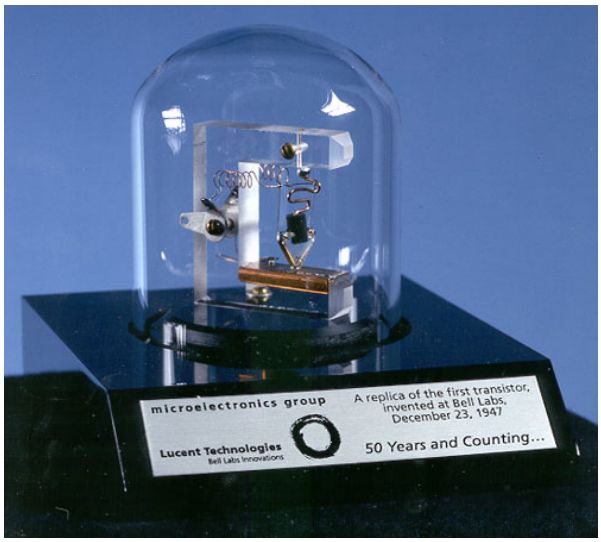
[2]This is true for pure states, but can be easily generalized to mixed states, see [7].

In Fig. 1.2, electrons flow from source to drain, but only when the "gate" has an applied voltage to it (effectively "lowering the barrier" for electrons to get through). This theory was sound and based on the recently developed quantum electron theory of metals developed by Wolfgang Pauli, Werner Heisenberg, Arnold Sommerfeld, Felix Bloch, and Rudolf Peierls [8]. And indeed, the people at Bell labs worked on this problem theoretically and experimentally for the beginning in the 1930s (for a full history, see [8]). Despite the strong foundations though, creating a field effect transistor turned out to be difficult, and in the process Brattain and Bardeen instead created the point-contact transistor.

1.2.2 The Point Contact Transistor



(a) Schematic of the point contact transistor[a].



(b) Replica of first transistor

[a]Licensed under Creative Commons CC0 1.0 Universal Public Domain Dedication.

Fig. 1.3: The point contact transistor.

The point-contact transistor, invented in 1947, worked quite differently from the field-effect design. Instead of using a voltage at a gate to control current flow, it used two very closely spaced metal contacts pressed against a semiconductor (typically germanium). One contact, called the emitter, would inject positive charge carriers (holes) into the semiconductor. The second contact, called the collector, would collect these carriers - but crucially, the amount of current that could flow through the collector could be controlled by small changes in the emitter current[3]. A schematic and image of a replica of the original device are illustrated in Fig. 1.3.

This amplification effect, where a small current controls a larger one, was revolutionary–though the exact physics behind it wasn't fully understood at the time. The key was that the metal contacts created special regions in the semiconductor where the positive carriers modified the barrier for current flow from the bulk material. While the detailed quantum mechanics is complex, you can think of it like creating "paths" that electrons prefer to take through the material, with the emitter current controlling how easily electrons can flow along these paths to the collector.

[3]For details on how this was made, see How the first transistor worked

While point-contact transistors were eventually superseded by more reliable and easier-to-manufacture designs, they represented a crucial breakthrough in electronics. They proved that solid-state devices could indeed amplify electrical signals. However, they were quite large. The original design for a field effect transistor would be needed, and they key resided in understanding and control the *surface physics* of semiconductors.

1.2.3 Surface physics and transistors

The key challenge in creating field effect transistors lay in understanding and controlling the surface properties of semiconductors. To understand why this was so difficult, let's break it down:

When a semiconductor crystal (like silicon) ends at a surface, something interesting happens. The regular pattern of atoms is suddenly interrupted - imagine a neat stack of blocks suddenly ending in mid-air. This interruption creates what we call "surface states" - special energy levels that electrons can occupy right at the surface of the material.

These surface states turned out to be extremely problematic for making transistors. Remember that in a field effect transistor, we want to control the flow of electrons using an electric field from the gate (see Fig. 1.2). However, these surface states acted like tiny electron traps, capturing and holding onto electrons. When electrons got stuck in these states, they effectively "screened" or blocked the electric field from the gate, preventing it from controlling the current flow through the semiconductor.

This screening effect was so strong that early attempts at field effect transistors simply didn't work–no matter how strong a voltage was applied to the gate, it couldn't effectively control the current flow. It was like trying to control a water flow with a valve, but having something constantly blocking the valve from moving.

The breakthrough came in the 1950s when researchers, particularly at Bell Labs, realized they needed to chemically "passivate" the semiconductor surface–essentially finding ways to neutralize these problematic surface states. The key discovery was that growing a thin layer of silicon dioxide (SiO) on silicon created a much more stable interface with far fewer problematic surface states. This oxide layer also served as an excellent insulator between the gate and the semiconductor.

This seemingly simple solution–growing an oxide layer–was actually a remarkable achievement that required precise control of material chemistry and manufacturing processes. It finally allowed the creation of practical field effect transistors, leading to the modern MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) that forms the backbone of today's electronics.

The success of this approach also highlights an important lesson in technology development: sometimes the biggest breakthroughs come not from changing the fundamental design, but from finding ways to control and manage the subtle physical effects that prevent a good design from working in practice.

1.2.4 Where are we with quantum computing?

Imagine that this course existed back in the 1950s and we called it "Computing Technology." Transistors were still coming online to enable computation at scale and we had both the information science and material theory to achieve it:

1. We knew what was needed to do universal classical computation.

2. We had the quantum theory of metals to describe how to build components (transistors) to achieve classical computation.

However, the engineering challenges took decades to resolve. It was only when we resolved those that computation as we currently envision it took off and we could have classical computers.

For quantum computing, we are in a similar situation:

1. We know what is needed to do universal quantum computation.
2. We have the quantum theory of photons, atoms, and superconductivity to achieve quantum computation.

However, as we will see in what follows, we have significant engineering challenges to achieve these in practice. While we will be largely concerned with the physics that make #2 possible in this course (and we'll touch on #1 for the first part of the course), we will pay attention to the strengths and weaknesses in the physics that lead to more pressing engineering challenges.

i Historical Parallel

Just as the theory of classical computation [6] preceded practical computers by decades, we now have the theory of quantum computation [5] but face significant engineering challenges. The key difference is that we're trying to control individual quantum systems rather than classical electrical currents.

1.3 Review of the Postulates of Quantum Mechanics

Quantum mechanics is built out of some basic postulates which are crucial to understand for quantum computation. When we talk about a "system" in these postulates, we will be thinking of two-level systems which we can label 0 or 1 (for our qubit). However, we will state them generally since many applications require some basic work to reduce the complicated system down to just the qubits we are interested in.

! Mathematical Notation Guide

Throughout this text, we'll use:

- $|\psi\rangle$ ("ket psi"): Represents a quantum state
- $\langle\phi|$ ("bra phi"): The dual vector to $|\phi\rangle$
- $\langle\phi|\psi\rangle$: The inner product between states
- $\otimes$: The tensor product operation

These notations provide a compact way to describe quantum systems.

1.3.1 Postulate I: The Hilbert space

A state in an isolated physical system S can be described by a set of *normalized state vectors*–$|\psi\rangle$ and all vectors related by a phase $|\psi'\rangle = e^{i\phi}|\psi\rangle$–belonging in the Hilbert space

$\mathcal{H}_S$.

importantly, a Hilbert space is equipped with an inner product (much like the dot product in three-dimensions) $\langle\alpha|\beta\rangle$.

We also need rules for attaching Hilbert spaces to one another. Afterall, we will need to use more than one qubit to do anything interesting.

> The Hilbert space of a composite system is the *tensor product* of the two individual systems $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$.

1.3.2 Postulate II: Physical Observables and Measurements

This postulate is also sometimes called the **Born rule**.

In order to measure the system (e.g., "where is the particle?" or "Is the qubit a 0 or 1?"), we need to know what *physical observables* are

> (i) Every physical observable a can be described as a Hermitian operator A acting in the Hilbert Space.

Formally, a Hermitian operator has $A = A^\dagger$ where $A^\dagger$ is the *conjugate transpose* of A. These operators have a whole set of orthonormal eigenstates $A|a_n\rangle = a_n|a_n\rangle$ for a *real* number a_n (orthonormal means $\langle a_n|a_m\rangle = \delta_{nm}$). These mathematical details are important for how we will perform measurements

> (ii) When a physical observable with operator A is measured on a normalized eigenstate $|\psi\rangle$, the result is an eigenvalue a_n of that operator with probability $p_n = |\langle a_n|\psi\rangle|^2$ (or in the case of a degeneracy d, $p_n = \sum_{i=1}^{d}|\langle a_n, i|\psi\rangle|^2$).

If we measure A repeatedly, we are naturally lead to the expectation value $\langle\psi|A|\psi\rangle = a = \sum_n a_n p_n$, the average result of repeated quantum mechanical measurements. Once a measurement is performed, however, the state is changed, for this we need an operator P_n which projects onto the eigenspace of A associated with the eigenvalue a_n.

> (iii) When a measurement of the observable A gives a results a_n, the state is changed to be the *normalized projection* of $|\psi\rangle$ to the eigenspace associated with a_n

$$|\psi\rangle \quad \Longrightarrow \quad \frac{P_n|\psi\rangle}{\sqrt{\langle\psi|P_n|\psi\rangle}}$$

If the eigenstates are not degenerate, then $|\psi\rangle \implies |a_n\rangle$. However, we will find that we will often have degenerate states, and in that case

$$|\psi\rangle \implies \frac{\sum_{i=1}^{d}\langle a_n, i|\psi\rangle\,|a_n, i\rangle}{\sqrt{\sum_{i=1}^{d}|\langle a_n, i|\psi\rangle|^2}}.$$

One comfortable with the braket notation might notice that within this, $P_n = \sum_{i=1}^{d}|a_n, i\rangle\langle a_n, i|$.

1.3.3 Postulate III: Time-evolution of a system

While we will talk about Hamiltonians, often in quantum computation we have gates that do not require these. In this case, we state this postulate as abstractly as possible

> The time evolution of a closed system from some time t_0 and state $|\psi_0\rangle$ to a final time t and state $|\psi\rangle$ can be described as a unitary transformation
>
> $$|\psi\rangle = U(t, t_0)\,|\psi_0\rangle\,.$$

This postulate is necessary for us to maintain probabilities. Unitary operators have the property that $UU^\dagger = U^\dagger U = \mathbb{1}$, and so

$$1 = \langle\psi_0|\psi_0\rangle = \langle\psi_0|U^\dagger U|\psi_0\rangle = \langle\psi|\psi\rangle\,.$$

In the case of time-independent *Hamiltonian dynamics,* the operator takes the form $U = e^{-iHt/\hbar}$ for a hermitian energy operator H called the *Hamiltonian.*

1.4 A Brief History of Computing: From Classical to Quantum

The concept that information is physical underlies both classical and quantum computation. Even in the earliest systems of record-keeping, we see physical objects encoding information:

- Kish Tablet (3500 BCE): A limestone tablet from Kish showing a record of pictographic writing.
- Quipu (2600–1900 BCE): A system of knotted ropes used by the Inca civilization for keeping records. The color, order, and number of knots all represented quantifiable or categorical data.

These examples highlight that from the very beginning, the act of storing and manipulating information has always had a physical basis—though the underlying physics often remained implicit for centuries.

1.4.1 Classical Computing Foundations

The paradigm of classical computing rests on a few fundamental ideas:

- **Boolean Logic & Universal Gates**: All classical computers can be built from a finite set of universal logical gates (e.g., {NAND}, {NOR}, or {AND, NOT}).

Most Boolean operations (AND, OR, NAND, etc.) are inherently irreversible: once a bit is erased or overwritten, the original state cannot be recovered from the output alone.

- **Extended (Physical) Church–Turing Thesis**: A probabilistic Turing machine can efficiently simulate any realistic physical model of computation with at most polynomial overhead.

This thesis, while unproven, underpins the expectation that classical computers (or at least classical models) are sufficient for simulating any physical system in principle. Quantum computation potentially challenges this with a polynomial in time algorithm (Shor's algorithm) that is exponential in time classically.

1.4.2 The Emergence of Quantum Information

While classical computing relies on bits that are strictly 0 or 1, quantum computing introduces powerful new concepts:

- **Superposition**: A quantum bit (qubit) can be in a linear combination of basis states (e.g., simultaneously "0" and "1" with certain complex amplitudes).
- **Entanglement**: Two or more qubits can become correlated in such a way that measuring one affects the outcomes for the others, even across vast distances.

The key question that launched the field of quantum information was whether these uniquely quantum properties—superposition and entanglement—could be exploited to perform computations more efficiently than any classical device.

In his seminal work, Simulating Physics with Computers [9], Richard Feynman observed that simulating quantum many-body systems on classical computers seems to require exponential resources. He posed the idea of harnessing genuine quantum systems themselves for simulation, planting the seeds for quantum computation as a research field.

Then, in a series of groundbreaking results between the 1980s and 1990s, researchers demonstrated that quantum computers could, in principle, outperform classical computers for certain tasks:

1. Deutsch (1985) [10]: Showed that it is possible to carry out a simple computational task on a quantum computer faster than any classical algorithm.
2. Deutsch–Jozsa (1992) [11]: Introduced a deterministic quantum algorithm that is exponentially faster (in the worst case) than any deterministic classical algorithm.
3. Bernstein–Vazirani (1992) [12]: Demonstrated a probabilistic quantum algorithm faster than any probabilistic classical algorithm.
4. Simon (1994) [13]: Provided a probabilistic quantum algorithm that is exponentially faster than any probabilistic classical algorithm for a specific promise problem.
5. Shor (1994) [14]: Showed how to factor integers efficiently, providing an exponential speedup over the best known classical methods. This result was particularly striking for cryptography, as factoring large numbers underpins many encryption schemes.

Alongside these algorithmic milestones, researchers began to delineate the limitations: not every problem can be exponentially sped up by quantum methods. For instance, Grover's algorithm (1996) [15] for unstructured search yields a quadratic speedup (from N to $\sqrt{N}$)—still better than classical, but not the exponential leap that Shor's algorithm provides for factoring.

1.4.3 Analog vs. Digital Quantum Simulation

As the field grew, quantum simulation branched into two approaches:

- Analog Quantum Simulation: Uses a controllable quantum system to mimic a target quantum system. The interactions in the simulator closely resemble the interactions in the system of interest.
- Digital Quantum Simulation: Decomposes a quantum evolution into a sequence of discrete gates (a "universal" set of quantum gates), akin to how classical digital computers function using logical gate operations.

Both approaches aim to exploit quantum mechanics to tackle problems in mathematics, physics, chemistry, and materials science that remain intractable for classical supercomputers.

Amid ongoing research, the interplay between classical and quantum paradigms remains a vibrant area of exploration. While classical computing infrastructure continues to be indispensable, quantum computing offers the promise of qualitatively new capabilities—provided we can tame the noise, errors, and fragilities inherent to quantum states.

1.5 Quantum Technologies

One major difference between the hindsight-history we have for classical computation and the current state of quantum computers is that are many platforms vying to enable quantum computation. There are arguments for and against each platform, and even arguments for using a combination of platforms. Here we give a list of some of the technologies and highlight the ones we will be surveying in this course.

Platforms covered in this course

- **Superconducting qubits**: Artificial atoms made from superconducting circuits that operate at ultra-low temperatures. Currently the most mature platform, used by companies like IBM and Google.
- **Trapped ions**: Individual atoms held in place by electromagnetic fields. Known for having very long coherence times and high-fidelity gates. Major players include IonQ and Quantinuum (formerly of Honeywell).
- **Photonic quantum computers**: Use particles of light (photons) as qubits. Can operate at room temperature and naturally interface with quantum communication systems. Being developed by companies like PsiQuantum and Xanadu.

Other platforms we will not cover

- **Silicon quantum dots**: Quantum bits made from individual electrons trapped in silicon, similar to classical semiconductor technology. Could potentially leverage existing manufacturing processes.
- **Neutral atoms and Rydberg arrays**: Individual neutral atoms arranged in arrays using laser beams. When excited to high-energy Rydberg states, atoms can interact strongly with their neighbors. Can create large numbers of identical qubits with programmable interactions. Companies like QuEra are pursuing this approach.
- **NV centers**: Quantum bits made from nitrogen-vacancy defects in diamond. Can operate at room temperature and have long coherence times, making them particularly promising for quantum sensing and networking applications.

A platform we will cover if time permits

- **Topological qubits**: A theoretical approach that would use special quantum states of matter to create error-protected qubits. Still in early research stages but could offer significant advantages if realized.

Current State of Quantum Computing

As of 2024, the largest quantum computers have around 50-100 physical qubits optimistically, but these are noisy and require error correction. For comparison, your smartphone has billions of classical bits. This highlights the early stage of quantum computing development and the engineering challenges ahead.

Each platform has its own advantages and challenges in terms of scalability, error rates, coherence times, and manufacturing complexity. The field is still evolving, and it's possible that different platforms may be optimal for different applications.

2 The Qubit

The fundamental building block of the classical computer was the bit: A 0 or 1 that could be manipulated by a classical computer (via transistors, see Section 1.2). In a similar manner, quantum computation has the "quantum bit" or just *qubit*, for short. This leads us to the linear algebra of 2×2 matrices, as we will see. Despite the apparent simplicity, we can already see many of the key features of quantum mechanics in this simple system.

⚠ Common Qubit Misconceptions

- A qubit is not just a probabilistic classical bit.
- You cannot directly access of the amplitudes of $|0\rangle$ and $|1\rangle$ through a single measurement.
- Superposition states collapse upon measurement.
- No-cloning theorem [16] means you cannot perfectly copy an unknown quantum state.
- Entanglement is not the same as classical correlation (see next chapter).

2.1 The Qubit Hilbert space

A qubit will be in two-dimensional complex vector space equipped with an inner product.

2.1.1 Qubit states

For the qubit, we associate two states with two different basis vectors: $|0\rangle$ and $|1\rangle$. This will be called the *computational basis*. The magic[1] of quantum mechanics is that a state need not be just one or the other, but could be *any* linear superposition of these

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{2.1}$$

In this, we have adopted the *bra-ket* notation due to Dirac. While it can be quite useful, we can write this in terms of matrices and vectors

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

This makes it clear that these states are *orthogonal* $\langle 0|1\rangle = 0$.

In this case we have

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}.$$

[1] Magic is a technical term in quantum computing, though we're using it in a colloquial sense here, see [17].

We also need the conjugate transpose, the Hermitian conjugate, of this vector, which will be a row-vector

$$\langle\psi| = \begin{bmatrix}\alpha^* & \beta^*\end{bmatrix}.$$

The key feature of quantum mechanics is that these states must be *normalized*, meaning that the probability of finding the system in any state must sum to 1. This means that

$$\langle\psi|\psi\rangle = \begin{bmatrix}\alpha^* & \beta^*\end{bmatrix}\begin{bmatrix}\alpha \\ \beta\end{bmatrix} = |\alpha|^2 + |\beta|^2 = 1.$$

In matrix notation, this is just the dot product of a vector with its complex conjugate.

Why Normalization Matters

The normalization condition $|\alpha|^2 + |\beta|^2 = 1$ isn't just mathematical convenience - it ensures probabilities add up to 100%! For example:

- $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ gives 50-50 chance of measuring 0 or 1
- $|\psi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$ gives 75% chance of 0 and 25% chance of 1

We can also write operators that act on these states. The simplest operator is the Pauli Z operator, which in matrix form is

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

When this operator acts on our basis states, we find

$$Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle.$$

This means that $|0\rangle$ and $|1\rangle$ are *eigenstates* of Z with eigenvalues $+1$ and -1 respectively. For a general state $|\psi\rangle$, measuring Z will yield either $+1$ or -1, with probabilities determined by $|\alpha|^2$ and $|\beta|^2$ respectively.

Example: Measuring a superposition state

Consider the state $|\psi\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$. When we measure this state in the Z basis:
Notice that this state is normalized since $(\frac{3}{5})^2 + (\frac{4}{5})^2 = 1$. If we measure this state in the computational basis:

- We'll get outcome $|0\rangle$ with probability $|\frac{3}{5}|^2 = 0.36$ (36%)
- We'll get outcome $|1\rangle$ with probability $|\frac{4}{5}|^2 = 0.64$ (64%)

After measurement, the state will collapse to either $|0\rangle$ or $|1\rangle$ with the above probabilities

Measurement Collapse in Practice

When we say a quantum state "collapses" upon measurement, what actually happens in the lab?

- For a superconducting qubit: We measure a voltage or current
- For an ion trap: We detect scattered photons

- For a photonic qubit: We count photons with a detector

Each technology has its own way of converting quantum information into classical signals!

2.1.2 Qubit operators

Since this is linear algebra, we can write a general operator $\mathcal{O}$ as a matrix

$$\mathcal{O} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

Often, we are interested in the eigenvalues and eigenstates of these operators $\mathcal{O}|\psi_i\rangle = \lambda_i |\psi_i\rangle$. Generically, we can find these by solving a polynomial equation

$$\det(\mathcal{O} - \lambda I) = 0.$$

Solving this step-by-step

$$\det(\mathcal{O} - \lambda I) = \begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0,$$

which gives us

$$(a-\lambda)(d-\lambda) - bc = 0.$$

This is a quadratic equation that we can solve:

$$\lambda^2 - (a+d)\lambda + (ad - bc) = 0.$$

The eigenvalues are therefore

$$\lambda_{\pm} = \frac{a + d \pm \sqrt{(a-d)^2 + 4bc}}{2}. \tag{2.2}$$

For quantum mechanical **observables** (see Section 1.3.2), we are particularly interested in *Hermitian* operators where $\mathcal{O} = \mathcal{O}^\dagger$,

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

This means that a and d must be real and $c = b^*$. In this case, the eigenvalues are always real, as we can see from the Eq. 2.2.

A particularly important class of operators are **unitary operators**, where $U^\dagger U = UU^\dagger = I$. These are what we use for time-evolution, see Section 1.3.3.

These operators preserve the inner product between states:

$$\langle U\psi | U\phi \rangle = \langle \psi | \phi \rangle$$

For a 2×2 matrix

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

the unitarity condition means that

$$\begin{bmatrix} a^* & c^* \\ b^* & d^* \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This gives us several conditions:

1. $|a|^2 + |c|^2 = 1$ (normalization of first column)
2. $|b|^2 + |d|^2 = 1$ (normalization of second column)
3. $ab^* + cd^* = 0$ (orthogonality of columns)

This immediately gives us some insight into these operators. If we define,

$$|\psi_1\rangle = \begin{bmatrix} a \\ c \end{bmatrix}, \quad |\psi_2\rangle = \begin{bmatrix} b \\ d \end{bmatrix},$$

then we have $\langle\psi_1|\psi_1\rangle = 1 = \langle\psi_2|\psi_2\rangle$ and $\langle\psi_1|\psi_2\rangle = 0$.

An important property of unitary operators is that their eigenvalues always have magnitude 1, meaning they can be written as $e^{i\theta}$ for some real θ. This makes them natural operators for describing quantum evolution.

> **Why Unitary?**
>
> Unitary operators are special because they:
>
> 1. Preserve the normalization of quantum states
> 2. Are reversible (have an inverse)
> 3. Represent physical operations that conserve probability
>
> This is why quantum gates must be unitary - they represent real physical processes that can be undone!

2.1.3 The Pauli operators

A particularly important set of operators are the Pauli operators. We've already seen the Pauli Z operator. The other two are[2]

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

These operators satisfy some important algebraic relations:

$$X^2 = Y^2 = Z^2 = I, \quad XY = iZ, \quad YZ = iX, \quad ZX = iY.$$

We can additionally start to see some logical operations begin to appear; X operates on the computational basis as a **NOT** gate

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle.$$

[2] In some literature, these are matrices are denoted by σ_x, σ_y, and σ_z and related to spin operators via $S_i = \frac{1}{2}\sigma_i$. This insight can help bridge the idea of these operators and the Bloch sphere.

Pauli Operators in Action

The Pauli operators represent quantum operations:

- X is like the classical NOT gate: flips between $|0\rangle$ and $|1\rangle$
- Z adds a phase: leaves $|0\rangle$ alone but negates $|1\rangle$
- $Y = iXZ$ combines both operations.

These simple operations are building blocks for more complex quantum algorithms!

Example: Applying operators

Let's apply the X (NOT) gate to our state $|\psi\rangle = (\frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle)$:

$$\begin{aligned} X|\psi\rangle &= X(\tfrac{3}{5}|0\rangle + \tfrac{4}{5}|1\rangle) \\ &= \tfrac{3}{5}X|0\rangle + \tfrac{4}{5}X|1\rangle \\ &= \tfrac{3}{5}|1\rangle + \tfrac{4}{5}|0\rangle \\ &= \tfrac{4}{5}|0\rangle + \tfrac{3}{5}|1\rangle \end{aligned}$$

The full set of Pauli operators, along with the identity, form a complete basis for 2×2 matrices, meaning we can write any operator as

$$\mathcal{O} = aI + bX + cY + dZ,$$

where a, b, c, and d are complex numbers. We can extract each of these numbers, mathematically, with a trace operation

$$\operatorname{tr}\mathcal{O} = 2a, \quad \operatorname{tr}\mathcal{O}X = 2b, \quad \operatorname{tr}\mathcal{O}Y = 2c, \quad \operatorname{tr}\mathcal{O}Z = 2d.$$

Note that separately, X, Y, and Z are Hermitian (and thus, observables). If $\mathcal{O}$ is an observable, then $\mathcal{O} = \mathcal{O}^\dagger$ immediately leads us to a, b, c, and d being all real.

We can put constraints on these coefficients for unitary operators as well, and we leave this as an exercise for the reader.

Finally, these operators have eigenstates as well, and we can define them as $X|\pm\rangle = \pm|\pm\rangle$ and $Y|\pm i\rangle = \pm|\pm i\rangle$, and they have the forms

$$\begin{aligned} |\pm\rangle &= \tfrac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle), \\ |\pm i\rangle &= \tfrac{1}{\sqrt{2}}(|0\rangle \pm i|1\rangle). \end{aligned}$$

We will often want to change our basis from $|0\rangle$ and $|1\rangle$ to $|+\rangle$ and $|-\rangle$. This is accomplished with something called the *Hadamard gate* (we'll call it H, not to be confused with a Hamiltonian) and it is created specifically to change from computational basis to the X basis: $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$. As a matrix it takes the form

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

! The Power of Hadamard

The Hadamard gate is one of the most important gates in quantum computing with useful properties:

1. It creates equal superpositions from computational basis states.
2. It's its own inverse ($H^2 = I$)
3. It's used in nearly every quantum algorithm
4. When applied to n qubits, it creates a superposition of all 2^n possible bit strings!

Example 3: The Hadamard Transform

The Hadamard gate is particularly important because it creates superposition states. Let's see what happens when we apply it to $|0\rangle$:

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

2.2 The Bloch sphere

The qubit itself is more than just a probability of being a 1 or a 0. A crucial bit of *quantum* information is the relative phase between the two states for instance

$$|\psi\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + e^{i\phi}|1\rangle),$$

and since all states are equivalent up to a *total* phase, we can write the amplitude of each state with a real number. In this case, if we set $|\psi\rangle = x|0\rangle + ye^{i\phi}|1\rangle$, then we have $\langle\psi|\psi\rangle = x^2 + y^2 = 1$ for normalization. This is the equation for a circle, and writing out $x = \cos(\theta/2)$ and $y = \sin(\theta/2)$[3], we are lead to an angular representation of our state.

$$|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\phi}|1\rangle\,. \tag{2.3}$$

Let's see how this state relates to our Pauli operators. If we calculate the expectation values of each operator:

[3]The divide-by-two for the angles will become clear as we go through this section.

$$
\begin{aligned}
\langle X \rangle &= \langle \psi | \, X \, | \psi \rangle \\
&= (\cos(\theta/2) \langle 0| + \sin(\theta/2) e^{-i\phi} \langle 1|) X (\cos(\theta/2) |0\rangle + \sin(\theta/2) e^{i\phi} |1\rangle) \\
&= (\cos(\theta/2) \langle 0| + \sin(\theta/2) e^{-i\phi} \langle 1|) (\cos(\theta/2) |1\rangle + \sin(\theta/2) e^{i\phi} |0\rangle) \\
&= \sin(\theta/2)\cos(\theta/2) e^{i\phi} + \cos(\theta/2)\sin(\theta/2) e^{-i\phi} \\
&= \sin(\theta/2)\cos(\theta/2)(e^{i\phi} + e^{-i\phi}) \\
&= 2\sin(\theta/2)\cos(\theta/2)\cos\phi \\
&= \sin\theta\cos\phi.
\end{aligned}
$$

We can carry out a similar calculation for Y and Z to obtain

$$
\begin{aligned}
\langle X \rangle &= \langle \psi | \, X \, | \psi \rangle = \sin\theta\cos\phi \\
\langle Y \rangle &= \langle \psi | \, Y | \psi \rangle = \sin\theta\sin\phi \\
\langle Z \rangle &= \langle \psi | \, Z \, | \psi \rangle = \cos\theta
\end{aligned}
$$

These expectation values give us coordinates , which are precisely the coordinates of a point on a unit sphere! This is why we call it the Bloch sphere. The angles θ and ϕ are the usual spherical coordinates.

The Bloch Sphere Geometry

- The north pole ($\theta = 0$) corresponds to $|0\rangle$
- The south pole ($\theta = \pi$) corresponds to $|1\rangle$
- The equator ($\theta = \pi/2$) contains equal superposition of computational basis states:
  - $\phi = 0$ gives $|+\rangle$ (positive x-axis)
  - $\phi = \pi$ gives $|-\rangle$ (negative x-axis)
  - $\phi = \pi/2$ gives $|+i\rangle$ (positive y-axis)
  - $\phi = 3\pi/2$ gives $|-i\rangle$ (negative y-axis)

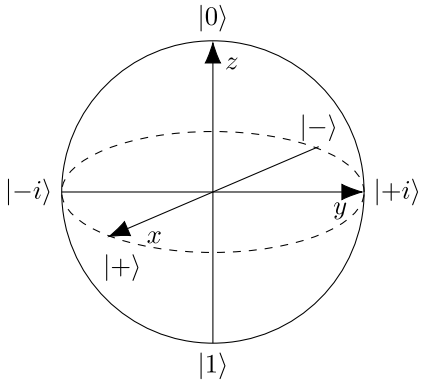


Fig. 2.1: The Bloch sphere showing eigenstates of X, Y, and Z Pauli operators

Example: Hadamard Gate on $|0\rangle$

The Hadamard gate H takes the state $|0\rangle$ (north pole) to $|+\rangle$ (on the equator at $\phi = 0$). In terms of the Bloch sphere coordinates, this means:

- Starting point: $\theta = 0$ (north pole)
- Ending point: $\theta = \pi/2$, $\phi = 0$ (positive x-axis)

The gate effectively rotates the state by 90° around the y-axis. Similarly, $H|1\rangle$ takes the south pole to $|-\rangle$ on the negative x-axis.

2.2.1 General Unitary Rotations

The Hadamard example shows how unitary gates can rotate states on the Bloch sphere. More generally, any single-qubit unitary operation can be thought of as a rotation of the Bloch sphere. Let's see how this works.

A general rotation around a unit vector $\vec{n} = (n_x, n_y, n_z)$ by angle θ is given by

$$R_{\vec{n}}(\theta) = \cos(\theta/2)I - i\sin(\theta/2)(n_x X + n_y Y + n_z Z).$$

For example:

- Rotation around z-axis:
$$R_z(\theta) = e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$
- Rotation around x-axis:
$$R_x(\theta) = e^{-i\theta X/2} = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$
- Rotation around y-axis:
$$R_y(\theta) = e^{-i\theta Y/2} = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

A remarkable fact is that any single-qubit unitary operation can be decomposed into three rotations around two different axes. This is known as the Euler angle decomposition:

$$U = e^{i\alpha} R_z(\phi) R_y(\theta) R_z(\psi)$$

where α, ϕ, θ, and ψ are real numbers. The global phase $e^{i\alpha}$ is often unimportant for quantum computing purposes.

Visualizing Rotations

The Euler angle decomposition has a nice geometric interpretation:

1. First rotation ($R_z(\psi)$): Rotate around z-axis

2. Second rotation ($R_y(\theta)$): Tilt to new latitude
3. Third rotation ($R_z(\phi)$): Rotate to final longitude
4. Global phase ($e^{i\alpha}$): Invisible in measurements

This is similar to how we specify points on Earth using latitude and longitude!

2.2.2 The Phase Gate

The phase gate (often denoted as S) is another important single-qubit gate that adds a phase of i to the $|1\rangle$ state while leaving $|0\rangle$ unchanged:

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

When acting on basis states:

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

The phase gate is equivalent to a $\pi/2$ rotation around the z-axis: $S = R_z(\pi/2)$. On the Bloch sphere, this corresponds to rotating a state by 90° around the z-axis.

Example: Phase Gate on Superposition

Let's see what happens when we apply S to an equal superposition state:

$$\begin{aligned} S|+\rangle &= S\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \\ &= \frac{1}{\sqrt{2}}(S|0\rangle + S|1\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \end{aligned}$$

This transforms $|+\rangle$ into $|+i\rangle$, rotating it from the positive x-axis to the positive y-axis on the Bloch sphere.

The phase gate is particularly important in quantum error correction and quantum algorithms where controlled phase operations are needed.

More generically, we can create any $R_z(\phi)$ gate to perform rotations about the z-axis (this is very useful for Shor's algorithm). However, a common variant is the T gate, which is simply $R_z(\pi/4)$ and is one of the minimal components needed to achieve universal quantum computation.

! Phase gate leaves computational basis states alone

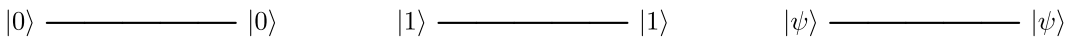
Note that these gates only change superposition of computational basis states, so $S|0\rangle = |0\rangle$ and $S|1\rangle = i|1\rangle$. (Similarly for any gate made from rotations about the z-axis.)

2.3 Quantum Circuits

Now that we've covered the key single-qubit operations, we can start to think about how to represent sequences of these operations graphically using quantum circuits. In quantum circuits:

- Qubits are represented as horizontal lines (called "wires", see Fig. 2.2).
- Gates are boxes or symbols placed on these wires (see Fig. 2.3 and Fig. 2.4).
- Time flows from left to right.
- Measurements are represented by meters (see Fig. 2.5).

Often, unless we are preparing a specific state for an algorithm, we may leave off the states from the ends of the "wire." This wire, Fig. 2.2, you can think of as the identity.



(a) Identity on $|0\rangle$ (b) Identity on $|1\rangle$ (c) Identity on $|\psi\rangle$

Fig. 2.2: Lines representing the evolution of a qubit (no gates applied)

While in quantum state evolution we apply operators right-to-left, the two operations $X|0\rangle = |1\rangle$ and $Z|+\rangle = |-\rangle$ are represented in Fig. 2.3.

> **!** Important
>
> The Pauli operators are both unitary and Hermitian, so they perform as quantum gates (which require unitary as Section 1.3.3 stipulates) and as observables (as Section 1.3.2 stipulates).

(a) Application of the X gate. (b) Application of the Z gate.

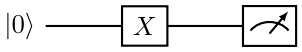
Fig. 2.3: Examples of the Pauli operators as gates.

The Hadamard and phase gates can also be mixed in, Fig. 2.4, and we can even introduce the meter symbol for measurements, Fig. 2.5.

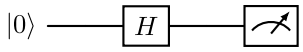
(a) Application of the Hadamard gate. (b) Application of the phase gate

Fig. 2.4: Examples of the Hadamard and phase gates.

Whenever measurements are performed, we will often transform to the computational basis $|0\rangle$ and $|1\rangle$ to perform a measurement. By playing tricks with unitary transformations, we can measure other observables by mixing unitaries and measurements of the computational basis; dependent on the platform, this may or may not be necessary. In terms of diagrams, you should assume meters are measurements in the computational basis unless it is noted otherwise.



(a) Will measure 1 with 100% probability



(b) Will measure 1 with 50% probability

Fig. 2.5: Examples of measurements. The meter is assumed to be measuring in the computational basis unless otherwise noted (i.e., measuring the Z operator).

i Reading Quantum Circuits

Quantum circuits are read from left to right, just like reading text. Each horizontal line represents a qubit's journey through time, and the boxes show what operations happen and when.
The measurement symbol at the end indicates when we extract classical information from our quantum system.

These diagrams give us a powerful visual language for describing quantum computations. Even complex algorithms can be broken down into sequences of these basic operations.

Example: Gate Sequences

Consider applying H then Z then H to $|0\rangle$:

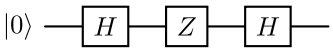


Fig. 2.6: Implementing an X gate with Z and H.

1. $H|0\rangle = |+\rangle$ (move to +x axis)
2. $Z|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ (rotate around z by 90°)
3. $H(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) = |1\rangle$ (move to -z axis)

This sequence effectively implements the X gate!

2.4 Noise and decoherence

A large part of the course will consist of finding physical systems where we can identify certain states as $|0\rangle$ and $|1\rangle$, and then proceed to figure out how to perform the gates we need for quantum algorithms. In this way, this course could be called "two-level systems and where to find them." However, these states are not pristine and so we need a way to talk about states that subject to noisy environments and loss of information.

In fact, environmental interactions can cause, amongst other issues:

- Loss of phase information (dephasing)
- Loss of energy (amplitude damping)
- Random bit flips

To properly describe these noise processes, we need to move beyond pure state descriptions and introduce the density matrix formalism.

2.4.1 The Density Matrix

For a pure quantum state $|\psi\rangle$, the density matrix is defined as

$$\rho = |\psi\rangle \langle\psi|$$

For example, the computational basis states have density matrices:

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

The density matrix also provides a convenient way to calculate expectation values of observables. For any observable A, the expectation value is given by:

$$\langle A \rangle = \text{tr}(A\rho)$$

For a pure state $|\psi\rangle$, this reduces to our familiar expression:

$$\text{tr}(A |\psi\rangle \langle\psi|) = \langle\psi| A |\psi\rangle$$

Example: Measuring Z with the Density Matrix

Consider measuring Z for the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$:
The density matrix is:

$$\rho_+ = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Using the formula for expecation values with $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$:

$$\langle Z \rangle = \text{tr}(Z\rho_+) = \text{tr}\left(\frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right) = 0$$

This matches what we expect: $|+\rangle$ has equal probability of measuring ± 1 for Z.

The real power of density matrices comes from describing *mixed states*, which are statistical mixtures of pure states:

$$\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i|$$

where p_i are classical probabilities ($p_i \geq 0$, $\sum_i p_i = 1$).

Properties of Density Matrices

Any valid density matrix must satisfy:

1. Hermiticity: $\rho = \rho^\dagger$

2. Positive semidefinite: $\langle\phi|\,\rho\,|\phi\rangle \geq 0$ for all $|\phi\rangle$
3. Unit trace: $\text{tr}(\rho) = 1$
4. For pure states: $\text{tr}(\rho^2) = 1$
5. For mixed states: $\text{tr}(\rho^2) < 1$

Quantum Operations with Density Matrices

Just as we can evolve pure states with unitary operators, we can evolve density matrices. For a unitary operation U, the density matrix transforms as:

$$\rho \to U\rho U^\dagger$$

This preserves all the properties of the density matrix we discussed above.

When we make a measurement, we use projection operators $\{P_m\}$ (like $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ for measuring in the computational basis). The probability of getting outcome m is:

$$p(m) = \text{tr}(P_m\rho)$$

After measuring and getting outcome m, the state "collapses" to:

$$\rho_m = \frac{P_m\rho P_m}{\text{tr}(P_m\rho)}$$

where the denominator ensures the trace remains 1.

i Quantum Channels

The density matrix formalism really shines when describing general quantum evolution, including noise. Any physical process (unitary or not) can be described by a quantum channel $\mathcal{E}$:

$$\rho \to \mathcal{E}(\rho) = \sum_k E_k\rho E_k^\dagger$$

The operators E_k (called Kraus operators) satisfy $\sum_k E_k^\dagger E_k = I$ to preserve probability. This includes both ideal unitary evolution (one $E_k = U$) and noise processes (multiple E_k), as we'll see in the next section.

Density Matrices and the Bloch Sphere

The density matrix formalism provides a beautiful geometric picture when combined with the Bloch sphere representation in Section 2.2. While pure states live on the surface of the Bloch sphere, mixed states live *inside* it. Any density matrix for a single qubit can be written as:

$$\rho = \frac{1}{2}(I + \vec{r}\cdot\vec{\sigma})$$

where $\vec{r} = (r_x, r_y, r_z)$ is called the Bloch vector and $\vec{\sigma} = (X, Y, Z)$ are the Pauli matrices.

The length of $\vec{r}$ determines how mixed the state is:

- For pure states, $|\vec{r}| = 1$ (surface of sphere)
- For mixed states, $|\vec{r}| < 1$ (inside sphere)
- For the maximally mixed state, $\vec{r} = 0$ (center of sphere)

This gives us an intuitive picture of decoherence: noise processes like dephasing and bit flips move the Bloch vector inward from the surface, representing the loss of quantum information. The maximally mixed state at the center represents complete loss of information–equal probabilities for all measurement outcomes.

> **Connection to Purity**
>
> The length of the Bloch vector is directly related to the purity $\text{tr}(\rho^2)$ we discussed earlier:
>
> $$\text{tr}(\rho^2) = \frac{1 + |\vec{r}|^2}{2}$$
>
> This confirms that pure states ($|\vec{r}| = 1$) have purity 1, while mixed states have purity less than 1.

2.4.2 Modeling Noise

A full discussion of different quantum channels can be found in [18]. Here, we discuss some of the ones relevant for loss of quantum information.

The language of density matrices allows us to describe some of the errors that can occur in a precise manner

Random bit-flips

If there is probability p that a bit is flipped, we can encode this within a change of the density matrix:

$$\rho \to (1-p)\rho + pX\rho X,$$

where the first term describes the probability that the density matrix remains unchanged and the second describe the process of randomly flipping a bit.

Dephasing

As we've already mentioned the phase between $|0\rangle$ and $|1\rangle$ is useful information that we will take advantage of. However, some physical processes can scramble this phase in a way that could be inherently difficult to parse, in that case we can describe these processes with "dephasing"

$$\rho \to (1-p)\rho + pZ\rho Z.$$

This process has a clear action on ρ when $p = 1/2$. To illustrate this, consider $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, this has a probability of measuring 0 of $|\alpha|^2$ and a probabilty of measuring 1 of $|\beta|^2$, but there is also phase information between α and β that tells us where on the Bloch sphere the state is located. Dephasing will eliminate this phase information.

To see this, the full density matrix before we apply dephasing is

$$\rho = \begin{bmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{bmatrix},$$

and after dephasing we will have the density matrix

$$\frac{1}{2}\rho + \frac{1}{2}Z\rho Z = \begin{bmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{bmatrix}.$$

The off-diagonal components carried the relative phase information and it is now entirely lost, leaving us with a classical mixture of $|0\rangle$ and $|1\rangle$

! Measurement as Dephasing

This process of losing phase information is exactly what happens when we measure a quantum state in a particular basis! When we measure in the computational basis ($\{|0\rangle, |1\rangle\}$), we are effectively performing complete dephasing - we destroy all phase information between the basis states and are left with only the classical probabilities. This is why measurement is often described as "collapsing" the quantum state into classical information.
This connection between measurement and dephasing illustrates a fundamental aspect of quantum mechanics: the act of gaining classical information about a quantum system necessarily destroys some of its quantum properties, as formalized in the measurement postulates we discussed earlier.

Example: Dephasing of a Superposition State

Consider the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ under dephasing:
Initial density matrix:

$$\rho_0 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

After dephasing with probability p:

$$\rho(p) = \frac{1}{2}\begin{bmatrix} 1 & (1-2p) \\ (1-2p) & 1 \end{bmatrix}$$

Complete dephasing ($p = \frac{1}{2}$) yields the maximally mixed state:

$$\rho(\tfrac{1}{2}) = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Depolarizing

The depolarizing channel represents one of the most severe forms of noise in quantum systems, often considered the "worst-case scenario" for quantum information. This channel transforms any input state into a mixture of itself and the maximally mixed state:

$$\rho \to (1-p)\rho + p\frac{I}{2},$$

where p represents the probability of depolarization. When $p = 1$, the state becomes completely mixed regardless of the input:

$$\rho \to \frac{1}{2} I.$$

Note that this has eliminated any information about $|\psi\rangle$!

Example: The Pauli Twirl

An interesting way to understand the depolarizing channel is through what's called the "Pauli twirl". The depolarizing channel can be written as a random application of Pauli operators:

$$\rho \to (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z).$$

This means we can implement depolarizing noise by randomly applying X, Y, or Z gates with probability $p/3$ each. This is particularly useful in quantum error correction, where we often want to simulate noise in a way that's easy to analyze.

Amplitude Damping

In many physical systems, one can lose information by having emission of energy. Physically, this could be an atom in some energy level and it emits a photon, falling to a lower energy level. One can imagine a process like this occuring with an operator

$$E_0 = \sqrt{\gamma}\,|0\rangle\langle 1|\,,$$

which describes the process of taking an occupied state $|1\rangle$ and transforming it to $|0\rangle$ with probability γ. To fully encode this into a change in the density matrix, we need one other operator: the chance that nothing happens. A natural guess would be

$$E_1 = |0\rangle\langle 0| + \sqrt{1-\gamma}\,|1\rangle\langle 1|\,,$$

which describes that there is unit probability of remaining $|0\rangle$ and probaiblity $1-\gamma$ of remaining in $|1\rangle$ if you start there.

The full operation on the density matrix is then

$$\rho \to E_1 \rho E_1^\dagger + E_0 \rho E_0^\dagger$$

Example: Amplitude Damping in Action

Let's see how amplitude damping affects a simple superposition state. Consider the initial state

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

which has density matrix

$$\rho = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

After applying amplitude damping with probability γ, the state becomes

$$\begin{aligned}\rho' &= E_1 \rho E_1^\dagger + E_0 \rho E_0^\dagger \\ &= \frac{1}{2}\begin{bmatrix} 1+\gamma & \sqrt{1-\gamma} \\ \sqrt{1-\gamma} & 1-\gamma \end{bmatrix}.\end{aligned}$$

We can see that:

- The probability of being in $|1\rangle$ decreases by γ
- The probability of being in $|0\rangle$ increases by γ
- The off-diagonal coherence terms decay by $\sqrt{1-\gamma}$

When $\gamma = 1$ (complete damping), the state becomes

$$\rho' = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = |0\rangle\langle 0|,$$

representing complete relaxation to the ground state.

Erasure Channel

Another important quantum channel is the erasure channel, which models the loss of a qubit to other accessible states. Unlike amplitude damping, where information leaks gradually, the erasure channel represents a complete loss of the qubit with probability ϵ. When this happens, the qubit is replaced by an "error state" that we denote as $|e\rangle$. Importantly, $|e\rangle$ exists in a different part of the Hilbert space than our qubit states - it's an additional state that flags that erasure has occurred.

! Orthogonality of Error State

The error state $|e\rangle$ is orthogonal to both $|0\rangle$ and $|1\rangle$, meaning $\langle e|0\rangle = \langle e|1\rangle = 0$. This is crucial as it represents a state completely outside our original qubit space but still within the system itself.

The erasure can then be represented by

$$\rho \mapsto (1-\epsilon)\rho + \epsilon\,|e\rangle\langle e|.$$

In particular, this process

- Preserves the state with probability $1-\epsilon$
- Erases it completely with probability ϵ, replacing it with the error state $|e\rangle$

Example: Erasure Channel in Action

Since the erasure channel operates in an enlarged Hilbert space that includes the error state (in this case, making it 3-dimensional), let's consider our superposition state embedded in this

expanded space:

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

After the erasure channel acts, the state becomes a mixed state:

$$\rho' = (1-\epsilon)\begin{bmatrix}1/2 & 1/2 & 0\\1/2 & 1/2 & 0\\0 & 0 & 0\end{bmatrix} + \epsilon\begin{bmatrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\frac{1-\epsilon}{2} & \frac{1-\epsilon}{2} & 0\\\frac{1-\epsilon}{2} & \frac{1-\epsilon}{2} & 0\\0 & 0 & \epsilon\end{bmatrix}$$

This represents that with probability $1-\epsilon$ we still have our original state in the upper-left 2×2 block, but with probability ϵ we have completely lost it to the error state, represented by the 1 in the bottom-right corner.

The erasure channel is particularly important in quantum communication and error correction because the error state exists within the system Hilbert space itself (not the environment). This additional state in the system allows us to track and flag when errors occur–making it especially useful for quantum error correction protocols, unlike other noise channels where the errors are unknown and harder to detect.

2.4.3 Physical Qubit Implementations

The quest for building practical quantum computers has led to several different approaches for implementing qubits. Each implementation aims to create a physical system that can reliably represent quantum states and allow for precise control and manipulation. We will explore in detail later what is needed for a good quantum device, but give a brief overview here of *what* the devices are.

Superconducting Qubits

- Qubit are energy levels of *charge* or *flux* within a superconducting circuit.
- Based on superconducting circuits using Josephson junctions.
- Operate at extremely low temperatures (~20 mK).
- Used by: IBM, Google, Rigetti.

Trapped Ion Qubits

- Qubits are electronic or nuclear states of individual ions
- Ions held in electromagnetic traps, manipulated by lasers
- Used by: IonQ, Honeywell-Quantinuum

Photonic Qubits

- Qubits are properites of light (e.g., polarization)
- Can operate at room temperature
- Used by: PsiQuantum, Xanadu

Semiconductor Quantum Dots

- Qubits can be encoded using either:
  - *charge* states (electron occupying different quantum dots)
  - *spin* states (up/down spin states of an electron)
- Created by confining electrons in semiconductor nanostructures
- Often called "artificial atoms" due to their discrete energy levels
- Active research at: Intel, TU Delft, Princeton, UNSW, CEA-Leti

NV Centers in Diamond

- The nitrogen vacancy defects in diamond provide energy levels accessible with light.
- Can operate at room temperature
- Applications in quantum sensing and networking

Topological Qubits

- Qubits are non-local and topologically protected states within an exotic (topological) state of matter.
- Most promising candidate: Majorana zero modes
- **Current Status**:
  - Active research at Microsoft and Delft
  - Recent progress in identifying Majorana signatures in nanowires
  - Debate continues over experimental evidence

As we'll discuss, the "perfect" qubit would combine long coherence times, fast gates, high fidelity, easy coupling to other qubits, and straightforward scalability. While each implementation has made significant progress, achieving all these properties simultaneously remains a major challenge in the field.

3 Multiple Qubits

Previously in Chapter 2 we discussed in detail how to understand a single qubit. While we saw some basic features, such as superposition and relative phase, it has not been apparent yet what we can do with these features. The power of these will really be unlocked by putting multiple qubits together. We will also begin to see hints of *quantum entanglement*.

However, there is just a practical concern: How much information can we store in a single qubit? With precise control, we have our answer in Section 2.2: the angles θ and ϕ on the Bloch sphere. Since algorithms are bit more greedy than that, we need to extend our space and the natural way to do that is to add more qubits. Of course, classically, we operate with many bits: Floats (real numbers) on most computers use 64 bits, and we often add, substract, multiply a lot of these numbers. But classically, when we have, for instance, two bits, there are four discrete states 00, 01, 10, and 11. As we will analyze in detail, quantum mechanically these will be four basis states that can make up a general quantum wave function

$$|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle\,. \tag{3.1}$$

3.1 Tensor Products and the multi-qubit Hilbert space

The equation above gives us an idea for how we ought to combine qubits. The tensor product will formalize this, but we will build it up "intuitively."

Imagine we have one big operator called "read-out" or $\mathcal{R}$; this operator will measure all of the qubits in the system, and its value will tell us *exactly* what the state of the system is in terms of the computational basis.

> **Two-Qubit Readout**
>
> The readout operator $\mathcal{R}$ maps the two-qubit basis states to unique numbers:
>
> - $\mathcal{R}\,|00\rangle = 0\,|00\rangle$
> - $\mathcal{R}\,|01\rangle = 1\,|01\rangle$
> - $\mathcal{R}\,|10\rangle = 2\,|10\rangle$
> - $\mathcal{R}\,|11\rangle = 3\,|11\rangle$
>
> This binary-to-decimal conversion helps us uniquely identify each computational basis state.

However, this operator $\mathcal{R}$ *must* be Hermitian to be a physical observable, and as a result its eigenstates span our Hilbert space. Well, upon readout, we know that each qubit can be in one of two states $b_i = 0$ or 1, and if we have $i = 1, \ldots, N$ qubits, there are 2^N possibilities.

We will return to this, but let us linger on two qubits. Notice that we can obtain the above states with an operation called the *tensor product*.

$\otimes$	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	$\|0\rangle \otimes \|0\rangle$	$\|0\rangle \otimes \|1\rangle$
$\|1\rangle$	$\|1\rangle \otimes \|0\rangle$	$\|1\rangle \otimes \|0\rangle$

This multiplication table is how we go from basis sets of single qubits to the basis set of two qubits. Very often, we will drop the $\otimes$ and simply write

$$|b_1 b_2\rangle \equiv |b_1\rangle \otimes |b_2\rangle \,.$$

Similarly, we can build up three qubit states by taking the tensor product of two-qubit states with a single qubit:

$\otimes$	$\|0\rangle$	$\|1\rangle$
$\|00\rangle$	$\|00\rangle \otimes \|0\rangle$	$\|00\rangle \otimes \|1\rangle$
$\|01\rangle$	$\|01\rangle \otimes \|0\rangle$	$\|01\rangle \otimes \|1\rangle$
$\|10\rangle$	$\|10\rangle \otimes \|0\rangle$	$\|10\rangle \otimes \|1\rangle$
$\|11\rangle$	$\|11\rangle \otimes \|0\rangle$	$\|11\rangle \otimes \|1\rangle$

again we can define

$$|b_1 b_2 b_3\rangle = |b_1 b_2\rangle \otimes |b_3\rangle \,,$$

and we proceed once more

$\otimes$	$\|0\rangle$	$\|1\rangle$
$\|000\rangle$	$\|000\rangle \otimes \|0\rangle$	$\|000\rangle \otimes \|1\rangle$
$\|001\rangle$	$\|001\rangle \otimes \|0\rangle$	$\|001\rangle \otimes \|1\rangle$
$\|010\rangle$	$\|010\rangle \otimes \|0\rangle$	$\|010\rangle \otimes \|1\rangle$
$\|011\rangle$	$\|011\rangle \otimes \|0\rangle$	$\|011\rangle \otimes \|1\rangle$
$\|100\rangle$	$\|100\rangle \otimes \|0\rangle$	$\|100\rangle \otimes \|1\rangle$
$\|101\rangle$	$\|101\rangle \otimes \|0\rangle$	$\|101\rangle \otimes \|1\rangle$
$\|110\rangle$	$\|110\rangle \otimes \|0\rangle$	$\|110\rangle \otimes \|1\rangle$
$\|111\rangle$	$\|111\rangle \otimes \|0\rangle$	$\|111\rangle \otimes \|1\rangle$

As we can see, every time we add a new qubit, the dimension of the space is multiplied by two. In general, we can can break N-qubits in the computational basis as

$$|b_1 b_2 \cdots b_N\rangle = |b_1\rangle \otimes |b_2\rangle \otimes \cdots \otimes |b_N\rangle$$

where $b_j = 0$ or 1. Since this is a basis, we can "count" the number of basis states to determine the dimension of the space to obtain for the Hilbert space of N-qubits $\mathcal{H}_N$,

$$\dim \mathcal{H}_N = 2^N.$$

Therefore, a general quantum state of N qubits can be written as a superposition of all possible computational basis states:

$$|\psi\rangle = \sum_{b_1,\ldots,b_N=0,1} \psi_{b_1 \cdots b_N} |b_1 \cdots b_N\rangle$$

where $\psi_{b_1 \cdots b_N}$ are complex coefficients satisfying the normalization condition

$$\sum_{b_1,\ldots,b_N=0,1} |\psi_{b_1 \cdots b_N}|^2 = 1.$$

This means that to fully specify a quantum state of N qubits, we need 2^N complex numbers (subject to normalization), which illustrates both the power and challenge of quantum computing - the exponential growth in the state space allows for massive parallel processing but also makes classical simulation difficult.

Two-qubit wave functions and entanglement

A general two-qubit wave function can be written as

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

where $|\psi_{00}|^2 + |\psi_{01}|^2 + |\psi_{10}|^2 + |\psi_{11}|^2 = 1$. When the two qubits are independent (a product state), we can write this as a tensor product of individual qubit states:

$$\begin{aligned} |\psi\rangle &= (\alpha_1 |0\rangle + \beta_1 |1\rangle) \otimes (\alpha_2 |0\rangle + \beta_2 |1\rangle) \\ &= \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\alpha_2 |10\rangle + \beta_1\beta_2 |11\rangle \end{aligned}$$

However, not all two-qubit states can be written as such a product! States that cannot be factored into a tensor product of individual qubit states are called *entangled states* - a crucial quantum resource we'll explore later.

The exponential growth in the state space has important implications for simulating quantum systems on classical computers. While we need 2^N complex numbers to specify an arbitrary quantum state, the situation becomes even more demanding when we consider operations on these states:

1. To represent an arbitrary quantum operation (unitary evolution) on N qubits, we need a $2^N \times 2^N$ unitary matrix. This requires storing and manipulating 2^{2N} complex numbers.
2. Even to compute the probability of a measurement outcome, we need to perform operations involving all 2^N amplitudes.

This exponential scaling is why classical computers struggle to simulate large quantum systems - the memory and computational requirements become overwhelming. For example:

- 10 qubits: $2^{10} = 1,024$ amplitudes, $2^{20} \approx 1$ million matrix elements
- 30 qubits: $2^{30} \approx 1$ billion amplitudes, $2^{60} \approx 10^{18}$ matrix elements
- 50 qubits: $2^{50} \approx 10^{15}$ amplitudes, $2^{100} \approx 10^{30}$ matrix elements

However, it's important to note that not all quantum computations require storing and manipulating the full state space. Many practical quantum algorithms and simulations exploit special properties:

- Some quantum states have special structure (like product states) that allow more efficient representations
- Many quantum operations act locally or have special symmetries that reduce the computational complexity
- Some quantum algorithms can be simulated using specialized techniques that avoid storing the full state vector

Nevertheless, the ability to access and manipulate this exponentially large state space can help us perform computations that classical computers would struggle with.

! Classical simulation vs. quantum measurement

When simulating quantum systems on classical computers, we have direct access to the full state vector - all the complex amplitudes $\psi_{b_1 \cdots b_N}$. This gives us complete information about the quantum state, allowing us to calculate any property without performing repeated measurements.
In contrast, real quantum computers are bound by the measurement postulates of quantum mechanics (Postulate II, Section 1.3.2). Each measurement:

1. Collapses the quantum state
2. Only returns eigenvalues of the measured observable
3. Must be repeated many times to estimate expectation values and state properties

This limitation of quantum hardware is why techniques like quantum state tomography are necessary - reconstructing the full quantum state requires performing many different measurements on multiple copies of the same state. Classical simulation sidesteps this fundamental quantum constraint, though at the cost of exponential classical resources.

Example: Building up three-qubit states

Consider how we build up the state $|101\rangle$:

1. Start with first two qubits: $|10\rangle$
2. Tensor with third qubit: $|10\rangle \otimes |1\rangle$
3. This gives: $|101\rangle$

We can verify this matches our counting:

- First qubit: $|1\rangle$ (second basis state)
- Second qubit: $|0\rangle$ (first basis state)
- Third qubit: $|1\rangle$ (second basis state)

Therefore in binary: 101, which is state number 5 in our computational basis (counting from 0).

Before we run head first into entanglement, let's take a minute to just do some counting to see that we are going to run into some trouble. For a single qubit, we have two complex numbers $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, but we need to normalize them $|\alpha|^2 + |\beta|^2 = 1$ and remove a phase. Thus, we have reduced our 2 complex numbers (= 4 real numbers) down to 4-1-1=2 real numbers. This is made explicit with the Bloch sphere where two real numbers (θ, ϕ) completely characterize the state. Therefore, do we only need $2 \times 2 = 4$ real numbers to describe a 4 qubit state? Well, let's count, we can see from Eq. 3.1 that we have 4 complex numbers (= 8 real numbers). But we must also impose normalization and remove an overall phase, reducing us down to 8-1-1 = 6 real numbers. But $6 > 4$; this is our first hint of something happening in our quantum system, we need more numbers to describe all of the states in a two qubit system than simply what we needed for two separate qubits.

Mathematical Note: State Space Geometry

For the mathematically inclined: The physical state space of an N-qubit system is complex projective space $\mathbb{CP}^{2^N-1}$. For two qubits, this means $\mathbb{CP}^3$, which is fundamentally different from $\mathbb{CP}^1 \times \mathbb{CP}^1$ (the space of two separate qubits). This geometric fact underlies why we need more parameters to describe entangled states - the state space has a richer structure than just the product of individual qubit spaces.

3.2 Entanglement

Entanglement is one of the most important phenomena in quantum mechanics without a clear classical antecedent. The term was first coined by Schrödinger in 1935 [19] in response to a famous paper by Einstein, Podolsky, and Rosen (EPR) [20]. It represents quantum correlations between particles that cannot be explained by a simple "lack of knowledge" by the observer. To get around this, it was thought there must be "hidden variables" to make quantum mechanics complete; thus, EPR used these quantum correlations to argue that quantum mechanics must be incomplete. Quantum mechanics appeared to allow "spooky action at a distance" that violated their ideas of locality and reality.

However, in 1964, John Stewart Bell [21] showed that quantum mechanics predicts correlations between entangled particles that are mathematically impossible to explain with any local hidden variable theory. Subsequent experiments have repeatedly confirmed these "Bell inequality violations," demonstrating that entanglement represents a fundamentally new kind of physical relationship not reducible to classical correlations (early experiments include [22] and [23]).

The existence of entanglement suggests that the quantum wave function represents more than just our knowledge about measurement probabilities - it appears to be a real physical object. More recent work by Matthew Pusey, Jonathan Barrett, and Terry Rudolph [24] has strengthened this view through their "PBR theorem," which shows that if quantum predictions are correct, then quantum states must be physically real rather than merely statistical.

Reality of the Wave Function

The PBR theorem (2012) [24] tells us something deep about quantum mechanics. To quote the paper,

> In conclusion, we have presented a no-go theorem, which—modulo assumptions—shows that models in which the quantum state is interpreted as mere information about an objective physical state of a system cannot reproduce the predictions of quantum theory. The result is in the same spirit as Bell's theorem, which states that no local theory can reproduce the predictions of quantum theory.

This provides strong support for viewing entanglement as a genuine physical phenomenon rather than just a limitation of our knowledge.

Let's explore what entanglement means mathematically and physically.

3.2.1 Bell States and Non-local Correlations

One of the simplest cases of quantum entanglement is the Bell state

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{3.2}$$

This state exhibits perfect correlations that persist regardless of the physical separation between the qubits. When we measure the first qubit:

- If we get $|0\rangle$, the state collapses to $|00\rangle$
- If we get $|1\rangle$, the state collapses to $|11\rangle$

The remarkable feature is that measuring either qubit instantly determines the state of the other qubit, even if they are separated by vast distances. For example:

1. Create $|\Phi^+\rangle$ and separate the qubits by sending one to Earth and one to Mars
2. Measure the Earth qubit $\rightarrow$ get result 0 or 1 with 50% probability
3. We know with 100% certainty that the Mars qubit will be measured to give the same result.
4. This happens faster than light could travel between the qubits.

! No faster-than-light communication

While entanglement appears to create "spooky action at a distance," it cannot be used to transmit information faster than light. This is because:

1. The measurement results are random
2. The person with the second qubit needs classical information about the first measurement to interpret their results
3. This classical information is still limited by the speed of light

i Bell state properties are basis independence

A remarkable property of the Bell state $|\Phi^+\rangle$ is that these perfect correlations persist no matter what basis we measure in. If we measure the first qubit in any basis and get some state $|\psi\rangle$, the second qubit will always be found in state $|\psi\rangle$ when measured in the same basis.
For example, if we measure the first qubit in the X basis: - If we get $|+\rangle$, the state collapses to $|{+}{+}\rangle$ - If we get $|-\rangle$, the state collapses to $|{-}{-}\rangle$
This is because we can rewrite $|\Phi^+\rangle$ in any basis and it maintains the same form:

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

This is true for any basis, not just Z and X.

These non-local correlations are fundamentally different from classical correlations, but how can we see that? The key ends up being: measurements that show quantum correlations and was Bell's central insight [21]. Instead of just looking at measurements of Z which are easily explained by a classical hidden variable, also perform measurements at other angles of the Bloch sphere.

i Mermin's Example

To see a simple example, read [25]. The essence of the idea presented there is to measure three different observables with the Bell state in Eq. 3.2,

$$Z, \quad -\frac{1}{2}Z + \frac{\sqrt{3}}{2}X, \quad -\frac{1}{2}Z - \frac{\sqrt{3}}{2}X,$$

with the results from these three measurements, there is no way to use classical probabilities of measurement outcomes to account for the distribution of results.

3.2.2 Mathematical definition and separability

A multi-qubit quantum state is entangled *if and only if* it cannot be written as a tensor product of individual qubit states[1]. For a two-qubit pure state $|\psi\rangle$, this means there do not exist single-qubit states $|\phi_1\rangle$ and $|\phi_2\rangle$ such that:

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle$$

For a general two-qubit state $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$, there is a simple condition for separability: the state is separable if and only if the determinant of its coefficient matrix is zero:

$$\begin{vmatrix} \alpha & \beta \\ \gamma & \delta \end{vmatrix} = \alpha\delta - \beta\gamma = 0$$

This can be proven by writing out the general form of a tensor product and matching coefficients. The classic examples of maximally entangled states are the Bell states:

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

For mixed states, the situation is more complex and requires measures like concurrence to fully characterize entanglement.

Example: Separable state condition

Consider a separable two-qubit state formed by the tensor product of two arbitrary single-qubit states:

$$|\phi_1\rangle = a|0\rangle + b|1\rangle \qquad |\phi_2\rangle = c|0\rangle + d|1\rangle$$

Their tensor product gives:

[1] This is true for pure states, but can be easily generalized to mixed states, see [7].

$$|\phi_1\rangle \otimes |\phi_2\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)$$
$$= ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$$

The coefficient matrix determinant is:

$$\begin{vmatrix} ac & ad \\ bc & bd \end{vmatrix} = (ac)(bd) - (ad)(bc) = abcd - abcd = 0$$

This confirms that any separable state satisfies the zero determinant condition. Conversely, if a state's coefficient matrix has non-zero determinant, it must be entangled.

Example: Checking for Entanglement

Let's examine two states to see if they're entangled:

1. Consider the state $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ The coefficient matrix is:

$$\begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{pmatrix}$$

   The determinant is $(1/\sqrt{2})(1/\sqrt{2}) - (0)(0) = 1/2 \neq 0$, so this state is entangled.

2. Consider the state $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ The coefficient matrix is:

$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & 0 \end{pmatrix}$$

   The determinant is $(1/\sqrt{2})(0) - (1/\sqrt{2})(0) = 0$, so this state is separable. Indeed, we can write it as $|\psi_2\rangle = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

3. Consider the state $|\psi_3\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ The coefficient matrix is:

$$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

   The determinant is $(1/2)(1/2) - (1/2)(1/2) = 0$, so this state is separable. We can verify this by rewriting it as: $|\psi_3\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

3.2.3 Reduced density matrices

For entangled states, we often want to describe the state of a single qubit within the two-qubit system. This is done using the reduced density matrix, obtained by "tracing out" the other qubit. The process, called the partial trace, gives us a density matrix that describes all measurable properties of the subsystem we're interested in.

For a two-qubit system in state $|\psi\rangle$, the density matrix is $\rho = |\psi\rangle\langle\psi|$. To get the reduced density matrix for the first qubit (ρ_A), we take the partial trace over the second qubit (B):

$$\rho_A = \mathrm{Tr}_B(\rho) = \sum_{b=0}^{1} \langle b_B|\rho|b_B\rangle$$

For a single term $\langle b_B|\rho|b_B\rangle$, this gives a 2×2 matrix acting on the first qubit:

$$\begin{aligned}\langle b_B|\rho|b_B\rangle &= \begin{pmatrix} \langle 0b|\rho|0b\rangle & \langle 0b|\rho|1b\rangle \\ \langle 1b|\rho|0b\rangle & \langle 1b|\rho|1b\rangle \end{pmatrix} \\ &= \begin{pmatrix} |\langle 0b|\psi\rangle|^2 & \langle 0b|\psi\rangle \langle 1b|\psi\rangle^* \\ \langle 1b|\psi\rangle \langle 0b|\psi\rangle^* & |\langle 1b|\psi\rangle|^2 \end{pmatrix}\end{aligned}$$

where $|b_B\rangle$ are the basis states of qubit B. Each element of ρ_A is a sum of two elements from the original density matrix, corresponding to tracing out qubit B in the computational basis.

Tensor notation for density matrices

In tensor notation, a two-qubit state $|\psi\rangle$ has components ψ_{ij} where i, j label the basis states of the first and second qubit. The density matrix elements are then $\rho_{ij,kl} = \psi_{ij}\psi^*_{kl}$, where the first pair of indices (i, j) corresponds to the ket and (k, l) to the bra.
The partial trace over qubit B corresponds to summing over matching indices for qubit B:

$$(\rho_A)_{ik} = \sum_j \rho_{ij,kj}$$

This tensor notation makes it clear why this operation is called a "trace" - we're summing over diagonal elements where the indices for system B match (j=j), just like in the usual matrix trace, while keeping the indices for system A (i,k) free.

Example: Reduced density matrix of a Bell state

Consider the Bell state $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Its density matrix is:

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

Let's find ρ_A step by step:

1. First, we compute the partial trace:

$$\begin{aligned}\rho_A &= \langle 0_B|\rho|0_B\rangle + \langle 1_B|\rho|1_B\rangle \\ &= \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \\ &= \frac{1}{2}I\end{aligned}$$

2. This shows that when we look at just one qubit of a maximally entangled pair:
   - It appears to be in a completely mixed state
   - We have equal probability of measuring 0 or 1
   - All quantum information is stored in the correlations between qubits

The reduced density matrix reveals a key feature of entanglement: while the total state is pure ($\rho^2 = \rho$), the subsystem state can be mixed ($\rho_A^2 \neq \rho_A$). This is a signature of entanglement - if we can only access one qubit of an entangled pair, we see a statistical mixture rather than a pure state.

Example: Partial Trace for a General Two-Qubit State

Suppose we have a general two-qubit pure state:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle.$$

The full density matrix is:

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} \begin{pmatrix} \alpha^* & \beta^* & \gamma^* & \delta^* \end{pmatrix}.$$

To find the reduced density matrix of the first qubit, $\rho_A = \mathrm{Tr}_B(\rho)$, group the basis states so that qubit B's index is traced out:

$$\rho_A = \begin{pmatrix} |\alpha|^2 + |\beta|^2 & \alpha\gamma^* + \beta\delta^* \\ \gamma\alpha^* + \delta\beta^* & |\gamma|^2 + |\delta|^2 \end{pmatrix}.$$

This 2×2 matrix captures all local measurements and observables on qubit A, regardless of the state of qubit B.

i Why Care About Reduced Density Matrices?

Reduced density matrices are crucial because they tell us what we can observe when we only have access to part of an entangled system. They help answer questions like:

1. **Local Measurements**: What results will we get if we only measure one qubit of an entangled pair?
2. **Quantum Information**: How much information is accessible locally vs. stored in correlations?
3. **Decoherence**: How does interaction with the environment affect our quantum system?

For example, in quantum teleportation, while the total state remains pure, the reduced density matrix of the transmitted qubit appears completely mixed until the classical information is received. This explains why teleportation cannot transmit information faster than light!

3.2.4 Quantifying entanglement

Several measures exist to quantify entanglement, each capturing different aspects:

1. **Von Neumann entropy**: For a pure bipartite state $|\psi\rangle$, the entanglement entropy is $S(\rho_A) = -\mathrm{Tr}(\rho_A \log_2 \rho_A)$ where ρ_A is the reduced density matrix of subsystem A. For two qubits, this ranges from 0 for separable states to 1 for maximally entangled states.

2. **Concurrence** [26]: For a two-qubit state ρ, defined as $C(\rho) = \max(0, \lambda_1 - \lambda_2 - \lambda_3 - \lambda_4)$ where λ_i are the square roots of eigenvalues of $\rho(Y \otimes Y)\rho^*(Y \otimes Y)$ in decreasing order (this reduces to $2|\alpha\delta - \beta\gamma|$ for a pure state $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$).
3. **Negativity**: Based on the partial transpose of the density matrix, providing a computable measure that captures the degree of entanglement. However, while negativity is zero for separable states, a zero negativity does not guarantee separability for some higher-dimensional systems - there exist entangled states with zero negativity.

These measures help quantify the "quantum-ness" of correlations and their potential utility in quantum information protocols. All of these are discussed in detail in [7].

We will find that many protocols of usefulness will produce entanglement in the system, though often only when in a particular basis. We will see that in the next section when we introduce operators on this Hilbert space

! Entanglement depends on the partition

When we talk about entanglement between subsystems A and B, it's crucial to understand that this depends entirely on how we choose to divide our total system into these subsystems. The same quantum state can appear entangled or unentangled depending on this choice of partition. For example, consider the state:

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

this is clearly an entangled Bell state, but our Hilbert space exists as (complex) four-dimensional space. I could "relabel" my states into a new basis (which we can define with a tilde), such that

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

and if we do that, we see that

$$|\psi\rangle = |\tilde{0}\tilde{0}\rangle\,,$$

a completely disentangled state! This is not to say that $|\psi\rangle$ is not entangled, it is. It is merely important to remember that entanglement is defined with respect to a certain partitioning of your Hilbert space. The space represented by $\tilde{0}$ and $\tilde{1}$ is a complex combination of 00, 01, 10, and 11. It is usually the physical situation which dictates how we partition our system (systems that are physically isolated from each other or for which we have single degrees of freedom which admit simple tensor products when considered with other systems).

When to Use Different Entanglement Measures

Each entanglement measure has its strengths:

1. **Von Neumann Entropy**
   - Best for: Pure bipartite states
   - Advantages: Clear physical interpretation, easy to calculate
   - Use when: You want to quantify how much quantum information is shared between subsystems
2. **Concurrence**
   - Best for: Two-qubit mixed states
   - Advantages: Can be directly calculated from density matrix
   - Use when: Working with noisy or mixed two-qubit states
3. **Negativity**
   - Best for: Higher-dimensional systems
   - Advantages: Easy to compute, works for mixed states
   - Use when: Dealing with larger systems or when you need a quick estimate of entanglement

Example: For the Bell state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$:

- Von Neumann Entropy = 1 (maximally entangled)
- Concurrence = 1 (maximally entangled)
- Negativity = 0.5 (maximum value for two qubits)

Three qubits can be entangled in two inequivalent ways

Three-qubit entanglement introduces fundamentally new features not present in two-qubit systems [27]. The most famous example is the GHZ state (named after Greenberger, Horne, and Zeilinger [28]):

$$|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

Unlike two-qubit entanglement, which has only one type of maximal entanglement (equivalent to Bell states), three-qubit systems can exhibit qualitatively different kinds of entanglement. The GHZ state above has the special property that measuring any one qubit immediately determines the state of the other two, but if you lose (trace out) any one qubit, the remaining two qubits are completely unentangled. This is fundamentally different from another type of three-qubit entanglement called the W state:

$$|W\rangle = \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle)$$

which maintains some two-qubit entanglement even after losing one qubit. These distinct classes of entanglement cannot be converted into each other using local operations and classical communication (LOCC).

3.3 Multi-qubit operations

With our exponentially big state space created and entanglement characterized, we can begin to think about how to translate our single qubit operations to multiple qubits and then how to build up operations that *use* multiple qubits. We will focus on two qubit gates since these will help us build up a set of universal gates for quantum computation.

3.3.1 Single-qubit gates

When we want to apply a single-qubit gate to one qubit in a multi-qubit system, we need to use tensor products to construct the appropriate operator. For a two-qubit system, if we want to apply a gate U to the first qubit, the full operator is:

$$U \otimes I$$

where I is the 2×2 identity matrix. Similarly, to apply U to the second qubit, we use:

$$I \otimes U$$

For example, applying the Pauli-X gate to the first qubit of a two-qubit system gives:

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

where the rows and columns correspond to the basis states in order $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. For example, the 1 in the third row, first column means $X \otimes I$ transforms $|00\rangle$ to $|10\rangle$, which flips the first qubit as expected.

Similarly, applying it to the second qubit gives:

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

Here, the 1 in the second row, first column shows $I \otimes X$ transforms $|00\rangle$ to $|01\rangle$, flipping only the second qubit as expected.

This pattern extends to more qubits. For an N-qubit system, to apply U to the kth qubit, we use:

$$\underbrace{I \otimes \cdots \otimes I}_{k-1} \otimes U \otimes \underbrace{I \otimes \cdots \otimes I}_{N-k}$$

where U appears in the kth position and I appears in all other positions. This construction ensures we affect only the target qubit while leaving all other qubits unchanged.

Example: H gate on second qubit of three-qubit system

Let's see how applying H to the second qubit of a three-qubit system works. The operator is:

$$I \otimes H \otimes I$$

Acting on the state $|000\rangle$:

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

As expected, only the middle qubit is put into an equal superposition of 0 and 1.

! Notational convenience for single-qubit operations

To avoid writing long chains of tensor products, we often use a subscript to indicate which qubit an operator acts on. For example, instead of writing

$$I \otimes X \otimes I \otimes I \otimes X$$

for a five-qubit system where we apply X to qubits 2 and 5, we can write this more compactly as

$$X_2 X_5$$

Similarly, applying the Hadamard gate to the third qubit of a four-qubit system would be written as

$$H_3$$

rather than $I \otimes I \otimes H \otimes I$. This notation is particularly helpful when describing quantum circuits involving many qubits.

3.3.2 Two-qubit gates

To begin to build up a full set of logical gates, we start with "controlled" gates. These are gates that will act as single qubit gates on one qubit (the target), but only if another qubit (the control) is in the $|1\rangle$ state.

We first introduce the two-qubit gate called the controlled-NOT (CNOT) gate, which flips the second qubit (target) when the first qubit (control) is in state $|1\rangle$[2].

The CNOT gate can be written as a 4×4 matrix acting on the two-qubit basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$:

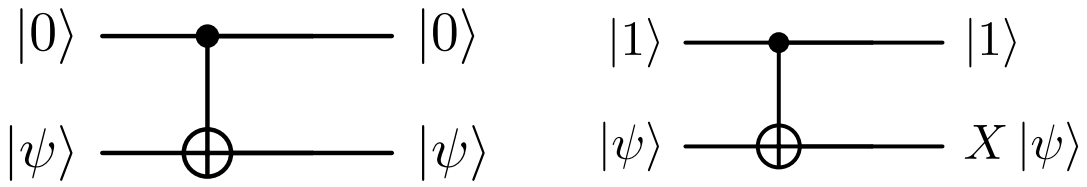
[2]This is also known as the CX-gate (the controlled-X gate).

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

We can understand this matrix by seeing how it acts on each basis state:

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

When the first (control) qubit is $|0\rangle$, the target qubit is unchanged. When the control qubit is $|1\rangle$, the target qubit is flipped (X gate applied). This has a circuit representation which we show below in Fig. 3.1.



(a) CNOT gate with control in $|0\rangle$

(b) CNOT gate with control in $|1\rangle$

Fig. 3.1: The CNOT gate behavior depends on the control qubit state. When the control is $|0\rangle$ (left), the target is unchanged. When the control is $|1\rangle$ (right), the target is flipped.

The CNOT gate's action on computational basis states appears simple - it either leaves them unchanged (when control is $|0$) or flips the target (when control is $|1$). However, when applied to superposition states, the CNOT can create entanglement.

For example, consider applying CNOT to a superposition state created by applying a Hadamard gate to the first qubit (circuit diagram in Fig. 3.2).

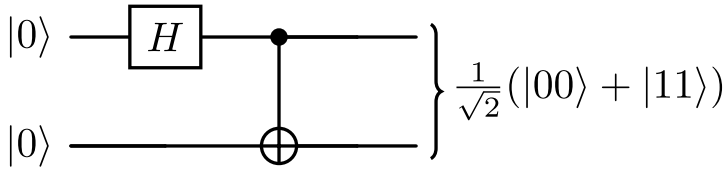


Fig. 3.2: The application of H followed by a CNOT gate can create the Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

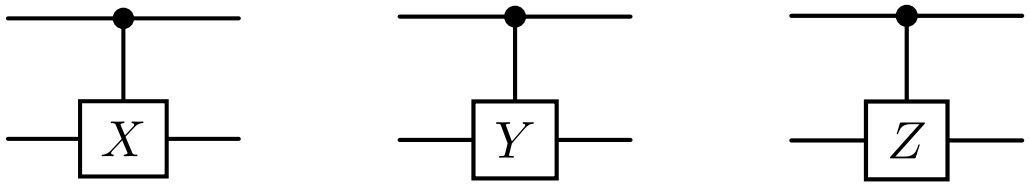
We can verify this mathematically:

$$\begin{aligned} |00\rangle \xrightarrow{H\otimes I} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}} \\ \xrightarrow{\text{CNOT}} \frac{|00\rangle + |11\rangle}{\sqrt{2}} \end{aligned}$$

The resulting state is the Bell state $|\Phi^+\rangle$ which we saw earlier - a maximally entangled state that cannot be written as a product of individual qubit states.

The CNOT gate is just one example of a controlled operation. More generally, we can create controlled versions of any single-qubit gate, where the target operation is applied only when the control qubit is $|1\rangle$. The most common controlled gates are:

- Controlled-X (CNOT): Flips the target qubit if control is $|1\rangle$, Fig. 3.3a
- Controlled-Z (CZ): Adds -1 phase if both qubits are $|1\rangle$, Fig. 3.3c
- Controlled-Y (CY): Applies Y rotation if control is $|1\rangle$, Fig. 3.3b



(a) CX (b) CY (c) CZ

Fig. 3.3: Circuit Notation for controlled-X (CX), controlled-Y (CY), and controlled-Z (CZ) gates. Note that CX=CNOT, so this is just an alternative circuit notation.

Any controlled gate can be constructed using CNOT gates and single-qubit operations. For example, the CZ gate can be implemented as:

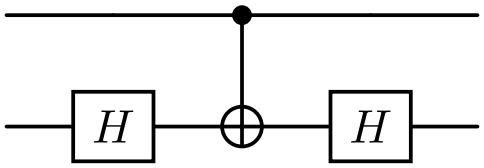


Fig. 3.4: The CZ gate can be decomposed into two Hadamards sandwiching a CNOT (CX) gate.

This notation is flexible as well, we can begin to apply any unitary matrix U onto a target qubit and have a control qubit for it. This generically will look like

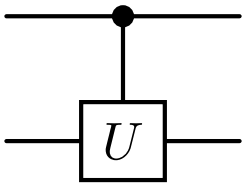


Fig. 3.5: An arbitrary controlled gate

Mathematically, we can write this out in a four-by-four matrix

$$\text{CU} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{pmatrix}$$

where U_{ij} are the matrix elements of the single-qubit unitary $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$.

Another important two-qubit gate is the SWAP gate, which exchanges the states of two qubits. The SWAP gate can be written as a four-by-four matrix:

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

Unlike controlled gates, the SWAP gate cannot create entanglement - it simply swaps the quantum states between the qubits. The circuit notation for a SWAP gate is:

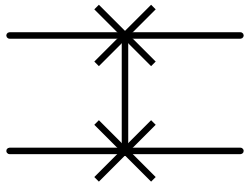


Fig. 3.6: SWAP gate circuit notation

We've seen how this can already create entanglement even though one qubit is merely acting as "control," and how we can "SWAP" two qubits. However, we can also write down more general four-by-four matrices on two qubits and if we have a generic four-by-four matrix U_4, we will write that circuit element as

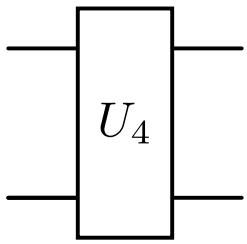


Fig. 3.7: A generic unitary on two qubits can be represented by a block spanning the two qubits

These two-qubit gates form the foundation for quantum computation with multiple qubits. The controlled operations, particularly the CNOT gate, are essential building blocks for quantum algorithms, while general two-qubit unitaries give us the full power to manipulate quantum states in ways impossible with just single-qubit operations. Understanding how these gates create and manipulate entanglement is crucial.

Example: Creating different entangled states

Let's see how different combinations of gates create distinct entangled states:

1. Bell state $|\Phi^+\rangle$:

```
|0 --H--•--|
```

```
          |
|0  -----X--|
```

$$|00\rangle \xrightarrow{H\otimes I} \frac{|00\rangle+|10\rangle}{\sqrt{2}} \xrightarrow{CNOT} \frac{|00\rangle+|11\rangle}{\sqrt{2}}$$

2. Bell state $|\Psi^+\rangle$:

```
|0  --H--•--|
         |
|0  --X--X--|
```

$$|00\rangle \xrightarrow{I\otimes X} |01\rangle \xrightarrow{H\otimes I} \frac{|01\rangle+|11\rangle}{\sqrt{2}} \xrightarrow{CNOT} \frac{|01\rangle+|10\rangle}{\sqrt{2}}$$

This shows how different gate sequences can create different types of entanglement. *Note: We have used the fixed-width font notation for circuit diagrams here.*

3.3.2.1 Universal gate sets

Just as classical computation can be performed using a small set of universal gates (like NAND or NOR), quantum computation can be achieved using a finite set of quantum gates that can approximate any unitary operation to arbitrary precision. This is known as a universal gate set.

A common universal gate set consists of:

1. The CNOT gate
2. Single-qubit rotations (or equivalently, any set of gates that can approximate any single-qubit rotation)

Remarkably, these are sufficient to construct any unitary operation on any number of qubits, though the construction may require many gates. This is analogous to how NAND gates can be used to build any classical logic circuit.

There are several alternative universal gate sets. Some common ones include:

- CZ (or CY) + single-qubit rotations
- CNOT + Hadamard + Phase (S) gate + T gate
- Toffoli + Hadamard

The choice of which universal gate set to use often depends on the physical implementation of the quantum computer. For example, some quantum computing architectures might naturally implement CZ gates rather than CNOT gates, making the CZ-based universal set more practical.

It's worth noting that while we can approximate any unitary to arbitrary precision with these gate sets, the number of gates required might grow exponentially with the desired precision. This is known as the Solovay-Kitaev theorem, which provides an algorithm for finding such approximations.

Example: Creating a GHZ State

A three-qubit GHZ state is

$$|\text{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}.$$

We can create this state using the following sequence of gates:

1. Start from $|000\rangle$
2. Apply a Hadamard gate to the first qubit:

$$|000\rangle \xrightarrow{H\otimes I\otimes I} \frac{|000\rangle + |100\rangle}{\sqrt{2}}$$

3. Apply two consecutive CNOTs, using the first qubit as the control:

$$\frac{|000\rangle + |100\rangle}{\sqrt{2}} \xrightarrow{\text{CNOT on qubits } 1\to 2} \frac{|000\rangle + |110\rangle}{\sqrt{2}}$$
$$\xrightarrow{\text{CNOT on qubits } 1\to 3} \frac{|000\rangle + |111\rangle}{\sqrt{2}}$$

Measuring any qubit in the computational basis will instantly project the entire system into either $|000\rangle$ or $|111\rangle$, illustrating the strong correlations present in multipartite entanglement.

3.3.3 Measurement

When measuring multiple qubits, we need to extend our understanding of single-qubit measurements. For a single qubit, measuring in the computational basis was straightforward - we would get either $|0\rangle$ or $|1\rangle$. However, in a multi-qubit system, measuring just one qubit introduces an important concept: partial measurements and degeneracy.

Consider measuring the first qubit of a two-qubit state in the computational basis. The measurement operators are:

$$P_0 = |0\rangle\langle 0| \otimes I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$P_1 = |1\rangle\langle 1| \otimes I = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

These operators are degenerate - for example, P_0 projects onto both $|00\rangle$ and $|01\rangle$ states. This means when we measure the first qubit and get 0, the second qubit remains in a quantum state.

Example: Partial Measurement

Consider the state:

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

If we measure the first qubit and get 0, the state collapses to:

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$$

The second qubit remains in a superposition!

When measuring multiple qubits, we need to be careful about how we calculate probabilities. For a single qubit, the probability of measuring $|0\rangle$ was simply $|\langle 0|\psi\rangle|^2$. However, with multiple qubits, we need to sum over all possible configurations of the unmeasured qubits.

For example, if we have a two-qubit state $|\psi\rangle$ and measure only the first qubit, the probability of getting 0 is:

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

where i runs over all possible states of the second qubit (0 and 1). This sum accounts for all ways we could get outcome 0 on the first qubit, regardless of what state the second qubit is in.

Example: Calculating Measurement Probabilities

Consider again the state:

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

The probability of measuring 0 on the first qubit is:

$$\begin{aligned} p(0) &= |\langle 00|\psi\rangle|^2 + |\langle 01|\psi\rangle|^2 \\ &= \left|\frac{1}{2}\right|^2 + \left|\frac{1}{2}\right|^2 \\ &= \frac{1}{2} \end{aligned}$$

This matches our intuition - the first qubit is equally likely to be 0 or 1.

3.3.3.1 Measuring General Observables

In practice, we may be restricted in what we can measure on a quantum computer, such as only in the computational (Z) basis. However, we often need to measure other observables. For example, we might want to measure the parity of two qubits (Z_1Z_2) or their correlation (X_1X_2).

To measure these observables, we need to transform our state before measurement. This is done by applying appropriate unitary operations that map our desired measurement basis to the computational basis.

For example, as we saw in the single qubit section, to measure in the X basis, we first apply H before measuring in the computational basis.

For two-qubit observables like Z_1Z_2, we can use controlled operations to map the eigenspaces to computational basis states. Here's how we might measure Z_1Z_2.

First, let's write out how Z_1Z_2 acts on the computational basis, separting out the +1 eigenvalues from the -1 eigenvalues

State	Z_1Z_2	Z_2
$\|00\rangle$	$+1$	$+1$
$\|01\rangle$	-1	-1
$\|10\rangle$	-1	$+1$
$\|11\rangle$	$+1$	-1

Notice that these operators have the same number of +1 and -1 eigenvalues. On top of that, the states $|00\rangle$ and $|01\rangle$ have the same eigenvalue, and $|10\rangle$ and $|11\rangle$ flips. This sounds like a CNOT gate, in fact if we have a unitary that takes $|10\rangle \mapsto |11\rangle$ and $|11\rangle \mapsto |10\rangle$, while leaving $|00\rangle$ and $|01\rangle$ alone, these operators match.

Let's build out a way to relate these operators which we will then use to measure *any* string Pauli matrices. With the above, we note that

$$\begin{aligned} |00\rangle &\mapsto |00\rangle\,, \\ |01\rangle &\mapsto |01\rangle\,, \\ |10\rangle &\mapsto |11\rangle\,, \\ |11\rangle &\mapsto |10\rangle \end{aligned}$$

will let us map $Z_1Z_2 \mapsto Z_2$. This has a simple matrix form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

In other words, the CNOT gate. Formally, this means that

$$\mathrm{CNOT}\, Z_1 Z_2\, \mathrm{CNOT} = Z_2, \tag{3.3}$$

which can be verified with matrix mulitplation. However, we can also note that if we apply $|b_1 b_2\rangle$ to both sides,

$$Z_1Z_2\mathrm{CNOT}\,|b_1b_2\rangle = \mathrm{CNOT}(-1)^{b_2}\,|b_1b_2\rangle = (-1)^{b_2}\mathrm{CNOT}\,|b_1b_2\rangle\,,$$

which verifies that $\mathrm{CNOT}\,|b_1b_2\rangle$ is an eigenvector of Z_1Z_2 with eigenvalue $(-1)^{b_2}$ (this assume Eq. 3.3 is correct).

Therefore, to *measure* Z_1Z_2, we perform CNOT, then measure the target qubit (number two). The circuit diagram looks like this

and if we want to not only measure Z_1Z_2 but ensure the system goes into an eigenstate of Z_1Z_2, we need to perform CNOT again

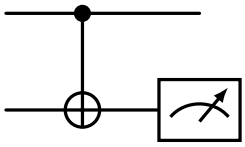
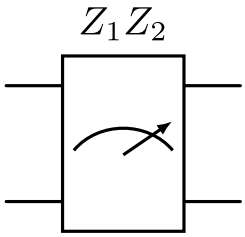
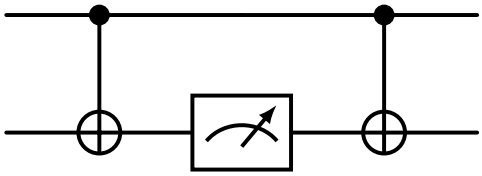


Fig. 3.8: Circuit to measure Z_1Z_2

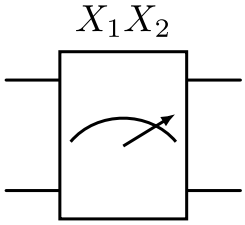


(a) Circuit to measure Z_1Z_2 with a single symbol

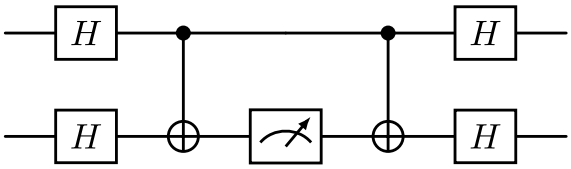


(b) Circuit to measure Z_1Z_2 with explicit gates

Fig. 3.9: Circuit diagrams for the Z_1Z_2 measurement. These two circuits are equivalent to each other.



(a) Circuit to measure X_1X_2 with a single symbol



(b) Circuit to measure X_1X_2 with explicit gates

Fig. 3.10: Circuit diagrams for the X_1X_2 measurement. These two circuits are equivalent to each other.

But what about measuring other Pauli strings? We can use the fact that $X = HZH$ and $Y = SXS^\dagger$ to convert any Pauli string measurement into a Z-type measurement. For example, to measure X_1X_2, we can apply Hadamard gates to both qubits, measure Z_1Z_2, and then apply Hadamard gates again:

This gives us a way to measure any Pauli string

1. Perform single-particle unitaries to convert it to only Z and I operators.
2. Find the permutation matrix that relates +1 and -1 eigenvalues of your Z-Pauli string to a single Z_n and apply that unitary matrix (this could be a complicated combination of CNOT gates)
3. Measure Z_n.
4. If the correct measured state is needed, undo the unitary in #2 followed by the single-particle unitaries in #1.

Pauli strings are particularly "simple." Measuring more complicated observables requires more thought and sometimes ancillary systems to assist in that measurement.

Example: Measuring X on One Qubit of a Two-Qubit System

Suppose we have a two-qubit state:

$$|\phi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle,$$

and we want to measure the first qubit in the X basis while leaving the second qubit unmeasured.

1. Recall $HXH = Z$. So to measure X on the first qubit, apply a Hadamard on that qubit to map the X basis to the Z basis:
$$(H \otimes I)|\phi\rangle.$$
2. Measure the first qubit in the computational basis (effectively measuring Z on the transformed state).
3. After the measurement, unapply the H if you want to restore the original basis of the first qubit.

This procedure effectively measures X_1 while leaving qubit 2 untouched (can it still be entangled?).

Example: Creating entanglement through measurement

Let's see how we can create entanglement through measurement. We'll start with the product state $|{+}{+}\rangle$ and use a Z_1Z_2 measurement to create a maximally entangled state.

1. Initial state:
$$|{+}{+}\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$
2. Measuring Z_1Z_2 projects onto the +1 or -1 eigenspaces:
   - +1 eigenspace: span$\{|00\rangle, |11\rangle\}$
   - -1 eigenspace: span$\{|01\rangle, |10\rangle\}$
3. After measurement:

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

4. If we get -1, we can apply X to the second qubit to transform to $|\Psi^+\rangle$:
$$(I \otimes X)\frac{|01\rangle + |10\rangle}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

This procedure creates the Bell state $|\Phi^+\rangle$ regardless of measurement outcome, demonstrating how measurement plus conditional corrections can generate entanglement from separable states.

3.4 Decoherence in multi-qubit systems

When dealing with multiple qubits, decoherence becomes even more challenging than in single-qubit systems. Not only can each qubit experience individual decoherence, but the interactions between qubits can create new pathways for errors. The main types of multi-qubit decoherence are:

1. **Independent decoherence**: Each qubit experiences its own local noise
2. **Correlated decoherence**: Environmental effects that simultaneously affect multiple qubits
3. **Cross-talk**: Unwanted interactions between qubits that should be isolated

For example, consider a two-qubit state that starts in a Bell state:

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

The density matrix for this pure state is:

$$|\Phi^+\rangle\langle\Phi^+| = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Under independent dephasing, each qubit loses phase coherence separately:

$$\rho(t) = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & e^{-2\gamma t} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ e^{-2\gamma t} & 0 & 0 & 1 \end{pmatrix}$$

which would represent exponential decay in entanglement over time; this could occur even with pure local noise. This highlights a crucial feature of quantum systems: even when noise acts independently on each qubit (i.e., local noise), it can destroy global quantum properties like entanglement. In other words, we don't need correlated noise to degrade entanglement - local noise channels are sufficient to compromise the quantum advantages that entanglement provides.

Example: Impact of different noise types

Consider a Bell state under different noise channels:

1. Amplitude damping on first qubit only:

$$\rho(t) = \begin{pmatrix} 1 - \frac{e^{-\gamma t}}{2} & 0 & 0 & \frac{e^{-\gamma t/2}}{\sqrt{2}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{e^{-\gamma t/2}}{\sqrt{2}} & 0 & 0 & \frac{e^{-\gamma t}}{2} \end{pmatrix}$$

2. Dephasing on both qubits:

$$\rho(t) = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & e^{-2\gamma t} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ e^{-2\gamma t} & 0 & 0 & 1 \end{pmatrix}$$

This shows how different noise channels affect the quantum correlations in distinct ways.

3.4.1 Gate Errors

In addition to decoherence during idle times, errors can occur during gate operations. These gate errors come in several forms:

1. **Systematic errors**: Consistent over/under-rotation of gates
2. **Random errors**: Fluctuations in gate parameters
3. **Cross-talk errors**: Gates affecting neighboring qubits
4. **Leakage errors**: System leaving the computational basis

For two-qubit gates like CNOT, errors are typically higher than single-qubit gates because:

- They require stronger interactions between qubits
- Take longer to implement
- Are more sensitive to timing and control errors (e.g.,some qubits have to wait while single-qubit operations "catch up").

Current State of Gate Fidelities

As of 2021, record error rates are:

- Single-qubit gates: 0.03%
- Two-qubit gates: 0.5%
- Measurement: 0.2%

These numbers are representative of state-of-the-art superconducting qubit platforms, as reported in [29].

4 Quantum Computing Algorithms

We now have a solid foundation in the building blocks of quantum computing. We understand how individual qubits work - their quantum states, how they evolve under unitary operations, and how we measure them. We've also explored how multiple qubits interact and become entangled, creating quantum correlations that have no classical analog.

With these fundamental tools in hand, we can now explore what makes quantum computing truly powerful - the algorithms that harness quantum mechanics to solve problems more efficiently than classical computers. These algorithms cleverly combine quantum superposition, interference, and entanglement to achieve computational speedups that would be impossible using classical bits alone.

The key insight is that quantum algorithms can process multiple computational paths simultaneously through superposition, while entanglement allows us to correlate these paths in useful ways. When we finally measure the system, quantum interference helps guide us toward the solution we seek. In the following sections, we'll examine several landmark quantum algorithms that demonstrate these principles and show how quantum computers can tackle important practical problems.

> **!** A Key Insight About Quantum Computing
>
> Scott Aaronson famously wrote on his blog: "If you take nothing else from this blog: quantum computers won't solve hard problems instantly by just trying all solutions in parallel."
> This gets at a crucial misconception about quantum computing. While quantum computers can indeed explore multiple computational paths simultaneously through superposition, we can't simply read out all those paths at the end. When we measure a quantum system, it collapses to just one classical outcome. The art of quantum algorithm design lies in cleverly using interference to amplify the paths leading to the solution we want, while suppressing paths leading to incorrect answers. This is why quantum algorithms require careful construction - we need to choreograph the quantum evolution so that when we finally measure, we're likely to get useful information.

4.1 Deutsch's Algorithm

Deutsch's algorithm, published in 1985 [10], represents the first quantum algorithm to demonstrate a speedup over classical computation. While the problem it solves is quite simple, it beautifully illustrates the key principles of quantum computation: superposition, interference, and parallelism.

4.1.1 The Problem Statement

Consider a black box (oracle) function $f : \{0, 1\} \to \{0, 1\}$ that takes a single bit as input and returns a single bit as output. There are only four possible such functions:

1. $f_1(x) = 0$ for all x (constant function)
2. $f_2(x) = 1$ for all x (constant function)
3. $f_3(x) = x$ (balanced function)
4. $f_4(x) = \text{NOT}(x)$ (balanced function)

The task is to determine whether f is constant (same output for all inputs) or balanced (equal number of 0s and 1s in output). Classically, this requires evaluating $f(0)$ and $f(1)$ - two function calls. Deutsch's algorithm solves this problem with a single quantum query.

What is an Oracle?

An oracle is a black box function that implements a specific operation without revealing its internal workings. In quantum computing, an oracle is typically provided as a unitary transformation that can be queried but whose implementation details are hidden. Oracles are crucial in algorithm analysis because they let us:

1. Focus on the number of queries needed rather than implementation details
2. Prove lower bounds on algorithmic complexity
3. Compare classical and quantum approaches on an equal footing

For example, in Deutsch's algorithm, we don't care how $f(x)$ is computed internally - we just care that we can access it as a quantum operation U_f that preserves quantum superposition.

4.1.2 Quantum Implementation

The Power of Hadamard Gates

The Hadamard transform is crucial in this algorithm because:

1. Initial H-gates create a superposition of all possible bit strings:

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

2. Final H-gates convert phase differences into bit values:
   - If a bit in s is 0: H maps $|0\rangle \to |0\rangle$
   - If a bit in s is 1: H maps $-|0\rangle \to |1\rangle$

This is why measuring after the final Hadamard gates directly reveals the hidden string s!

The quantum circuit requires two qubits:

- An input qubit initialized to $|0\rangle$
- An auxiliary qubit initialized to $|1\rangle$

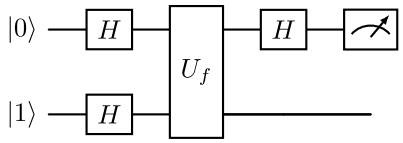


Fig. 4.1: Circuit for Deutsch algorithm

The algorithm proceeds as follows:

1. Apply Hadamard gates to both qubits:

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

2. Apply the quantum oracle U_f, which implements:

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

   where $\oplus$ denotes addition modulo 2.

3. Apply another Hadamard gate to the first qubit.

4. Measure the first qubit.

The interference in Deutsch's algorithm occurs in the final Hadamard transformation. To see this explicitly, let's track how the phases accumulate:

! Phase Kickback

Phase kickback is a phenomenon in quantum computing where the phase shift caused by an operation on one qubit (often called the target qubit) is "kicked back" onto another qubit (often called the control qubit). This happens particularly in controlled operations when the target qubit is in an eigenstate of the operation.
In Deutsch's algorithm, we have four possibilities in the code basis

1. $|0\rangle |y\rangle \mapsto |0\rangle |y+0\rangle$ and $|1\rangle |y\rangle \mapsto |1\rangle |y+0\rangle$. In this case, nothing changes and

$$|+\rangle |-\rangle \mapsto |+\rangle |-\rangle .$$

2. $|0\rangle |y\rangle \mapsto |0\rangle |y+1\rangle$ and $|1\rangle |y\rangle \mapsto |1\rangle |y+1\rangle$. In this case, both basis states get a phase kickback, and

$$|+\rangle |-\rangle \mapsto -|+\rangle |-\rangle .$$

   Yet an overall phase does not affect the state.

3. $|0\rangle |y\rangle \mapsto |0\rangle |y+0\rangle$ and $|1\rangle |y\rangle \mapsto |1\rangle |y+1\rangle$. In this case, only the $|1\rangle$ basis state gets a phase kickback, and

$$|+\rangle |-\rangle \mapsto |0\rangle |-\rangle - |1\rangle |-\rangle = |-\rangle |-\rangle .$$

4. $|0\rangle |y\rangle \mapsto |0\rangle |y+1\rangle$ and $|1\rangle |y\rangle \mapsto |1\rangle |y+0\rangle$. In this case, only the $|0\rangle$ basis state gets a phase kickback, and

$$|+\rangle |-\rangle \mapsto -|0\rangle |-\rangle + |1\rangle |-\rangle = -|-\rangle |-\rangle .$$

Even though the extra qubit is not changed by the oracle, it provides an important function with its -1 phase, allowing it to change the phase of other qubits from $|+\rangle$ to $|-\rangle$ in a strategic way to allow interference.
Generically, when U_f acts on $|x\rangle|-\rangle$, it transforms the state as follows:

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

The phase factor $(-1)^{f(x)}$ appears on the input qubit $|x\rangle$. This phase encodes whether $f(x)$ is 0 or 1. This is how the global properties of f are transferred to the first qubit.

After the oracle U_f acts on the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, the first qubit (ignoring the auxiliary qubit, which remains in the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$), becomes:

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

The key here is that the function values $f(0)$ and $f(1)$ are now encoded as *phases* in front of the basis states $|0\rangle$ and $|1\rangle$ (this is **phase kickback**, see above callout box). Let's consider the two possibilities for $f(x)$: constant or balanced.

- **Constant Functions:** If $f(x)$ is constant, then $f(0) = f(1)$. The state is then either $\pm\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$, depending on whether $f(x) = 0$ or $f(x) = 1$.
- **Balanced Functions:** If $f(x)$ is balanced, then $f(0) \neq f(1)$. This means one of the function values is 0 and the other is 1. The state is then either $\pm\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

Now we apply the final Hadamard gate to the first qubit. Recall that H transforms the basis states as:

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

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

Applying the Hadamard gate H to the first qubit transforms the state based on whether $f(x)$ is constant or balanced:

- **Constant Functions:** If $f(0) = f(1) = c$, the state before the Hadamard gate is $\frac{(-1)^c}{\sqrt{2}}(|0\rangle + |1\rangle)$. Applying H yields:

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

  Therefore, we *always* measure $|0\rangle$.

- **Balanced Functions:** If $f(0) \neq f(1)$, the state before the Hadamard gate is $\pm\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Applying H yields:

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

  Therefore, we *always* measure $|1\rangle$.

The final Hadamard gate creates an interference effect, channeling the global property of f (constant or balanced) into a single, easily measurable property of the first qubit.

In summary, if f is constant, the final state of the first qubit will be $|0\rangle$. If f is balanced, the final state will be $|1\rangle$.

Example: Walking through Deutsch's Algorithm

Let's analyze how the algorithm behaves for each possible function, focusing on the state of the *first qubit* after the oracle and after the final Hadamard gate.

1. **For $f_1(x) = 0$ (constant):**
   - Initial state (first qubit): $|+\rangle$
   - State after oracle (first qubit): $\frac{1}{\sqrt{2}}[(-1)^{f_1(0)}|0\rangle + (-1)^{f_1(1)}|1\rangle] = |+\rangle$
   - Apply final H gate: $H|+\rangle = |0\rangle$
   - Measurement: Always $|0\rangle$ ⇒ Constant
2. **For $f_2(x) = 1$ (constant):**
   - Initial state (first qubit): $|+\rangle$
   - State after oracle (first qubit): $\frac{1}{\sqrt{2}}[(-1)^{f_2(0)}|0\rangle + (-1)^{f_2(1)}|1\rangle] = -|+\rangle$
   - Apply final H gate: $H(-|+\rangle) = -|0\rangle$
   - Measurement: Always $|0\rangle$ ⇒ Constant
3. **For $f_3(x) = x$ (balanced):**
   - Initial state (first qubit): $|+\rangle$
   - State after oracle (first qubit): $\frac{1}{\sqrt{2}}[(-1)^{f_3(0)}|0\rangle + (-1)^{f_3(1)}|1\rangle] = |-\rangle$
   - Apply final H gate: $H|-\rangle = |1\rangle$
   - Measurement: Always $|1\rangle$ ⇒ Balanced

4.1.3 Key Insights

Several quantum principles make this speedup possible:

1. **Superposition**: The initial Hadamard transforms create a superposition that allows us to evaluate $f(0)$ and $f(1)$ simultaneously.
2. **Phase Kickback**: The auxiliary qubit, initialized to $|1\rangle$ and transformed by Hadamard, enables the function's output to be encoded in the phase of the first qubit.
3. **Interference**: The final Hadamard transform converts the phase difference into a measurable bit value.

Historical Context

Deutsch's algorithm was the first to demonstrate that quantum computers could solve certain problems faster than classical computers. While the problem itself is artificial, the techniques it introduced - particularly the use of quantum parallelism and interference - became fundamental building blocks for more practical quantum algorithms.

Understanding Balanced vs Constant Functions

Consider a single-bit function $f(x)$. There are only four possible functions:
Constant Functions:

- $f_1(x) = 0$ for all x
- $f_2(x) = 1$ for all x

Balanced Functions:

- $f_3(x) = x$ (identity)
- $f_4(x) = \text{NOT}(x)$

The quantum algorithm can distinguish between these cases with just one query, while classically we need two queries to be certain!

4.1.4 Mathematical Details

Let's examine how the state evolves through the circuit. After the initial Hadamard gates, we have:

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

After applying U_f, the state becomes:

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

The final Hadamard on the first qubit gives:

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

For constant functions, $f(0) = f(1)$, leading to a measurement of $|0\rangle$. For balanced functions, $f(0) \neq f(1)$, leading to a measurement of $|1\rangle$.

This elegant algorithm demonstrates how quantum interference can extract global properties of a function (constant vs. balanced) with fewer queries than classically possible.

4.2 Deutsch–Jozsa Algorithm

The Deutsch-Jozsa algorithm, published in 1992 [11], generalizes Deutsch's algorithm to handle functions with n-bit inputs. It provides one of the first examples of an exponential speedup over classical deterministic algorithms, though for a somewhat artificial problem.

4.2.1 The Problem Statement

Consider a black box function $f : \{0,1\}^n \to \{0,1\}$ that takes n bits as input and returns a single bit. We are promised that f is either:

- Constant: returns the same value (0 or 1) for all inputs
- Balanced: returns 0 for exactly half of the inputs and 1 for the other half

The task is to determine whether f is constant or balanced.

> **i** Classical vs. Quantum Complexity
>
> Classically, in the worst case, we need to evaluate $f(x)$ for $2^{n-1}+1$ different inputs to be certain of the answer. This is because we might need to check more than half the possible inputs before finding two different outputs. The Deutsch-Jozsa algorithm solves this problem with just one quantum query!

4.2.2 Quantum Implementation

The quantum circuit requires:

- n qubits for the input register, initialized to $|0\rangle^{\otimes n}$
- One auxiliary qubit initialized to $|1\rangle$

The Deutsch-Jozsa algorithm can be summarized as follows:

1. Apply Hadamard gates to all input qubits and the auxiliary qubit.
2. Apply the quantum oracle U_f.
3. Apply Hadamard gates to the input qubits.
4. Measure the input qubits.

To understand the algorithm more deeply, let's analyze each step mathematically.

4.2.3 Mathematical Analysis

1. **Initialization and Hadamard Transform:**

   We begin with the state $|0\rangle^{\otimes n}|1\rangle$. Applying Hadamard gates to all $n+1$ qubits transforms the state as follows:

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

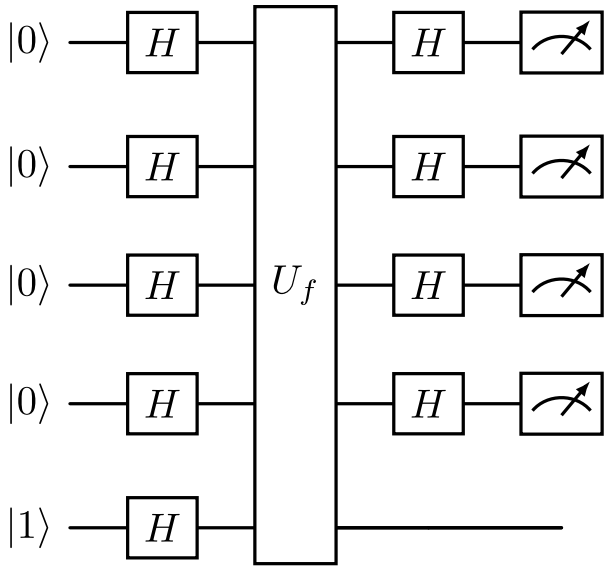


Fig. 4.2: The implementation of the Deutsch-Jozsa Algorithm for four bits.

2. **Application of the Quantum Oracle:**

   The quantum oracle U_f is applied, which implements the transformation:

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

   This transforms the state to:

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

   where $\overline{f(x)}$ represents the complement of $f(x)$. This can be rewritten via phase kickback as:

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

3. **Final Hadamard Transform:**

   Hadamard gates are applied to the n input qubits. To see the effect, we apply the Hadamard transform to each qubit. Recall that $H|x\rangle = \frac{1}{\sqrt{2}} \sum_{z \in \{0,1\}} (-1)^{x \cdot z} |z\rangle$. Applying this to all n qubits, we get:

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

   where $x \cdot z = x_1 z_1 \oplus x_2 z_2 \oplus \cdots \oplus x_n z_n$. Therefore, the state becomes:

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

Rearranging the summation, we have:

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

Constant Function Case:

If $f(x)$ is a constant function, meaning $f(x) = c$ for all x, the state simplifies to:

$$\pm |0\rangle^{\otimes n} \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

The sign depends on the value of c. Measuring the first n qubits will yield $|0\rangle^{\otimes n}$ with certainty.

Balanced Function Case:

If $f(x)$ is a balanced function, the interference is such that the amplitude of the $|0\rangle^{\otimes n}$ state becomes zero. Therefore, when measuring the first n qubits, the probability of obtaining $|0\rangle^{\otimes n}$ is 0. This means we are guaranteed to measure a state other than $|0\rangle^{\otimes n}$.

4. **Measurement:**

   Finally, we measure the n input qubits. If the measurement result is $|0\rangle^{\otimes n}$, the function is constant. Otherwise, if any of the qubits are measured to be $|1\rangle$, the function is balanced.

> **Superposition from Product of Plus States**
>
> Applying a Hadamard gate to each qubit in the $|0\rangle^{\otimes n}$ state creates a superposition of all possible n-bit strings. This is because:
>
> $$\begin{aligned} H^{\otimes n} |0\rangle^{\otimes n} &= H|0\rangle \otimes H|0\rangle \otimes \cdots \otimes H|0\rangle \\ &= |+\rangle^{\otimes n} \\ &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \end{aligned}$$
>
> Each qubit is in an equal superposition of $|0\rangle$ and $|1\rangle$, and since the qubits are independent, the overall state is a superposition of all 2^n possible combinations of $|0\rangle$ and $|1\rangle$.
> For $n = 1$:
>
> $$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \sum_{x \in \{0,1\}} |x\rangle$$
>
> For $n = 2$:

$$
\begin{aligned}
H^{\otimes 2}|00\rangle &= H|0\rangle \otimes H|0\rangle \\
&= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\
&= \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \\
&= \frac{1}{2} \sum_{x \in \{0,1\}^2} |x\rangle
\end{aligned}
$$

For $n = 3$:

$$
\begin{aligned}
H^{\otimes 3}|000\rangle &= H|0\rangle \otimes H|0\rangle \otimes H|0\rangle \\
&= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\
&= \frac{|000\rangle + |001\rangle + |010\rangle + |011\rangle}{2\sqrt{2}} \\
&\quad + \frac{|100\rangle + |101\rangle + |110\rangle + |111\rangle}{2\sqrt{2}} \\
&= \frac{1}{2\sqrt{2}} \sum_{x \in \{0,1\}^3} |x\rangle
\end{aligned}
$$

The amplitude of the $|0\rangle^{\otimes n}$ state after the final Hadamard transform is given by:

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

This sum evaluates to:

- ± 1 if $f(x)$ is a constant function.
- 0 if $f(x)$ is a balanced function.

If the sum is ± 1, this means that the amplitude of the $|0\rangle^{\otimes n}$ state is non-zero, and when we measure the qubits, we will obtain the $|0\rangle^{\otimes n}$ state with certainty (or its opposite, depending on the sign). Conversely, if the sum is 0, the amplitude of the $|0\rangle^{\otimes n}$ state is zero, meaning that when we measure the qubits, we are guaranteed *not* to obtain the $|0\rangle^{\otimes n}$ state. This implies that at least one of the qubits must be in the $|1\rangle$ state.

! Illustrating Quantum Parallelism

The unitary operator U_f performs:

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

When applied to a superposition of all possible inputs:

$$
\frac{1}{\sqrt{2^n}} \sum_x |x\rangle |-\rangle \to \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle |-\rangle
$$

This single operation effectively computes $f(x)$ for all 2^n inputs simultaneously! This is a key feature of quantum computation.

4.2.4 Key Insights

1. **Quantum Parallelism**: The algorithm processes all possible inputs simultaneously through superposition.
2. **Interference**: The final Hadamard transforms create interference patterns that distinguish between constant and balanced functions.
3. **Promise Problem**: The algorithm's exponential speedup relies crucially on the promise that f is either constant or balanced. Without this promise, the quantum advantage disappears.

i Example with n=2

For n=2, the classical algorithm needs to evaluate $f(00)$, $f(01)$, and possibly $f(10)$ to determine if f is constant or balanced. The quantum algorithm still requires just one query regardless of n. Consider a balanced function $f(00) = f(01) = 0$ and $f(10) = f(11) = 1$:

1. Initial state after Hadamards: $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)(|0\rangle - |1\rangle)$
2. After oracle: $\frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)(|0\rangle - |1\rangle)$
3. Final state after Hadamards: $|\psi\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ where $|\psi\rangle$ has zero amplitude for $|00\rangle$:

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

$$H^{\otimes 2}|\psi\rangle = |10\rangle$$

The Deutsch-Jozsa algorithm, while solving an artificial problem, demonstrates the potential power of quantum computation and introduces techniques that appear in many more practical quantum algorithms.

4.3 Bernstein–Vazirani Algorithm

Building upon the Deutsch and Deutsch-Jozsa algorithms, the Bernstein-Vazirani algorithm [12] tackles a different problem: instead of determining a global property of a function (like whether it's constant or balanced), it aims to learn a hidden string encoded within the function's behavior. While the Deutsch-Jozsa algorithm distinguishes between constant and balanced functions, the Bernstein-Vazirani algorithm identifies a specific string s that defines the function $f(x) = s \cdot x \pmod 2$. This algorithm showcases how quantum computation can be used to extract specific information from a function with just a single query, offering an exponential speedup compared to classical approaches.

4.3.1 Problem Statement

The Bernstein-Vazirani algorithm is designed to solve the following problem:

Given a hidden string $s \in \{0,1\}^n$, and a function $f : \{0,1\}^n \to \{0,1\}$ such that $f(x) = s \cdot x \pmod 2$, where $s \cdot x$ is the bitwise dot product of s and x, the goal is to find the hidden string s.

Classically, determining s requires n queries to the function f. The Bernstein-Vazirani algorithm can find s with just one quantum query.

4.3.2 Quantum Implementation

The quantum circuit consists of:

- n input qubits initialized to $|0\rangle^{\otimes n}$
- One auxiliary qubit initialized to $|1\rangle$

The algorithm proceeds as follows:

1. Apply Hadamard gates to all n input qubits and the auxiliary qubit.
2. Apply the quantum oracle U_f, defined as $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$.
3. Apply Hadamard gates to the n input qubits.
4. Measure the n input qubits. The measurement outcome will be the hidden string s.

4.3.3 Step-by-step state evolution with Hadamard gates

Let's break down how the quantum state evolves as Hadamard gates are applied. This will give you a clearer picture of how the algorithm works.

1. **Initialization:** We start with n qubits in the $|0\rangle$ state and one auxiliary qubit in the $|1\rangle$ state:

$$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle = |00...0\rangle \otimes |1\rangle$$

2. **Applying Hadamards:** We apply Hadamard gates to all $n+1$ qubits. Recall that the Hadamard gate transforms $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Applying $H^{\otimes n}$ to the first n qubits transforms $|0\rangle^{\otimes n}$ into an equal superposition of all possible n-bit strings:

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

Applying a Hadamard gate to the auxiliary qubit transforms $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Therefore, after applying the Hadamard gates, the state becomes:

$$\begin{aligned} |\psi_1\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x \in \{0,1\}^n} |x\rangle (|0\rangle - |1\rangle) \end{aligned}$$

3. **Applying the Quantum Oracle:** The quantum oracle U_f acts as $|x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$. Applying U_f to $|\psi_1\rangle$ gives:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle |f(x) \oplus 1\rangle$$

Since $f(x) = s \cdot x \pmod 2$, we can rewrite this as:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} |x\rangle |s \cdot x \oplus 1\rangle$$

We can rewrite $|s \cdot x \oplus 1\rangle$ as $(-1)^{s\cdot x}|1\rangle$, so the state becomes:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (-1)^{s\cdot x} |x\rangle (|0\rangle - |1\rangle) \tag{4.1}$$

4. **Applying Hadamards Again:** We apply Hadamard gates to the first n qubits again. Let's consider what happens if we apply $H^{\otimes n}$ to the state $|s\rangle$.

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

Notice that this is Eq. 4.1, and since $H^2 = I$, we can invert this to see that

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

5. **Measurement:** Measuring the first n qubits yields the hidden string s with certainty. The auxiliary qubit is left in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, which we can ignore after the measurement.

In summary, the algorithm cleverly uses Hadamard gates and the quantum oracle to transform the initial state into a state where measuring the first n qubits directly reveals the hidden string s. The key is the interference caused by the Hadamard gates and the phase kickback from the oracle, which amplifies the amplitude of the state corresponding to the hidden string.

i Classical vs. Quantum Speed

The Bernstein-Vazirani algorithm showcases a significant speedup *for this specific problem* compared to its classical counterpart. Classically, determining the hidden string s would require n queries to the function $f(x)$, while the quantum algorithm achieves the same result with just a single query to the quantum oracle U_f. This demonstrates an exponential speedup highlighting the power of quantum computation for specific tasks.

Bernstein-Vazirani Algorithm Example (n=2, s=10)

Let's consider the Bernstein-Vazirani algorithm for $n = 2$ with the hidden string $s = 10$. The function $f(x)$ computes the dot product of x and s modulo 2: $f(x) = (s \cdot x) \pmod 2$. In our case, $f(x) = (10 \cdot x) \pmod 2$.

1. Initialization: We begin with a three-qubit state, with the first two qubits in state $|0\rangle$ and the auxiliary qubit in state $|1\rangle$:

$$|\psi_0\rangle = |001\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle$$

2. Apply Hadamard Gates: Apply Hadamard gates to each qubit. Recall that $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Applying $H^{\otimes 3}$ to $|\psi_0\rangle$:

$$\begin{aligned}
|\psi_1\rangle &= H^{\otimes 3}|001\rangle \\
&= (H \otimes H \otimes H)(|0\rangle \otimes |0\rangle \otimes |1\rangle) \\
&= (H|0\rangle) \otimes (H|0\rangle) \otimes (H|1\rangle) \\
&= \left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right] \otimes \left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right] \otimes \left[\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right] \\
&= \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\
&= \frac{1}{\sqrt{8}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)(|0\rangle - |1\rangle)
\end{aligned}$$

3. Apply Quantum Oracle (Uf): Apply the quantum oracle U_f which acts as $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$. We use phase kickback by initializing the auxiliary qubit to $|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Thus, the oracle effectively applies a phase $(-1)^{f(x)}$ to the computational basis states of the input qubits: $|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$. For $s = 10$, we have: $f(00) = (10) \cdot (00) = 0$ $f(01) = (10) \cdot (01) = 0$ $f(10) = (10) \cdot (10) = 1$ $f(11) = (10) \cdot (11) = 1$ Applying the oracle U_f to $|\psi_1\rangle$ and utilizing phase kickback, we get:

$$\begin{aligned}
|\psi_2\rangle &= U_f|\psi_1\rangle \\
&= \frac{1}{\sqrt{8}}\left[(-1)^{f(00)}|00\rangle + (-1)^{f(01)}|01\rangle + (-1)^{f(10)}|10\rangle + (-1)^{f(11)}|11\rangle\right](|0\rangle - |1\rangle) \\
&= \frac{1}{\sqrt{8}}\left[|00\rangle + |01\rangle - |10\rangle - |11\rangle\right](|0\rangle - |1\rangle)
\end{aligned}$$

4. Apply Hadamard Gates Again: Apply Hadamard gates to the first two qubits (input qubits) only, $H^{\otimes 2} \otimes I$ to $|\psi_2\rangle$:

$$\begin{aligned}
|\psi_3\rangle &= (H^{\otimes 2} \otimes I)|\psi_2\rangle \\
&= (H \otimes H \otimes I)\left[\frac{1}{\sqrt{8}}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)(|0\rangle - |1\rangle)\right] \\
&= |10\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)
\end{aligned}$$

5. Measurement: Measuring the first two qubits in the computational basis will yield the state $|10\rangle$ with probability 1. This directly reveals the hidden string $s = 10$. The auxiliary qubit is left in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$, but we are only interested in the measurement outcome of the first two qubits.

4.3.4 Connection to Deutsch-Jozsa Algorithm

The Bernstein-Vazirani algorithm can be seen as a generalization of the Deutsch-Jozsa algorithm. While Deutsch-Jozsa distinguishes between constant and balanced functions, Bernstein-Vazirani identifies a specific hidden string. If we consider the Deutsch-Jozsa problem as determining whether the dot product $s \cdot x$ is always 0 (constant function) or not (balanced function), Bernstein-Vazirani extends this to find the specific string s that defines this dot product. Both algorithms leverage the power of quantum parallelism and interference to achieve an exponential speedup over classical algorithms for specific problems related to function evaluation.

4.4 Simon's Algorithm

Simon's algorithm, introduced by Daniel Simon in 1994 [13], solves a problem that exhibits an exponential speedup over the best-known classical algorithm. It distinguishes itself as one of the early quantum algorithms demonstrating a significant advantage, albeit for a specific problem.

4.4.1 Problem Statement

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$ with the following promise:

- There exists a secret string $s \in \{0,1\}^n$ such that for all $x, y \in \{0,1\}^n$, $f(x) = f(y)$ if and only if $x \oplus y = 0^n$ or $x \oplus y = s$, where $\oplus$ denotes bitwise XOR.

In simpler terms, f is two-to-one, and it maps x and $x \oplus s$ to the same value. The goal is to find the hidden string s.

Classically, solving this problem requires $O(2^{n/2})$ queries to f using a collision-finding algorithm. Simon's algorithm solves it in *polynomial time* with $O(n)$ queries.

4.4.2 Quantum Implementation

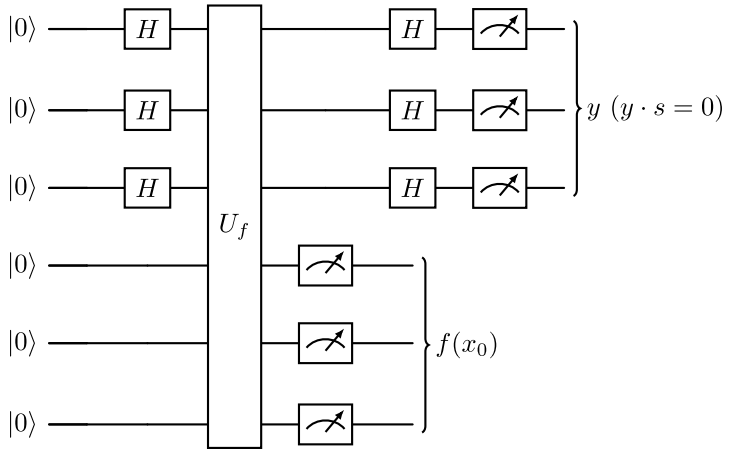


Fig. 4.3: The Simon's algorithm reveals an y such that $s \cdot y = 0$.

The quantum circuit for Simon's algorithm involves the following steps:

1. **Initialization:** Start with two registers of n qubits each, both initialized to $|0\rangle^{\otimes n}$.

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

2. **Superposition:** Apply a Hadamard gate to each qubit in the first register. This creates an equal superposition of all possible n-bit strings.

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

3. **Query the Oracle:** Apply the quantum oracle U_f that implements the function f: $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$.

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

4. **Measure the Second Register:** Measure the second register. This yields a value $f(x_0)$ for some $x_0 \in \{0,1\}^n$. Due to the property of f, we know that $f(x_0) = f(x_0 \oplus s)$. Therefore, the first register collapses to an equal superposition of $|x_0\rangle$ and $|x_0 \oplus s\rangle$:

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle) |f(x_0)\rangle$$

We can ignore the second register from now on, as it is no longer needed. Consider only the first register, which is now in the state:

$$|\psi_3'\rangle = \frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle)$$

5. **Apply Hadamard Transform:** Apply a Hadamard gate to each qubit in the first register.

$$|\psi_4\rangle = H^{\otimes n}\left[\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle)\right]$$

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

$$|\psi_4\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{y\in\{0,1\}^n}\left[(-1)^{x_0\cdot y} + (-1)^{(x_0\oplus s)\cdot y}\right]|y\rangle$$

Simplifying the exponent, we have $(x_0 \oplus s)\cdot y = (x_0\cdot y)\oplus(s\cdot y)$. Thus:

$$|\psi_4\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{y\in\{0,1\}^n}(-1)^{x_0\cdot y}\left[1 + (-1)^{s\cdot y}\right]|y\rangle$$

Notice that if $s\cdot y = 1$, then the term in the brackets becomes $1 + (-1) = 0$. Therefore, we only get non-zero amplitudes for y such that $s\cdot y = 0$.

6. **Measure the First Register:** Measure the first register. The measurement outcome y will satisfy the equation $s\cdot y = 0 \pmod 2$.
7. **Repeat:** Repeat steps 2-6 $O(n)$ times to obtain $n-1$ linearly independent equations of the form $s\cdot y_i = 0 \pmod 2$.
8. **Solve the System of Equations:** Solve the system of linear equations to find the hidden string s. This can be done efficiently using classical Gaussian elimination.

Example

Let's work through an example of Simon's algorithm for $n = 3$ qubits. Suppose the hidden string is $s = 110$ (though we don't know this yet).
For a hidden string $s = 110$, the valid measurements that satisfy $s\cdot y = 0 \pmod 2$ are $y \in \{001, 110, 111\}$.
After running steps 2-6 of Simon's algorithm once, we might measure $y_1 = 001$ in the first register. This gives us our first equation:

$$s\cdot y_1 = 0 \pmod 2$$

Substituting $y_1 = 001$, we get:

$$s_1\cdot 0 + s_2\cdot 0 + s_3\cdot 1 = 0 \pmod 2$$

$$s_3 = 0$$

So we've learned that the third bit of our hidden string is 0.
Running the algorithm a second time, we might measure $y_2 = 110$. This gives us our second equation:

$$s\cdot y_2 = 0 \pmod 2$$

Substituting $y_2 = 110$, we get:

$$s_1 \cdot 1 + s_2 \cdot 1 + s_3 \cdot 0 = 0 \pmod 2$$

$$s_1 + s_2 = 0 \pmod 2$$

Since $s_1 + s_2 = 0 \pmod 2$, either both s_1 and s_2 are 0, or both are 1. Given that $s \neq 000$ (by the promise that f is two-to-one) and $s_3 = 0$, we must have $s_1 = s_2 = 1$.
Therefore, $s = 110$, which is indeed our hidden string.
This example demonstrates how Simon's algorithm allows us to efficiently determine the hidden string by collecting and solving a system of linear equations.

i Simon's Secret: How Identical Outputs Reveal the Hidden String

Simon's algorithm cleverly exploits the function's special property: $f(x) = f(x \oplus s)$.

1. **Superposition:** The initial Hadamard gates create a superposition of all possible inputs:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

2. **Oracle's Role:** The oracle maps each input to its corresponding output:

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

3. **The Magic:** Because $f(x) = f(x \oplus s)$, measuring the second register (the output) *entangles* pairs of inputs (x and $x \oplus s$) that produce the *same* output. This entanglement creates a special interference pattern in the first register that, after more steps, allows us to extract the hidden string s.

4.4.3 Quantum Speedup

Simon's algorithm provides an exponential speedup over classical algorithms for finding the hidden string s. The classical algorithm requires $O(2^{n/2})$ queries, while Simon's algorithm requires only $O(n)$ quantum queries and polynomial classical post-processing. This demonstrates the potential of quantum computers to solve certain problems much more efficiently than their classical counterparts.

4.5 Quantum Phase Estimation

Quantum Phase Estimation (QPE) is a crucial quantum algorithm used to estimate the eigenvalue (or phase) of an eigenvector of a unitary operator. It serves as a building block for many other quantum algorithms, including Shor's algorithm for factoring and algorithms for simulating quantum systems.

4.5.1 Problem Statement

Given a unitary operator U and an eigenstate $|\psi\rangle$ such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, the goal of QPE is to estimate the value of θ. Here, θ is the phase we want to estimate, and it lies in the range $[0, 1)$.

4.5.2 Quantum Implementation

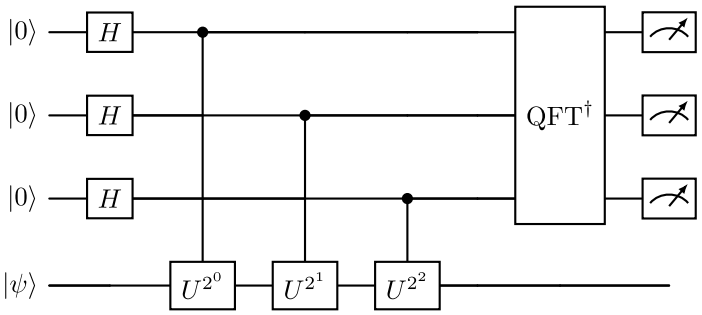


Fig. 4.4: QPE circuit with 3 qubits in the control register (for 3 digit accuracy)

The QPE algorithm uses two registers:

1. **The Control Register:** This register consists of t qubits, initialized to $|0\rangle^{\otimes t}$. The number of qubits t determines the precision of the phase estimation.
2. **The Eigenstate Register:** This register holds the eigenstate $|\psi\rangle$ of the unitary operator U.

The algorithm proceeds as follows:

1. **Initialization:** Prepare the control register in the state $|0\rangle^{\otimes t}$ and the eigenstate register in the state $|\psi\rangle$.

$$|\psi_0\rangle = |0\rangle^{\otimes t}|\psi\rangle$$

2. **Superposition:** Apply a Hadamard gate to each qubit in the control register to create an equal superposition.

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

3. **Controlled Unitary Operations:** Apply controlled-U operations, where U is applied 2^j times conditioned on the j-th qubit (counting from 0) of the control register. This means applying U^{2^j} when the j-th qubit is $|1\rangle$.

$$|\psi_2\rangle = \frac{1}{\sqrt{2^t}}\sum_{x=0}^{2^t-1}|x\rangle U^x|\psi\rangle = \frac{1}{\sqrt{2^t}}\sum_{x=0}^{2^t-1}|x\rangle e^{2\pi i\theta x}|\psi\rangle$$

We can rewrite this as:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^t}} \left(\sum_{x=0}^{2^t-1} e^{2\pi i \theta x} |x\rangle \right) |\psi\rangle$$

4. **Inverse Quantum Fourier Transform (QFT†):** Apply the inverse Quantum Fourier Transform (QFT†) to the control register. The QFT† transforms the state:

$$\text{QFT}^\dagger |x\rangle = \frac{1}{\sqrt{2^t}} \sum_{y=0}^{2^t-1} e^{-2\pi i x y / 2^t} |y\rangle$$

Applying QFT† to the control register of $|\psi_2\rangle$ yields:

$$|\psi_3\rangle = \frac{1}{2^t} \sum_{x=0}^{2^t-1} \sum_{y=0}^{2^t-1} e^{2\pi i \theta x} e^{-2\pi i x y / 2^t} |y\rangle |\psi\rangle$$

The amplitude of each state $|y\rangle$ in the control register is:

$$\alpha_y = \frac{1}{2^t} \sum_{x=0}^{2^t-1} e^{2\pi i x(\theta - y/2^t)}$$

This amplitude is large when $y/2^t$ is close to θ.

5. **Measurement:** Measure the control register in the computational basis. The measurement outcome y provides an estimate of θ as $\tilde{\theta} = y/2^t$.

i Quantum Fourier Transform (QFT) and Factorization

The Quantum Fourier Transform (QFT) is the quantum analogue of the discrete Fourier transform (DFT) and is a crucial component in many quantum algorithms, including Shor's algorithm and quantum phase estimation. The QFT transforms a quantum state from the computational basis to the Fourier basis and vice versa. A key feature of the QFT is that it can be factored into a product of single-qubit rotations, enabling its efficient implementation on a quantum computer. Mathematically, the QFT is defined as the following transformation on a quantum state $|x\rangle$ with N basis states, where $N = 2^n$ for n qubits:

$$\text{QFT} |x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y / N} |y\rangle$$

Let's express x and y in their binary representations: $x = x_1 x_2 ... x_n$ and $y = y_1 y_2 ... y_n$, where x_i and y_i are the i-th bits of x and y, respectively. Then we can rewrite the QFT as:

$$\text{QFT} |x_1 x_2 ... x_n\rangle = \frac{1}{\sqrt{N}} \sum_{y_1=0}^{1} \sum_{y_2=0}^{1} ... \sum_{y_n=0}^{1} e^{2\pi i x (\sum_{k=1}^{n} y_k 2^{-k})} |y_1 y_2 ... y_n\rangle$$

$$= \frac{1}{\sqrt{N}} \sum_{y_1=0}^{1} \sum_{y_2=0}^{1} \dots \sum_{y_n=0}^{1} \prod_{k=1}^{n} e^{2\pi i x y_k 2^{-k}} |y_1 y_2 \dots y_n\rangle$$

$$= \frac{1}{\sqrt{N}} \bigotimes_{k=1}^{n} \sum_{y_k=0}^{1} e^{2\pi i x y_k 2^{-k}} |y_k\rangle$$

$$= \frac{1}{\sqrt{N}} \bigotimes_{k=1}^{n} \left(|0\rangle + e^{2\pi i x 2^{-k}} |1\rangle\right)$$

$$= \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i x 2^{-1}} |1\rangle\right) \left(|0\rangle + e^{2\pi i x 2^{-2}} |1\rangle\right) \dots \left(|0\rangle + e^{2\pi i x 2^{-n}} |1\rangle\right)$$

This factorization shows that the QFT can be implemented by applying a series of single-qubit rotations conditioned on the values of other qubits. Specifically, the k-th qubit is rotated by an angle that depends on the values of the qubits $x_1, x_2, ..., x_n$. This decomposition is crucial for constructing the QFT circuit efficiently.

Key Properties and Implications:

1. **Unitary Transformation:** The QFT is a unitary transformation, meaning it preserves the norm of quantum states and can be implemented as a quantum circuit.
2. **Efficient Implementation:** The QFT can be implemented on a quantum computer using $O(n^2)$ quantum gates, where n is the number of qubits. Note that unlike a classical FFT, direct readout of the quantum state does not give the full Fourier transform due to the nature of quantum measurement.
3. **Period Finding:** The QFT is particularly useful for finding periodic patterns in quantum states, which is a key step in Shor's algorithm for factoring integers.
4. **Phase Estimation:** The QFT is used in the quantum phase estimation algorithm to accurately estimate the eigenvalues of unitary operators.

The inverse Quantum Fourier Transform (QFT†) reverses the transformation:

$$\text{QFT}^\dagger |x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{-2\pi i x y/N} |y\rangle$$

It also has an efficient implementation based on a similar factorization.

i Unpacking the Quantum Fourier Transform: Controlled Phase Shifts

The Quantum Fourier Transform (QFT) relies on a series of *controlled phase shift* gates to create interference patterns.

- **Controlled Phase Shift Gates:** These gates apply a phase shift to a target qubit, conditioned on the state of a control qubit. The amount of the phase shift is different for each gate. For example:
  - R_2 gate: Adds a phase shift of $\pi/2$ (90 degrees).
  - R_3 gate: Adds a phase shift of $\pi/4$ (45 degrees).
  - In general, R_k gate: Adds a phase shift of $\pi/2^{k-1}$ radians.
- **Binary Representation Analogy:** Think of these phase shifts as encoding information in

binary. Each qubit represents a "digit" in the binary representation of a frequency. The earlier qubits in the circuit represent the more significant bits, and later qubits represent the less significant bits.

- **Creating Interference:** By carefully controlling these phase shifts, the QFT creates a complex interference pattern that allows us to extract the frequency components of a quantum state. This is crucial for algorithms like Shor's algorithm, where we need to find the period of a function.

4.5.3 Accuracy

The accuracy of the phase estimation depends on the number of qubits t in the control register. If θ can be written exactly as $j/2^t$ for some integer j, then the measurement outcome will be exactly j with probability 1. Otherwise, the measurement will yield an approximation $\tilde{\theta}$ of θ. With high probability, the estimated phase $\tilde{\theta}$ satisfies $|\theta - \tilde{\theta}| \leq 2^{-t}$. Therefore, to achieve an accuracy of 2^{-n}, we need $t = n + O(\log(1/\epsilon))$ qubits in the control register to succeed with probability at least $1 - \epsilon$.

4.5.4 Example

Suppose $U|\psi\rangle = e^{2\pi i(1/4)}|\psi\rangle$, so $\theta = 1/4 = 0.25$. Let's use $t = 2$ qubits in the control register. The possible measurement outcomes are $00, 01, 10, 11$, corresponding to estimates $0/4 = 0$, $1/4 = 0.25$, $2/4 = 0.5$, and $3/4 = 0.75$. The algorithm will ideally measure $|01\rangle$ with high probability, giving the correct estimate $\tilde{\theta} = 0.25$.

4.5.5 Applications

QPE is a fundamental algorithm with numerous applications:

- **Shor's Algorithm:** Used to find the order of an element, which is a crucial step in factoring integers.
- **Quantum Simulation:** Used to estimate the energy levels of quantum systems.
- **Estimating Eigenvalues:** Used in various quantum machine learning algorithms.

QPE showcases the power of quantum computation by efficiently estimating the phase of an eigenvalue, a task that can be difficult or impossible for classical computers in certain scenarios.

4.6 Grover's Algorithm

Grover's algorithm [15] is a quantum search algorithm for finding a specific item in an unsorted database of N items with only $O(\sqrt{N})$ queries to the database, offering a quadratic speedup over classical search. The key idea is to use a quantum oracle to mark the target item and then apply a diffusion operator to amplify the probability of measuring the target. We can understand the power of Grover's algorithm by considering the geometric interpretation of the oracle and diffusion operators as rotations on a Bloch sphere.

4.6.1 The Unstructured Search Problem

Suppose we have a function $f(x)$ that takes an input x from a set of N items, where $f(x) = 1$ if x is the item we are searching for (the "target") and $f(x) = 0$ otherwise. Our goal is to find the value of x such that $f(x) = 1$. Classically, in the worst case, we might have to check every item in the database, requiring $O(N)$ queries.

> **Relating number of elements to qubits**
>
> For a database of N items, the number of qubits required to represent each item is $n = \lceil \log_2 N \rceil$.

4.6.2 Quantum Oracle

The quantum oracle, O, is a unitary operator that marks the target state. If the target state is $|w\rangle$, the oracle acts as:

$$O|x\rangle = \begin{cases} -|x\rangle & \text{if } x = w \\ |x\rangle & \text{if } x \neq w \end{cases}$$

This can be written more generally as $O = I - 2|w\rangle\langle w|$, where I is the identity operator. Geometrically, the oracle O reflects the state about the hyperplane orthogonal to $|w\rangle$.

4.6.3 The Diffusion Operator

The diffusion operator, D, is defined as:

$$D = 2|\psi\rangle\langle\psi| - I$$

where $|\psi\rangle$ is the uniform superposition over all states:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

The diffusion operator D reflects the state about the mean amplitude.

> **Demystifying the Grover Diffusion Operator: "Inversion About the Mean"**
>
> The Grover diffusion operator ($D = 2|\psi\rangle\langle\psi| - I$) might seem mysterious, but it's essentially an "inversion about the mean" of the amplitudes. Here's how to think about it:
>
> 1. **Calculate the Mean Amplitude:** Imagine all the amplitudes of the quantum states as numbers. Calculate the average (mean) of these numbers. Since we start with a uniform superposition, this mean is initially close to zero.
> 2. **Invert Around the Mean:** For each state's amplitude:

- Calculate the difference between the amplitude and the mean.
- Subtract this difference *again* from the mean. This "inverts" the amplitude around the mean. Mathematically, if a_i is the amplitude of state $|i\rangle$ and $\bar{a}$ is the mean amplitude, the new amplitude a'_i is calculated as:

$$a'_i = \bar{a} - (a_i - \bar{a}) = 2\bar{a} - a_i$$

Visual Analogy: Imagine a seesaw balanced at the mean amplitude ($\bar{a}$). The diffusion operator flips each amplitude to the opposite side of the seesaw, relative to the balance point. This has the effect of *decreasing* the amplitudes of the majority of states and *increasing* the amplitude of the marked state(s).
Why this works: The marked state gets a larger "boost" because its amplitude is initially *negative* (due to the oracle's phase flip), making it lower than the mean. This process, repeated multiple times, amplifies the marked state until it dominates the superposition.

4.6.4 Grover Iteration as a Rotation

The Grover iteration $G = D \cdot O$ can be understood as a rotation within the two-dimensional subspace spanned by the target state $|w\rangle$ and the superposition of non-target states. Let's define $|\psi_\perp\rangle$ as the normalized superposition of all states orthogonal to the target state $|w\rangle$. If the uniform superposition is $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$ and the target state is $|w\rangle$, we can express $|\psi\rangle$ in terms of $|w\rangle$ and $|\psi_\perp\rangle$.

$$|\psi\rangle = \frac{1}{\sqrt{N}} |w\rangle + \sqrt{1 - \frac{1}{N}} |\psi_\perp\rangle .$$

Notice that there is a low chance $1/N$ of getting w from this wavefunction as it currently stands.

In the basis $\{|\psi_\perp\rangle, |w\rangle\}$, the Grover operator G takes the form of a rotation matrix:

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

This matrix represents a rotation by an unknown angle θ on the Bloch sphere. If we let $\sin(\theta/2) = \frac{2\sqrt{N-1}}{N}$, then

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

The goal is to rotate the initial state $|\psi\rangle$ as close as possible to the target state $|w\rangle$. After k iterations, the state is:

$$G^k |\psi_\perp\rangle = \cos(k\theta/2)|\psi_\perp\rangle + \sin(k\theta/2)|w\rangle$$

To maximize the probability of measuring $|w\rangle$, we want $\sin(k\theta)$ to be as close to 1 as possible, which means $k\theta \approx \pi$.

Since $\sin(\theta/2) = \frac{2\sqrt{N-1}}{N} \approx \frac{2\sqrt{N}}{N} = \frac{2}{\sqrt{N}}$ for large N, we have $\theta/2 \approx \frac{2}{\sqrt{N}}$, so $\theta \approx \frac{4}{\sqrt{N}}$.

Therefore, $k \approx \frac{\pi}{\theta} \approx \frac{\pi}{4}\sqrt{N}$.

Thus, the optimal number of iterations is approximately $\frac{\pi}{4}\sqrt{N}$.

4.6.5 Measurement

After applying the Grover iteration approximately $\frac{\pi}{4}\sqrt{N}$ times, measuring the state will yield the target state $|w\rangle$ with high probability.

4.6.6 Quantum Speedup

Grover's algorithm achieves a quadratic speedup compared to classical search. While a classical search requires $O(N)$ queries in the worst case, Grover's algorithm finds the target with $O(\sqrt{N})$ quantum queries. This speedup demonstrates the power of quantum computation for solving search problems.

4.7 Shor's Algorithm (High-Level Overview)

Shor's algorithm is a quantum algorithm for factoring large integers [14]. It has exponential speedup compared to the best-known classical factoring algorithm. Factoring large numbers is a computationally hard problem that underlies the security of many public-key cryptosystems, such as RSA. Shor's algorithm combines classical number theory with quantum computation to efficiently find the prime factors of an integer.

The algorithm consists of two main parts:

1. **Classical Pre-processing:** This involves reducing the factoring problem to the problem of finding the period of a function. This step leverages number-theoretic results but is executed on a classical computer.
2. **Quantum Period Finding:** This is the core quantum part of the algorithm. It uses the quantum Fourier transform (QFT) to efficiently find the period of the function defined in the classical pre-processing step. The quantum period finding provides an exponential speedup compared to classical period-finding algorithms.
3. **Classical Post-processing:** Once the period is found using the quantum computation, a classical computer performs some post-processing steps, using number theory, to deduce the factors of the original integer.

The quantum part of Shor's algorithm relies on the following key quantum computing concepts:

- **Quantum Fourier Transform (QFT):** The QFT is used to find the period of a function by identifying the dominant frequencies in its Fourier transform. The QFT can be implemented efficiently on a quantum computer.
- **Superposition:** Superposition is used to evaluate the function at multiple points simultaneously.
- **Quantum Measurement:** Measurement is used to extract the period information from the quantum state.

In summary, Shor's algorithm leverages the QFT to perform period finding exponentially faster than any known classical algorithm, which allows us to factor large integers efficiently. The combination of classical pre- and post-processing with the quantum period-finding routine makes Shor's algorithm a powerful tool with significant implications for cryptography.

4.7.1 Quantum Period Finding in Shor's Algorithm

The quantum period-finding algorithm is the heart of Shor's algorithm, providing the exponential speedup over classical methods. Here's a breakdown of how it works:

1. **Problem Setup:** We are given an integer N that we want to factor. We choose a random number a such that $1 < a < N$ and $\gcd(a, N) = 1$. We define a function $f(x) = a^x \mod N$. The goal is to find the period r of this function, which is the smallest positive integer such that $f(x + r) = f(x)$ for all x. In other words, $a^r \equiv 1 \pmod{N}$.

2. **Quantum Registers:** We use two quantum registers:

   - Register 1 (input register): This register consists of n qubits, where n is chosen such that $N^2 \leq 2^n < 2N^2$. This register will hold the input values x for the function $f(x)$. It is initialized to $|0\rangle^{\otimes n}$.
   - Register 2 (output register): This register consists of m qubits, where m is large enough to store the possible values of $f(x)$. It is initialized to $|0\rangle^{\otimes m}$.

3. **Superposition:** Apply a Hadamard gate to each qubit in Register 1 to create an equal superposition of all possible input values:

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle^{\otimes m}$$

4. **Function Evaluation (Quantum Oracle):** Apply a quantum oracle U_f that performs the modular exponentiation: $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$. This creates the entangled state:

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

5. **Quantum Fourier Transform (QFT):** Apply the QFT to Register 1. This transforms the state as follows:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i x y / 2^n} |y\rangle \right) |f(x)\rangle$$

   Rearranging the summation, we get:

$$|\psi_2\rangle = \frac{1}{2^n} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} e^{2\pi i x y / 2^n} |y\rangle |f(x)\rangle$$

6. **Measurement:** Measure Register 2. This collapses the superposition to a specific value $f(x_0)$. The state of Register 1 becomes a superposition of states $|x\rangle$ such that $f(x) = f(x_0)$. Since $f(x)$ is periodic with period r, the register 1 will contain a superposition of states $|x_0\rangle, |x_0 + r\rangle, |x_0 + 2r\rangle, \ldots$

7. **Second Measurement:** Measure Register 1. The measurement result y will be close to an integer multiple of $2^n/r$. That is, $y \approx c \cdot \frac{2^n}{r}$ for some integer c.

8. **Classical Post-processing (Continued Fractions):** Use the continued fractions algorithm to find the best rational approximation of $y/2^n$. This gives us a candidate for the period r.
9. **Verification:** Check if the candidate r is indeed the period by verifying that $a^r \equiv 1 \pmod{N}$. If it is not, repeat the algorithm.
10. **Factoring:** If r is even, compute $\gcd(a^{r/2}+1, N)$ and $\gcd(a^{r/2}-1, N)$. These are likely to be non-trivial factors of N. If the factors are trivial or r is odd, go back to step 1 and choose a different random a.

The QFT step is crucial because it transforms the periodic function $f(x)$ into a state where the period can be efficiently extracted through measurement and classical post-processing. The exponential speedup arises from the ability of the QFT to identify the period in a single quantum computation, whereas classical period-finding algorithms would require exponentially many evaluations of the function.

! Why Modular Exponentiation?

Modular exponentiation (computing $a^x \mod N$) is crucial because:

1. It's periodic: For some period r, $a^{x+r} \equiv a^x \pmod{N}$
2. This period r is related to the factors of N
3. The quantum computer can find r efficiently using the Quantum Fourier Transform (QFT)

Key insight: We don't need to know all values of $a^x \mod N$, we just need to find how often they repeat!
Example: For $N = 15$, $a = 7$

- $7^1 \mod 15 = 7$
- $7^2 \mod 15 = 4$
- $7^3 \mod 15 = 13$
- $7^4 \mod 15 = 1 \leftarrow$ Period found!

4.8 HHL Algorithm

The HHL algorithm, named after Harrow, Hassidim, and Lloyd, is a quantum algorithm that solves linear systems of equations [30]. Given a matrix A and a vector $\vec{b}$, the goal is to find the vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. While solving linear systems is a ubiquitous task in classical computing, the HHL algorithm offers a potential exponential speedup under certain conditions, making it a landmark result in quantum algorithm design.

4.8.1 The Linear Systems Problem

In its most general form, a linear system of equations is represented as:

$$A\vec{x} = \vec{b}$$

where:

- A is an $N \times N$ matrix.
- $\vec{x}$ is an N-dimensional vector we want to find.
- $\vec{b}$ is an N-dimensional vector.

Classically, solving this problem typically requires $O(N^3)$ time using Gaussian elimination or $O(N^{2.373})$ using more advanced algorithms. For large N, these algorithms can be computationally expensive.

4.8.2 Quantum Solution Approach

The HHL algorithm provides a quantum solution to this problem with a runtime complexity of $O(\log(N))$, under certain assumptions. Here's a high-level overview of the algorithm:

1. **State Preparation:** The input vector $\vec{b}$ is loaded into a quantum state $|\vec{b}\rangle$. This step assumes that we have a way to efficiently prepare the state, which is a crucial requirement for the algorithm's speedup.
2. **Hamiltonian Simulation:** The matrix A is represented as a Hamiltonian, and we perform Hamiltonian simulation to implement the time evolution operator e^{iAt} for some time t. This step requires A to be Hermitian. If A is not Hermitian, it can be transformed into a Hermitian matrix by considering the augmented system:

$$\begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$$

3. **Quantum Phase Estimation (QPE):** QPE is applied to estimate the eigenvalues of A. This involves applying the time evolution operator e^{iAt} multiple times and using the quantum Fourier transform to extract the eigenvalues. The result is a quantum state where the eigenvalues are stored in a separate register.
4. **Controlled Rotation:** A controlled rotation is performed based on the estimated eigenvalues. This rotation effectively multiplies each eigenvector component by the inverse of its corresponding eigenvalue.
5. **Uncomputation:** The QPE is uncomputed to remove the eigenvalue register, leaving the solution in the original register.
6. **Measurement:** Finally, we measure the quantum state to obtain the solution vector $\vec{x}$.

4.8.3 Mathematical Details

Let's break down the key steps with some mathematical details:

- **Eigenvalue Decomposition:** Assume A is Hermitian and has the eigenvalue decomposition:

$$A = \sum_{j=1}^{N} \lambda_j |u_j\rangle\langle u_j|$$

  where λ_j are the eigenvalues and $|u_j\rangle$ are the corresponding eigenvectors.

- **Input State:** The input state $|\vec{b}\rangle$ can be expressed in terms of the eigenvectors of A:

$$|\vec{b}\rangle = \sum_{j=1}^{N} b_j |u_j\rangle$$

where $b_j = \langle u_j|\vec{b}\rangle$ are the coefficients.

- **Solution State:** The solution vector $\vec{x}$ can also be expressed in terms of the eigenvectors:

$$|\vec{x}\rangle = A^{-1}|\vec{b}\rangle = \sum_{j=1}^{N} \frac{b_j}{\lambda_j} |u_j\rangle$$

The goal of the HHL algorithm is to prepare this state $|\vec{x}\rangle$.

- **Quantum Phase Estimation:** QPE allows us to estimate the eigenvalues λ_j. After QPE, we have a state like:

$$\sum_{j=1}^{N} b_j |\lambda_j\rangle |u_j\rangle$$

where $|\lambda_j\rangle$ is a quantum register holding an estimate of the eigenvalue λ_j.

- **Controlled Rotation:** We then perform a controlled rotation. This rotation is applied to an ancilla qubit conditioned on the estimated eigenvalue $|\lambda_j\rangle$. The state becomes:

$$\sum_{j=1}^{N} b_j |\lambda_j\rangle \left(C\frac{1}{\lambda_j}|0\rangle + S|1\rangle \right) |u_j\rangle$$

Here, $|0\rangle$ and $|1\rangle$ represent the states of the ancilla qubit, and C and S are chosen such that $C^2/\lambda_j^2 + S^2 = 1$, ensuring normalization. The crucial point is that the amplitude of the $|0\rangle$ state is proportional to $1/\lambda_j$. Measuring the ancilla qubit in the $|0\rangle$ state projects the system (approximately) onto the desired solution:

$$\sum_{j=1}^{N} \frac{b_j}{\lambda_j} |u_j\rangle \propto |\vec{x}\rangle$$

4.8.4 Assumptions and Limitations

The HHL algorithm's exponential speedup comes with several caveats:

1. **Sparse Matrix:** The matrix A must be sparse, meaning it has a small number of non-zero elements per row. This is necessary for efficient Hamiltonian simulation.
2. **State Preparation:** Efficiently preparing the input state $|\vec{b}\rangle$ is crucial. If this step is classically hard, the overall algorithm will not provide a speedup.

3. **Condition Number:** The condition number of A, denoted as $\kappa(A)$, affects the algorithm's runtime. A large condition number indicates that A is ill-conditioned, which can lead to numerical instability and increased runtime. The complexity scales as $O(\kappa \log(N))$.
4. **Logarithmic Dependence:** The algorithm's runtime depends logarithmically on the size of the system, N. This is where the exponential speedup comes from compared to classical algorithms that scale polynomially with N. However, the logarithmic speedup is only advantageous for sufficiently large N.
5. **Output:** The HHL algorithm outputs a quantum state $|\vec{x}\rangle$, not the explicit values of the solution vector $\vec{x}$. Extracting specific elements of $\vec{x}$ requires additional measurements, which can affect the overall runtime.

4.8.5 Applications

Despite its limitations, the HHL algorithm has potential applications in various fields:

1. **Finite Element Analysis:** Solving partial differential equations using finite element methods often involves solving large, sparse linear systems.
2. **Fluid Dynamics:** Simulating fluid flow can require solving linear systems to determine the velocity and pressure fields.
3. **Circuit Analysis:** Analyzing electrical circuits involves solving linear systems to determine the currents and voltages in the circuit.
4. **Machine Learning:** Some machine learning algorithms, such as linear regression and support vector machines, involve solving linear systems.

4.8.6 Conclusion

The HHL algorithm [30] is a significant achievement in quantum algorithm design, demonstrating the potential for exponential speedups in solving linear systems of equations. While it has limitations and requires specific conditions to be met, it opens up new possibilities for tackling computationally challenging problems in science and engineering. Understanding its assumptions and limitations is crucial for assessing its applicability to specific problems and for developing future quantum algorithms that can overcome these challenges.

4.9 Hybrid Quantum-Classical Algorithms

4.9.1 Variational Quantum Eigensolver (VQE)

The Variational Quantum Eigensolver (VQE) is a hybrid quantum-classical algorithm used to find the ground state (minimum energy) of a quantum system. Many problems in quantum chemistry, materials science, and condensed matter physics involve finding the ground state energy of a molecule or material, which is classically intractable for large systems. VQE leverages a quantum computer to prepare a trial wave function (called an "ansatz") $|\psi(\theta)\rangle$ and measure its energy $\langle\psi(\theta)|H|\psi(\theta)\rangle$, where H is the Hamiltonian of the system, and θ represents a set of adjustable parameters. A classical optimization

algorithm then adjusts the parameters θ of the ansatz to minimize the energy. The key idea is to variationally find the eigenvector corresponding to the smallest eigenvalue, which represents the ground state.

4.9.2 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) is another hybrid quantum-classical algorithm designed to find approximate solutions to combinatorial optimization problems. These are problems where the goal is to find the best solution from a finite set of possibilities (e.g., the traveling salesman problem, graph partitioning). QAOA uses a quantum computer to explore the solution space and a classical computer to optimize the parameters that control the quantum evolution. The algorithm alternates between applying operators related to the problem's cost function and a mixing operator, with the goal of converging to a state that encodes a good solution. Like VQE, QAOA relies on variational techniques, where the parameters of a quantum circuit are optimized using classical methods.

4.10 Overview and comparison with classical

Here's a table summarizing the classical and quantum complexities of the algorithms discussed, along with whether they are deterministic or probabilistic. These can all be more precisely defined by number of gates and other factors not mentioned here, for more details see the Quantum Algorithm Zoo:

Algorithm	Classical Complexity (Deterministic)	Classical Complexity (Probabilistic)	Quantum Complexity	Deterministic or Probabilistic
Deutsch's Algorithm	2 queries	N/A	1 query	Deterministic
Deutsch-Jozsa Algorithm	$\mathcal{O}(2^{n-1})$	1 query	$\mathcal{O}(1)$	Deterministic
Bernstein-Vazirani Algorithm	n queries	N/A	1 query	Deterministic
Simon's Algorithm	$\mathcal{O}(2^{n/2})$	N/A	$\mathcal{O}(n)$	Probabilistic
Quantum Phase Estimation (QPE)	Exponential (worst-case)	Exponential (worst-case)	$\mathcal{O}(m)$ controlled operations for m bits	Probabilistic
Grover's Algorithm	$\mathcal{O}(N)$	N/A	$\mathcal{O}(\sqrt{N})$	Probabilistic
Shor's Algorithm	$\mathcal{O}(\exp(N))$	$\mathcal{O}(\exp(N))$	$\mathcal{O}(\text{poly}(N))$	Probabilistic
HHL Algorithm	$\mathcal{O}(N^3)$ or $\mathcal{O}(N^\omega)$	N/A	$\mathcal{O}(\log(N))$	Probabilistic

Notes:

- **N:** Input size (e.g., number of items in a database for Grover's, number to be factored for Shor's, dimension of the matrix for HHL).
- **n:** Number of qubits.
- **t:** Number of qubits in the control register for QPE, related to desired accuracy.

- **exp(N):** Exponential in N.
- **poly(N):** Polynomial in N.
- **HHL Conditions:** The HHL algorithm's exponential speedup is contingent on several factors: the matrix A being sparse, efficient state preparation of $|b\rangle$, and a low condition number.
- **Probabilistic vs. Deterministic:** Deterministic algorithms always produce the correct answer. Probabilistic algorithms have a chance of error, but the probability of error can be made arbitrarily small by repeating the algorithm.
- **Speedup:** This column indicates the type of speedup the quantum algorithm offers compared to the best-known classical algorithm.

This table provides a general overview. The precise complexities and speedups can vary depending on the specific implementation and problem instance.

Interlude: Hardwire requirements

Before we hop into the platforms where we might be able to perform these algorithms to solve problems, we need to specify what we want.

DiVincenzo's Criteria for Quantum Computation

In 2000, David DiVincenzo outlined a set of criteria [31] that any physical system must meet to be considered a viable quantum computer. These criteria serve as a benchmark for evaluating different quantum computing platforms and guiding research efforts. The criteria can be summarized as follows:

1. **Scalable qubits:** A practical quantum computer must be able to scale to a large number of qubits. The number of qubits required depends on the complexity of the problem being solved, but fault-tolerant quantum computation generally requires thousands or even millions of qubits.
2. **Initialization:** The ability to initialize the qubits to a known state, such as $|0\rangle^{\otimes n}$, is crucial. This provides a well-defined starting point for quantum algorithms.
3. **Long coherence times:** Qubits must maintain their quantum coherence for a sufficiently long time to allow complex quantum operations to be performed. Decoherence, the loss of quantum information to the environment, is a major obstacle to quantum computation. Coherence times must be much longer than the gate operation times.
4. **Universal gate set:** A universal set of quantum gates is required to perform arbitrary quantum computations. A universal gate set is a small set of gates that can be combined to approximate any other quantum gate. Examples include the Hadamard gate, the CNOT gate, and single-qubit rotation gates.
5. **Measurement:** The ability to accurately measure the state of the qubits is essential for extracting the results of a quantum computation. Measurements must be reliable and have high fidelity.

In addition to these five criteria, DiVincenzo also proposed two criteria for quantum communication:

6. **Interconvertible quantum and classical bits:** The ability to convert quantum information into classical information and vice versa is important for interfacing with classical computers and communication networks.
7. **Faithful transmission of qubits:** The ability to transmit qubits reliably between different locations is essential for building distributed quantum computers and quantum networks.

Meeting these criteria is a significant challenge, and different quantum computing platforms are at various stages of development with respect to each criterion. Superconducting qubits, trapped ions, neutral atoms, quantum dots, and topological qubits are among the leading candidates for building practical quantum computers. Each platform has its own strengths and weaknesses, and ongoing research is focused on overcoming the challenges associated with each approach [29].

Measurement-Based Quantum Computation

It's worth noting that some quantum computing platforms employ a different paradigm called measurement-based quantum computation (MBQC). Unlike the gate-based approach we've primarily discussed, MBQC relies on creating a highly entangled resource state, often a cluster state, and then performing computation by making a series of single-qubit measurements. The specific measurements performed and their order determine the quantum algorithm being executed. While the underlying physics and qubit requirements are similar, the programming and control aspects of MBQC differ significantly from gate-based quantum computation.

Magic State Distillation

Achieving a universal gate set is crucial for implementing arbitrary quantum algorithms. While some quantum computing platforms can directly implement a universal gate set, others have a limited set of native gates. In such cases, additional resources and techniques are required to achieve universality. One common approach is to use *magic state distillation*.

The Need for Magic States

A restricted gate set, such as those that are Clifford gates, can perform a limited set of quantum computations efficiently. Clifford gates consist of Hadamard, CNOT, and phase gates. The Gottesman-Knill theorem states that quantum circuits consisting only of Clifford gates, preparation of computational basis states (e.g., $|0\rangle$), and measurement in the computational basis can be efficiently simulated on a classical computer. Therefore, to achieve true quantum supremacy, we need non-Clifford gates.

Magic state distillation is a technique used to create high-fidelity *magic states*, which can then be used along with Clifford gates to implement non-Clifford gates, such as the T gate (also known as the $\pi/8$ gate). The T gate adds a phase of $\pi/4$ to the $|1\rangle$ state:

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

A magic state, often denoted as $|A\rangle$, is a specific quantum state that, when combined with Clifford gates through a process called *state injection*, allows for the implementation of non-Clifford gates.

By adding the T gate to the Clifford gates, we obtain a universal gate set.

Resource Overhead and Error Correction

Magic state distillation introduces significant resource overhead in terms of the number of physical qubits and quantum gates required. Generating high-fidelity magic states is a computationally intensive process, often requiring multiple rounds of distillation and significant post-selection to obtain the desired fidelity. This overhead can be a limiting factor in the overall performance and scalability of quantum algorithms.

Furthermore, magic state distillation is often intertwined with quantum error correction. Since the distillation process itself is susceptible to errors, robust error correction schemes are necessary to protect the fragile quantum information during distillation. The interplay between magic state distillation and error correction adds another layer of complexity and resource requirements.

Despite these challenges, for many quantum computing platforms with restricted native gate sets, magic state distillation remains the most viable path to achieving universality and unlocking the full potential of fault-tolerant quantum computation. Ongoing research focuses on developing more efficient distillation protocols and error correction codes to reduce the resource overhead and improve the overall performance of quantum algorithms.

References

[1] F. Arute et al., Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[2] Google Quantum AI and Collaborators et al., Quantum error correction below the surface code threshold, Nature (2024).

[3] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff, Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits, arXiv:1910.09534 (2019).

[4] F. Pan, K. Chen, and P. Zhang, Solving the Sampling Problem of the Sycamore Quantum Circuits, Physical Review Letters **129**, 090502 (2022).

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Anniversary edition (Cambridge University Press, Cambridge ; New York, 2011).

[6] A. M. Turing, On Computable Numbers, with an Application to the Entscheidungsproblem, Proceedings of the London Mathematical Society **s2-42**, 230 (1937).

[7] R. Horodecki, M. Horodecki, and K. Horodecki, Quantum entanglement, Reviews of Modern Physics **81**, 865 (2009).

[8] L. Hoddeson, The Discovery of the Point-Contact Transistor, Historical Studies in the Physical Sciences **12**, 41 (1981).

[9] R. P. Feynman, Simulating physics with computers, International Journal of Theoretical Physics **21**, 467 (1982).

[10] D. Deutsch, Quantum theory, the Church–Turing principle and the universal quantum computer, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **400**, 97 (1985).

[11] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences **439**, 553 (1992).

[12] E. Bernstein and U. Vazirani, Quantum Complexity Theory, SIAM Journal on Computing **26**, 1411 (1997).

[13] D. R. Simon, On the Power of Quantum Computation, SIAM Journal on Computing **26**, 1474 (1997).

[14] P. W. Shor, *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (IEEE Comput. Soc. Press, Santa Fe, NM, USA, 1994), pp. 124–134.

[15] L. K. Grover, *A Fast Quantum Mechanical Algorithm for Database Search*, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC '96* (ACM Press, Philadelphia, Pennsylvania, United States, 1996), pp. 212–219.

[16] W. K. Wootters and W. H. Zurek, A single quantum cannot be cloned, Nature **299**, 802 (1982).

[17] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, Physical Review A **71**, 022316 (2005).

[18] M. M. Wilde, *Quantum Information Theory*, 2nd ed (Cambridge university press, Cambridge, 2017).

[19] E. Schrödinger, Discussion of Probability Relations between Separated Systems, Mathematical Proceedings of the Cambridge Philosophical Society **31**, 555 (1935).

[20] A. Einstein, B. Podolsky, and N. Rosen, Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?, Physical Review **47**, 777 (1935).

[21] J. S. Bell, On the Einstein Podolsky Rosen paradox, Physics Physique Fizika **1**, 195 (1964).

[22] S. J. Freedman and J. F. Clauser, Experimental Test of Local Hidden-Variable Theories, Physical Review Letters **28**, 938 (1972).

[23] A. Aspect, J. Dalibard, and G. Roger, Experimental Test of Bell's Inequalities Using Time-Varying Analyzers, Physical Review Letters **49**, 1804 (1982).

[24] M. F. Pusey, J. Barrett, and T. Rudolph, On the reality of the quantum state, Nature Physics **8**, 475 (2012).

[25] N. D. Mermin, Bringing home the atomic world: Quantum mysteries for anybody, American Journal of Physics **49**, 940 (1981).

[26] W. Wootters, Entanglement of formation and concurrence, Quantum Information and Computation **1**, 27 (2001).

[27] W. Dür, G. Vidal, and J. I. Cirac, Three qubits can be entangled in two inequivalent ways, Physical Review A **62**, 62314 (2000).

[28] D. M. Greenberger, M. A. Horne, A. Shimony, and A. Zeilinger, Bell's theorem without inequalities, American Journal of Physics **58**, 1131 (1990).

[29] N. P. De Leon, K. M. Itoh, D. Kim, K. K. Mehta, T. E. Northup, H. Paik, B. S. Palmer, N. Samarth, S. Sangtawesin, and D. W. Steuerman, Materials challenges and opportunities for quantum computing hardware, Science **372**, eabb2823 (2021).

[30] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, Physical Review Letters **103**, 150502 (2009).

[31] D. P. DiVincenzo, *Topics in Quantum Computers*, in *Mesoscopic Electron Transport*, edited by L. L. Sohn, L. P. Kouwenhoven, and G. Schön (Springer Netherlands, Dordrecht, 1997), pp. 657–677.