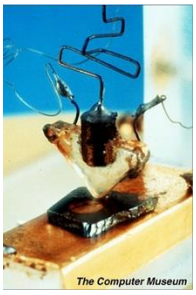
# Snapshot of modern classical computers

1936: “On computable numbers, with an application to the Entscheidungsproblem”, Alan Turing




1947: First transistor (Bell Labs)

1958: First integrated circuit


1975: Altair 8800, one of the first micro computers

1981: Osborne 1, first true mobile computer

1989: first Macintosh

# Brief history of quantum computing

1980s: Richard Feynman

- Classical computers are very inefficient in simulating quantum systems ($e^N$)
- Computers are physical objects
- Why not creating computers following quantum laws?
- They will efficiently simulate at least themselves, maybe more, thus will be faster than any classical computer

Richard Feynman

*On quantum physics and computer simulation*

. . . there is plenty of room to make [computers] smaller. . . . nothing that I can see in the physical laws . . . says the computer elements cannot be made enormously smaller than they are now. In fact, there may be certain advantages.
—1959

Might I say immediately . . . we always have had a great deal of difficulty in understanding the world view that quantum mechanics represents. . . . I cannot define the real problem, therefore I suspect there’s not a real problem, but I’m not sure there’s no real problem.

I mentioned . . . the possibility . . . of things being affected not just by the past, but also by the future, and therefore that our probabilities are in some sense “illusory.” We only have the information from the past and we try to predict the next step, but in reality it depends upon the near future . . .I’m trying to get . . . you people who think about computer-simulation possibilities to . . . digest . . . the real answers of quantum mechanics and see if you can’t invent a different point of view than the physicists . . .
. . . the discovery of computers and the thinking about computers has turned out to be extremely useful in many branches of human reasoning. For instance, we never really understood how lousy our understanding of languages was, the theory of grammar and all that stuff, until we tried to make a computer which would be able to understand language . . . I . . . was hoping that the computer-type thinking would give us some new ideas . . .
. . . trying to find a computer simulation of physics seems to me to be an excellent program to follow out. . . . the real use of it would be with quantum mechanics. . . . Nature isn’t classical . . . and if you want to make a simulation of Nature, you’d better make it quantum mechanical, and by golly it’s a wonderful problem, because it doesn’t look so easy.
—1981

Feynman, R. 1959. There’s Plenty of Room at the Bottom. Talk given at the annual meeting of the American Physical Society at Caltech. (Excerpt reprinted with permission from Caltech’s *Engineering and Science*.)
———. 1981. Simulating Physics with Computers. Keynote address delivered at the MIT Physics of Computation Conference. Published in *Int. J. Theor. Phys.* **21** (6/7), 1982. (Excerpts reprinted with permission from the *International Journal of Theoretical Physics*.)

# Brief history of quantum computing

1980: Paul Benioff describes the first QM model of computation

1985: David Deutsch describes first universal QC

1992: Deutsch-Jozsa algorithm

1993: Simon's algorithm

1994: Shor's algorithm

1995: Monroe & Wineland realize the first quantum gate (CNOT) with trapped ions

1996: Grover's algorithm

1998: First realization of a quantum algorithm (Deutsch-Jozsa), with NMR

1999: Nakamura and Tsai demonstrate superconducting qubits

2000: Fahri et al. propose Adiabatic Quantum Computation

2000: Raussendorf et al: One way (measurement based) quantum computing

2001: Shor's algorithm implemented to factorize 15

2014: Fahri et al. QAOA (Quantum Approximate Optimization Algorithm)

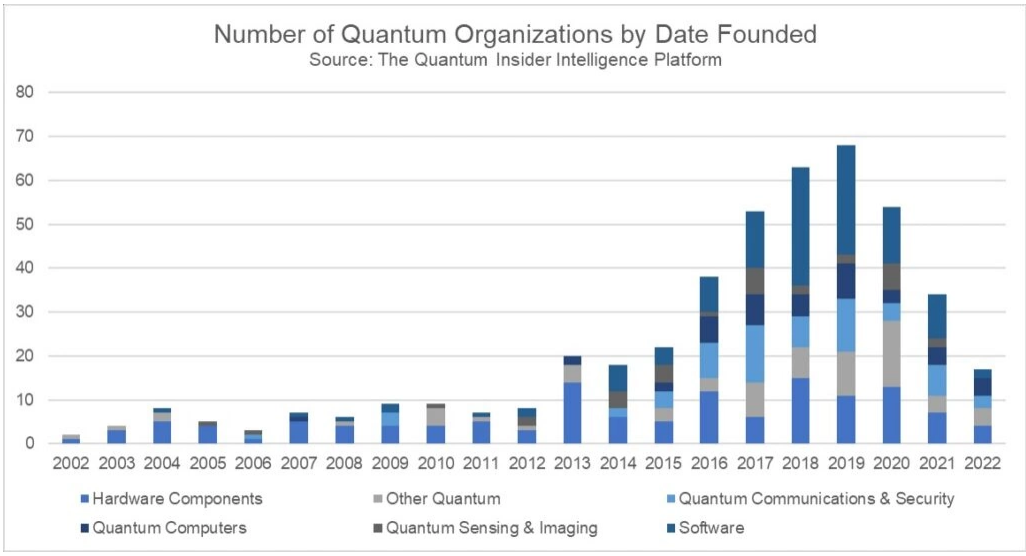
2016: IBM Quantum Experience

2019: Quantum supremacy by Google (?)

2023: First tests of error correcting schemes (Google)

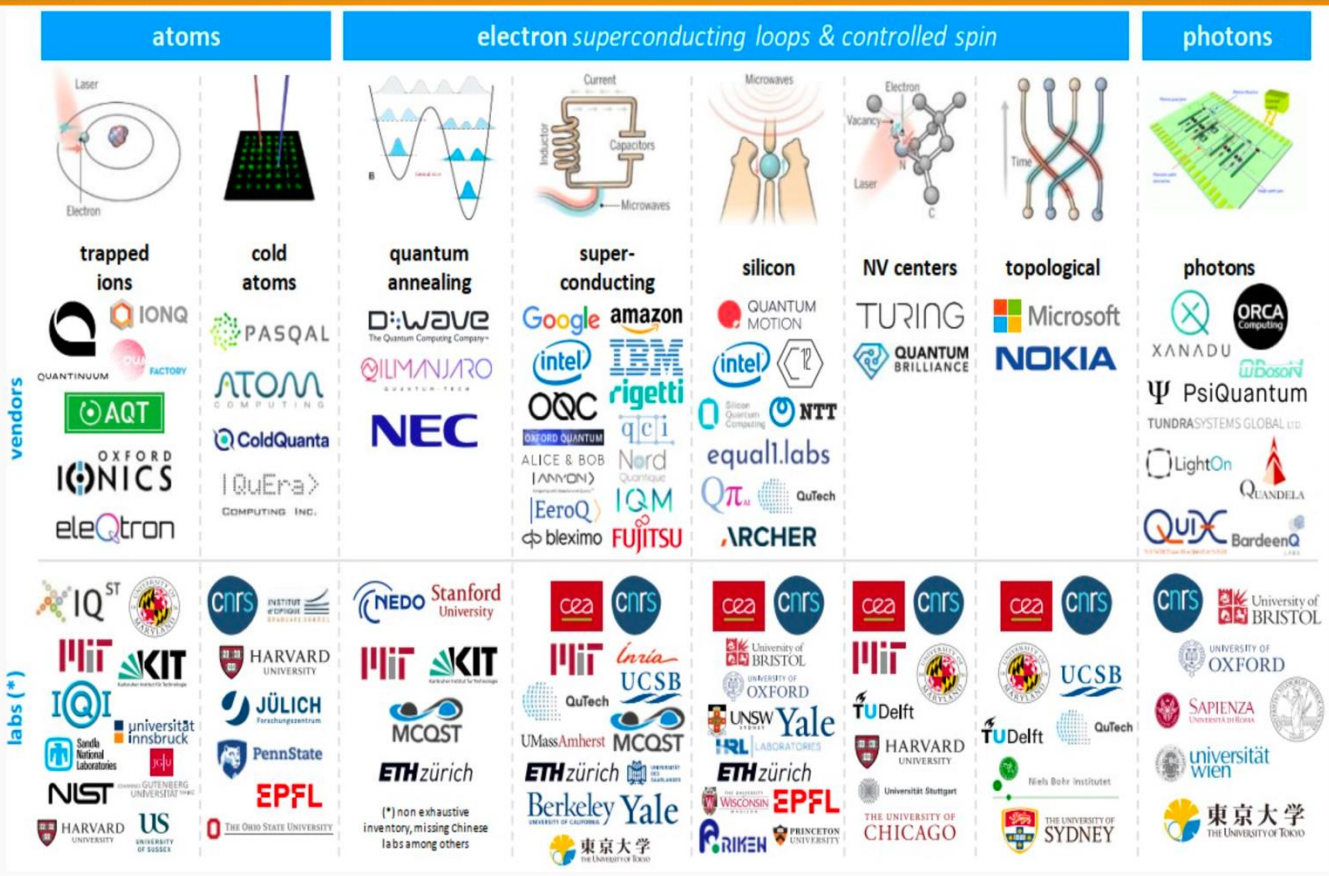
(from "Timeline of Quantum Computing", Wikipedia)

# The Rise of Quantum Computing Companies

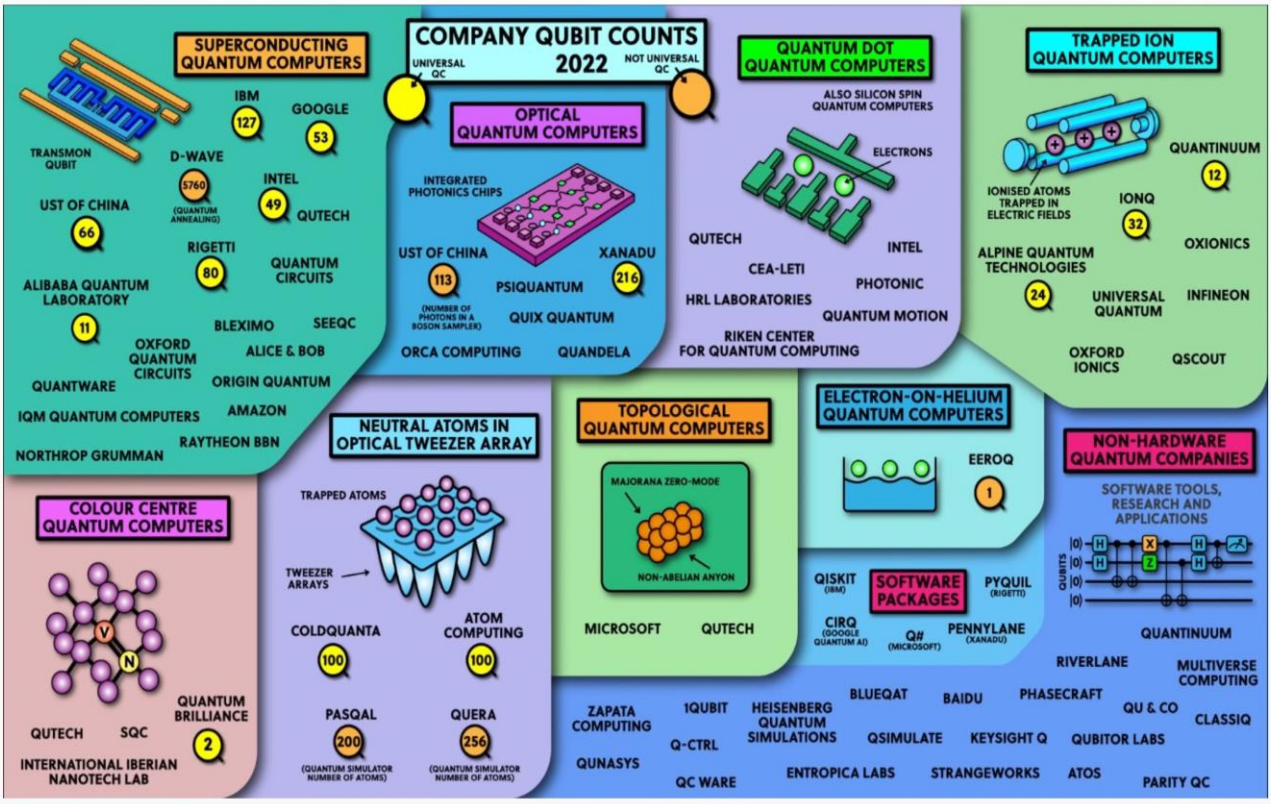



*Source: The Quantum Insider Intelligence Platform*

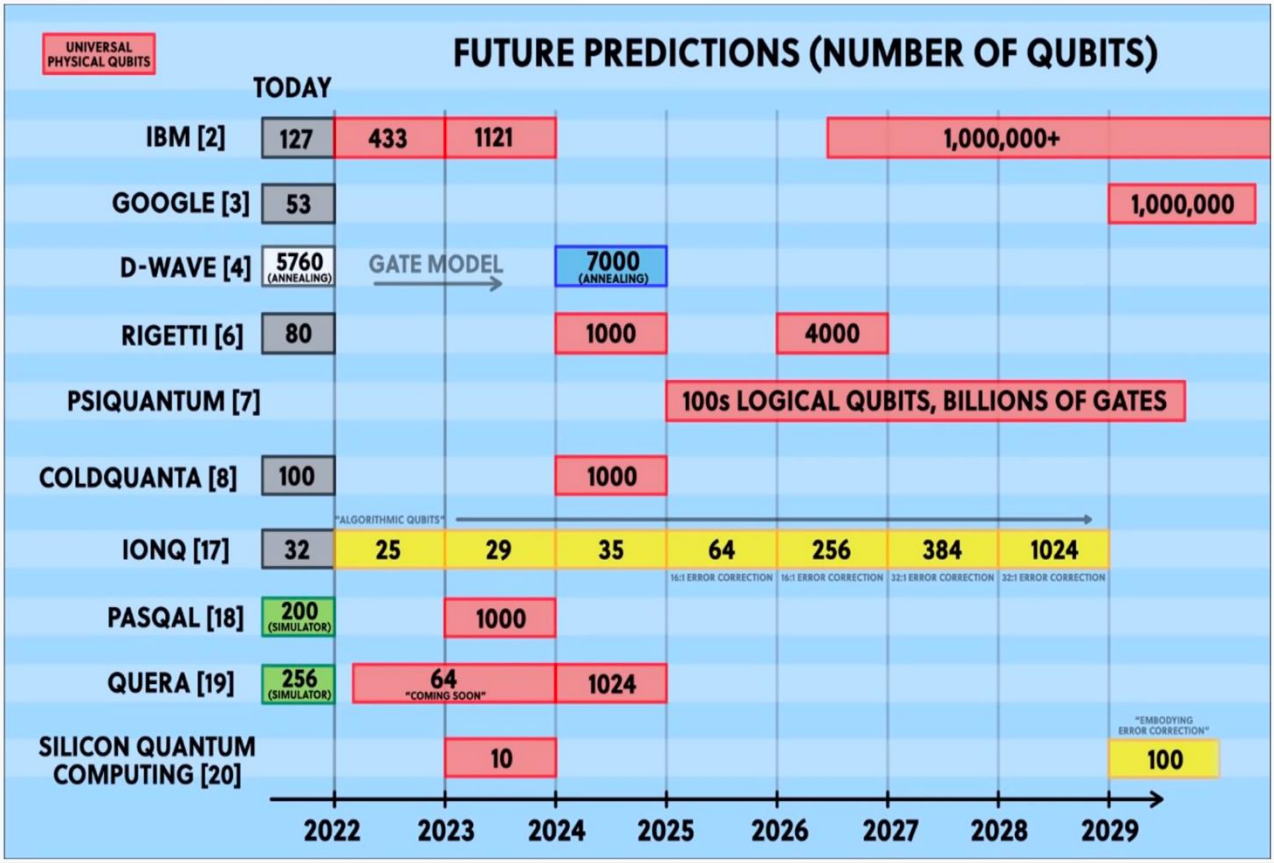
# Qubit Platforms



# Qubit Counts (2021)




# Qubit Count Predictions (2030)



# Physical realization of Quantum Computers

Superconducting qubits: In superconductors, the basic charge carriers are pairs of electrons (known as Cooper pairs), rather than single fermions as found in typical conductors. These implement superconducting electronic circuits using superconducting qubits as artificial atoms; the two logic states are the ground state and the excited state. Superconducting quantum computing devices are typically designed in the radio-frequency spectrum, cooled in dilution refrigerators below 15 mK (millikelvins) and addressed with conventional electronic instruments, e.g. frequency synthesizers and spectrum analyzers.

The largest number of qubits is about 433 (IBM)

Companies:

1. IBM
2. Google
3. Intel
4. Rigetti

Trapped ions: the ions are suspended in free space using electromagnetic fields. Qubits are stored in stable electronic states of each ion, and quantum information can be transferred through the collective quantized motion of the ions in a shared trap (interacting through the Coulomb force). Lasers are applied to induce coupling between the qubit states (for single qubit operations) or coupling between the internal qubit states and the external motional states (for entanglement between qubits).

The largest number of particles to be controllably entangled is about 20-30 trapped ions.

Companies:

1. Quantinuum (2021 – Cambridge UK)
2. IonQ (2015 – Maryland USA)
3. Quantum Factory (2018 – Munich DE)
4. Alpine Quantum Technologies (2018 – Austria AT)
5. Oxford Ionics (2019 – Begbroke UK)
6. EleQtron (2020 – Siegen DE)

Neutral atoms: the atoms are trapped in optical lattices, and manipulated with lasers. Qubits are encoded in the internal states. To turn on interactions between qubits, researchers target a pair of adjacent atoms with a laser pulse that excites one of them to a high-energy state called a Rydberg state, in which a valence electron orbits far from the nucleus. The Rydberg atom's strong electric dipole interactions prevent the laser from also exciting its neighbour, an effect known as a Rydberg blockade, but it's impossible to know which of the atoms was excited. The result is a single excitation shared between two qubits that can't be described separately—the characteristic feature of entanglement.

The largest number of particles to be controllably entangled is about 10 neutral atoms. Overall they can control hundred of atoms.

Companies:

1. Pasqual (2019 – Paris Region FR)
2. Atom Computing (2018 – Berkeley USA)
3. ColdQuanta (2007 – Boulder USA)
4. Quera Computing (2018 – Boston USA)

Photons: It is a type of quantum computing that uses photons as a representation of qubits. The main advantages are simple components, the ability to run a variety of quantum operations, and most importantly, photonic quantum computers can perform at room temperature, which reduces the size of the extreme cooling systems.

Companies:

1. Xanandu Quantum Technologies (2016 – Canada)
2. ORCA Computing (2019 – London UK)
3. PsiQuantum (2015 – Silicon Valley USA)
4. TundraSystem Global (2014 – Cardiff UK)
5. Quandela (2017 – Paris FR)
6. QuiX Quantum (2019 – Enschede NL)

# Cloud-based Quantum Computing

IBM Q Experience (superconducting qubits)

Xanadu (photonic quantum computer)

Forest by Rigetti Comuting (superconducting qubits)

Several simulators of quantum computers

# Classical computation

Several models studied for the theory of classical computation

- Turing machines
- High-level programmable languages
- Boolean circuits

So far, the **Boolean circuit model** is by far the easiest model to generalize to quantum computation, being the closest to physical implementation. We will review it very briefly.

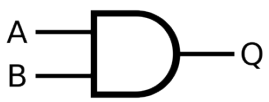
# Boolean circuit model

**Proposition**: Any Boolean function

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

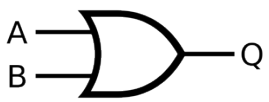
is computable by a Boolean circuit C using just AND, OR and NOT **gates** (in other words, AND, OR, NOT are **universal** for classical computation)

AND $A \wedge B$



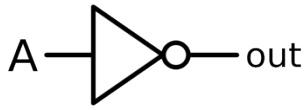

| INPUT | | OUTPUT |
|---|---|---|
| A | B | A AND B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR $A \vee B$




| INPUT A | B | OUTPUT A + B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

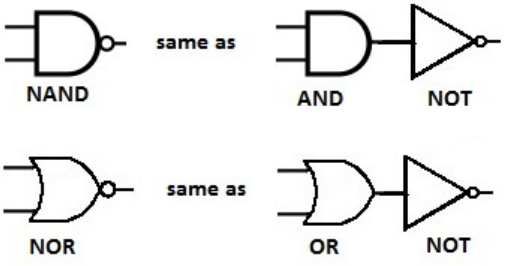
NOT $\neg A$




| INPUT | OUTPUT |
|---|---|
| A | NOT A |
| 0 | 1 |
| 1 | 0 |

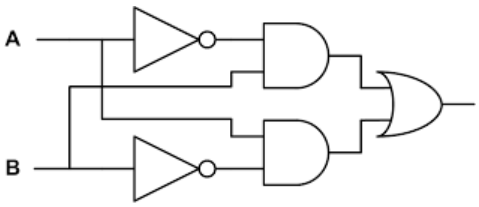
# Example 1: NAND, NOR, XOR

Pronunciation: ex-or




| INPUT | | OUTPUT |
|---|---|---|
| A | B | A NAND B |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| INPUT | | OUTPUT |
|---|---|---|
| A | B | A NOR B |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

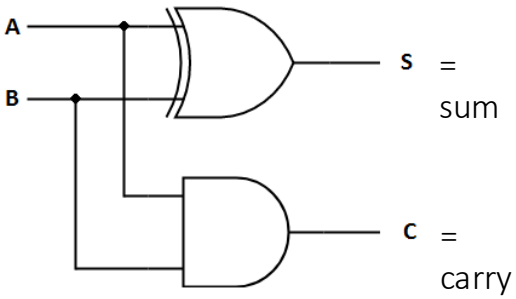



=

XOR $A \oplus B$

| INPUT | | OUTPUT |
|---|---|---|
| A | B | A XOR B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Example 2: Half adder




| Input | | Output | |
|---|---|---|---|
| A | B | Sum | Carry |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Note the **elements of a circuit**:

- Wires
- Gates
- Input on the left
- Output on the right

**Size of a circuit** = number of gates

**DUPE gate**: duplicates bits

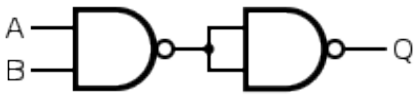
# NAND is universal

The number of fundamental gates can be reduced

**Proposition**: The NAND and DUPE gates are universal for computation

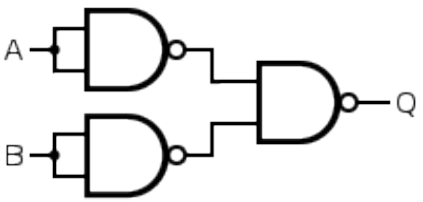
**Desired AND Gate**


**NAND Construction**

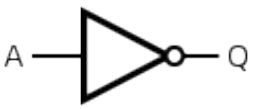



**Desired OR Gate**


**NAND Construction**




**Desired NOT Gate**




**NAND Construction**

# Reversible Computation

Logical gates are not always reversible:

- NOT is reversible
- AND is irreversible

| INPUT | OUTPUT |
|---|---|
| A | NOT A |
| 0 | 1 |
| 1 | 0 |

| INPUT | | OUTPUT |
|---|---|---|
| A | B | A AND B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

The laws of Physics are reversible, therefore if computation is implemented physically, it should be written in terms of reversible gates ➜ **Universal reversible computation** should be possible, there should exists a **universal set of reversible gates.**

This problem was studied in the '60s and '70s by Landauer e Bennett in connection with **thermodynamics**.

They were considering whether it is possible to have circuits made only of reversible gates, thus dissipating no energy. This was thought to be an important issue at that time. In fact now supercomputers needs **heavy cooling systems**. Yet it is not the most pressing one.

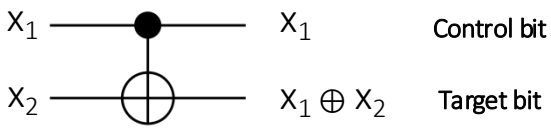
Reversible computation is important in the context of **quantum computation**, because – as we will see – quantum circuits need to be reversible in order to work properly.

# Reversible gates - CNOT gate

**Definition:** A Boolean gate G is said to be reversible if it has the same number of inputs and outputs, and its mapping is bijective.

Some important new reversible gates

CNOT =

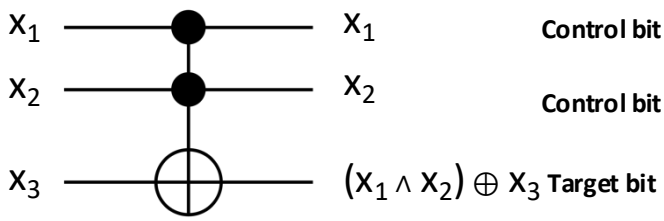



| INPUT | | OUTPUT | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

If the control bit is 0, the target bit is left unchanged, otherwise it is flipped

# CCNOT gate

**CCNOT** =




| INPUT | | | OUTPUT | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

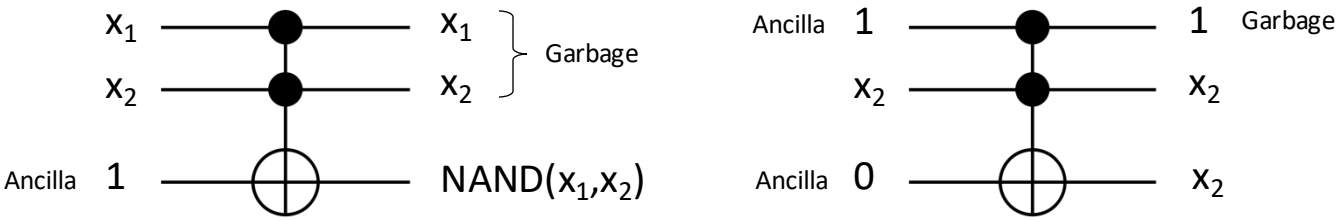
A NOT gate is applied to the target bit only if both control bits are 1, otherwise it is left unchanged. This is also called **Toffoli gate.**

Comments:

- With the same logic, one can build the CCCNOT = $C^3$NOT gate and in general the **$C^n$NOT gate**.
- The CNOT and CCNOT are their own inverse. If applied twice, they give the **identity**. This is not always the case.

# Universal reversible gates

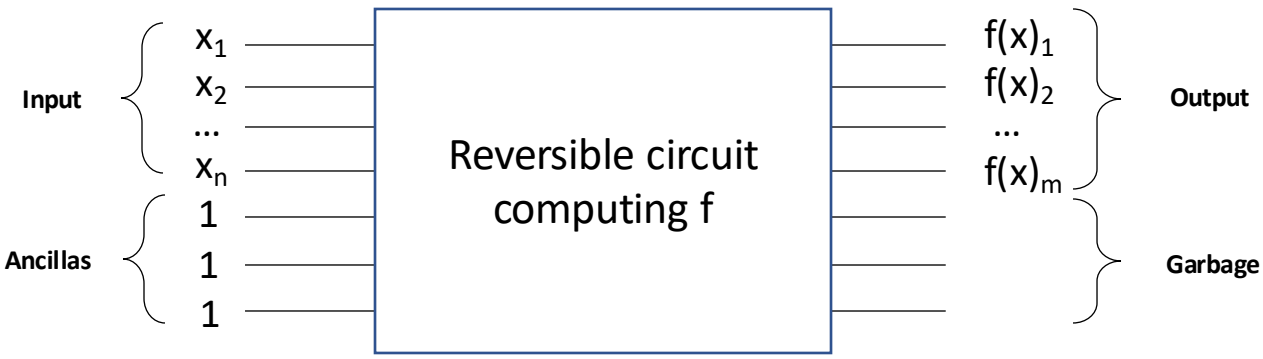
CCNOT can be used to simulate NAND and DUPE




Theorem: The **CCNOT gate is universal**, assuming that ancilla inputs and garbage outputs are allowed. Any standard Boolen circuit can be **efficiently** transformed into a reversible circuit.

# Reversible circuit

So far ancillas were sometimes 0 sometimes 1. They can be initialized to the same value, let's say 1, by means of a NOT gate. A reversible circuit computing $f: \{0,1\}^n \rightarrow \{0,1\}^m$ will then look as follows




The number of inputs and outputs is the same; the number of wires never changes. In fact, we can stop thinking about wires and think about each bit being carried in its own register, keeping its identity throughout the computation.

# Probabilistic (randomized) computation

For finite probabilistic models, see

Van Kampen "Stochastic process in Physics and Chemistry", chapter V.2

https://arxiv.org/pdf/2209.14902 (section 4.3)

https://arxiv.org/pdf/1405.0303 (section 2)

We can open to the possibility that the value of a bit is not known with certainty

0 or 1 ⟶ { 0 with probability $p_1$ ; 1 with probability $p_2$

deterministic bit — random bit

Note: the physics has not changed, we simply do not know the value of the bit.

The mathematical model changes, though. There are some computational tasks which we know how to provably solve efficiently using randomized computation (like generating prime numbers) but which we do not know how to provably solve efficiently using deterministic computation.

However there **should not** be any fundamental difference between the two models of computation, since they are based on the same physics.

We will introduce a **new notation** to deal with probabilistic computation, which will bring us a bit closer to quantum computation.

| Basic notation | Vector notation | Abstract (Dirac) notation |
|---|---|---|
| 0 | $\begin{pmatrix}1\\0\end{pmatrix}$ | $\|0\rangle$ |
| 1 | $\begin{pmatrix}0\\1\end{pmatrix}$ | $\|1\rangle$ |
| 0 with probability $p_1$<br>1 with probability $p_2$ | $\begin{pmatrix}p_1\\p_2\end{pmatrix}$ | $p_1\|0\rangle + p_2\|1\rangle$ |

# Gates in the new notation: the NOT gate

In the new notation, **gates** are represented by **matrices**

$$\textbf{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Then

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

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

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} p_2 \\ p_1 \end{pmatrix}$$

For all other gates, we need to understand how to represent two and more bits.

# Two (and more) random bits

With two bits, we have four possible states

$$00 \rightarrow \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \qquad 01 \rightarrow \begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix}$$

$$10 \rightarrow \begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \qquad 11 \rightarrow \begin{pmatrix}0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix}$$

Tensor product (we'll come back on this soon)

# Two-bit gates: the AND gate

$$\textbf{AND} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note: it is not a square matrix, because the gate is not reversible

$$00 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0 \qquad 01 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0$$

$$10 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0 \qquad 11 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1$$

# Two-bit gates: the CNOT gate

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

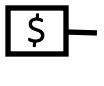
Note: it is a square matrix, because the gate is reversible

$$00 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = 00 \qquad 01 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = 01$$

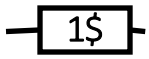
$$10 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 11 \qquad 11 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = 10$$

# A truly probabilistic gate

We introduce two new gates

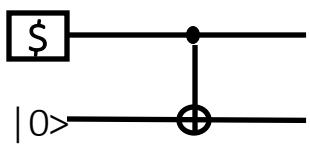
COIN =  $\rightarrow \begin{pmatrix} ½ \\ ½ \end{pmatrix}$

It has no input and a single bit output. It generates randomly either a 0 or a 1, with probability ½ each. It is like **fair coin tossing.**

1COIN =  $= \begin{pmatrix} 1 & ½ \\ 0 & ½ \end{pmatrix}$

If the input bit is 0, it is left unchanged. If it is 1, it is replaced by a COIN.

# Example 1



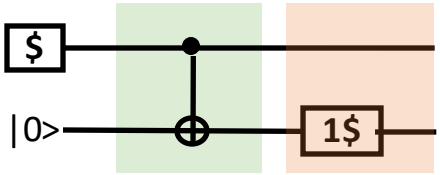

With probability ½ the input bit 00 and with probability ½ it is 10. In the first case the CNOT will leave in unchanged, in the second case it will changed into 11.

In mathematical terms

$$\begin{pmatrix} ½ \\ ½ \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} ½ \\ 0 \\ ½ \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ½ \\ 0 \\ ½ \\ 0 \end{pmatrix} = \begin{pmatrix} ½ \\ 0 \\ 0 \\ ½ \end{pmatrix} = ½ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + ½ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

00 11

# Example 2




$$\begin{pmatrix} ½ \\ ½ \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} ½ \\ 0 \\ ½ \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ½ \\ 0 \\ ½ \\ 0 \end{pmatrix} = \begin{pmatrix} ½ \\ 0 \\ 0 \\ ½ \end{pmatrix} \rightarrow \begin{pmatrix} 1 & ½ & 0 & 0 \\ 0 & ½ & 0 & 0 \\ 0 & 0 & 1 & ½ \\ 0 & 0 & 0 & ½ \end{pmatrix} \begin{pmatrix} ½ \\ 0 \\ 0 \\ ½ \end{pmatrix} = \begin{pmatrix} ½ \\ 0 \\ ¼ \\ ¼ \end{pmatrix}$$

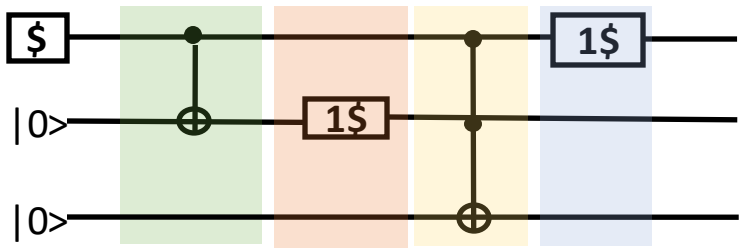
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & ½ \\ 0 & ½ \end{pmatrix}$$

Using the Dirac notation ($|0> \otimes |0> = |00>$, and same for others)

½ |00> + ½ |10> → ½ |00> + ½ |11> → ½ |00> + ½ ( ½ |10> + ½ |11>)

= ½ |00> + ¼ |10> + ¼ |11> = $\begin{pmatrix} ½ \\ 0 \\ ¼ \\ ¼ \end{pmatrix}$

# Example 3




½ |000> + ½ |100> ➜ ½ |000> + ½ |110>

➜ ½ |000> + ½ ( ½ |100> + ½ |110>) = ½ |000> + ¼ |100> + ¼ |110>

➜ ½ |000> + ¼ |100> + ¼ |111>

➜ ½ |000> + ¼ ( ½ |000> + ½ |100>) + ¼ ( ½ |011> + ½ |111>)

= $^5/_8$ |000> + $^1/_8$ |100> + $^1/_8$ |011> + $^1/_8$ |111>

# Comment 1

We used the formalism of **linear algebra** for probabilistic computation because "**ignorance propagates linearly**".

If a physical system is either in state x with probability p or in state y with probability q, and x evolves into X and y into Y, then at the end the system will be in state X with probability p or in state Y with probability q. In Dirac notation:

p|x> + q|y> ➜ p|X> + q|Y> = p T[|x>] + q T[|y>] = T[p|x> + q|y>]

The evolution **operator T is linear**, and can be represented by a matrix.

# Comment 2

**Measurements simply reveal the true state of the system**, which was unknown to us before the measurement. **After the measurement**, the information about **the state of the system changes**, and with it the probability distribution. With reference to the previous example

1. We measure the three bits and find 000:

$$^5/_8\,|000\rangle + {}^1/_8\,|100\rangle + {}^1/_8\,|011\rangle + {}^1/_8\,|111\rangle \;\rightarrow\; |000\rangle$$

This happens with probability $^5/_8$

2. We measure the first bit and find 0; this happens with probability $^5/_8 + {}^1/_8 = {}^3/_4$

$$^5/_8\,|000\rangle + {}^1/_8\,|100\rangle + {}^1/_8\,|011\rangle + {}^1/_8\,|111\rangle \;\rightarrow\; \frac{{}^5/_8\,|000\rangle + {}^1/_8\,|011\rangle}{{}^3/_4}$$

$$= {}^5/_6\,|000\rangle + {}^1/_6\,|011\rangle$$

We can call it "**collapse**" of the probability. It is not a real physical phenomenon. It is **Bayes rule**: P(A|B) = P(B|A) P(A) / P(B). In our case:

$$P(|000\rangle|\text{“0”}) = P(\text{“0”}||000\rangle)\,P(|000\rangle) / P(\text{”0”}) = 1 \times {}^5/_8 \div {}^3/_4 = {}^5/_6$$

# Rules of probabilistic classical computation

1. The **state** of a single probabilistic bit is given by a vector in $R^2$, or in Dirac notation:

$|x\rangle = p|0\rangle + q|1\rangle$, with $p,q \in \mathbb{R}$, and $p+q=1$.

The **coefficients** give the **probabilities** for the bit to have that value.

States for **multiple bits** are constructed via **tensor product** of $R^2$

Two bits: $|xy\rangle = |x\rangle \otimes |y\rangle$

Three bits: $|xyz\rangle = |x\rangle \otimes |y\rangle \otimes |z\rangle$, and so on

Why tensor products, and not – for example – Cartesian product?

Take for example three bits. There are 8 possible configurations: 000, 001, 010, 011, 100, 101, 110, 111. The register can be in any of these 8 states, and the information propagates linearly (without interference among the states), therefore they behave like **linearly independent states**.

This means that one needs 8 basis states in the vector space, which is what is provided by the tensor product, not by the Cartesian product.

# Rules of probabilistic classical computation

2. **Gates** are implemented by **linear operators**, i.e. matrices.

Gates can be either reversible (square invertible matrices) or irreversible (for example rectangular matrices).

As we saw that computation can always be made reversible, without loss of generality we can say that gates are implemented by linear invertible operators (**NxN invertible stochastic matrices**).

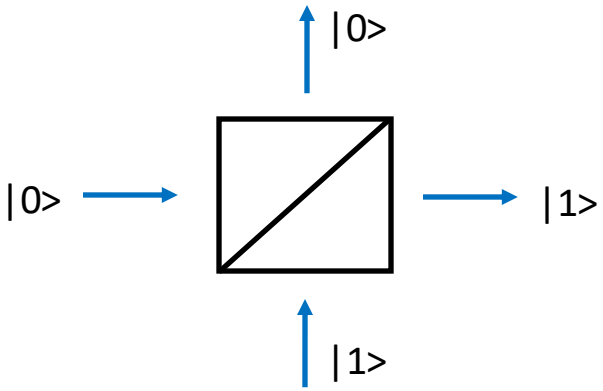
Of course, they have to preserve probabilities.

3. **Measurements** are updates of information. The states changes according to Bayes rule ("collapse" of the state)

As we will see, the rules of quantum computation are almost similar, but with fundamental differences.

# Preview of Quantum Computation

**Beam splitters (BS)** are optical devices, which split the path of a photon in two: once a photon has entered, there is ½ probability that it goes one way, and ½ probability that it goes the other way. It is a **probabilistic gate.**

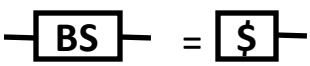



If we associate the value of the bit to the path of the photon (instead of the voltage as in standard computers), then we have

$|0\rangle \rightarrow \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle$

$|1\rangle \rightarrow \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle$

Then




$|x\rangle \rightarrow \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle$

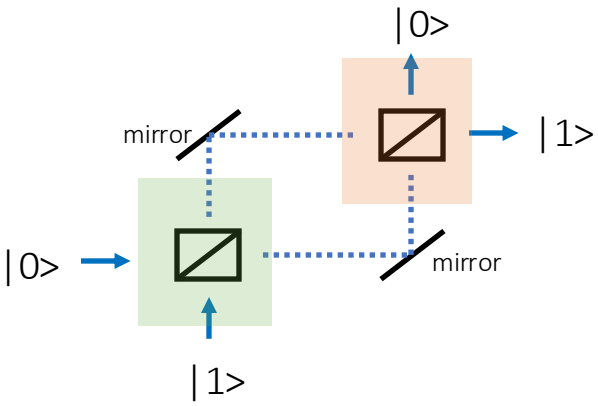
Whatever the input state, it generates an equal weighted distributions of 0 and 1. The matrix representation is:

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

In fact: $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$ since $p+q=1$

# Preview of Quantum Computation

But now we can do the following optical construction:



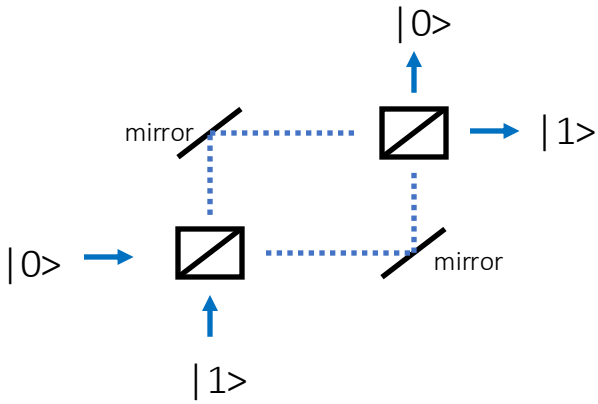

This is equivalent to the following circuit


Since

$$\begin{pmatrix} ½ & ½ \\ ½ & ½ \end{pmatrix} \begin{pmatrix} ½ & ½ \\ ½ & ½ \end{pmatrix} = \begin{pmatrix} ½ & ½ \\ ½ & ½ \end{pmatrix}$$

In a classical picture (coin tossing), this makes perfectly sense

But this is not what happens. What happens it:




$|0\rangle \rightarrow |0\rangle$

$|1\rangle \rightarrow |1\rangle$

How is this possible? The answer is that photons are quantum: they cannot be thought as particles which follow **one path or the other**. They are more like waves, which split in two, **interfere** and then recombine

# Preview of Quantum Computation

We will see how this is described by quantum mechanics, but the essence is the following: how can we **destroy probabilities**?

We have to justify

$|0\rangle \rightarrow \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle \rightarrow |0\rangle$

first BS second BS

Instead of

$|0\rangle \rightarrow \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle \rightarrow \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle$

first BS second BS

We destroy probabilities with negative (in general, complex) numbers. But what does it mean to have negative probabilities? The solution of QM is:

Bit $\rightarrow$ $p|0\rangle + q|1\rangle$ with $p,q \in \mathbb{R}^+$ and $p+q=1$

probabilities

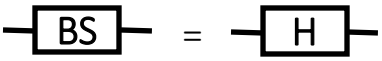
changed into

Qubit $\rightarrow$ $a|0\rangle + b|1\rangle$ with $a,b \in \mathbb{C}$ and $|a|^2 + |b|^2 = 1$

amplitudes Probabilities
(they remain always positive)

# Preview of Quantum Computation

The BS is mathematically described by




Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ !!!

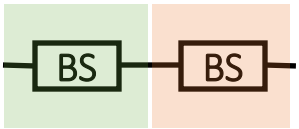
Then


$|0\rangle \rightarrow {}^{1}/_{\sqrt{2}}\,|0\rangle + {}^{1}/_{\sqrt{2}}\,|1\rangle$

$|1\rangle \rightarrow {}^{1}/_{\sqrt{2}}\,|0\rangle - {}^{1}/_{\sqrt{2}}\,|1\rangle$

In both cases, probabilities are 50% of getting the value 0 or 1

But now



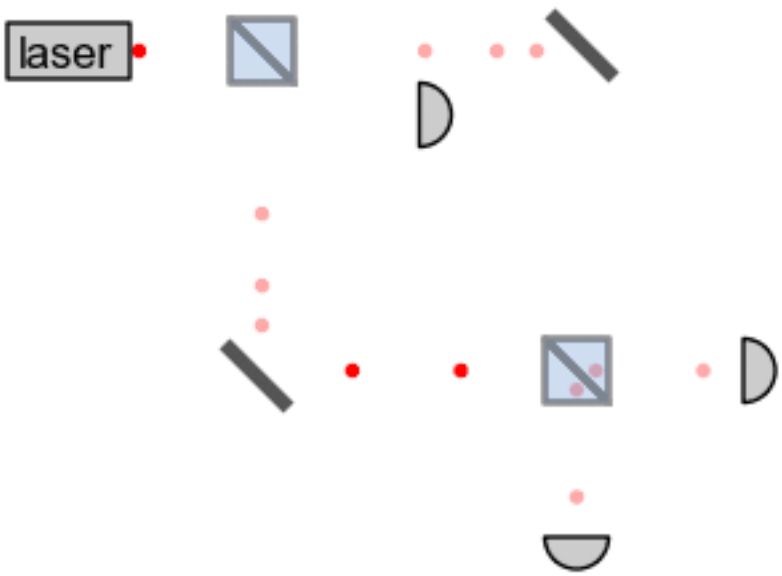

$= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

After the second BS, the bit takes the initial value

What happens physically is that the **photon** behaves like a **wave**. There can be **constructive interference**, which mathematically is expressed by amplitudes **adding**, and **destructive** interference, which mathematically is expressed by amplitudes **subtracting**. This is the role of negative numbers.

# Preview of Quantum Computation

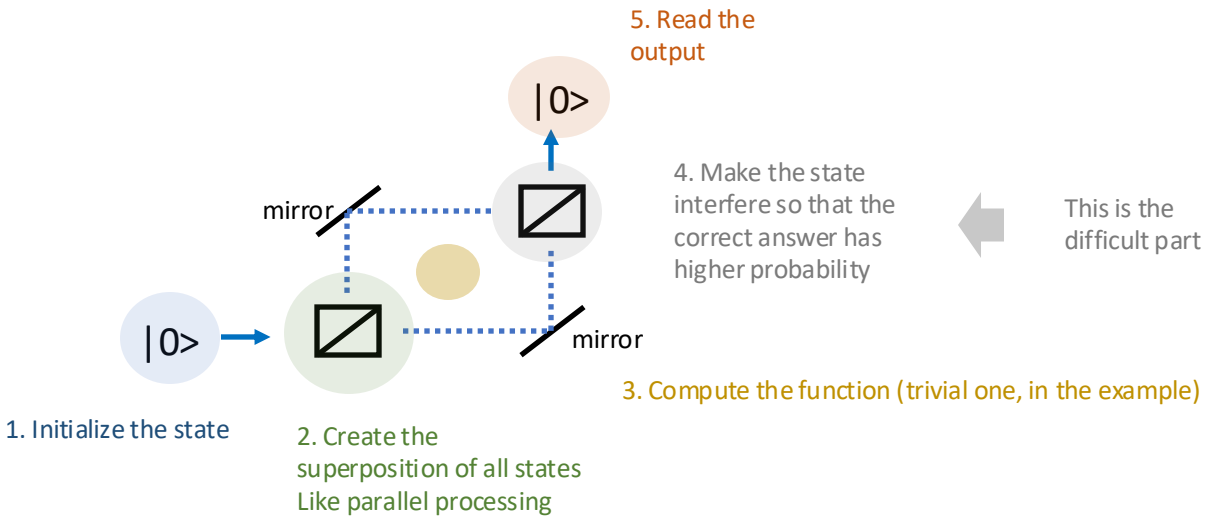
The surprising thing is that if we measure the photon right after the first BS and before it enters the second one, we will not find it half here and half there, as it would happen with classical waves. It will always be **either here or there**, and the wave behaviour is destroyed.




Understanding what this means brings into the **foundations of quantum mechanics**, which is beyond the scope of the present course.

# Quantum Computation

The essence of a quantum computation is the following




*The art of quantum computing is to make the different terms of the superposition interfere in such a way to maximize the correct answer, in a number of steps which is smaller than for any classical algorithm*.

# Complex linear algebra

The basic mathematical objects in Quantum Mechanics are **state vectors** and **linear operators** (matrices). Because the theory is fundamentally linear, and the probability amplitudes are complex numbers, the mathematics underlying quantum mechanics is **complex linear algebra**.

# Vectors

**Vectors** are members of a complex vector space, or **Hilbert space**, with an associated inner product.

It is important to remember that these abstract mathematical objects represent physical things, and should be considered independent of the particular representations they are given by choosing a particular basis.

State vectors in quantum mechanics are written in **Dirac notation**. The basic object is the **ket-vector** $|\psi\rangle$, which (given a particular basis) can be represented as a *column vector*. The adjoint of a ket-vector is a **bra-vector** $\langle\psi|$, represented as a *row vector*.

$$|\psi\rangle = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix}, \quad \langle\psi| = (\alpha_1^* \cdots \alpha_N^*) = |\psi\rangle^\dagger$$

If the vector $|\psi\rangle$ is normalized, that means

$$\langle\psi|\psi\rangle = \sum_{j=1}^{N} |\alpha_j|^2 = 1$$

## Inner and Outer Products.

Given two vectors $|\psi\rangle$ and $|\varphi\rangle$,

$$|\psi\rangle = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix}, \quad |\phi\rangle = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_N \end{pmatrix},$$

the **inner product** between them is written

$$\langle\phi|\psi\rangle = (\beta_1^* \cdots \beta_N^*) \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \sum_j \beta_j^* \alpha_j$$

The inner product is independent of the choice of basis. $\langle\varphi|\psi\rangle$ is called a *bracket*. Note that $\langle\varphi|\psi\rangle = \langle\psi|\varphi\rangle^*$, and for a normalized vector $\langle\psi|\psi\rangle = 1$. If two vectors are orthogonal then $\langle\psi|\varphi\rangle = 0$.

It is also handy to write the **outer product** (also sometimes called a *dyad*):

$$|\psi\rangle\langle\phi| = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} (\beta_1^* \cdots \beta_N^*) = \begin{pmatrix} \alpha_1\beta_1^* & \cdots & \alpha_1\beta_N^* \\ \vdots & \ddots & \vdots \\ \alpha_N\beta_1^* & \cdots & \alpha_N\beta_N^* \end{pmatrix}$$

A dyad $|\psi\rangle\langle\varphi|$ is a **linear operator**. As we shall see, it is common (and often convenient) to write more general operators as linear combinations of dyads.

## Orthonormal bases.

An orthonormal basis for an $N$-dimensional space has $N$ vectors that satisfy

$$\langle i|j\rangle = \delta_{ij}, \quad \sum_{j=1}^{N} |j\rangle\langle j| = \hat{I}.$$

It is normally most convenient to choose a particular orthonormal basis $\{|j\rangle\}$ and work in terms of it. **As long as one works within a fixed basis, one can treat state vectors as column vectors and operators as matrices**.

# Linear operators

A **linear operator** *O* transforms states to states such that

$$\hat{O}(a|\psi\rangle + b|\phi\rangle) = a\hat{O}|\psi\rangle + b\hat{O}|\phi\rangle$$

for all states $|\psi\rangle$, $|\varphi\rangle$ and complex numbers *a, b*. Given a choice of basis $\{|j\rangle\}$, an operator can be represented by a **matrix**

$$\hat{O} = \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{pmatrix} \equiv [a_{ij}]$$

where

$$\langle i|\hat{O}|j\rangle = a_{ij},$$

are called **matrix elements**. The operator can be written as a sum over outer products

$$\hat{O} = \sum_{ij} a_{ij}|i\rangle\langle j|$$

The matrix representation depends on the choice of basis. We will only be dealing with orthonormal bases in this class.

*Three operations on operators are of particular relevance in Quantum Mechanics: the trace, the commutator and the Hermitian conjugation.*

## The trace.

The trace of an operator is the sum of the diagonal elements:

$$\mathrm{Tr}\{\hat{O}\} = \sum_j \langle j|\hat{O}|j\rangle = \sum_j a_{jj}.$$

A traceless operator has Tr{O} = 0. The trace is *independent of the choice of basis*. If $\{|j\rangle\}$ and $\{|\varphi_k\rangle\}$ are both orthonormal bases, then

$$\mathrm{Tr}\{\hat{O}\} = \sum_j \langle j|\hat{O}|j\rangle = \sum_k \langle \phi_k|\hat{O}|\phi_k\rangle$$

The trace also has the useful *cyclic property:*

$$\mathrm{Tr}\{\hat{A}\hat{B}\} = \mathrm{Tr}\{\hat{B}\hat{A}\}$$

This applies to products of any number of operators:

$$\mathrm{Tr}\{\hat{A}\hat{B}\hat{C}\} = \mathrm{Tr}\{\hat{C}\hat{A}\hat{B}\} = \mathrm{Tr}\{\hat{B}\hat{C}\hat{A}\}$$

This invariance implies that Tr{$|\varphi\rangle\langle\psi|$} = $\langle\psi|\varphi\rangle$.

## The Commutator.

Matrix multiplication is *noncommutative*, in general. That is, in general AB differs from BA. Given two operators A and B the *commutator* is [A,B] = AB – BA.

[A,B] = 0 if and only if a and B commute. Occasionally, one will encounter matrices that *anticommute*: AB = - BA. For example, the Pauli matrices anticommute with each other. In these cases, it is sometimes helpful to define the *anticommutator*:

$$\{\hat{A}, \hat{B}\} \equiv \hat{A}\hat{B} + \hat{B}\hat{A}.$$

## Hermitian Conjugation.

One of the most important operations in complex linear algebra is *Hermitian conjugation*. The Hermitian conjugate $O^\dagger$ is the *complex conjugate* of the *transpose* of an operator O. If in a particular basis

$$O = [a_{ij}] \qquad \text{then} \qquad O^\dagger = [a_{ji}^*]$$

Hermitian conjugation works similarly to transposition in real linear algebra: $(AB)^\dagger = B^\dagger A^\dagger$. When applied to state vectors, $(|\psi\rangle)^\dagger = \langle\psi|$. Similarly, for dyads $(|\psi\rangle\langle\varphi|)^\dagger = |\varphi\rangle\langle\psi|$.

Note that Hermitian conjugation is *not* linear, but rather is *antilinear*:

$$(a\hat{O})^\dagger = a^*\hat{O}^\dagger, \qquad (a|\psi\rangle)^\dagger = a^*\langle\psi|.$$

*We now introduce the most relevant type of operators for Quantum Mechanics: normal operators, Hermitian operators, unitary operators, and projection operators.*

## Normal operators.

A *normal operator* satisfies $O^\dagger O = O\, O^\dagger$. **Operators are *diagonalizable* if and only if they are normal.** That is, for normal O we can always find an orthonormal basis $\{|\varphi_j\rangle\}$ such that

$$\hat{O} = \sum_j \lambda_j |\phi_j\rangle\langle\phi_j|, \quad \mathrm{Tr}\{\hat{O}\} = \sum_j \lambda_j$$

and any diagonalizable operator must be normal. This is called the **spectral theroem**.

These values $\lambda_j$ are the *eigenvalues* of O and $\{|\varphi_j\rangle\}$ the corresponding *eigenvectors*, $O|\varphi_j\rangle = \lambda_j |\varphi_j\rangle$. If O is *nondegenerate*—i.e., all the $\lambda_j$ are distinct—then the eigenvectors are unique (up to a phase). Otherwise there is some freedom in choosing this *eigenbasis*.

If two normal operators A and B commute, it is possibleto find an eigenbasis which simultaneously diagonalizes *both* of them. (The converse is also true.)

## Hermitian operators.

One very useful class of operators are the *Hermitian operators H* that satisfy $H = H^\dagger$. These are the complex analogue of *symmetric* matrices. They are **obviously normal**: $H^\dagger H = H^2 = H\, H^\dagger$. The eigenvalues of a Hermitian matrix are always *real*.

An example are the Pauli matrices

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

It is easy to check that any 2 x 2 Hermitian matrix can be written as a linear combination of the Pauli matrices and the identity

Consider a 2 x 2 matrix

$$O = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

Hemiticity requires

$$O = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = O^{\dagger} = \begin{pmatrix} \alpha^* & \gamma^* \\ \beta^* & \delta^* \end{pmatrix}$$

Therefore

$$\begin{aligned} \alpha &= a + d \\ \beta &= b - ic \\ \gamma &= b + ic \\ \delta &= a - d \end{aligned}$$

where a, b, c, d are real numbers. Therefore we have

$$\hat{O} = a\hat{I} + b\hat{\sigma}_x + c\hat{\sigma}_y + d\hat{\sigma}_x$$

## Unitary Operators.

A *unitary* operator satisfies $U^\dagger U = U\,U^\dagger = I$. It is clearly a normal operator. All of its eigenvalues have unit norm; that is, $|\lambda_j| = 1$ for all $j$. This means that

$$\lambda_j = \exp(i\vartheta_j)$$

for real $0 \leq \vartheta_j < 2\pi$.

**There is a correspondence between Hermitian and unitary operators**: for every unitary operator U there is an Hermitian operator H such that

$$U = \exp(iH).$$

(We will clarify what $\exp(iH)$ means shortly.)

A unitary operator is equivalent to a **change of basis**. This is easy to check: if $\{|j\rangle\}$ is an orthonormal basis, then so is $\{U|j\rangle\}$:

$$(\hat{U}|i\rangle)^\dagger(\hat{U}|j\rangle) = \langle i|\hat{U}^\dagger\hat{U}|j\rangle = \langle i|j\rangle = \delta_{ij}$$

For spin-1/2, the most general $2 \times 2$ unitary can be written (up to a global phase)

$$\hat{U} = \cos(\theta/2)\hat{I} + i\sin(\theta/2)\vec{n}\cdot\hat{\vec{\sigma}}$$

where n is again a real unit three-vector $(n_x, n_y, n_z)$ and sigma = ($\text{sigma}_x$, $\text{sigma}_y$, $\text{sigma}_z$). In the Bloch sphere picture, U is a rotation by $\vartheta$ about the axis n.

## Orthogonal projectors.

An *orthogonal projector P is* an Hermitian operator that obeys $P^2 = P$. All the eigenvalues of P are either 0 or 1. The *complement* of a projector 1–P is also a projector. Note that $|\psi\rangle\langle\psi|$ is a projector for *any* normalized state vector $|\psi\rangle$. Such a projector is called *one-dimensional*; a projector more generally has dimension d = Tr[P].

In dimension 2, *any* projector can be written in the form

$$\hat{\mathcal{P}} = \left(\hat{I} + \vec{n}\cdot\hat{\vec{\sigma}}\right)/2 = |\psi_{\vec{n}}\rangle\langle\psi_{\vec{n}}|,$$

where n is a (real) unit three-vector ($n_x$, $n_y$, $n_z$) (i.e., with $n_x^2 + n_y^2 + n_z^2 = 1$) and sigma are the three Pauli matrices .In the Bloch sphere picture, n is the direction in space, and P the projector onto the state $|\psi_n\rangle$ that is spin up along that axis.

## Operator space.

The space of all operators on a particular Hilbert space of dimension $N$ is itself a Hilbert space of dimension $N^2$; sometimes this fact can be very useful. If A and B operators, so is aB + bB for any complex *a, b*.

One can define an inner product on operator space. The most commonly used one is $(A, B) \equiv \mathrm{Tr}\{A^\dagger B\}$ (the *Frobenius* or **Hilbert-Schmidt inner product**).

It is easy to see that $(A, B) = (B, A)^*$, and $(A, A) \geq 0$ with equality only for $A = 0$. With respect to this inner product, the Pauli matrices together with the identity form an orthogonal basis for all operators on 2D Hilbert space.

A linear transformation on operator space is often referred to as a *superoperator*.

## Functions of Operators.

It is common to consider a function of an operator $f(O)$, where $f$ is ordinarily a function on the complex numbers, which is itself an operator.

Suppose that $f(x)$ is defined by a Taylor series: $f(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)^2 + \cdots$. For the operator version, we write

$$f(\hat{O}) = c_0\hat{I} + c_1(\hat{O} - x_0\hat{I}) + c_2(\hat{O} - x_0\hat{I})^2 + \cdots$$

In practice, only the case $x_0 = 0$ is of interest, in which case:

$$f(\hat{O}) = c_0\hat{I} + c_1\hat{O} + c_2\hat{O}^2 + \ldots$$

For particular functions and operators, this series can sometimes be summed explicitly. If $O$ is **normal**, we simplify by writing $O$ in diagonal form:

$$\hat{O} = \sum_j \lambda_j |\phi_j\rangle\langle\phi_j|$$

It is easy to show that

$$\hat{O}^n = \sum_j \lambda_j^n |\phi_j\rangle\langle\phi_j|$$

And therefore

$$f(\hat{O}) = \sum_j \left[\sum_n c_n \lambda_j^n\right] |\phi_j\rangle\langle\phi_j| = \sum_j f(\lambda_j)|\phi_j\rangle\langle\phi_j|$$

In particular, for **projectors**, $P^n = P$ for all n greater or equal to 1. Then

$$\begin{aligned} f(\hat{P}) &= c_0\hat{I} + c_1\hat{P} + c_2\hat{P} + \ldots \\ &= c_0\hat{I} \pm c_0\hat{P} + c_1\hat{P} + c_2\hat{P} + \ldots \\ &= f(0)[\hat{I} - \hat{P}] + f(1)\hat{P} \end{aligned}$$

For **idempotent** operators obeying $\hat{O}^2 = \hat{I}$ (such as the Pauli matrices), one can sum the even and odd term separately. In particular:

$$\begin{aligned} e^{i\theta\hat{O}} &= \hat{I} + i\theta\hat{O} - \frac{\theta^2\hat{O}^2}{2} - i\frac{\theta^3\hat{O}^3}{3!} + \frac{\theta^4\hat{O}^4}{4!} + i\frac{\theta^5\hat{O}^5}{5!} - \dots \\ &= \hat{I} + i\theta\hat{O} - \frac{\theta^2}{2}\hat{I} - i\frac{\theta^3}{3!}\hat{O} + \frac{\theta^4}{4!}\hat{I} + i\frac{\theta^5}{5!}\hat{O} - \dots \\ &= \left(1 - \frac{\theta^2}{2} + \frac{\theta^4}{4!} - \dots\right)\hat{I} + i\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots\right)\hat{O} \\ &= \cos(\theta)\hat{I} + i\sin(\theta)\hat{O} \end{aligned}$$

As an application, let a be n real, three-dimensional unit vector and $\vartheta$ a real number. Then

$$(\vec{n}\cdot\vec{\sigma})^2 = 1$$

where we used the property of the Pauli matrices

$$(\vec{n}\cdot\vec{\sigma})(\vec{m}\cdot\vec{\sigma}) = (\vec{n}\cdot\vec{m})\hat{I} + i(\vec{n}\times\vec{n})\cdot\vec{\sigma}$$

As such:

$$e^{i\theta\vec{n}\cdot\vec{\sigma}} = \cos(\theta)\hat{I} + i\sin(\theta)\vec{n}\cdot\vec{\sigma}$$

As in the previous example, the most commonly-used function is the exponential

$$\exp(x) = 1 + x + x^2/2! + \cdots,$$

but others also occur from time to time:

$$\cos(x) = 1 - x^2/2 + x^4/4! - \cdots$$

$$\sin(x) = x - x^3/3! + x^5/5! - \cdots$$

$$\log(1 + x) = x - x^2/2 + x^3/3 - \cdots$$

**Exercise 2.34:** Find the square root and logarithm of the matrix

$$A = \begin{bmatrix} 4 & 3 \\ 3 & 4 \end{bmatrix}$$

Since the matrix is: $4\hat{I} + 3\hat{\sigma}_x$

Then the eigenvectors are those of the Pauli matrix and the eigenvalues equal to 4+3 = 7 and 4-3 = 1. The eigenprojectors are:

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

$$P_- = \frac{1}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Therefore:

$$\sqrt{A} = \sqrt{7}P_+ + P_- = \frac{1}{4}\begin{bmatrix} \sqrt{7}+1 & \sqrt{7}-1 \\ \sqrt{7}-1 & \sqrt{7}+1 \end{bmatrix}$$

As a mater of fact, if one takes the square, the original matrix A is recovered.

## Tensor products.

The tensor (or Kronecker) product is a way of combining two Hilbert spaces to produce a higher dimensional space. Let $|\psi\rangle$ be a state in a $D_1$-dimensional Hilbert space $\mathrm{H}_1$ and $|\varphi\rangle$ be a state in a $D_2$-dimensional Hilbert space $\mathrm{H}_2$. Then we define $|\psi\rangle \otimes |\varphi\rangle$ to be a state in the $D_1 D_2$-dimensional space $\mathrm{H}_1 \otimes \mathrm{H}_2$. Such a state is called a *product state*. Any state in this larger space can be written as a linear combination of product states

$$|\Psi\rangle = \sum_\ell \alpha_\ell |\psi_\ell\rangle \otimes |\phi_\ell\rangle$$

where $|\psi_\ell\rangle \in \mathrm{H}_1$ and $|\varphi_\ell\rangle \in \mathrm{H}_2$.

What are the properties of this product?

$$(a|\psi\rangle + b|\psi'\rangle) \otimes |\phi\rangle = a|\psi\rangle \otimes |\phi\rangle + b|\psi'\rangle \otimes |\phi\rangle.$$

$$|\psi\rangle \otimes (a|\phi\rangle + b|\phi'\rangle) = a|\psi\rangle \otimes |\phi\rangle + b|\psi\rangle \otimes |\phi'\rangle.$$

We need also to define bra-vectors, and the inner product:

$$(|\psi\rangle \otimes |\phi\rangle)^\dagger = \langle\psi| \otimes \langle\phi|.$$

$$(\langle\psi'| \otimes \langle\phi'|)(|\psi\rangle \otimes |\phi\rangle) = \langle\psi'|\psi\rangle\langle\phi'|\phi\rangle.$$

If $\{|j\rangle_1\}$ is a basis for $H_1$ and $\{|k\rangle_2\}$ is a basis for $H_2$ then $\{|j\rangle_1 \bigotimes |k\rangle_2\}$ is a basis for $H_1 \bigotimes H_2$. Given two states in $H_1$ and $H_2$

$$|\psi\rangle = \sum_{j=1}^{D_1} \alpha_j |j\rangle_1, \quad |\phi\rangle = \sum_{k=1}^{D_2} \beta_k |k\rangle_2,$$

in terms of this basis

$$|\psi\rangle \otimes |\phi\rangle = \sum_{j,k} \alpha_j \beta_k |j\rangle_1 \otimes |k\rangle_2.$$

Generic states in $H_1 \bigotimes H_2$ are *not* product states:

$$|\Psi\rangle = \sum_{j,k} t_{jk} |j\rangle_1 \otimes |k\rangle_2.$$

## Operator Tensor Products.

If $A$ is an operator on $H_1$ and B on $H_2$, we construct a similar product to get a new operator $A \bigotimes B$ on $H_1 \bigotimes H_2$. Its properties are similar to tensor products of states:

$$(a\hat{A} + b\hat{A}') \otimes \hat{B} = a\hat{A} \otimes \hat{B} + b\hat{A}' \otimes \hat{B}.$$

$$\hat{A} \otimes (a\hat{B} + b\hat{B}') = a\hat{A} \otimes \hat{B} + b\hat{A} \otimes \hat{B}'.$$

$$(\hat{A} \otimes \hat{B})^\dagger = \hat{A}^\dagger \otimes \hat{B}^\dagger.$$

$$(\hat{A} \otimes \hat{B})(\hat{A}' \otimes \hat{B}') = \hat{A}\hat{A}' \otimes \hat{B}\hat{B}'.$$

$$\mathrm{Tr}\{\hat{A} \otimes \hat{B}\} = \mathrm{Tr}\{\hat{A}\}\mathrm{Tr}\{\hat{B}\}.$$

We can also apply these tensor product operators to tensor product states:

$$(\hat{A} \otimes \hat{B})(|\psi\rangle \otimes |\phi\rangle) = \hat{A}|\psi\rangle \otimes \hat{B}|\phi\rangle.$$

$$(\langle\psi| \otimes \langle\phi|)(\hat{A} \otimes \hat{B}) = \langle\psi|\hat{A} \otimes \langle\phi|\hat{B}.$$

A general operator on $H_1 \otimes H_2$ is *not* a product operator, but can be written as a linear combination of product operators:

$$\hat{O} = \sum_\ell \hat{A}_\ell \otimes \hat{B}_\ell.$$

**Tensor products play an important role in quantum mechanics! They describe how the Hilbert spaces of subsystems are combined.**

**Matrix representation.**

What does the matrix representation of a tensor product look like? If $|\psi\rangle$ has amplitudes $(\alpha_1, \cdots, \alpha_{D_1})$ and $|\varphi\rangle$ has amplitudes $(\beta_1, \cdots, \beta_{D_2})$, the state $|\psi\rangle \otimes |\varphi\rangle$ in $H_1 \otimes H_2$ is represented as a $D_1D_2$-dimensional column vector:

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix} \alpha_1|\phi\rangle \\ \alpha_2|\phi\rangle \\ \vdots \\ \alpha_{D_1}|\phi\rangle \end{pmatrix} = \begin{pmatrix} \alpha_1\beta_1 \\ \alpha_1\beta_2 \\ \vdots \\ \alpha_1\beta_{D_2} \\ \alpha_2\beta_1 \\ \vdots \\ \alpha_{D_1}\beta_{D_2} \end{pmatrix}$$

Example

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \otimes \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \times 2 \\ 1 \times 3 \\ 2 \times 2 \\ 2 \times 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 6 \end{bmatrix}$$

Similarly, if $A = [a_{ij}]$ and $B = [b_{ij}]$ then

$$\hat{A} \otimes \hat{B} = \begin{pmatrix} a_{11}\hat{B} & \cdots & a_{1D_1}\hat{B} \\ \vdots & \ddots & \vdots \\ a_{D_1 1}\hat{B} & \cdots & a_{D_1 D_1}\hat{B} \end{pmatrix}$$

Example

$$\hat{I} \otimes \hat{I} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \hat{Z} \otimes \hat{I} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

$$\hat{I} \otimes \hat{X} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \hat{X} \otimes \hat{Y} = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix}$$

For brevity, $|\psi\rangle \otimes |\varphi\rangle$ is often written $|\psi\rangle|\varphi\rangle$ or even $|\psi\varphi\rangle$. One can similarly condense the notation for operators; but one should be very careful not to confuse A⊗B with AB.

## Example: Two Spins

Given two states in the $Z$ basis,

$$|\psi\rangle = \alpha_1 |\uparrow\rangle + \alpha_2 |\downarrow\rangle \text{ and } |\varphi\rangle = \beta_1 |\uparrow\rangle + \beta_2 |\downarrow\rangle$$

we can write

$$|\psi\rangle \otimes |\varphi\rangle = \alpha_1\beta_1 |\uparrow\uparrow\rangle + \alpha_1\beta_2 |\uparrow\downarrow\rangle + \alpha_2\beta_1 |\downarrow\uparrow\rangle + \alpha_2\beta_2 |\downarrow\downarrow\rangle.$$

A general state of two spins would be

$$|\Psi\rangle = t_{11} |\uparrow\uparrow\rangle + t_{12} |\uparrow\downarrow\rangle + t_{21} |\downarrow\uparrow\rangle + t_{22} |\downarrow\downarrow\rangle.$$

As a column vector this is

$$|\Psi\rangle = \begin{pmatrix} t_{11} \\ t_{12} \\ t_{21} \\ t_{22} \end{pmatrix}$$

# The Postulates of Quantum Mechanics

We have reviewed the mathematics (complex linear algebra) necessary to understand quantum mechanics. We will now see how the *physics* of quantum mechanics fits into this mathematical framework.

We are really defining the *structure* of a quantum theory. All physical theories based on quantum mechanics share this common structure. Later in the course, we will see how this mathematical structure is realized for realistic systems. For now, we will use our simple example of the spin-1/2 particle to illustrate these ideas.

# Postulate 1: State space

Every physical system has an associated Hilbert space H of some dimension D, known as the state space of that system; and the system is completely described by its state vector, which is a unit vector in the state space.

If we choose a particular basis for the Hilbert space $|j\rangle$, $j = 1, \ldots, D$, a state can be written in the form

$$|\psi\rangle = \sum_{j=1}^{D} \alpha_j |j\rangle, \quad \text{where} \sum_j |\alpha_j|^2 = 1.$$

In the case of spin-1/2, we can write any state in terms of the basis "spin up" and "spin down" along the Z axis:

$$|\psi\rangle = \alpha_1 |\uparrow_Z\rangle + \alpha_2 |\downarrow_Z\rangle.$$

The overall phase of the state has no physical meaning, so $|\psi\rangle$ and $e^{i\vartheta}|\psi\rangle$ represent the same physical state.

The choice of basis relates to possible measurements of the system. As we will see, each basis is associated with a particular measurement (or group of compatible measurements), and each basis vector with a particular measurement outcome.

For spin-1/2 each basis is associated with a particular direction in space along which the component of the spin could be measured. So $\{|\uparrow_Z\rangle, |\downarrow_Z\rangle\}$ is associated with measurement of the component of spin along the *Z* axis, with the basis vectors corresponding to spin up or down.

Similarly, the bases $\{|\uparrow_X\rangle, |\downarrow_X\rangle\}$ and $\{|\uparrow_Y\rangle, |\downarrow_Y\rangle\}$ represent other possible measurements.

# Postulate 2: Unitary time evolution

The time-evolution of a closed system is described by a unitary transformation,

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

where U is independent of the initial state.

In fact, the time-evolution of the state is given by the Schrödinger equation

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

where $H(t)$ is an Hermitian operator (the *Hamiltonian*) that describes the *energy* of the system. How does this relate to unitary transformations?

This is easiest to see if H is a fixed operator (i.e., constant in time). In that case, a solution to Schrödinger's equation is

$$|\psi(t_2)\rangle = \exp(-i\hat{H}(t_2 - t_1)/\hbar)|\psi(t_1)\rangle.$$

The operator $-H(t_2 - t_1)/hbar$ is Hermitian, so the operator

$$\hat{U}(t_2, t_1) = \exp(-i\hat{H}(t_2 - t_1)/\hbar)$$

is unitary, as asserted.

Suppose there is a uniform magnetic field in the $Z$ direction. Then states with spin up and down along the $Z$ axis have different energies. This is represented by a Hamiltonian

$$\hat{H} = \begin{pmatrix} E_0 & 0 \\ 0 & -E_0 \end{pmatrix} \equiv E_0 \hat{Z},$$

where $E_0$ is proportional to the strength of the magnetic field. If $|\psi\rangle = \alpha|\uparrow_Z\rangle + \beta|\downarrow_Z\rangle$ at $t = 0$, then

$$|\psi(t)\rangle = \alpha e^{-iE_0 t/\hbar}|\uparrow_Z\rangle + \beta e^{iE_0 t/\hbar}|\downarrow_Z\rangle.$$

This type of evolution is equivalent to a steady rotation about the $Z$ axis, called *precession*.

## Examples

**EXAMPLE 2.1** Let us consider a time-independent Hamiltonian

$$H = -\frac{\hbar}{2}\omega\sigma_x. \tag{2.8}$$

Suppose the system is in the eigenstate of $\sigma_z$ with the eigenvalue $+1$ at time $t = 0$;

$$|\psi(0)\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The wave function $|\psi(t)\rangle$ $(t > 0)$ is then found from Eq. (2.5) to be

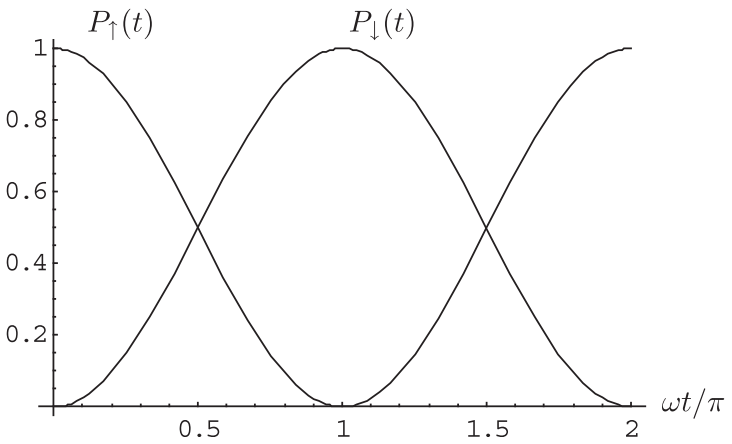
$$|\psi(t)\rangle = \exp\left(i\frac{\omega}{2}\sigma_x t\right)|\psi(0)\rangle. \tag{2.9}$$

The matrix exponential function in this equation is evaluated with the help of Eq. (1.44) and we find

$$|\psi(t)\rangle = \begin{pmatrix} \cos\omega t/2 & i\sin\omega t/2 \\ i\sin\omega t/2 & \cos\omega t/2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\omega t/2 \\ i\sin\omega t/2 \end{pmatrix}. \tag{2.10}$$

Suppose we measure the observable $\sigma_z$. Note that $|\psi(t)\rangle$ is expanded in terms of the eigenvectors of $\sigma_z$ as

$$|\psi(t)\rangle = \cos\frac{\omega}{2}t|\sigma_z = +1\rangle + i\sin\frac{\omega}{2}t|\sigma_z = -1\rangle.$$




The state oscillates among the two eigenstates. Why? What should happen to not have the oscillation? What are the probabilities of outcomes of measurements?

Next let us take the initial state

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

which is an eigenvector of $\sigma_x$ (and hence the Hamiltonian) with the eigenvalue $+1$. We find $|\psi(t)\rangle$ in this case as

$$|\psi(t)\rangle = \begin{pmatrix} \cos\omega t/2 & i\sin\omega t/2 \\ i\sin\omega t/2 & \cos\omega t/2 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{e^{i\omega t/2}}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{2.11}$$

Therefore the state remains in its initial state at an arbitrary $t > 0$. This is an expected result since the system at $t = 0$ is an eigenstate of the Hamiltonian.

**EXERCISE 2.2** Let us consider a Hamiltonian

$$H = -\frac{\hbar}{2}\omega\sigma_y. \tag{2.12}$$

Suppose the initial state of the system is

$$|\psi(0)\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.13}$$

(1) Find the wave function $|\psi(t)\rangle$ at later time $t > 0$.
(2) Find the probability for the system to have the outcome $+1$ upon measurement of $\sigma_z$ at $t > 0$.
(3) Find the probability for the system to have the outcome $+1$ upon measurement of $\sigma_x$ at $t > 0$.

Now let us formulate Example 2.1 and Exercise 2.2 in the most general form. Consider a Hamiltonian

$$H = -\frac{\hbar}{2}\omega \hat{\boldsymbol{n}} \cdot \boldsymbol{\sigma}, \tag{2.14}$$

where $\hat{\boldsymbol{n}}$ is a unit vector in $\mathbb{R}^3$. The time-evolution operator is readily obtained, by making use of the result of Proposition 1.2, as

$$U(t) = \exp(-iHt/\hbar) = \cos\frac{\omega}{2}t\ I + i(\hat{\boldsymbol{n}} \cdot \boldsymbol{\sigma}) \sin\frac{\omega}{2}t. \tag{2.15}$$

Suppose the initial state is

$$|\psi(0)\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

for example. Then we find

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle = \begin{pmatrix} \cos(\omega t/2) + in_z \sin(\omega t/2) \\ i(n_x + in_y) \sin(\omega t/2) \end{pmatrix}. \tag{2.16}$$

The reader should verify that $|\psi(t)\rangle$ is normalized at any instant of time $t > 0$.

If $|\psi\rangle$ is an eigenstate of *H*, $|\uparrow_z\rangle$ or $|\downarrow_z\rangle$, the only effect is a change in the global phase of the state which has no physical consequences:

$$|\psi(t)\rangle = e^{-iE_0t/\hbar}|\uparrow_Z\rangle \quad \text{or} \quad e^{+iE_0t/\hbar}|\downarrow_Z\rangle$$

Because of this, we call these *stationary states*. Similarly, we could have a uniform field in the *X* direction or the *Y* direction. In these cases, the Hamiltonians would be $E_0X$ or $E_0Y$, and the stationary states would be the X or Y eigenstates.

If the Hamiltonian $H(t)$ is not constant, the situation is more complicated; but the time evolution in every case is still given by a unitary transformation.

# Controlling the Hamiltonian

A common situation in quantum information is when we have some control over the Hamiltonian of the system. For instance, we could turn on a uniform magnetic field in the *Z* direction, leave it on for a time $\tau$, and then turn it off. In that case, the state will have evolved by

$$|\psi\rangle \rightarrow \exp(i\theta\hat{Z})|\psi\rangle \equiv \hat{U}|\psi\rangle,$$

where $\vartheta = -E_0\tau/hbar$. In this case, we say we have "performed a unitary transformation U on n the system." The Hamiltonian has the time dependence $H(t) = f(t)E_0Z$, where $f(t) = 1$ for $0 \leq t \leq \tau$ and $f(t) = 0$ otherwise.

# Postulate 3: Measurement

An observable is a measurable quantity, which is associated with an Hermitian operator $O = O^\dagger$. In measuring an observable O,the possible measurement outcomes are given by the eigenvalues $\lambda_j$; these occur with probabilities equal to the square of the amplitude for that outcome, and the system is left in an eigenstate of O.

Suppose one is given a system in a state $|\psi\rangle$, and wishes to make a measurement of an observable $\hat{O}$. By the spectral theorem

$$\hat{O} = \sum_{j=1}^{M} \lambda_j \hat{\mathcal{P}}_j, \quad \sum_{j=1}^{M} \hat{\mathcal{P}}_j = \hat{I}, \quad \hat{\mathcal{P}}_j \hat{\mathcal{P}}_k = \delta_{jk} \hat{\mathcal{P}}_j.$$

$M \leq D$ and the projectors $P_j$ are a decompostion of the identity.

The outcome $\lambda_j$ occurs with probability $p_j = \langle\psi| P_j |\psi\rangle$ $\langle P_j\rangle$, and the system is left in the (renormalized) state

$$|\psi'\rangle = \hat{\mathcal{P}}_j |\psi\rangle / \sqrt{p_j}.$$

This is called *Born's Rule*. The expectation value of the measurement outcomes is $\langle O\rangle = \langle\psi | O|\psi\rangle$.

If $M = D$, so the $P_j$ are projectors $|\varphi_j\rangle\langle\varphi_j|$ onto eigen- vectors of O, then we can write $|\psi\rangle$ in the eigenbasis:

$$|\psi\rangle = \sum \alpha_j |\phi_j\rangle.$$

Outcome $\lambda_j$ occurs with probability $p_j = |\alpha_j|^2$ and the system is afterwards left in the state $|\varphi_j\rangle$.

## Expectation values.

Given a state $|\psi\rangle$ and an opertor O, we can calculate a number

$$\langle \hat{O} \rangle \equiv \langle \psi | \hat{O} | \psi \rangle = \langle \psi | \left( \hat{O} | \psi \rangle \right)$$

which is called *the expectation value of O in the state* $|\psi\rangle$. Given a particular choice of basis, we can express this number in terms of the elements of O and $|\psi\rangle$:

$$\langle \hat{O} \rangle = \sum_{i,j} \alpha_i^* a_{ij} \alpha_j$$

As we will see, when O is Hermitian, its expectation value gives the average result of some measurement on a system in the state $|\psi\rangle$.

# Postulate 4: Composite systems

If a composite system is composed of subsystems A and B which have associated Hilbert spaces $H_A$ and $H_B$, then the associated Hilbert space of the joint system is the tensor product space

$H_A \otimes H_B$ .

Everything we have already learned about quantum mechanics generalizes to the case where the system is part of a composite system. Let's go through them point by point.

# Acting on a Subsystem

1. If subsystem A is in state $|\psi\rangle$ and subsystem B is in state $|\varphi\rangle$, then the *joint system* is in the product state $|\psi\rangle \otimes |\varphi\rangle$.

2. If $U_A$ is a unitary transformation which acts on subsystem A, then

$$U_A \otimes I$$

is the corresponding unitary for the joint system. Similarly, if $U_B$ acts on B, then $I \otimes U_B$ is the unitary for the joint system.

$$(\hat{U}_A \otimes \hat{I})(|\psi\rangle \otimes |\phi\rangle) = (\hat{U}_A|\psi\rangle) \otimes |\phi\rangle.$$

$$(\hat{I} \otimes \hat{U}_B)(|\psi\rangle \otimes |\phi\rangle) = |\psi\rangle \otimes (\hat{U}_B|\phi\rangle).$$

3. If A is an observable for subsystem A, then $A \otimes I$ is the corresponding observable for the joint system.

Note that $A \otimes I$ and $I \otimes B$ *always* commute, and therefore are compatible observables. Physically, this means that measurements on different subsystems can always be done simultaneously.

# Treating Subsystems Jointly

While we can extend the results for single systems to composite systems, there are many more possible states, evolutions and measurements that are allowed.

Most states $|\Psi\rangle$ of a composite system are *not* product states. For an example with two spin-1/2 systems,

$$|\Psi\rangle = \frac{1}{\sqrt{2}}|\uparrow\downarrow\rangle - \frac{1}{\sqrt{2}}|\downarrow\uparrow\rangle$$

is not a product state. For this joint state, we cannot assign well-defined states to the subsystems. Such a joint state is called *entangled*.

A consequence of entanglement is that measurements on the subsystems will in general be *correlated*.

# Interactions

We can also perform joint unitaries U on both systems at once. If the initial state is a product $|\psi\rangle \otimes |\varphi\rangle$, then after applying a joint unitary the system will in general be entangled. Correlations are produced by *interaction* between the spins.

For example, we might have a Hamiltonian of the form

$$\hat{H} = E_0 \hat{Z} \otimes \hat{Z}.$$

The unitary operators $\exp(i\vartheta H)$ produced by this Hamiltonian will *not* be product operators, even though H itself *is* a product operator. So initial product states will evolve to become entangled.

# Joint Measurements

Finally, we can measure observables O which are *not* product operators. The measurement process follows the same rules we have already seen: an eigenvalue of O will occur with some probability, and the system will be left in the corresponding eigenstate of *O*. However, since O is not a product operator, the eigenstates of O will in general be *entangled* states.

This means that entanglement can be produced by joint measurements even if the initial state $|\psi\rangle \otimes |\varphi\rangle$ was a product.

# The Qubit

The bit is the fundamental concept of classical computation and classical information. Just as a classical bit has a state – either 0 or 1 – a qubit also has a state. Two possible states for a qubit are the states $|0\rangle$ and $|1\rangle$, which as you might guess correspond to the states 0 and 1 for a classical bit. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$. It is also possible to form **linear combinations of states**, often called superpositions:

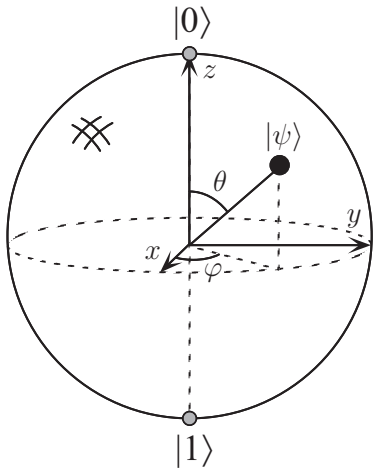
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad\qquad |\alpha|^2 + |\beta|^2 = 1$$

The special states $|0\rangle$ and $|1\rangle$ are known as computational basis states, and form an orthonormal basis for this vector space .

Because $|\alpha|^2 + |\beta|^2 = 1$, we may rewrite the above relation as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle.$$

Up to an global phase factor, which has no physical significance. The numbers θ and ϕ define a point on the unit three-dimensional sphere, often called the **Bloch sphere.**



# Multiple Qubits

Suppose we have **two qubits**. If these were two classical bits, then there would be four possible states, 00, 01, 10, and 11. Correspondingly, a two qubit system has four computational basis states denoted $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. A pair of qubits can also exist in **superpositions of these four states**, so the quantum state of two qubits involves associating a complex coefficient – sometimes called an amplitude – with each computational basis state, such that the state vector describing the two qubits is

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

Similar to the case for a single qubit, the measurement result x (= 00, 01, 10 or 11) occurs with probability $|\alpha_x|^2$, with the state of the qubits after the measurement being $|x\rangle$. The condition that probabilities sum to one is therefore expressed by the normalization condition

$$\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1$$

More generally, we may consider a system of n qubits. The computational basis states of this system are of the form $|x_1x_2 \ldots x_n\rangle$, and so a quantum state of such a system is specified by $2^n$ amplitudes.

Two ways of denoting the qubits

**Binary basis**: sequence of 0 and 1, i.e. $|x_{n-1}\, x_{n-2} \ldots x_0\rangle$
**Decimal basis**: $|x\rangle$, with $x = x_{n-1}\, 2^{n-1} + x_{n-2}\, 2^{n-2} + \ldots + x_0$

Examples
$|10\rangle = |\, 1 \times 2^1 + 0 \times 2^0\,\rangle = |2\rangle$
$|101\rangle = |\, 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\,\rangle = |6\rangle$

The set

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

is an orthonormal basis of a two-qubit system and is called the **Bell basis**. Each vector is called the Bell state or the Bell vector. Note that all the Bell states are entangled.

**EXERCISE.** The Bell basis is obtained from the binary basis {|00⟩, |01⟩, |10⟩, |11⟩} by a unitary transformation. Write down the unitary transformation explicitly.

Among **three-qubit entangled states**, the following two states are important for various reasons and hence deserve special names. The state

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

is called the **Greenberger-Horne-Zeilinger state** and is often abbreviated as the **GHZ state**. Another important three-qubit state is the **W state**

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

# Quantum Computation

A quantum computation is a collection of the following three elements:

- A register or a set of registers,
- A unitary matrix U, which is tailored to execute a given quantum algorithm
- Measurements to extract information we need.

More formally, we say **a quantum computation is the set {H,U,{$M_m$}}**, where H = $C^{2n}$ is the Hilbert space of an n-qubit register, U ∈ U($2^n$) represents the quantum algorithm and {$M_m$} is the set of measurement operators. The hardware along with equipment to control the qubits is called a quantum computer.

# Single Qubit Quantum Gates

$$I = |0\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$Y = |0\rangle\langle 1| - |1\rangle\langle 0| = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = -i\sigma_y,$$

$$X = |1\rangle\langle 0| + |0\rangle\langle 1| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma_x,$$

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \sigma_z.$$

The transformation $I$ is the trivial (identity) transformation, while $X$ is the negation (NOT), $Z$ the phase shift and $Y = XZ$ the combination of them. It is easily verified that these gates are unitary.

Exercise: Find the Hamiltonian that implements these gates, and show how they are implemented.

Three other quantum gates will play a large part in what follows, the Hadamard gate (denoted $H$), phase gate (denoted $S$), and $\pi/8$ gate (denoted $T$):

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}; \quad T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}. \tag{4.2}$$

A couple of useful algebraic facts to keep in mind are that $H = (X+Z)/\sqrt{2}$ and $S = T^2$. You might wonder why the $T$ gate is called the $\pi/8$ gate when it is $\pi/4$ that appears in the definition. The reason is that the gate has historically often been referred to as the $\pi/8$ gate, simply because up to an unimportant global phase $T$ is equal to a gate which has $\exp(\pm i\pi/8)$ appearing on its diagonals.

$$T = \exp(i\pi/8)\begin{bmatrix} \exp(-i\pi/8) & 0 \\ 0 & \exp(i\pi/8) \end{bmatrix}. \tag{4.3}$$

Nevertheless, the nomenclature is in some respects rather unfortunate, and we often refer to this gate as the $T$ gate.

The Pauli matrices give rise to three useful classes of unitary matrices when they are exponentiated, the *rotation operators* about the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes, defined by the equations:

$$R_x(\theta) \equiv e^{-i\theta X/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{4.4}$$

$$R_y(\theta) \equiv e^{-i\theta Y/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{4.5}$$

$$R_z(\theta) \equiv e^{-i\theta Z/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{4.6}$$

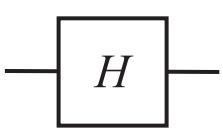
The **Hadamard gate** or the **Hadamard transformation** $H$ is an important unitary transformation defined by

$$\begin{aligned} U_{\mathrm{H}} : |0\rangle &\rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle &\rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{4.9}$$

It is used to generate a superposition state from $|0\rangle$ or $|1\rangle$. The matrix representation of $H$ is

$$U_{\mathrm{H}} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\langle 0| + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\langle 1| = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4.10}$$

A Hadamard gate is depicted as



# Hadamard-Walsh Gate

There are numerous important applications of the Hadamard transformation. All possible $2^n$ states are generated, when $U_{\rm H}$ is applied on each qubit of the state $|00\ldots0\rangle$:

$$\begin{aligned}
&(H\otimes H\otimes\ldots\otimes H)|00\ldots0\rangle \\
&=\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes\ldots\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle) \\
&=\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle. \qquad (4.11)
\end{aligned}$$

Therefore, we produce a superposition of all the states $|x\rangle$ with $0\leq x\leq 2^n-1$ simultaneously. This action of $H$ on an $n$-qubit system is called the **Walsh transformation**, or **Walsh-Hadamard transformation**, and denoted as $W_n$. Note that

$$W_1=U_{\rm H},\quad W_{n+1}=U_{\rm H}\otimes W_n. \qquad (4.12)$$

# Exercises

**Exercise 4.7:** Show that $XYX=-Y$ and use this to prove that $XR_y(\theta)X=R_y(-\theta)$.

**Exercise 4.8:** An arbitrary single qubit unitary operator can be written in the form

$$U=\exp(i\alpha)R_{\hat{n}}(\theta) \qquad (4.9)$$

for some real numbers $\alpha$ and $\theta$, and a real three-dimensional unit vector $\hat{n}$.

1. Prove this fact.
2. Find values for $\alpha$, $\theta$, and $\hat{n}$ giving the Hadamard gate $H$.
3. Find values for $\alpha$, $\theta$, and $\hat{n}$ giving the phase gate

$$S=\begin{bmatrix}1&0\\0&i\end{bmatrix}. \qquad (4.10)$$

**Exercise 4.13: (Circuit identities)** It is useful to be able to simplify circuits by inspection, using well-known identities. Prove the following three identities:

$$HXH=Z;\quad HYH=-Y;\quad HZH=X. \qquad (4.18)$$

**Exercise 4.14:** Use the previous exercise to show that $HTH=R_x(\pi/4)$, up to a global phase.

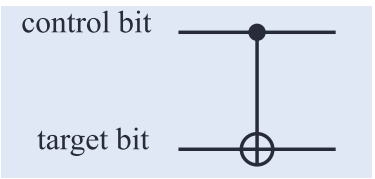
# Two qubit gates: CNOT Gate

The **CNOT** (**controlled-NOT**) gate is a two-qubit gate, which plays quite an important role in quantum computation. The gate flips the second qubit (the **target qubit**) when the first qubit (the **control qubit**) is $|1\rangle$, while leaving the second bit unchanged when the first qubit state is $|0\rangle$.

$$U_{\text{CNOT}} : |00\rangle \mapsto |00\rangle,\ |01\rangle \mapsto |01\rangle,\ |10\rangle \mapsto |11\rangle,\ |11\rangle \mapsto |10\rangle.$$

$$U_{\text{CNOT}} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X,$$

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




Let $\{|i\rangle\}$ be the basis vectors, where $i \in \{0, 1\}$. The action of CNOT on the input state $|i\rangle|j\rangle$ is written as $|i\rangle|i \oplus j\rangle$, where $i \oplus j$ is an addition mod 2, that is, $0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1$ and $1 \oplus 1 = 0$.

The following identity holds

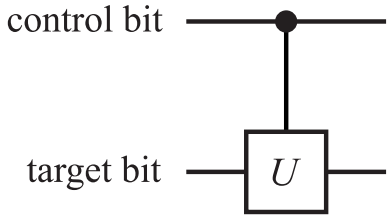
$$\begin{aligned} \text{CNOT} &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X \\ &= \frac{1}{2}(I + Z) \otimes I + \frac{1}{2}(I - Z) \otimes X \\ &= \frac{1}{2} I \otimes (I + X) + \frac{1}{2} Z \otimes (I - X) \end{aligned}$$

# Control-U Gate

More generally, we consider a controlled-$U$ gate,

$$V = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U, \tag{4.7}$$

in which the target bit is acted on by a unitary transformation $U$ only when the control bit is $|1\rangle$. This gate is denoted graphically as



# Swap Gate

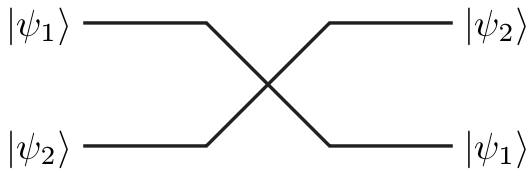
The SWAP gate acts on a tensor product state as

$$U_{\mathrm{SWAP}}|\psi_1,\psi_2\rangle = |\psi_2,\psi_1\rangle. \tag{4.14}$$

The explict form of $U_{\mathrm{SWAP}}$ is given by

$$\begin{aligned} U_{\mathrm{SWAP}} &= |00\rangle\langle 00| + |01\rangle\langle 10| + |10\rangle\langle 01| + |11\rangle\langle 11| \\ &= \begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \end{pmatrix}. \end{aligned} \tag{4.15}$$

Needless to say, it works as a linear operator on a superposition of states. The SWAP gate is expressed as



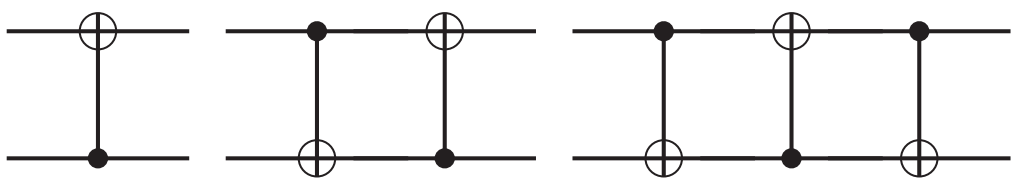

Note that the SWAP gate is a special gate which maps an arbitrary tensor product state to a tensor product state. In contrast, most two-qubit gates map a tensor product state to an entangled state.

# Exercise

**EXERCISE 4.1** Show that the $U_{\mathrm{CNOT}}$ cannot be written as a tensor product of two one-qubit gates.

**EXERCISE 4.2** Let $(a|0\rangle + b|1\rangle) \otimes |0\rangle$ be an input state to a CNOT gate. What is the output state?

**EXERCISE 4.3** (1) Find the matrix representation of the "upside down" CNOT gate (a) in the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.




(2) Find the matrix representation of the circuit (b).
(3) Find the matrix representation of the circuit (c). Find the action of the circuit on a tensor product state $|\psi_1\rangle \otimes |\psi_2\rangle$.

Given the outcome of Exercise 4.3(c) and the mathematical expression of the CNOT gate, one can write

$$\begin{aligned} \text{SWAP} &= \text{CNOT}\ \overline{\text{CNOT}}\ \text{CNOT} \\ &= \frac{1}{2}(I \otimes I + Z \otimes Z) + \frac{1}{2} X \otimes X (I \otimes I - Z \otimes Z) \end{aligned}$$

This expression (using the relation Y = XZ) can be rewritten as:

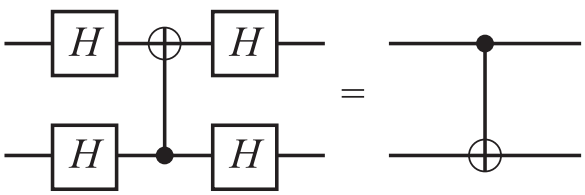
$$\text{SWAP} = \frac{1}{2}(I \otimes I + X \otimes X - Y \otimes Y + Z \otimes Z)$$

Given the relation between the gates X, Y, Z and the Pauli matrices, we have also:

$$\text{SWAP} = \frac{1}{2}(I \otimes I + \bar{\sigma} \otimes \bar{\sigma}) = \frac{1}{2}(I \otimes I + \sigma_x \otimes \sigma_x + \sigma_y \otimes \sigma_y + \sigma_z \otimes \sigma_z)$$

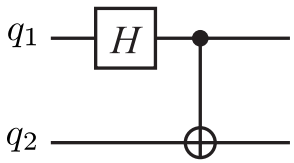
# Exercise

**EXERCISE 4.5** Show that the two circuits below are equivalent:




This exercise shows that the control bit and the target bit in a CNOT gate are interchangeable by introducing four Hadamard gates.

**EXERCISE 4.6** Let us consider the following quantum circuit




(4.13)

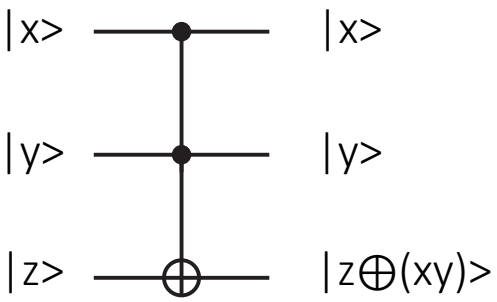
where $q_1$ denotes the first qubit, while $q_2$ denotes the second. What are the outputs for the inputs $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$?

# 3 qubit gate: CCNOT (Toffoli) Gate

The **CCNOT** (**Controlled-Controlled-NOT**) gate has three inputs, and the third qubit flips when and only when the first two qubits are both in the state $|1\rangle$. The explicit form of the CCNOT gate is

$$U_{\text{CCNOT}} = (|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I + |11\rangle\langle 11| \otimes X. \tag{4.8}$$

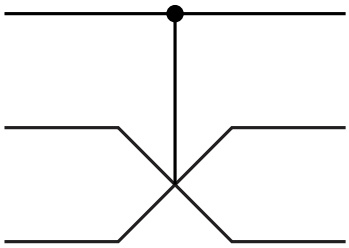
This gate is graphically expressed as




The CCNOT gate is also known as the **Toffoli gate**.

# Fredkin Gate

The controlled-SWAP gate



is also called the **Fredkin gate**. It flips the second (middle) and the third (bottom) qubits when and only when the first (top) qubit is in the state $|1\rangle$. Its explicit form is

$$U_{\text{Fredkin}} = |0\rangle\langle 0| \otimes I_4 + |1\rangle\langle 1| \otimes U_{\text{SWAP}}. \tag{4.17}$$

# Exercise

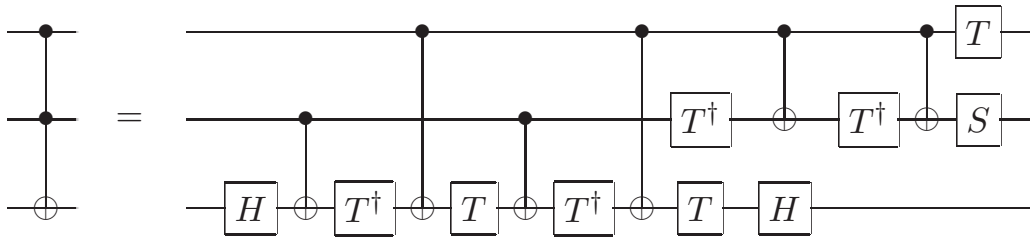



Figure 4.9. Implementation of the Toffoli gate using Hadamard, phase, controlled-NOT and $\pi/8$ gates.

**Exercise 4.24:** Verify that Figure 4.9 implements the Toffoli gate.

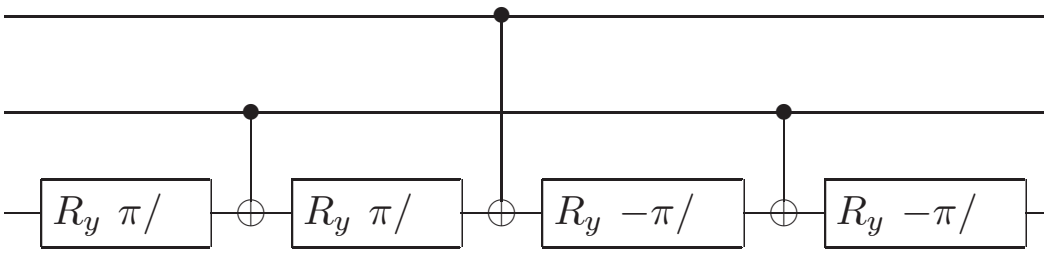
**Exercise 4.25: (Fredkin gate construction)** Recall that the Fredkin (controlled-swap) gate performs the transform

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.30}$$

(1) Give a quantum circuit which uses three Toffoli gates to construct the Fredkin gate (*Hint*: think of the swap gate construction – you can control each gate, one at a time).
(2) Show that the first and last Toffoli gates can be replaced by CNOT gates.
(3) Now replace the middle Toffoli gate with the circuit in Figure 4.8 to obtain a Fredkin gate construction using only six two-qubit gates.
(4) Can you come up with an even simpler construction, with only five two-qubit gates?

# Exercise

**Exercise 4.26:** Show that the circuit:




differs from a Toffoli gate only by relative phases. That is, the circuit takes $|c_1, c_2, t\rangle$ to $e^{i\theta(c_1,c_2,t)}|c_1, c_2, t \oplus c_1 \cdot c_2\rangle$, where $e^{i\theta(c_1,c_2,t)}$ is some relative phase factor. Such gates can sometimes be useful in experimental implementations, where it may be much easier to implement a gate that is the same as the Toffoli up to relative phases than it is to do the Toffoli directly.

**Exercise 4.27:** Using just CNOTs and Toffoli gates, construct a quantum circuit to perform the transformation

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.31}$$

This kind of partial cyclic permutation operation will be useful later, in Chapter 7.

# Recovering classical Gates

The (classical) Toffoli gate is universal, therefore it reproduces all reversible and irreversible classical gates. Its quantum version generalizes the classical gates into quantum gates.

In general:

1. Take a classical gate. If irreversible, consider its reversible variant.
2. Define the quantum counterpart so that on the computational basis it acts as the reversible classical gate.
3. Extend it by linearity to the whole space.

The gate thus obtained is the quantum generalization of the classical gate.

In summary, we have shown that all the classical logic gates, NOT, AND, OR, XOR and NAND gates, may be obtained from the CCNOT gate. Thus all the classical computation may be carried out with a quantum computor. Note, however, that these gates belong to a tiny subset of the set of unitary matrices.

# Exercise

**Exercise 4.36:** Construct a quantum circuit to add two two-bit numbers $x$ and $y$ modulo 4. That is, the circuit should perform the transformation $|x, y\rangle \rightarrow |x, x + y \bmod 4\rangle$.

What the circuit should do is the following

| | | | 0 | | 1 | | 2 | | 3 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $y_1$ | $y_0$ | $y_1$ | $y_0$ | $y_1$ | $y_0$ | $y_1$ | $y_0$ |
| | | | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| | $x_1$ | $x_0$ | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |

We see that:

$x_0 \rightarrow x_0$

$x_1 \rightarrow x_1$

$y_0 \rightarrow x_0 \oplus y_0$

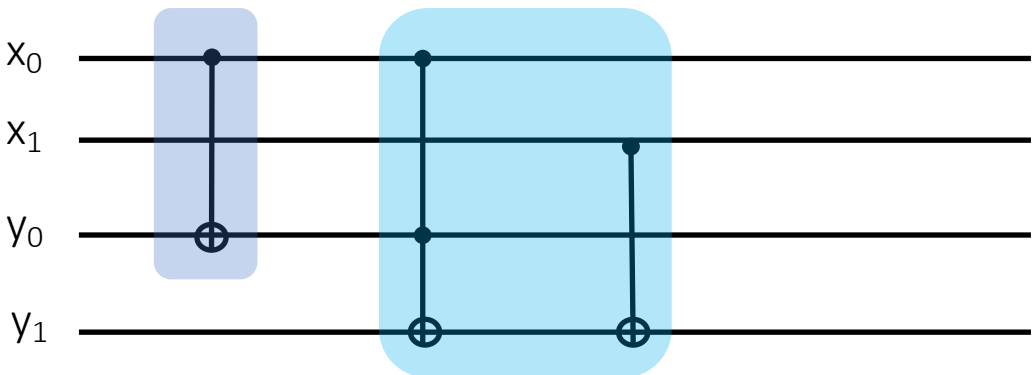
$y_1 \rightarrow x_1 \oplus y_1 \oplus (x_0\, y_0)$

This is implemented by a CNOT gate

$y_1 \oplus (x_0\, y_0)$ is implemented by a CCNOT gate

the rest by a CNOT gate

The circuit then is



# Universal Quantum Gates

Like in the classical case, there exist a **universal set of quantum gates**.

We will now show that

- Single qubit gates
- CNOT gate

are universal for quantum computation.

# Two-level unitary matrix

We will prove the following Lemma before stating the main theorem. Let us start with a definition. A **two-level unitary matrix** is a unitary matrix which acts non-trivially only on two vector components. Suppose $V$ is a two-level unitary matrix. Then $V$ has the same matrix elements as those of the unit matrix except for certain four elements $V_{aa}, V_{ab}, V_{ba}$ and $V_{bb}$. An example of a two-level unitary matrix is

$$V = \begin{pmatrix} \alpha^* & 0 & 0 & \beta^* \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\beta & 0 & 0 & \alpha \end{pmatrix}, \quad (|\alpha|^2 + |\beta|^2 = 1),$$

where $a = 1$ and $b = 4$.

**LEMMA 4.1** Let $U$ be a unitary matrix acting on $\mathbb{C}^d$. Then there are $N \leq d(d-1)/2$ two-level unitary matrices $U_1, U_2, \ldots, U_N$ such that

$$U = U_1 U_2 \ldots U_N. \tag{4.46}$$

# Proof of lemma: d = 3

*Proof.* The proof requires several steps. It is instructive to start with the case $d = 3$. Let

$$U = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & j \end{pmatrix}$$

be a unitary matrix. We want to find two-level unitary matrices $U_1, U_2, U_3$ such that

$$U_3 U_2 U_1 U = I.$$

Then it follows that

$$U = U_1^\dagger U_2^\dagger U_3^\dagger.$$

(Never mind the daggers! If $U_k$ is two-level unitary, $U_k^\dagger$ is also two-level unitary.) We prove the above decomposition by constructing $U_k$ explicitly.

(i) Let

$$U_1 = \begin{pmatrix} \frac{a^*}{u} & \frac{b^*}{u} & 0 \\ -\frac{b}{u} & \frac{a}{u} & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where $u = \sqrt{|a|^2 + |b|^2}$. Verify that $U_1$ is unitary. Then we obtain

$$U_1 U = \begin{pmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{pmatrix},$$

where $a', \ldots, j'$ are some complex numbers, whose details are not necessary. Observe that, with this choice of $U_1$, the first component of the second row vanishes.

(ii) Let

$$U_2 = \begin{pmatrix} \frac{a'^*}{u'} & 0 & \frac{c'^*}{u'} \\ 0 & 1 & 0 \\ -\frac{c'}{u'} & 0 & \frac{a'}{u'} \end{pmatrix} = \begin{pmatrix} a'^* & 0 & c'^* \\ 0 & 1 & 0 \\ -c' & 0 & a' \end{pmatrix},$$

where $u' = \sqrt{|a'|^2 + |c'|^2} = 1$. Then

$$U_2 U_1 U = \begin{pmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{pmatrix},$$

where the equality $d'' = g'' = 0$ follows from the fact that $U_2 U_1 U$ is unitary, and hence the first row must be normalized.

(iii) Finally let

$$U_3 = (U_2 U_1 U)^\dagger = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e''^* & f''^* \\ 0 & h''^* & j''^* \end{pmatrix}.$$

Then, by definition, $U_3 U_2 U_1 U = I$ is obvious. This completes the proof for $d = 3$.

The moral of the lemma is that with N two-level unitary matrices there are enough degrees of freedom to play with, to reproduce any unitary matrix of dimension d.

# Proof of lemma: any d

Suppose $U$ is a unitary matrix acting on $\mathbb{C}^d$ with a general dimension $d$. Then by repeating the above arguments, we find two-level unitary matrices $U_1, U_2, \ldots, U_{d-1}$ such that

$$U_{d-1} \ldots U_2 U_1 U = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ & & \ldots & \ldots & \\ 0 & * & * & \ldots & * \end{pmatrix},$$

namely the $(1,1)$ component is unity and other components of the first row and the first column vanish. The number of matrices $\{U_k\}$ to achieve this form is the same as the number of zeros in the first column, hence $(d-1)$.

We then repeat the same procedure to the $(d-1) \times (d-1)$ block unitary matrix using $(d-2)$ two-level unitary matrices. After repeating this, we finally decompose $U$ into a product of two-level unitary matrices

$$U = V_1 V_2 \ldots V_N,$$

where $N \leq (d-1) + (d-2) + \ldots + 1 = d(d-1)/2$. ∎

# Exercise

**EXERCISE 4.12** Let $U$ be a general $4 \times 4$ unitary matrix. Find two-level unitary matrices $U_1, U_2$ and $U_3$ such that

$$U_3 U_2 U_1 U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix}.$$

**EXERCISE 4.13** Let

$$U = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}. \tag{4.47}$$

Decompose $U$ into a product of two-level unitary matrices. Esercizio 4.37 del Nielsen Chuang

# Universality theorem

**THEOREM 4.2** (Barenco *et al.*) The set of single qubit gates and CNOT gate are universal. Namely, any unitary gate acting on an $n$-qubit register can be implemented with single qubit gates and CNOT gates.

Proof. Thanks to the previous lemma, it suffices to prove the theorem for a **two-level unitary matrix**, acting non trivially on two qubits s and t.

$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}, \quad (a, b, c, d \in \mathbb{C})$$

In the example ($2^3$ dim matrix):
s = 000 and t = 111

$$s = s_{n-1}2^{n-1} + \ldots + s_1 2 + s_0 \qquad t = t_{n-1}2^{n-1} + \ldots + t_1 2 + t_0$$

**Step 1.** The two-level unitary matrix U can be reduced to a 2x2 unitary matrix.

$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix} \quad \Rightarrow \quad \tilde{U} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}$$

Define the **Gray code** connecting s and t. It is a sequence of binary numbers such that adjacent numbers differ only by one bit. In our case s = 000 and t = 111; an example of Gray code is

$$\begin{array}{c} \\ g_1 = \\ g_2 = \\ g_3 = \\ g_4 = \end{array} \begin{array}{ccc} q_1 & q_2 & q_3 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}$$

If s and t differ in p bits, the shortest Gray code is made of p+1 elements

The strategy now is to find gates providing the sequence of state changes

$$|s\rangle = |g_1\rangle \rightarrow |g_2\rangle \rightarrow \ldots \rightarrow |g_{m-1}\rangle$$

Then $g_{m-1}$ and $g_m$ differ only in one bit, which is identified with the single qubit on which U acts. After having applied the U gate, we bring things back. In our example:

$$|s\rangle = |000\rangle \longrightarrow |100\rangle \longrightarrow |110\rangle = |11\rangle \otimes |0\rangle$$
$$|t\rangle \qquad\qquad\qquad\qquad\qquad\qquad |11\rangle \otimes |1\rangle$$

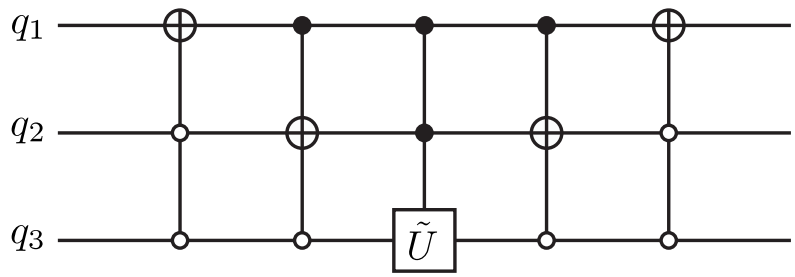
DO the Gary code

Act with the 2x2 gate $\tilde{U}$

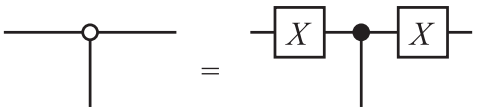
UNDO the Gary code

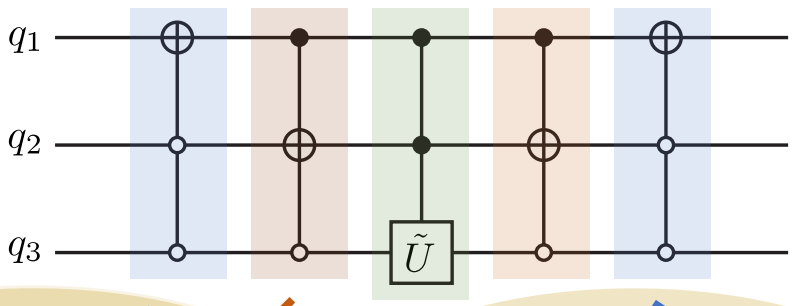
# Universality theorem: d = 3 example

$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix} =$$




Where we defined




$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}$$

It changes
$|x00\rangle \rightarrow |x^{\neg}00\rangle$

It changes
$|1x0\rangle \rightarrow |1x^{\neg}0\rangle$

DO

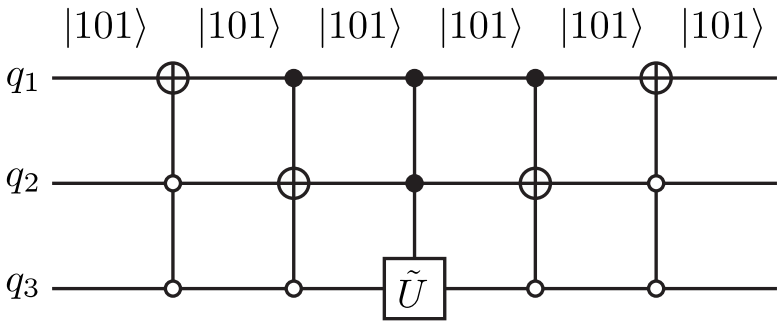
It changes
$|1x0\rangle \rightarrow |1x^{\neg}0\rangle$

It changes
$|1x0\rangle \rightarrow |1x^{\neg}0\rangle$

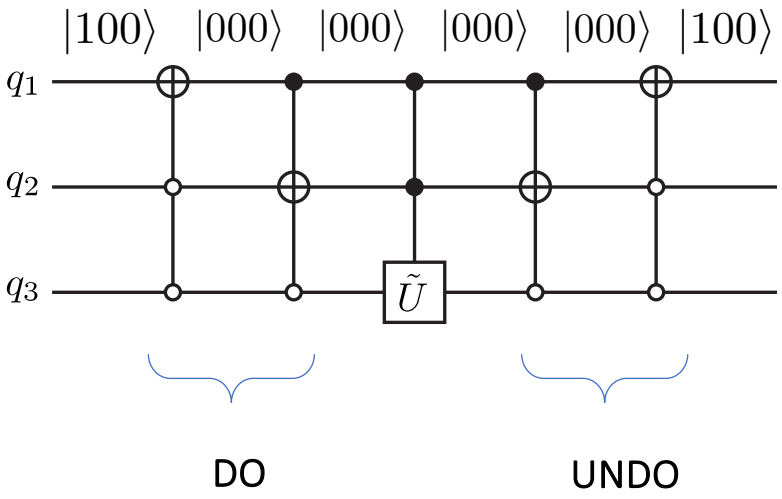
UNDO

| | $q_1$ | $q_2$ | $q_3$ |
|---|---|---|---|
| $g_1 =$ | 0 | 0 | 0 |
| $g_2 =$ | 1 | 0 | 0 |
| $g_3 =$ | 1 | 1 | 0 |
| $g_4 =$ | 1 | 1 | 1 |

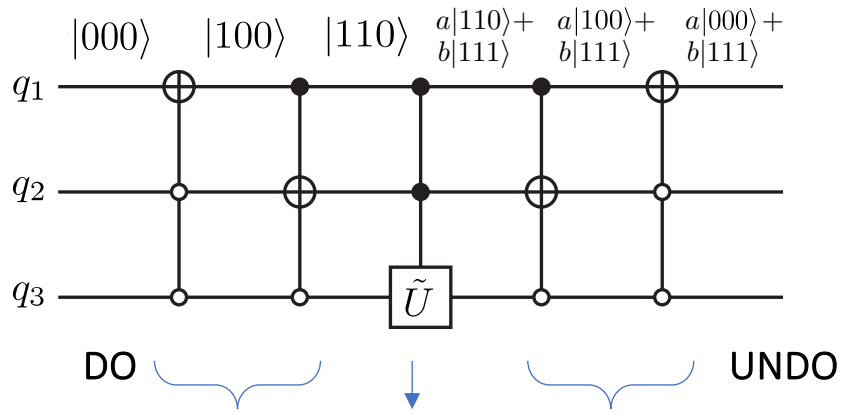
Let us consider the effect on a qubit different from |s> and |t>, for example the qubit |101>




Or the qubit |100>




While on |000>




The gate acts only on the third qubit

# Exercises

**EXERCISE 4.14** (1) Find the shortest Gray code which connects 000 with 110.
(2) Use this result to find a quantum circuit, such as Fig. 4.5, implementing a two-level unitary gate

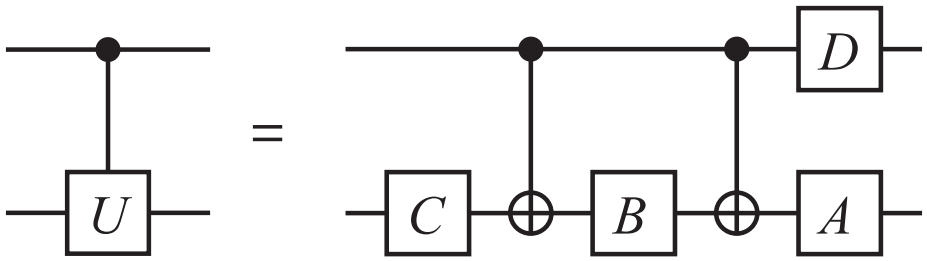
$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & c & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \tilde{U} \equiv \begin{pmatrix} a & c \\ b & d \end{pmatrix} \in U(2).$$

**Exercise 4.39:** Find a quantum circuit using single qubit operations and CNOTs to implement the transformation

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 & 0 & 0 & c \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & 0 & d \end{bmatrix}, \tag{4.60}$$

It will be shown next that all the gates in the above circuit can be implemented with single-qubit gates and CNOT gates, which proves the universality of these gates.

**Step 2.** The controlled-U gate is decomposed in the CNOT gate and single qubit gates




**LEMMA 4.2** Let $U \in \mathrm{SU}(2)$. Then there exist $\alpha, \beta, \gamma \in \mathbb{R}$ such that $U = R_z(\alpha)R_y(\beta)R_z(\gamma)$, where

$$R_z(\alpha) = \exp(i\alpha\sigma_z/2) = \begin{pmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{pmatrix},$$
$$R_y(\beta) = \exp(i\beta\sigma_y/2) = \begin{pmatrix} \cos(\beta/2) & \sin(\beta/2) \\ -\sin(\beta/2) & \cos(\beta/2) \end{pmatrix}.$$

*Proof.* After some calculation, we obtain

$$R_z(\alpha)R_y(\beta)R_z(\gamma) = \begin{pmatrix} e^{i(\alpha+\gamma)/2}\cos(\beta/2) & e^{i(\alpha-\gamma)/2}\sin(\beta/2) \\ -e^{i(-\alpha+\gamma)/2}\sin(\beta/2) & e^{-i(\alpha+\gamma)/2}\cos(\beta/2) \end{pmatrix}. \quad (4.53)$$

Any $U \in \mathrm{SU}(2)$ may be written in the form

$$U = \begin{pmatrix} a & b \\ -b^* & a^* \end{pmatrix} = \begin{pmatrix} \cos\theta e^{i\lambda} & \sin\theta e^{i\mu} \\ -\sin\theta e^{-i\mu} & \cos\theta e^{-i\lambda} \end{pmatrix}, \quad (4.54)$$

where we used the fact that $\det U = |a|^2 + |b|^2 = 1$. Now we obtain $U = R_z(\alpha)R_y(\beta)R_z(\gamma)$ by making identifications

$$\theta = \frac{\beta}{2}, \lambda = \frac{\alpha+\gamma}{2}, \mu = \frac{\alpha-\gamma}{2}. \quad (4.55)$$

∎

**LEMMA 4.3** Let $U \in \mathrm{SU}(2)$. Then there exist $A, B, C \in \mathrm{SU}(2)$ such that $U = AXBXC$ and $ABC = I$, where $X = \sigma_x$.

*Proof.* Lemma 4.2 states that $U = R_z(\alpha)R_y(\beta)R_z(\gamma)$ for some $\alpha, \beta, \gamma \in \mathbb{R}$. Let

$$A = R_z(\alpha)R_y\left(\frac{\beta}{2}\right), B = R_y\left(-\frac{\beta}{2}\right)R_z\left(-\frac{\alpha+\gamma}{2}\right), C = R_z\left(-\frac{\alpha-\gamma}{2}\right).$$

Then

$$\begin{aligned} AXBXC &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)XR_y\left(-\frac{\beta}{2}\right)R_z\left(-\frac{\alpha+\gamma}{2}\right)XR_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)\left[XR_y\left(-\frac{\beta}{2}\right)X\right]\left[XR_z\left(-\frac{\alpha+\gamma}{2}\right)X\right]R_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)R_y\left(\frac{\beta}{2}\right)R_z\left(\frac{\alpha+\gamma}{2}\right)R_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y(\beta)R_z(\gamma) = U, \end{aligned}$$

where use has been made of the identities $X^2 = I$ and $X\sigma_{y,z}X = -\sigma_{y,z}$.

It is also verified that

$$\begin{aligned} ABC &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)R_y\left(-\frac{\beta}{2}\right)R_z\left(-\frac{\alpha+\gamma}{2}\right)R_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y(0)R_z(-\alpha) = I. \end{aligned}$$

This proves the Lemma. ∎

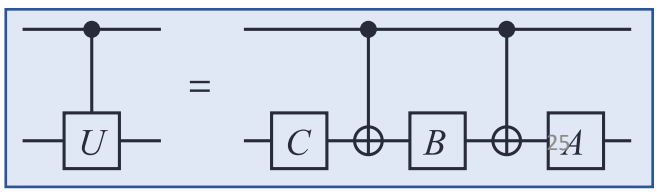
**LEMMA 4.4** Let $U \in \mathrm{SU}(2)$ be factorized as $U = AXBXC$ as in the previous Lemma. Then the controlled-$U$ gate can be implemented with at most three single-qubit gates and two CNOT gates (see Fig. 4.8).

*Proof.* The proof is almost obvious. When the control bit is 0, the target bit $|\psi\rangle$ is operated by $C, B$ and $A$ in this order so that

$$|\psi\rangle \mapsto ABC|\psi\rangle = |\psi\rangle,$$

while when the control bit is 1, we have

$$|\psi\rangle \mapsto AXBXC|\psi\rangle = U|\psi\rangle.$$



# From SU(2) to (2)

So far, we have worked with $U \in \mathrm{SU}(2)$. To implement a general $U$-gate with $U \in \mathrm{U}(2)$, we have to deal with the phase. Let us first recall that any $U \in \mathrm{U}(2)$ is decomposed as $U = e^{i\alpha}V,\ V \in \mathrm{SU}(2), \alpha \in \mathbb{R}$.

**LEMMA 4.5** Let

$$\Phi(\phi) = e^{i\phi} I = \begin{pmatrix} e^{i\phi} & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

and

$$D = R_z(-\phi)\Phi\left(\frac{\phi}{2}\right) = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix} \begin{pmatrix} e^{i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}.$$

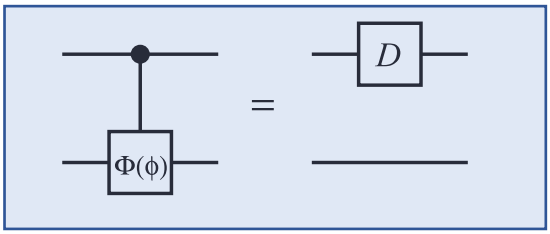
Then the controlled-$\Phi(\phi)$ gate is expressed as a tensor product of single qubit gates as

$$U_{\mathrm{C}\Phi(\phi)} = D \otimes I. \tag{4.56}$$

.

*Proof.* The LHS is

$$\begin{aligned} U_{\mathrm{C}\Phi(\phi)} &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes \Phi(\phi) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes e^{i\phi} I \\ &= |0\rangle\langle 0| \otimes I + e^{i\phi}|1\rangle\langle 1| \otimes I, \end{aligned}$$




while the RHS is

$$\begin{aligned} D \otimes I &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \otimes I \\ &= \left[|0\rangle\langle 0| + e^{i\phi}|1\rangle\langle 1|\right] \otimes I = U_{\mathrm{C}\Phi(\phi)}, \end{aligned}$$

which proves the lemma. ∎

**EXERCISE 4.15** Let us consider the controlled-$V_1$ gate $U_{\mathrm{C}V_1}$ and the controlled-$V_2$ gate $U_{\mathrm{C}V_2}$. Show that the controlled-$V_1$ gate followed by the controlled-$V_2$ gate is the controlled-$V_2V_1$ gate $U_{\mathrm{C}(V_2V_1)}$ as shown in Fig. 4.10.

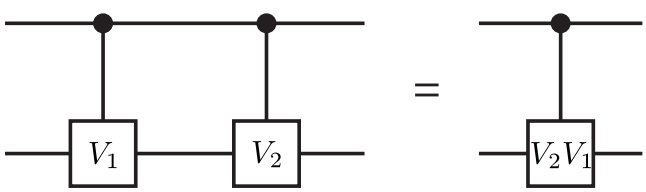



**FIGURE 4.10**
Equality $U_{\mathrm{C}V_2}U_{\mathrm{C}V_1} = U_{\mathrm{C}(V_2V_1)}$.

# Controlled-U gate with U in U(2)

**PROPOSITION 4.1** Let $U \in \mathrm{U}(2)$. Then the controlled-$U$ gate $U_{CU}$ can be constructed by at most four single-qubit gates and two CNOT gates.

*Proof.* Let $U = \Phi(\phi)AXBXC$. According to the exercise above, the controlled-$U$ gate is written as a product of the controlled-$\Phi(\phi)$ gate and the controlled-$AXBXC$ gate. Moreover, Lemma 4.5 states that the controlled-$\Phi(\phi)$ gate may be replaced by a single-qubit phase gate acting on the first qubit. The rest of the gate, the controlled-$AXBXC$ gate is implemented with three SU(2) gates and two CNOT gates as proved in Lemma 4.3. Therefore we have the following decomposition:

$$U_{CU} = (D \otimes A)U_{\mathrm{CNOT}}(I \otimes B)U_{\mathrm{CNOT}}(I \otimes C), \tag{4.57}$$

where

$$D = R_z(-\phi)\Phi(\phi/2)$$

and use has been made of the identity $(D \otimes I)(I \otimes A) = D \otimes A$. ∎

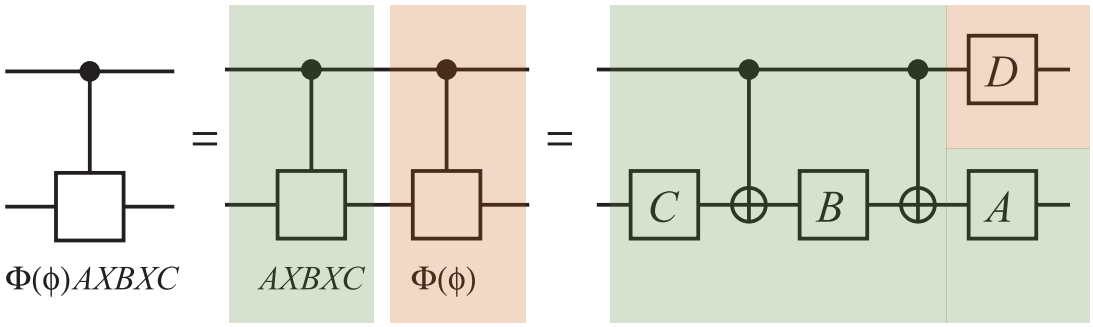
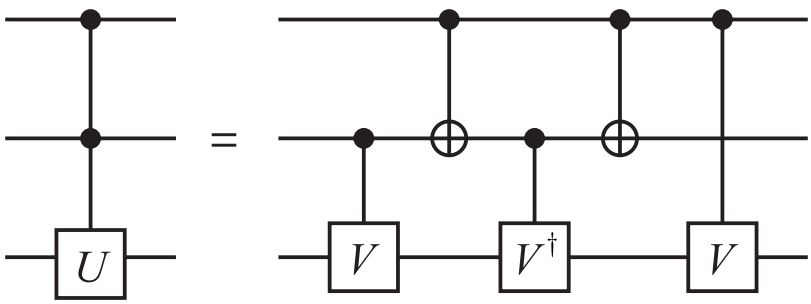



**FIGURE 4.11**
Controlled-$U$ gate is implemented with at most four single-qubit gates and two CNOT gates.

**Step 3.** The CCNOT gate and its variants are implemented with CNOT gates and its variants

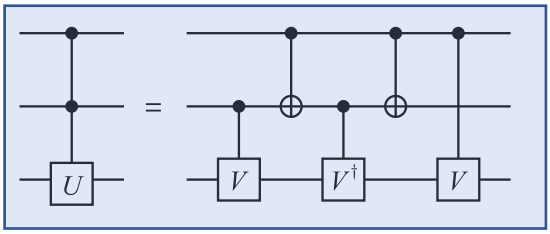



**LEMMA 4.6** The two quantum circuits in Fig. 4.12 are equivalent, where $U = V^2$.

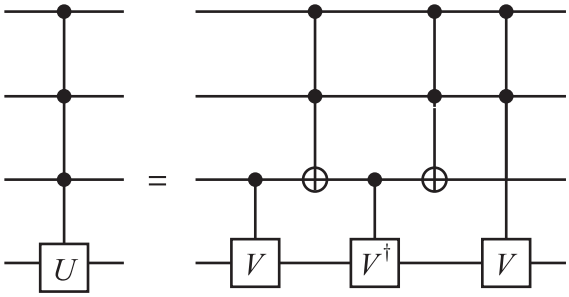
*Proof.* If both the first and the second qubits are 0 in the RHS, all the gates are ineffective and the third qubit is unchanged; the gate in this subspace acts as $|00\rangle\langle 00| \otimes I$. In case the first qubit is 0 and the second is 1, the third qubit is mapped as $|\psi\rangle \mapsto V^\dagger V|\psi\rangle = |\psi\rangle$; the gate is then $|01\rangle\langle 01| \otimes I$. When the first qubit is 1 and the second is 0, the third qubit is mapped as $|\psi\rangle \mapsto VV^\dagger|\psi\rangle = |\psi\rangle$; hence the gate in this subspace is $|10\rangle\langle 10| \otimes I$. Finally let both the first and the second qubits be 1. Then the action of the gate on the third qubit is $|\psi\rangle \mapsto VV|\psi\rangle = U|\psi\rangle$; namely the gate in this subspace is $|11\rangle\langle 11| \otimes U$. Thus it has been proved that the RHS of Fig. 4.12 is

$$(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I + |11\rangle\langle 11| \otimes U, \tag{4.58}$$

namely the controlled-controlled-$U$ gate. ∎



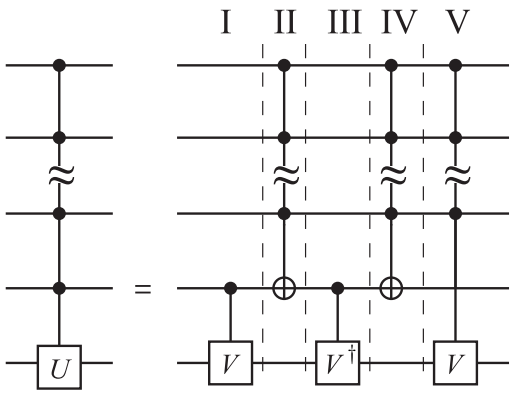
**EXERCISE 4.17** Show that the circuit in Fig. 4.13 is a controlled-$U$ gate with three control bits, where $U = V^2$.




**PROPOSITION 4.2** The quantum circuit in Fig. 4.14 with $U = V^2$ is a decomposition of the controlled-$U$ gate with $n-1$ control bits.

The proof of the above proposition is very similar to that of Lemma 4.6 and Exercise 4.17 and is left as an exercise to the readers.

Theorem 4.2 has been now proved. ∎




order. The above controlled-$U$ gate with $(n-1)$ control bits requires $\Theta(n^2)$ elementary gates.*† Let us write the number of the elementary gates required to construct the gate in Fig. 4.14 by $C(n)$. Construction of layers I and III requires elementary gates whose number is independent of $n$. It can be shown that the number of the elementary gates required to construct the controlled NOT gate with $(n-2)$ control bits is $\Theta(n)$ [14]. Therefore layers II and IV require $\Theta(n)$ elementary gates. Finally the layer V, a controlled-$V$ gate with $(n-2)$ control bits, requires $C(n-1)$ basic gates by definition. Thus we obtain a recursion relation

$$C(n) - C(n-1) = \Theta(n). \tag{4.59}$$

The solution to this recursion relation is

$$C(n) = \Theta(n^2). \tag{4.60}$$

Therefore, implementation of a controlled-$U$ gate with $U \in \mathrm{U}(2)$ and $(n-1)$ control bits requires $\Theta(n^2)$ elementary gates.

# Algorithms with oracle

Suppose we are supplied with a quantum **oracle** – a black box whose internal workings are not important at this stage – with the ability to recognize solutions to a given problem by computing a suitable function f. More precisely, the oracle is a unitary operator, $U_f$, defined by its action on the computational basis:

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

where $\bigoplus$ is addition mod 2.

Suppose $U_f$ acts on the input which is a **superposition of many states**. Since $U_f$ is a linear operator, it acts simultaneously on all the vectors that constitute the superposition. **Thus the output is also a superposition of all the results**

$$U_f : \sum_x |x\rangle|0\rangle \mapsto \sum_x |x\rangle|f(x)\rangle.$$

All values of the function computed at once. Very easy!!
But… **measurements will make the wave function collapse giving only one output**. No advantage.

The goal of a quantum algorithm is to operate in such a way that the particular outcome we want to observe has a larger probability to be measured than the other outcomes.

# Deutsch Algorithm

Let $f : \{0,1\} \to \{0,1\}$ be a binary function. Note that there are only four possible $f$, namely

$$f_1 : 0 \mapsto 0,\ 1 \mapsto 0, \quad f_2 : 0 \mapsto 1,\ 1 \mapsto 1,$$
$$f_3 : 0 \mapsto 0,\ 1 \mapsto 1, \quad f_4 : 0 \mapsto 1,\ 1 \mapsto 0.$$

The first two cases, $f_1$ and $f_2$, are called *constant*, while the rest, $f_3$ and $f_4$, are *balanced*. If we only have classical resources, we need to evaluate $f$ twice to tell if $f$ is constant or balanced. There is a quantum algorithm, however, with which it is possible to tell if $f$ is constant or balanced with a single evaluation of $f$, as was shown by Deutsch [2].

First we need to turn the classical function f(x) into a quantum one.

To this purpose we define the quantum oracle

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

The algorithm is structured as follows.

1. Start with the state $|01\rangle$.
2. Apply an Hadamard on both qubits: $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$
3. Apply the operator $U_f$ implementing the function

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

**Quantum parallelism**: all values computed at once

# Deutsch Algorithm

4. Apply an Hadamard to the first qubit

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

If the function is constant, the two blue terms are equal, the state reduces to

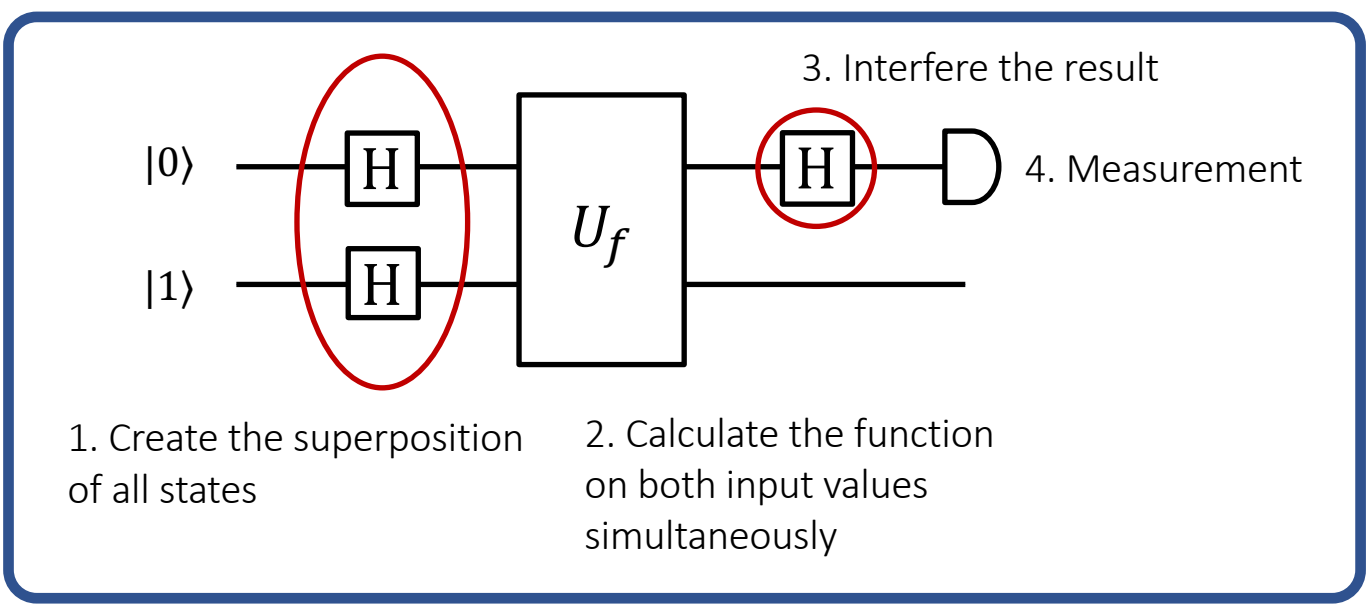
$$\frac{1}{\sqrt{2}}|0\rangle(|f(0)\rangle - |\neg f(0)\rangle)$$

If the function is balanced, the two blue terms are opposite, the state reduces to

$$\frac{1}{\sqrt{2}}|1\rangle(|f(0)\rangle - |\neg f(0)\rangle)$$

Therefore **the measurement of the first qubit tells us whether f is constant or balanced.**

5. Measure the first qubit to determine f. The full algorithm is:



# Deutsch-Jozsa Algorithm

Let us first define the **Deutsch-Jozsa problem**. Suppose there is a binary function

$$f : S_n \equiv \{0, 1, \ldots, 2^n - 1\} \to \{0, 1\}.$$

We require that $f$ be either *constant* or *balanced* as before. When $f$ is constant, it takes a constant value 0 or 1 irrespetive of the input value $x$. When it is balanaced the value $f(x)$ for the half of $x \in S_n$ is 0, while it is 1 for the rest of $x$. In other words, $|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$, where $|A|$ denotes the number

It is clear that we need at least $2^{n-1} + 1$ steps, in the worst case with classical manipulations, to make sure if $f(x)$ is constant or balanced with 100% confidence. It will be shown below that the number of steps reduces to a single step if we are allowed to use a quantum algorithm.

## The **oracle** for the Deutsch-Jozsa algorithm is always the same

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

1. Prepare an $(n+1)$-qubit register in the state $|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$. First $n$ qubits work as input qubits, while the $(n+1)$st qubit serves as a "scratch pad." Such qubits, which are neither input qubits nor output qubits, but work as a scratch pad to store temporary information are called **ancillas** or **ancillary qubits**.

2. Apply the Walsh-Hadamard transforamtion to the register. Then we have the state

$$U_{\rm H}^{\otimes n+1}|\psi_0\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n} \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

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

3. Apply the oracle. The state changes into

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

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

Although the gate $U_f$ is applied once for all, it is applied to *all* the $n$-qubit states $|x\rangle$ simultaneously.

# Deutsch-Jozsa Algorithm

4. The Walsh-Hadamard transformation is applied on the first $n$ qubits next. We obtain

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

## On the Hadamard gate

It is instructive to write the action of the one-qubit Hadamard gate in the following form,

$$U_{\rm H}|x\rangle=\frac{1}{\sqrt{2}}(|0\rangle+(-1)^x|1\rangle)=\frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle,$$

where $x\in\{0,1\}$, to find the resulting state. The action of the Walsh-Hadamard transformation on $|x\rangle=|x_{n-1}\dots x_1x_0\rangle$ yields

$$\begin{aligned}W_n|x\rangle&=(U_{\rm H}|x_{n-1}\rangle)(U_{\rm H}|x_{n-2}\rangle)\dots(U_{\rm H}|x_0\rangle)\\&=\frac{1}{\sqrt{2^n}}\sum_{y_{n-1},y_{n-2},\dots,y_0\in\{0,1\}}(-1)^{x_{n-1}y_{n-1}+x_{n-2}y_{n-2}+\dots+x_0y_0}\\&\quad\times|y_{n-1}y_{n-2}\dots y_0\rangle\\&=\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{x\cdot y}|y\rangle,\end{aligned}$$

where $x\cdot y=x_{n-1}y_{n-1}\oplus x_{n-2}y_{n-2}\oplus\dots\oplus x_0y_0$.

We then have

$$=\frac{1}{2^n}\left(\sum_{x,y=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot y}|y\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle).$$

# Deutsch-Jozsa Algorithm

5. The first $n$ qubits are measured. Suppose $f(x)$ is constant. Then

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

up to an overall phase. Now let us consider the summation

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{x\cdot y}$$

with a fixed $y \in S_n$. Clearly it vanishes since $x \cdot y$ is 0 for half of $x$ and 1 for the other half of $x$ unless $y = 0$. Therefore the summation yields $\delta_{y0}$. Now the state reduces to

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

and the measurement outcome of the first $n$ qubits is always $00\ldots0$.

Suppose $f(x)$ is balanced next. The probability amplitude of $|y=0\rangle$ in $|\psi_3\rangle$ is proportional to

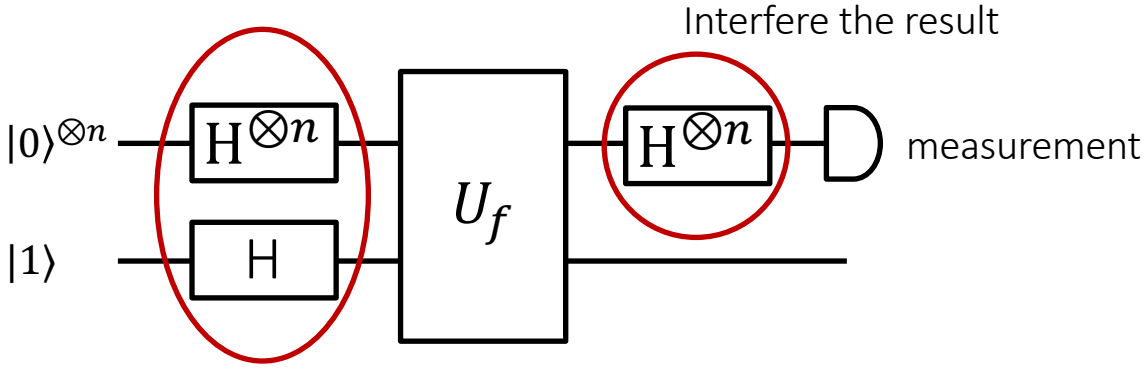
$$\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot 0}=\sum_{x=0}^{2^n-1}(-1)^{f(x)}=0.$$

Therefore the probability of obtaining measurement outcome $00\ldots0$ for the first $n$ qubits vanishes. In conclusion, the function $f$ is constant if we obtain $00\ldots0$ upon the meaurement of the first $n$ qubits in the state $|\psi_3\rangle$, and it is balanced otherwise.

Example with 3 qubits.

Take y = 110. Then
$x\cdot y = x_2 \oplus x_1$

| x | $x_2 \oplus x_1$ |
|---|---|
| 000 | 0 |
| 001 | 0 |
| 010 | 1 |
| 011 | 1 |
| 100 | 1 |
| 101 | 1 |
| 110 | 0 |
| 111 | 0 |




Calculate the function on both input values simultaneously

# Bernstein-Vazirani Algorithm

The **Bernstein-Vazirani algorithm** is a special case of the Deutsch-Jozsa algorithm, in which $f(x)$ is given by $f(x) = c \cdot x$, where $c = c_{n-1}c_{n-2}\dots c_0$ is an $n$-bit binary number [4]. Our aim is to find $c$ with the smallest number of evaluations of $f$. If we apply the Deutsch-Jozsa algorithm with this $f$, we obtain

$$|\psi_3\rangle = \frac{1}{2^n}\left[\sum_{x,y=0}^{2^n-1}(-1)^{c\cdot x}(-1)^{x\cdot y}|y\rangle\right]\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Let us fix $y$ first. If we take $y = c$, we obtain

$$\sum_x (-1)^{c\cdot x}(-1)^{x\cdot c} = \sum_x (-1)^{2c\cdot x} = 2^n.$$

If $y \neq c$, on the other hand, there will be the same number of $x$ such that $c \cdot x = 0$ and $x$ such that $c \cdot x = 1$ in the summation over $x$ and, as a result, the probability amplitude of $|y \neq c\rangle$ vanishes. By using these results, we end up with

$$|\psi_3\rangle = |c\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

We are able to tell what $c$ is by measuring the first $n$ qubits.

# Exercise

**EXERCISE 5.1** Let us take $n = 2$ for definiteness. Consider the following cases and find the final wave function $|\psi_3\rangle$ and evaluate the measurement outcomes and their probabilities for each case.
(1) $f(x) = 1\ \forall x \in S_2$.
(2) $f(00) = f(01) = 1, f(10) = f(11) = 0$.
(3) $f(00) = 0, f(01) = f(10) = f(11) = 1$. (This function is neither constant nor balanced.)

**EXERCISE 5.2** Consider the Bernstein-Vazirani algorithm with $n = 3$ and $c = 101$. Work out the quantum circuit depicted in Fig. 5.2 to show that the measurement outcome of the first three qubits is $c = 101$.

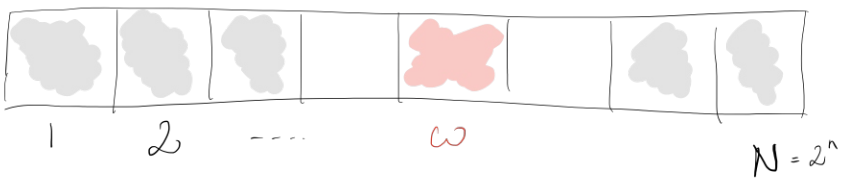
# Grover's search algorithm

Suppose there is a stack of $N = 2^n$ files, randomly placed, that are numbered by $x \in S_n \equiv \{0, 1, \ldots, N-1\}$. Our task is to find an algorithm which picks out a particular file which satisfies a certain condition.

In mathematical language, this is expressed as follows. Let $f : S_n \to \{0, 1\}$ be a function defined by

$$f(x) = \begin{cases} 1 & (x = z) \\ 0 & (x \neq z), \end{cases}$$

where $z$ is the address of the file we are looking for. It is assumed that $f(x)$ is *instantaneously* calculable, such that this process does not require any computational steps. A function of this sort is often called an oracle as noted in Chapter 5. Thus, the problem is to find $z$ such that $f(z) = 1$, given a function $f : S_n \to \{0, 1\}$ which assumes the value 1 only at a single point.




Clearly we have to check one file after another in a classical algorithm, and it will take $O(N)$ steps on average. It is shown below that it takes only $O(\sqrt{N})$ steps with Grover's algorithm. This is accomplished by *amplifying* the amplitude of the vector $|z\rangle$ while cancelling that of the vectors $|x\rangle$ $(x \neq z)$.

We first needs to implement the function f(x) quantum mechanically. The **oracle** $U_f$ is defined in the usual way

$$U_f|x, y\rangle = |x, y \oplus f(x)\rangle \qquad \text{with } f(x) = \text{either 0 or 1}$$

In particular we have

$$U_f|x\rangle \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle] = (-1)^{f(x)}|x\rangle \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle]$$

# Grover's search algorithm

Therefore, without loss of generality, we can neglect the last qubit and assume

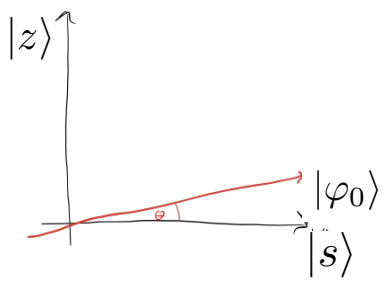
$$U_f|x\rangle = (-1)^{f(x)}|x\rangle$$

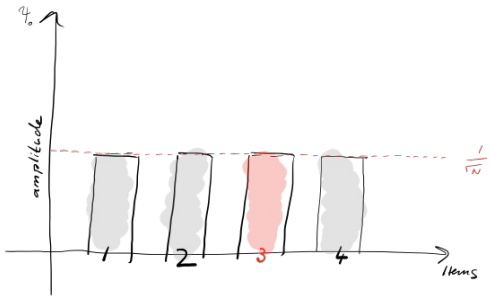
on the computational basis. We see that if $x$ is an unmarked item, the oracle does nothing to the state. It flips the phase for the marked item. It is easy to see that

$$U_f = I - 2|z\rangle\langle z|$$

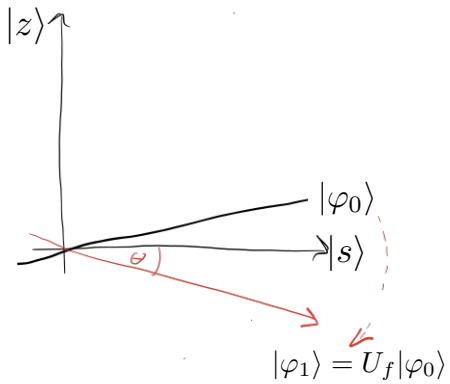
**Step 1:** Create an initially **equal weighted superposition** of all states (this is done with N Hadamard gates):

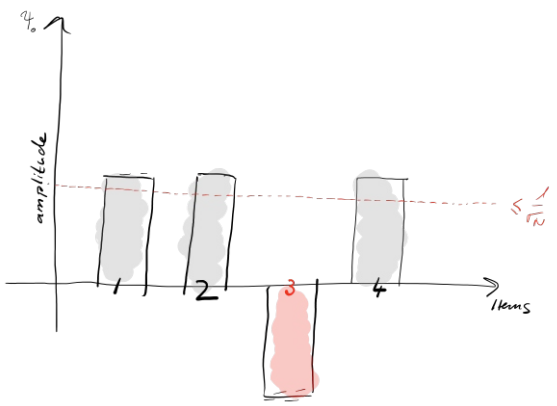
$$|\varphi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$



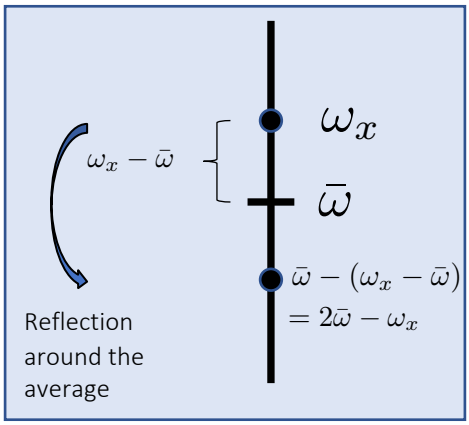




**Step 2: Apply the oracle $U_f$**. Geometrically this corresponds to a reflection of the state $|z\rangle$ about $|s\rangle$. This transformation means that the amplitude in front of the $|z\rangle$ state becomes negative, which in turn means that the average amplitude has been lowered.



# Grover's search algorithm




Step 3: Apply the gate

$$D = -I + 2|\varphi_0\rangle\langle\varphi_0|.$$

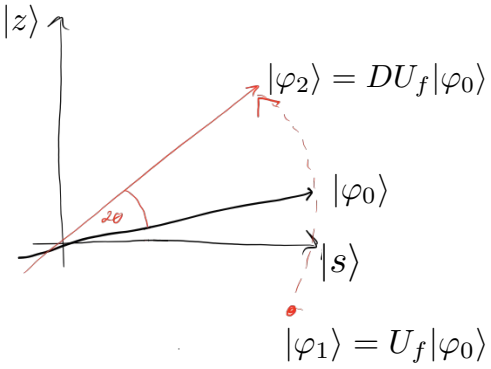
The action of the gate is the following

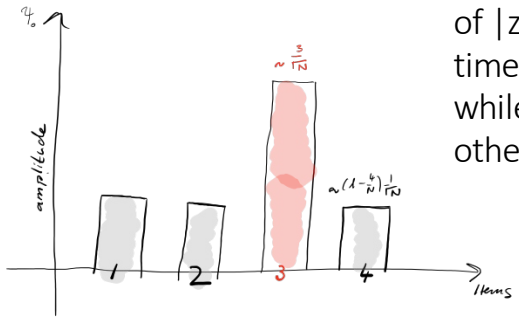
$$|\varphi\rangle = \sum_{x=0}^{N-1} \omega_x |x\rangle \;\rightarrow\; D|\varphi\rangle = \left[\frac{2}{N}\sum_{x,y=0}^{N-1}|x\rangle\langle y|\right]\sum_{z=0}^{N-1}\omega_z|z\rangle - \sum_{x=0}^{N-1}\omega_x|x\rangle$$

$$= \frac{2}{N}\left[\sum_{x=0}^{N-1}|x\rangle\right]\left[\sum_{y=0}^{N-1}\omega_y\right] - \sum_{x=0}^{N-1}\omega_x|x\rangle = \sum_{x=0}^{N-1}(2\bar{\omega} - \omega_x)|x\rangle$$

with $\bar{\omega} = \frac{1}{N}\sum_{x=0}^{N-1}\omega_x$ average

In our case we get




Since the average amplitude has been lowered by the first reflection, this transformation boosts the negative amplitude of $|z\rangle$ to roughly three times its original value, while it decreases the other amplitudes.

# Grover's search algorithm

**Step 3:** go to step 2 an repeat the application of $U_f$ and D a sufficient number of times. Let us call $G_f = D\, U_f$.

**PROPOSITION 7.2** Let us write

$$G_f^k|\varphi_0\rangle = a_k|z\rangle + b_k \sum_{x\neq z} |x\rangle$$

with the initial condition

$$a_0 = b_0 = \frac{1}{\sqrt{N}}.$$

Then the coefficients $\{a_k, b_k\}$ for $k \geq 1$ satisfy the recursion relations

$$a_k = \frac{N-2}{N} a_{k-1} + \frac{2(N-1)}{N} b_{k-1}, \tag{7.18}$$

$$b_k = -\frac{2}{N} a_{k-1} + \frac{N-2}{N} b_{k-1} \tag{7.19}$$

for $k = 1, 2, \ldots$.

*Proof.* It is easy to see the recursion relations are satified for $k = 1$
Let $G_f^{k-1}|\varphi_0\rangle = a_{k-1}|z\rangle + b_{k-1}\sum_{x\neq z}|x\rangle$. Then

$$\begin{aligned}
G_f^k|\varphi_0\rangle =& G_f \left( a_{k-1}|z\rangle + b_{k-1}\sum_{x\neq z}|x\rangle \right) \\
=& \left(-I + 2|\varphi_0\rangle\langle\varphi_0|\right) \left( -a_{k-1}|z\rangle + b_{k-1}\sum_{x\neq z}|x\rangle \right) \\
=& -b_{k-1}\sum_{x\neq z}|x\rangle + a_{k-1}|z\rangle + \frac{2}{\sqrt{N}}(N-1)b_{k-1}|\varphi_0\rangle - \frac{2a_{k-1}}{\sqrt{N}}|\varphi_0\rangle \\
=& -b_{k-1}\sum_{x\neq z}|x\rangle + a_{k-1}|z\rangle + \frac{2}{N}(N-1)b_{k-1}\sum_{x}|x\rangle - \frac{2a_{k-1}}{N}\sum_{x}|x\rangle \\
=& \left[\frac{N-2}{N}a_{k-1} + \frac{2(N-1)}{N}b_{k-1}\right]|z\rangle + \left[-\frac{2}{N}a_{k-1} + \frac{N-2}{N}b_{k-1}\right]\sum_{x\neq z}|x\rangle,
\end{aligned}$$

and proposition is proved. ∎

# Grover's search algorithm

**PROPOSITION 7.3** The solutions of the recursion relations in Proposition 7.2 are explicitly given by

$$a_k = \sin[(2k+1)\theta], \quad b_k = \frac{1}{\sqrt{N-1}} \cos[(2k+1)\theta], \tag{7.20}$$

for $k = 0, 1, 2, \ldots$, where

$$\sin\theta = \sqrt{\frac{1}{N}}, \quad \cos\theta = \sqrt{1 - \frac{1}{N}}. \tag{7.21}$$

*Proof.* Let $c_k = \sqrt{N-1} b_k$. The recursion relations (7.18) and (7.19) are written in a matrix form,

$$\begin{pmatrix} a_k \\ c_k \end{pmatrix} = M \begin{pmatrix} a_{k-1} \\ c_{k-1} \end{pmatrix}, \; M = \begin{pmatrix} (N-2)/N & 2\sqrt{N-1}/N \\ -2\sqrt{N-1}/N & (N-2)/N \end{pmatrix} = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ -\sin 2\theta & \cos 2\theta \end{pmatrix}.$$

Note that $M$ is a rotation matrix in $\mathbb{R}^2$, and its $k$th power is another rotation matrix corresponding to a rotation angle $2k\theta$. Thus the above recursion relation is easily solved to yield

$$\begin{pmatrix} a_k \\ c_k \end{pmatrix} = M^k \begin{pmatrix} a_0 \\ c_0 \end{pmatrix} = \begin{pmatrix} \cos 2k\theta & \sin 2k\theta \\ -\sin 2k\theta & \cos 2k\theta \end{pmatrix} \begin{pmatrix} \sin\theta \\ \cos\theta \end{pmatrix} = \begin{pmatrix} \sin[(2k+1)\theta] \\ \cos[(2k+1)\theta] \end{pmatrix}.$$

Replacing $c_k$ by $b_k$ proves the proposition. ∎

We have proved that the application of $G_f$ $k$ times on $|\varphi_0\rangle$ results in the state

$$G_f^k |\varphi_0\rangle = \sin[(2k+1)\theta] |z\rangle + \frac{1}{\sqrt{N-1}} \cos[(2k+1)\theta] \sum_{x \neq z} |x\rangle. \tag{7.22}$$

Measurement of the state $U_f^k |\varphi_0\rangle$ yields $|z\rangle$ with the probability

$$P_{z,k} = \sin^2[(2k+1)\theta]. \tag{7.23}$$

**STEP 4** Our final task is to find the $k$ that maximizes $P_{z,k}$. A rough estimate for the maximizing $k$ is obtained by putting

$$(2k+1)\theta = \frac{\pi}{2} \to k = \frac{1}{2}\left(\frac{\pi}{2\theta} - 1\right). \tag{7.24}$$

# Grover's search algorithm

**PROPOSITION 7.4** Let $N \gg 1$ and let

$$m = \left\lfloor \frac{\pi}{4\theta} \right\rfloor, \tag{7.25}$$

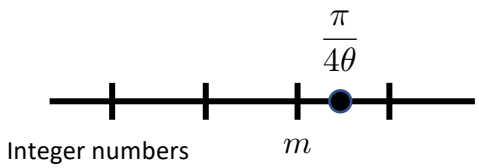
where $\lfloor x \rfloor$ stands for the floor of $x$. The file we are searching for will be obtained in $G_f^m |\varphi_0\rangle$ with the probability

$$P_{z,m} \geq 1 - \frac{1}{N} \tag{7.26}$$

and

$$m = O(\sqrt{N}). \tag{7.27}$$

This is the number of times we repeat the algorithm, which grows with the square root of N




*Proof.* Equation (7.25) leads to the inequality $\pi/4\theta - 1 < m \leq \pi/4\theta$. Let us define $\tilde{m}$ by

$$(2\tilde{m}+1)\theta = \frac{\pi}{2} \rightarrow \tilde{m} = \frac{\pi}{4\theta} - \frac{1}{2}.$$

Observe that $m$ and $\tilde{m}$ satisfy

$$|m - \tilde{m}| \leq \frac{1}{2}, \tag{7.28}$$

from which it follows that

$$|(2m+1)\theta - (2\tilde{m}+1)\theta| = \left|(2m+1)\theta - \frac{\pi}{2}\right| \leq \theta. \tag{7.29}$$

Considering that $\theta \sim 1/\sqrt{N}$ is a small number when $N \gg 1$ and $\sin x$ is monotonically increasing in the neighborhood of $x = 0$, we obtain

$$0 < \sin|(2m+1)\theta - \pi/2| < \sin\theta$$

$$= \cos[(2m+1)\theta]$$

or

$$\cos^2[(2m+1)\theta] \leq \sin^2\theta = \frac{1}{N}. \tag{7.30}$$

Thus it has been shown that

$$P_{m,z} = \sin^2[(2m+1)\theta] = 1 - \cos^2[(2m+1)\theta] \geq 1 - \frac{1}{N}. \tag{7.31}$$

It also follows from $\theta > \sin\theta = 1/\sqrt{N}$ that

$$m = \left\lfloor \frac{\pi}{4\theta} \right\rfloor \leq \frac{\pi}{4\theta} \leq \frac{\pi}{4}\sqrt{N}. \tag{7.32}$$

∎

# Grover's search algorithm

It is important to note that this quantum algorithm takes only $O(\sqrt{N})$ steps and this is much faster than the classical counterpart which requires $O(N)$ steps.

We now show how to implement the gates

# Selective Phase Rotation Transform

**DEFINITION 6.2** (**Selective Phase Rotation Transform**) Let us define a kernel

$$K_n(x,y) = e^{i\theta_x}\delta_{xy}, \quad \forall x,y \in S_n, \tag{6.43}$$

Selective phase rotation is then defined by the following unitary operator

$$U|x\rangle = \sum_{y=0}^{N-1} K(y,x)|y\rangle.$$

The matrix representations for $K_1$ and $K_2$ are

$$K_1 = \begin{pmatrix} e^{i\theta_0} & 0 \\ 0 & e^{i\theta_1} \end{pmatrix}, \quad K_2 = \begin{pmatrix} e^{i\theta_0} & 0 & 0 & 0 \\ 0 & e^{i\theta_1} & 0 & 0 \\ 0 & 0 & e^{i\theta_2} & 0 \\ 0 & 0 & 0 & e^{i\theta_3} \end{pmatrix}.$$

# Selective Phase Rotation Transform

The implementation of $K_n$ is achieved with the universal set of gates as follows. Take $n = 2$, for example. The kernel $K_2$ has been given above. This is decomposed as a product of two two-level unitary matrices as

$$K_2 = A_0 A_1, \tag{6.45}$$

where

$$A_0 = \begin{pmatrix} e^{i\theta_0} & 0 & 0 & 0 \\ 0 & e^{i\theta_1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\theta_2} & 0 \\ 0 & 0 & 0 & e^{i\theta_3} \end{pmatrix}. \tag{6.46}$$

Note that

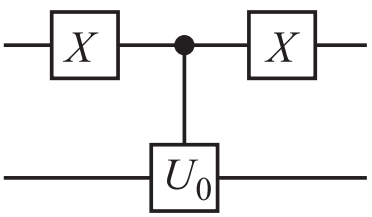
$$A_0 = |0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes I, \quad U_0 = \begin{pmatrix} e^{i\theta_0} & 0 \\ 0 & e^{i\theta_1} \end{pmatrix},$$
$$A_1 = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U_1, \quad U_1 = \begin{pmatrix} e^{i\theta_2} & 0 \\ 0 & e^{i\theta_3} \end{pmatrix}.$$

Thus $A_1$ is realized as an ordinary controlled-$U_1$ gate while the control bit is negated in $A_0$. Then what we have to do for $A_0$ is to negate the control bit first and then to apply ordinary controlled-$U_0$ gate and finally to negate the control bit back to its input state. In summary, $A_0$ is implemented as in

Fig. 6.5. In fact, it can be readily verified that the gate in Fig. 6.5 is written as

$$\begin{aligned} &(X \otimes I)(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U_0)(X \otimes I) \\ &= X|0\rangle\langle 0|X \otimes I + X|1\rangle\langle 1|X \otimes U_0 = |1\rangle\langle 1| \otimes I + |0\rangle\langle 0| \otimes U_0 = A_0. \end{aligned}$$

Thus these gates are implemented with the set of universal gates. In fact, the order of $A_i$ does not matter since $[A_0, A_1] = 0$.



# Back on Grover's search algorithm

We need to prove that the D gate used to perform the quantum search can be implemented efficiently. We now show that

$$D = W_n R_0 W_n, \tag{7.6}$$

where $W_n$ is the Walsh-Hadamard transform,

$$W_n(x,y) = \frac{1}{\sqrt{N}}(-1)^{x\cdot y}, \quad (x, y \in S_n) \tag{7.7}$$

and $R_0$ is the selective phase rotation transform defined by

$$R_0(x,y) = e^{i\pi(1-\delta_{x0})}\delta_{xy} = (-1)^{1-\delta_{x0}}\delta_{xy}. \tag{7.8}$$

Proof

$$\langle x|D|y\rangle = \langle x|\left[-I + 2|\varphi_o\rangle\langle\varphi_0|\right]|y\rangle = -\delta_{xy} + \frac{2}{N} \qquad |\varphi_0\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$$

$$\begin{aligned}\langle x|W_n R_0 W_n|y\rangle = \sum_{u,v}\langle x|W_n|u\rangle\langle u|R_0|v\rangle\langle v|W_n|y\rangle &= \frac{1}{N}\sum_{u,v}(-1)^{x\cdot u}(-1)^{1-\delta_{u0}}\delta_{uv}(-1)^{v\cdot y}. \\ &= \frac{1}{N}\sum_{u}(-1)^{x\cdot u}(-1)^{y\cdot u}(-1)^{1-\delta_{u0}} \\ &= \frac{1}{N}\left[1 - \sum_{u\neq 0}(-1)^{x\cdot u}(-1)^{y\cdot u}\right] = A\end{aligned}$$

**x = y:** $\quad A = \frac{1}{N}\left[1 - \sum_{u\neq 0}\right] = \frac{1}{N}\left[1-(N-1)\right] = -1 + \frac{2}{N}$

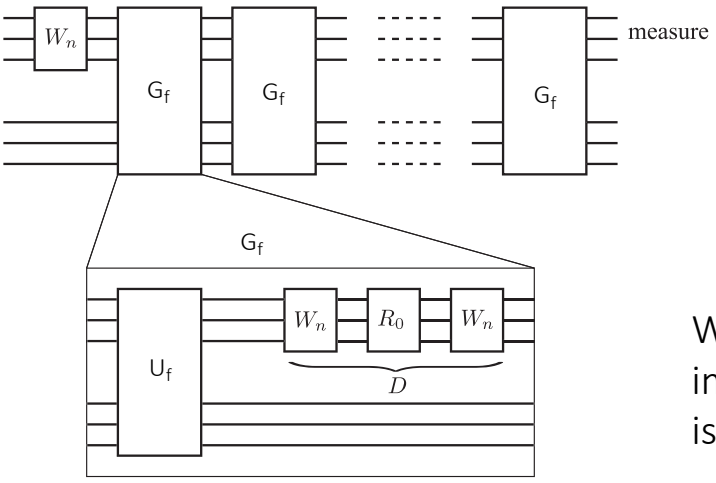
**x ≠ y**. As discussed in relation to the Deutsch-Jozsa algorithm

$$\sum_{u=0}^{N-1}(-1)^{x\cdot u} = 0 \;\rightarrow\; \sum_{u\neq 0}^{N-1}(-1)^{x\cdot u} = -1$$

Therefore: $A = \frac{1}{N}\left[1-(-1)\right] = \frac{2}{N}$

# Back on Grover's search algorithm

Therefore the D gate can be implemented efficiently. The overall circuit is




We are not interested on how to implement the oracle $U_f$ since this is supposed to be given

**Optimality of Grover's algorithm**. We have shown that a quantum computer can search N items, consulting the search oracle only O(√N) times. One can prove that no quantum algorithm can perform this task using fewer than
O(√N) accesses to the search oracle, and thus the algorithm we have demonstrated is optimal.

For a proof, see e.g. Nielsen – Chuang p. 269.

# The Quantum Fourier Transform

The **discrete Fourier transform** takes as input a vector of complex numbers, $x_0, \ldots, x_{N-1}$ where the length N of the vector is a fixed parameter. It outputs the transformed data, a vector of complex numbers $y_0, \ldots, y_{N-1}$, defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N}$$

In the **quantum Fourier transform**, we do a DFT on the amplitudes of a quantum state

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle$$

where the amplitudes $y_k$ are the discrete Fourier transform of the amplitudes $x_j$. On the **computational basis** it reads

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

It is not obvious from the definition, but this transformation is a **unitary transformation**, and thus can be implemented as the dynamics for a quantum computer.

Since the output amplitudes are a linear combination of the input amplitudes, it is a linear operator. Its form is easy to write down

$$\hat{F} = \sum_{j,k=0}^{N-1} \frac{e^{2\pi ijk/N}}{\sqrt{N}} |k\rangle\langle j|$$

It is easy to check that it is unitary

$$\begin{aligned} \hat{F}^\dagger \hat{F} &= \frac{1}{N} \sum_{j,k,j',k'} e^{2\pi i(j'k'-jk)/N} |j\rangle\langle j'| \delta_{kk'} \\ &= \frac{1}{N} \sum_{j,k,j'} e^{2\pi i(j'-j)k/N} |j\rangle\langle j'| \\ &= \sum_{j,j'} |j\rangle\langle j'| \delta_{jj'} = \sum_j |j\rangle\langle j| = \hat{I}. \end{aligned}$$

The Fourier transform lets us define a new basis: $|\tilde{x}\rangle = F\,|x\rangle$, where $\{|x\rangle\}$ is the usual computational basis. This basis has a number of **interesting properties**. Every vector $|\tilde{x}\rangle$ is an equally weighted superposition of all the computational basis states:

$$\begin{aligned} |\langle\tilde{x}|y\rangle|^2 &= \langle y|\tilde{x}\rangle\langle\tilde{x}|y\rangle = \langle y|\hat{F}|x\rangle\langle x|\hat{F}^\dagger|y\rangle \\ &= \frac{e^{2\pi ixy/N}}{\sqrt{N}} \frac{e^{-2\pi ixy/N}}{\sqrt{N}} = \frac{1}{N}. \end{aligned}$$

From the point of view of physics, the relationship of this basis to the computational basis is analogous to that between the momentum and position bases of a particle.

Recall that the Hadamard transform could also turn computational basis states into equally weighted superpositions of all states. But it left all amplitudes real, while the amplitudes of $|\tilde{x}\rangle$ are complex.

In **matrix representation**:

$$\mathbf{QFT} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \qquad \omega = e^{2\pi i/N}$$

In particular, for **1 qubit**:

$$\mathbf{QFT} = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

while, for **2 qubits**:

$$\mathbf{QFT} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

# Circuit for the Quantum Fourier Transform

At this point we specialize to the case of **n qubits**, so the dimension is $N = 2^n$.

We have seen that the quantum Fourier transform is a unitary operator. Therefore, by our earlier results, there is a quantum circuit which implements it. However, **there is no guarantee that this circuit will be efficient**! A general unitary requires a circuit with a number of gates exponential in the number of bits.

Very fortunately, in this case an efficient circuit does exist. (Fortunately, because the Fourier transform is at the heart of the most impressive quantum algorithms!)

**Binary expression: $j \to j_1 j_2 \ldots j_n$**

$$j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n$$

For example, for n = 2 we have $j = 2\, j_1 + j_2$, therefore

$$0 \to 00$$
$$1 \to 01$$
$$2 \to 10$$
$$3 \to 11$$

as usual. We also consider the **binary fraction**

$$0.j_1 j_2 \ldots j_n = j_1/2 + j_2/4 + \cdots + j_n/2^n = j/2^n$$

The key insight into designing a circuit for the Fourier transform is to notice that it can be written in a **product form**

$$|j_1,\ldots,j_n\rangle \rightarrow \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right)}{2^{n/2}}$$

The proof is the following

$$\begin{aligned}
|j\rangle &\rightarrow \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n}|k\rangle \\
&= \frac{1}{2^{n/2}}\sum_{k_1=0}^{1}\cdots\sum_{k_n=0}^{1} e^{2\pi i j\left(\sum_{l=1}^{n} k_l 2^{-l}\right)}|k_1\ldots k_n\rangle \\
&= \frac{1}{2^{n/2}}\sum_{k_1=0}^{1}\cdots\sum_{k_n=0}^{1}\bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}}|k_l\rangle \\
&= \frac{1}{2^{n/2}}\bigotimes_{l=1}^{n}\left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}}|k_l\rangle\right] \\
&= \frac{1}{2^{n/2}}\bigotimes_{l=1}^{n}\left[|0\rangle + e^{2\pi i j 2^{-l}}|1\rangle\right] \\
&= \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right)}{2^{n/2}}
\end{aligned}$$

In going from the third to the fourth line, we used the identity

$$\begin{aligned}
\sum_{k_1=0}^{1}\cdots\sum_{k_n=0}^{1}\bigotimes_{l=1}^{n} f_{k_l} &= \sum_{k_1=0}^{1}\cdots\sum_{k_n=0}^{1} f_{k_1} f_{k_2}\ldots f_{k_n} \\
&= \bigotimes_{l=1}^{n}\sum_{k_l=0}^{1} f_{k_l} = \sum_{k_1=0}^{1} f_{k_1}\sum_{k_2=0}^{1} f_{k_2}\cdots\sum_{k_n=0}^{1} f_{k_n}
\end{aligned}$$

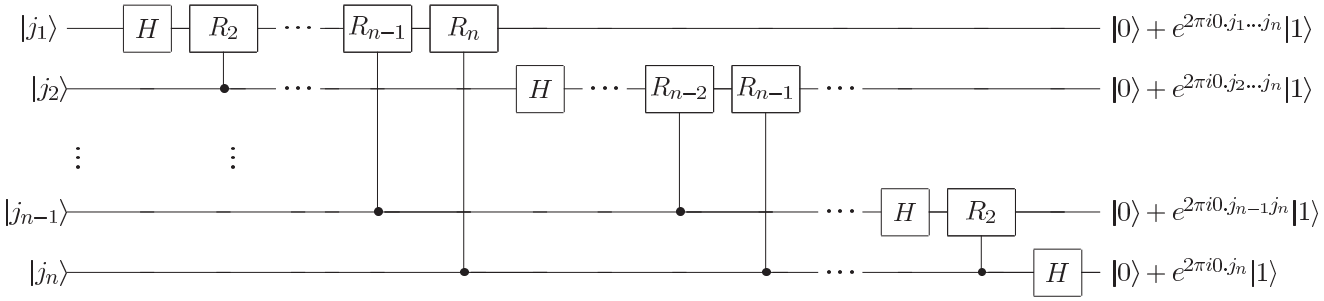
The unitary

$$|0,1\rangle \rightarrow (|0\rangle \pm \exp(i\theta)|1\rangle)/\sqrt{2}$$

is a Hadamard followed by a Z-rotation by θ. In the expression above the rotation depends on the values of the other bits. So **we should expect to be able to build the Fourier transform out of Hadamard and controlled-phase rotation gates.** Define the rotation

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

The circuit implementing the QFT then is




Let us see if it works. Applying the Hadamard gate to the first bit produces the state

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)|j_2 \ldots j_n\rangle$$

since $e^{2\pi i\, 0.j_1} = -1$ when $j_1 = 1$, and is +1 otherwise. Applying the controlled-$R_2$ gate produces the state

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right)|j_2 \ldots j_n\rangle$$

We continue applying the controlled-$R_3$, $R_4$ through $R_n$ gates, each of which adds an extra bit to the phase of the co-efficient of the first $|1\rangle$. At the end of this procedure we have the state

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right)|j_2 \ldots j_n\rangle$$

Next, we perform a similar procedure on the second qubit. The Hadamard gate puts us in the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right)|j_3 \ldots j_n\rangle$$

and the controlled-$R_2$ through $R_{n-1}$ gates yield the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle\right)|j_3 \ldots j_n\rangle$$

We continue in this fashion for each qubit, giving a final state

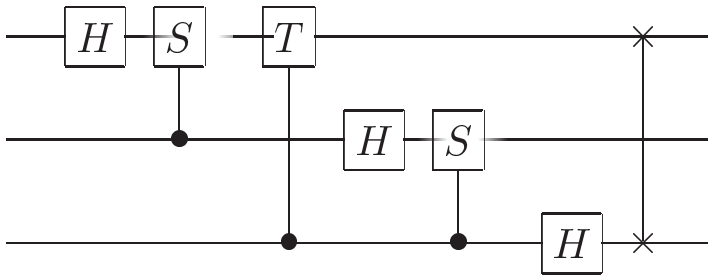
$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)$$

If necessary, swap operations omitted from the Figure for clarity, can be used to reverse the order of the qubits. After the swap operations, the state of the qubits is

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)$$

**Box 5.1: Three qubit quantum Fourier transform**

For concreteness it may help to look at the explicit circuit for the three qubit quantum Fourier transform:




Recall that $S$ and $T$ are the phase and $\pi/8$ gates (see page xxiii). As a matrix the quantum Fourier transform in this instance may be written out explicitly, using $\omega = e^{2\pi i/8} = \sqrt{i}$, as

$$\frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix}. \quad (5.19)$$

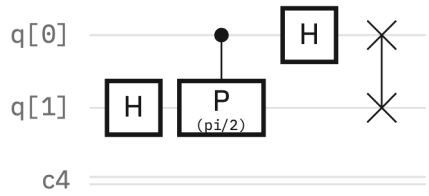
**How many gates does this circuit use?** We start by doing a Hadamard gate and n − 1 conditional rotations on the first qubit – a total of n gates. This is followed by a Hadamard gate and n − 2 conditional rotations on the second qubit, for a total of n + (n − 1) gates. Continuing in this way, we see that n+(n−1)+···+1 = n(n+1)/2 gates are required plus the gates involved in the swaps. At most n/2 swaps are required, and each swap can be accomplished using three CNOT gates. Therefore, **this circuit provides a $\Theta(n^2)$ algorithm for performing the quantum Fourier transform**.

In contrast, the best classical algorithms for computing the discrete Fourier transform on $2^n$ elements are algorithms such as the **Fast Fourier Transform (FFT),** which compute the discrete Fourier transform using **$\Theta(n2^n)$ gates**.

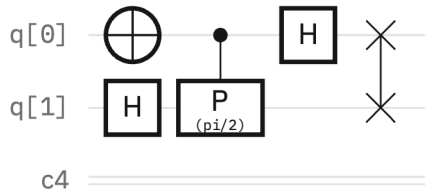
But remember, we cannot measure particular values of the amplitudes! The quantum Fourier Transform is useful only as a piece of another algorithm.

## Exercise on IBM Quantum Composer.

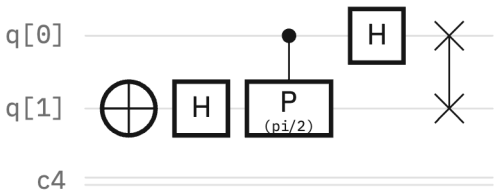
### N = 2



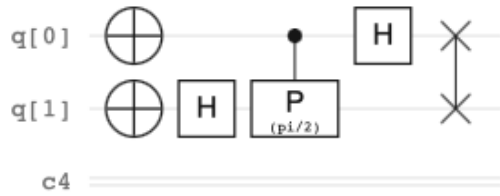

Opuput state
[ 0.5+0j, 0.5+0j, 0.5+0j, 0.5+0j ]




Opuput state
[ 0.5+0j, 0+0.5j,-0.5+0j, 0-0.5j ]



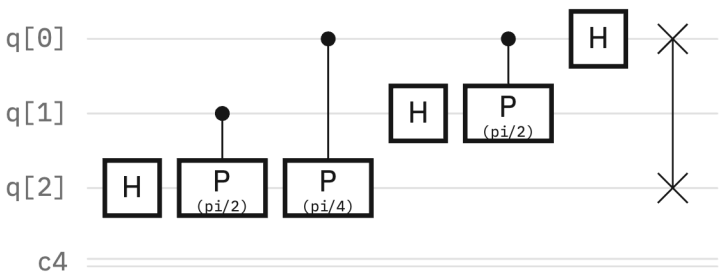

Opuput state
[ 0.5+0j,-0.5+0j, 0.5+0j,-0.5+0j ]




Opuput state
[ 0.5+0j, 0-0.5j, -0.5+0j, 0+0.5j ]

This corresponds to the matrix seen before

### N = 3

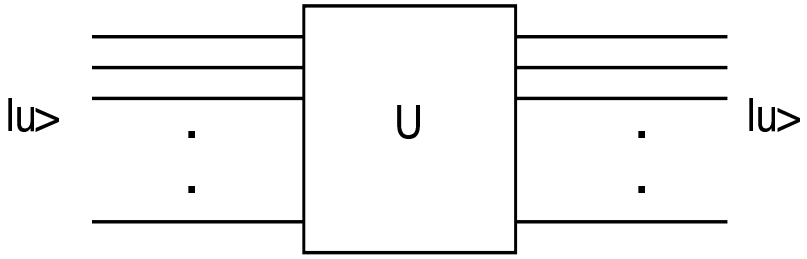


# Phase estimation algorithm

**The problem**: Suppose we are given a unitary operator U on n qubits, which has a known eigenstate $|u\rangle$ with an unknown eigenvalue $e^{2\pi i\phi}$, with $\phi$ in [0,1). We wish to find the phase $\phi$ with some precision.

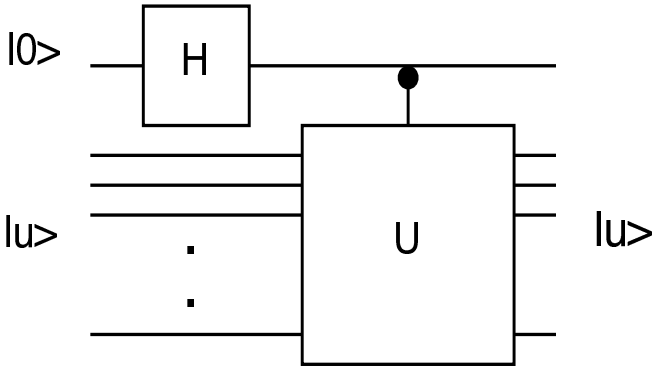
By itself, the phase estimation algorithm is a solution to a rather artificial problem. But this solution turns out to be useful as a piece of several other algorithms, to solve much more natural and important problems.

The first thing we might try is to prepare n q-bits in the state $|u\rangle$, and carry out the unitary transformation U on them:




Is there a measurement on the bits which will give us information about the phase $\phi$? The answer, of course, is no: $\hat{U}$ just produces an **overall phase** on the state, with no observable consequences.

We need to generate relative phases, which can be measured. This can be done in the following way.



The initial state changes as follows:

$$\begin{aligned} |0\rangle|u\rangle &\rightarrow \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]|u\rangle \\ &\rightarrow \frac{1}{\sqrt{2}}[|0\rangle|u\rangle + |1\rangle e^{2\pi i\varphi}|u\rangle] = \frac{1}{\sqrt{2}}[|0\rangle + e^{2\pi i\varphi}|1\rangle]|u\rangle \end{aligned}$$

Now we have a **relative phase** among the two qubits of the computational basis, which can be measured for example by first applying an Hadamard gate to the first qubit, whose state then becomes:

$$\frac{1 + e^{2\pi i\varphi}}{2}|0\rangle + \frac{1 - e^{2\pi i\varphi}}{2}|1\rangle$$

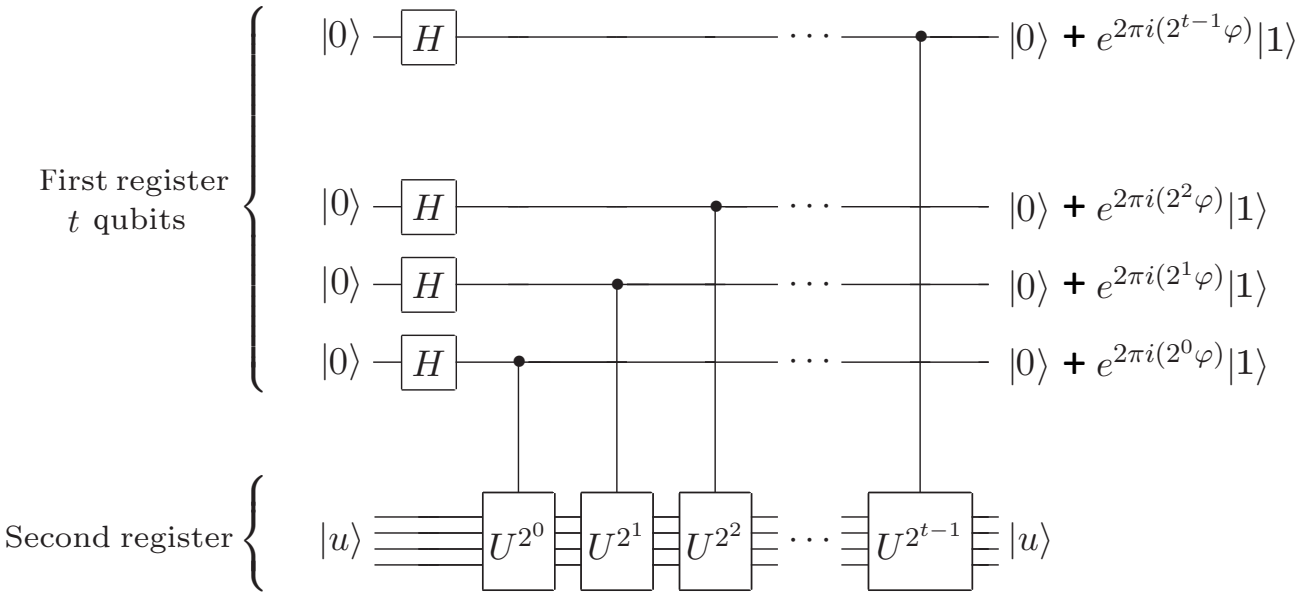
and by making a measurement on the computational basis we have the output probabilities:

$$\mathbb{P}[0] = \frac{1 + \cos 2\pi\varphi}{2}, \qquad \mathbb{P}[1] = \frac{1 - \cos 2\pi\varphi}{2}$$

We can run this circuit several times to recover the phase $\phi$. Unfortunately, the convergence of this algorithm is very slow. After N repetitions, the accuracy in the estimate of the phase $\phi$ is $N^{-1/2}$. If we wish to know $\phi$ with m bits accuracy, then $N^{1/2} = 2^m$, which means $N = 2^{2m}$: the number of repetitions grows exponentially with the number of bits of accuracy.

A smarter solution is provided by the following algorithm.

The **first stage** of the algorithm is:




(normalization factors $2^{-1/2}$ have been omintted)

How we choose t depends on two things: the number of digits of accuracy we wish to have in our estimate for ϕ, and with what probability we wish the phase estimation procedure to be successful.

The final state of the first register is easily seen to be

$$\frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{t-2}\varphi}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 2^{0}\varphi}|1\rangle\right)$$

$$= \frac{1}{2^{t/2}}\sum_{k_1=0}^{1}\cdots\sum_{k_t=0}^{1} e^{2\pi i\varphi(k_1 2^{t-1}+k_2 2^{t-2}+\ldots+k_t 2^0)}|k_1 k_2\ldots k_t\rangle = \frac{1}{2^{t/2}}\sum_{k=0}^{2^t-1} e^{2\pi i\varphi k}|k\rangle .$$

Suppose ϕ may be expressed exactly in t bits, as $\phi = 0.\phi_1 \ldots \phi_t$. Then

$$\begin{aligned} e^{2\pi i\phi} &= e^{2\pi i 0.\phi_1\ldots\phi_t}. \\ e^{4\pi i\phi} &= e^{2\pi i\phi_1.\phi_2\ldots\phi_t} = e^{2\pi i\phi_1 + 2\pi i 0.\phi_2\ldots\phi_t} = e^{2\pi i 0.\phi_2\ldots\phi_t} \\ e^{2^j\pi i\phi} &= e^{2\pi i 0.\phi_j\ldots\phi_t}. \end{aligned}$$

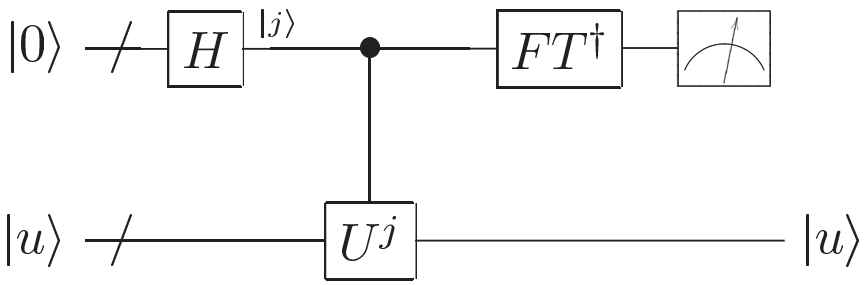
The output state can be rewritten as

$$\frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 0.\varphi_t}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.\varphi_{t-1}\varphi_t}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 0.\varphi_1\varphi_2\cdots\varphi_t}|1\rangle\right)$$

which is the Quantum Fourier Transform of the state $|\varphi_1 \ldots \varphi_t\rangle$

Therefore the **second stage** of the algorithm is to apply the inverse Quantum Fourier Transform to the output of the first state, and one recovers exactly the bits of the binary fraction for ϕ.

The full phase estimation algorithm is:




The above analysis applies to the ideal case, where ϕ can be written exactly with a t bit binary expansion. What happens when this is not the case?

Applying the inverse QFT to the state in the previous page produces the state

$$\frac{1}{2^t}\sum_{k,l=0}^{2^t-1} e^{\frac{-2\pi i k l}{2^t}} e^{2\pi i \varphi k}|l\rangle = \frac{1}{2^t}\sum_{k,l=0}^{2^t-1} e^{-\frac{2\pi i}{2^t}(l-2^t\varphi)k}|l\rangle$$

We see again that if $\phi = 0.\phi_1 \ldots \Phi_t$, then $2^t\,\phi$ is an integer and the sum over k returns a Kronecker delta, forcing it to be equal to $2^t\,\phi$. The final state is $|\phi_1 \cdots \phi_t\rangle = |2^t\phi\rangle$

If this is not the case, the coefficient associated to the state |l> is:

$$= \frac{1}{2^t} \sum_{k=0}^{2^t-1} e^{-\frac{2\pi i}{2^t}(l-2^t\varphi)k} = \frac{1}{2^t} \frac{1-e^{2\pi i(l-2^t\varphi)}}{1-e^{2\pi i(l-2^t\varphi)/2^t}}$$

And its square modulus is

$$\frac{1}{2^{2t}} \frac{1-\cos[2\pi(l-2^t\varphi)]}{1-\cos[2\pi(l-2^t\varphi)/2^t]}$$

The above function is sharply peaked around the closest l to $2^t$ ϕ. More precisely, it can be shown (see Nielsen & Chuang) that to successfully obtain ϕ accurate to n bits with probability of success at least 1 − ε, we choose

$$t = n + \left\lceil \log\left(2 + \frac{1}{2\epsilon}\right) \right\rceil$$

The number of qubits needed to run the algorithm with the desired accuracy grows linearly. Assuming that the controlled-$U^{2^j}$ unitaries are given by oracles (and hence free), the complexity of the algorithm is basically that of the Quantum Fourier Transform, $O(t^2)$. We have an exponential advantage with respect to the naïve algorithm we first tried.

However, if we have to perform circuits for the controlled-$U^{2^j}$ unitaries, than things change. Even if we have an efficient circuit for controlled-U, we need efficient circuits for all the controlled-$U^{2^j}$ gates as well; just repeating the controlled-U $2^j$ times will make the complexity exponential.

There is another somewhat artificial assumption as well. It is assumed that we don't know the eigenvalue $e^{2\pi i\phi}$, but that we can prepare the eigenvector |u⟩. While this may sometimes be true, in most cases it will not be.

The phase estimation algorithm then is:

**Inputs:** (1) A black box wich performs a controlled-$U^j$ operation, for integer $j$, (2) an eigenstate $|u\rangle$ of $U$ with eigenvalue $e^{2\pi i\varphi_u}$, and (3) $t = n + \lceil \log\left(2 + \frac{1}{2\epsilon}\right) \rceil$ qubits initialized to $|0\rangle$.

**Outputs:** An $n$-bit approximation $\widetilde{\varphi_u}$ to $\varphi_u$.

**Runtime:** $O(t^2)$ operations and one call to controlled-$U^j$ black box. Succeeds with probability at least $1 - \epsilon$.

**Procedure:**

| | | |
|---|---|---|
| **1.** | $\|0\rangle\|u\rangle$ | initial state |
| **2.** | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|u\rangle$ | create superposition |
| **3.** | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle U^j\|u\rangle$ | apply black box |
| | $= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j\varphi_u}\|j\rangle\|u\rangle$ | result of black box |
| **4.** | $\rightarrow \|\widetilde{\varphi_u}\rangle\|u\rangle$ | apply inverse Fourier transform |
| **5.** | $\rightarrow \widetilde{\varphi_u}$ | measure first register |

**Example 1**. Consider the unitary operator (X gate)

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

with eigenstate $|+\rangle = \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$ and we know the relative eigenvalue is a phase $\lambda = e^{2\pi i\theta}$.
The goal is to find the phase, with 1 bit precision.

1. The initial state is:

$$|0\rangle|+\rangle$$

2. We apply a Hadamard to the first qubit to create the superposition:

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

3. We apply the controlled-U gate once: the state remains unchanged

4. The inverse Fourier transform, which amounts to an Hadamard, brings the state back to >

$$|0\rangle|+\rangle$$

5. Measuring the first register gives 0, from which we learn that the phase, in binary fraction, is 0.0 (to 1 bit accuracy).

This is correct: we know that the eigenvalue is 1, therefore the phase is 0.

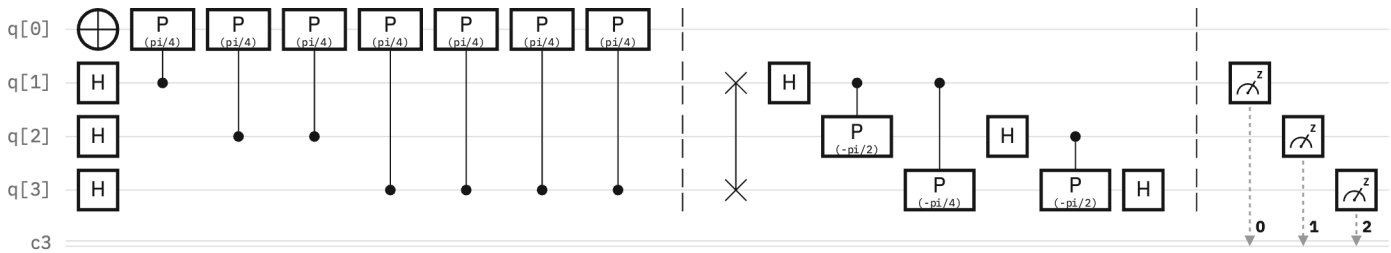
Exercise 1. Consider the unitary operator (T gate)

$$T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}$$

with eigenstate |1>. Write down the circuit to find the associated eigenvalue (which is assumed to be a phase) with 3 bit precision.

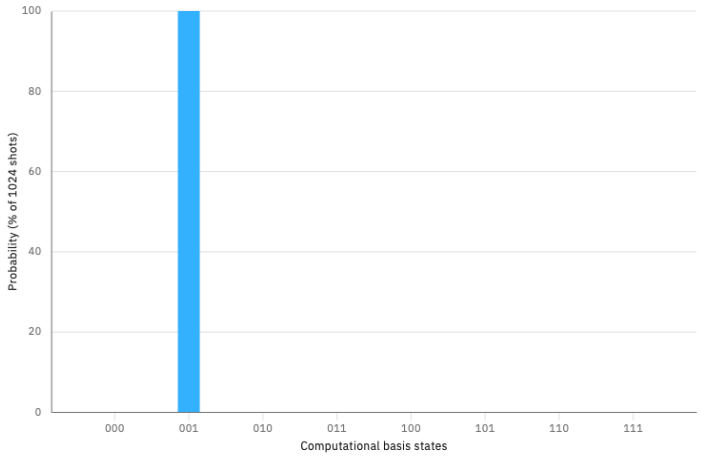
Solution: since the associated eigenvalue is $e^{i\pi/4}$, the desired phase is 1/8, which corresponds to the binary fraction 0.001. Therefore the algorithm returns 001, corresponding to the exact value of the phase.

On the IBM Quantum Composer, it looks as follows

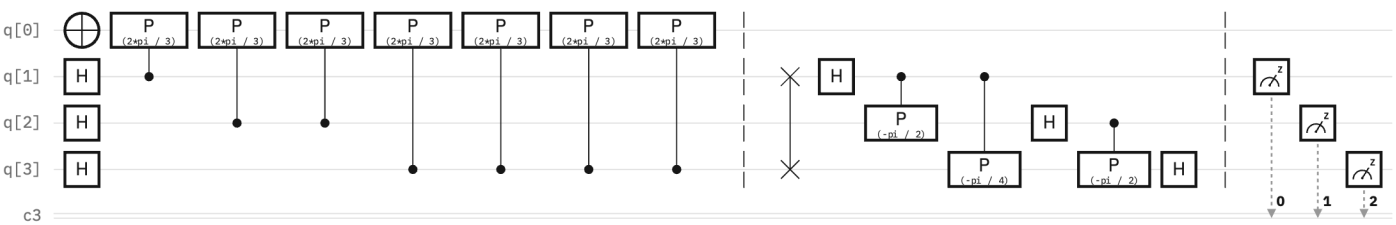



Remember that qubits are ordered from bottom to top.
(use “freeform alignment” to place the gate as desired)

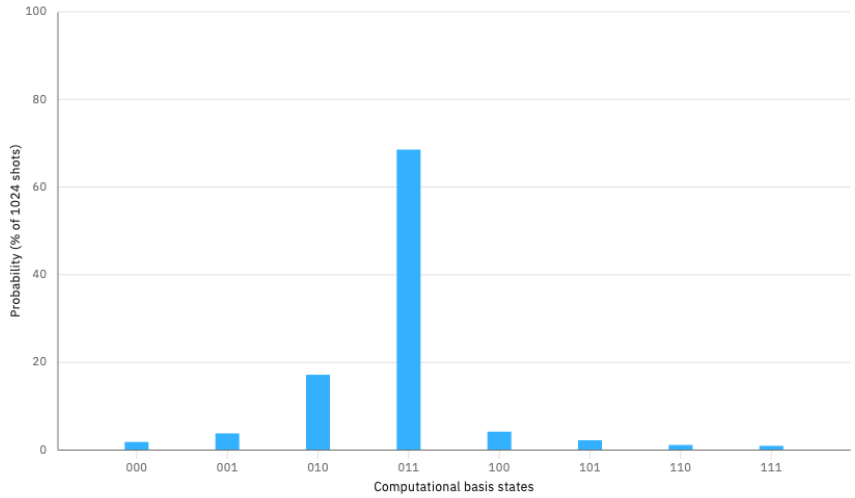
The **output probabilities** are




**Exercise 2**. Consider the same case as before, with a phase $\phi = 1/3 = 0{,}33333$ (not binary fraction). With 3-bit precision the algorithm is similar to the one before:



The output probabilities are

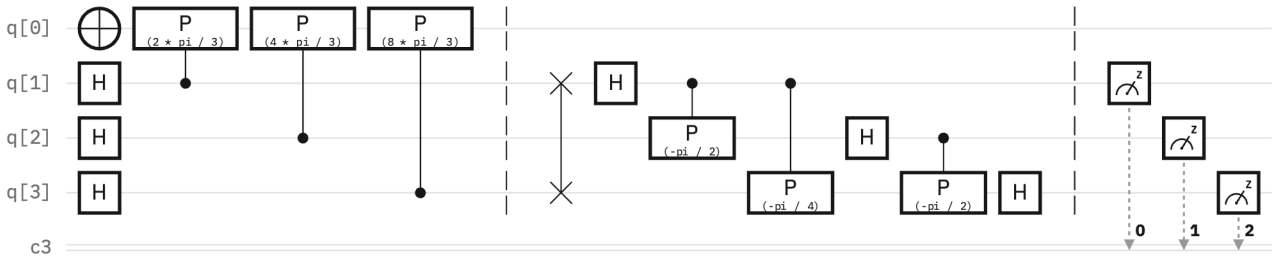



Most likely outcome: 011 → binary fraction 0.011 corresponding to a phase $\phi = 1/4 + 1/8 = 0{,}375$ → 0,042 difference from exact result (off by 13%).

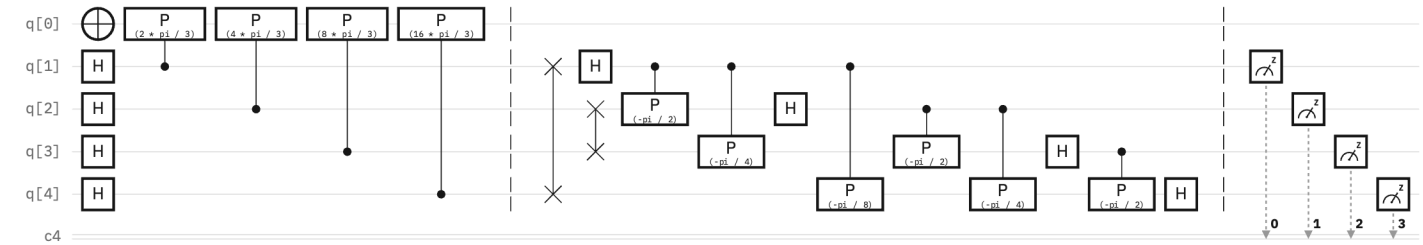
The second most likely outcome is: 010 → binary fraction 0.010 corresponding to a phase $\phi = 1/4 = 0{,}25$ → 0,083 difference from exact result (off by 25%).

The true phase lies in between, closer to the most likely outcome.

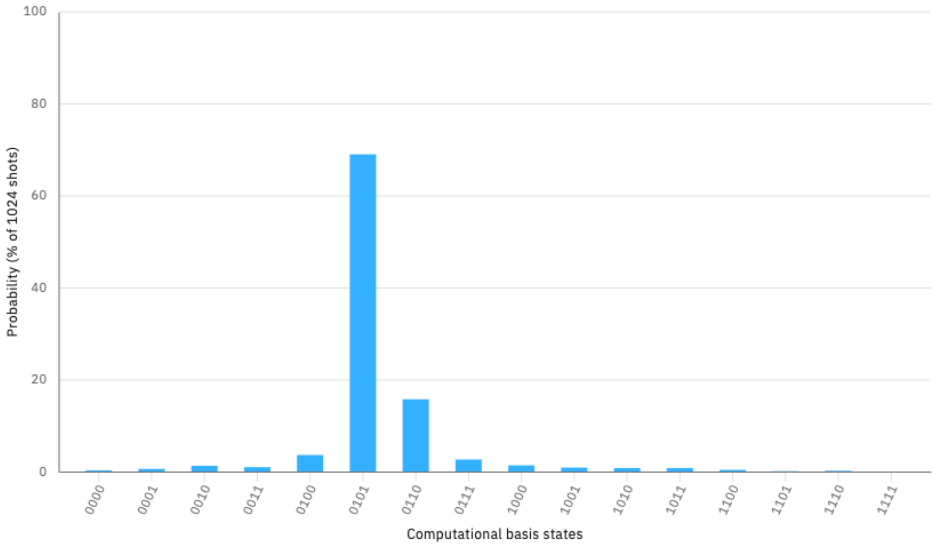
The circuit can be written more shortly as follows



**Exercise 3**. Same as before, but with 4-bit precision. The circuit is




The output probabilities are

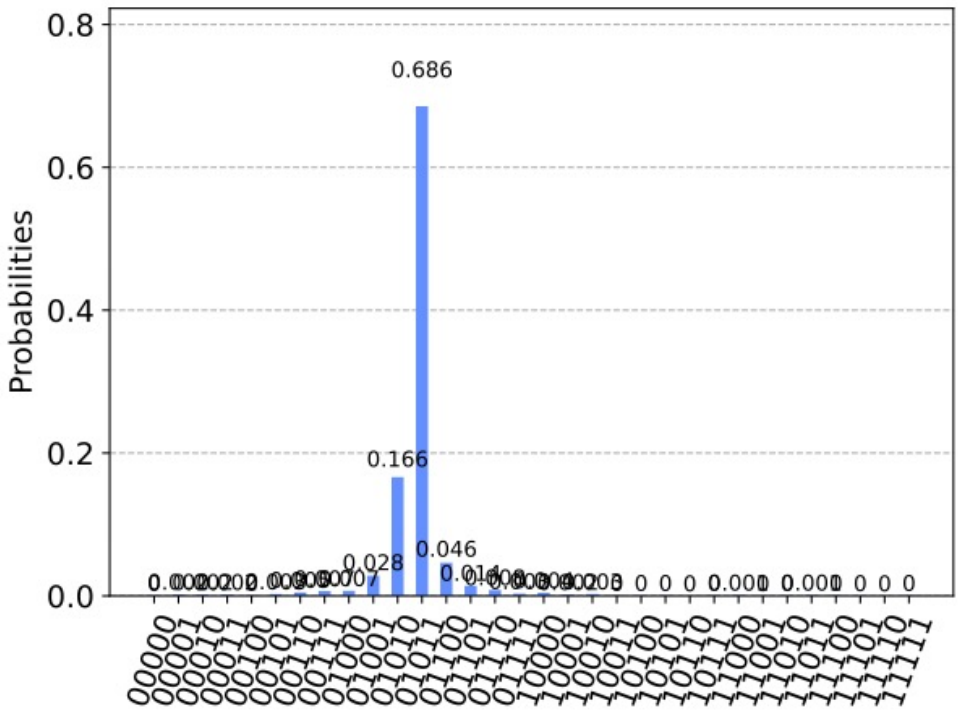



Most likely outcome: 0101 → binary fraction 0.0101 corresponding to a phase $\phi = 1/4 + 1/16 = 0{,}313$ → 0,02 difference from exact result (off by 6%).

The second most likely outcome is: 0110 → binary fraction 0.0110 corresponding to a phase $\phi = 1/4 + 1/8 = 0{,}375$ → 0,042 difference from exact result (off by 13%).

We see an improvement with respect to the case with 3 bits

**Exercise 4**. Same as before, but with 5-bit precision. The outcome probabilities are




Most likely outcome: 01011 → binary fraction 0.01011 corresponding to a phase $\phi$ = 1/4 + 1/16 + 1/32 = 0,344 → 0,011 difference from exact result (off by 3%).

The second most likely outcome is: 01010 → binary fraction 0.01010 corresponding to a phase $\phi$ = 1/4 + 1/16 = 0,313 → 0,02 difference from exact result (off by 6%).

Again, we see an improvement with respect to the previous cases.

# Order finding algorithm

**Definition of order:** For positive integers x and N , x < N , with no common factors, the order of x modulo N is defined to be the least positive integer, r, such that $x^r = 1 (\text{mod } N)$.

**Order finding problem: to determine the order for some specified x and N.**

Order-finding is believed to be a hard problem on a classical computer. The quantum algorithm for order-finding is just the phase estimation algorithm applied to the unitary operator

$$U|y\rangle \equiv |xy (\text{mod } N)\rangle$$

with y ∈ {0, 1, … $2^L$-1}, with L to be defined later. Note that here and below, when N ≤ y ≤ $2^L$ −1, we use the convention that xy(mod N) is just y again. That is, U only acts non-trivially when 0 ≤ y ≤ N − 1.

Example: N = 15, x = 7. Then:

U |0> = |0>
U |1> = |7>
U |2> = |14>
U |3> = |6>
U |4> = |13>
U |5> = |5>
U |6> = |12>
U |7> = |4>

U |8> = |11>
U |9> = |3>
U |10> = |10>
U |11> = |2>
U |12> = |9>
U |13> = |1>
U |14> = |8>
U |15> = |15>

U |16> = |16>
U |17> = |17>
and so on

A simple calculation shows that the states defined by

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^k \bmod N\rangle$$

for integer $0 \leq s \leq r-1$ are eigenstates of U, since

$$\begin{aligned} U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^{k+1} \bmod N\rangle \\ &= \exp\left[\frac{2\pi i s}{r}\right] |u_s\rangle \, . \end{aligned}$$

(this because $x^r \bmod N = 1$, by definition).

Using the phase estimation procedure allows us to obtain, with high accuracy, the corresponding eigenvalues $\exp^{2\pi i s/r}$, from which we can obtain the order r with a little bit more work.

There are three important requirements to be met in order for the algorithm to be efficient:

- We must have efficient procedures to implement a controlled-$U^{2^j}$ operation for any integer j.
- We must be able to efficiently prepare an eigenstate $|u_s\rangle$ with a non-trivial eigenvalue.
- We must be able to obtain the desired answer, r, from the result of the phase estimation algorithm, $\phi \approx s/r$.

We analyse the three elements separately.

**Implementation of the controlled-$U^{2j}$ operation: modular exponentiation.** The following relation holds:

$$\begin{aligned} |z\rangle|y\rangle &\rightarrow |z\rangle U^{z_t 2^{t-1}} \ldots U^{z_1 2^0}|y\rangle \\ &= |z\rangle|x^{z_t 2^{t-1}} \times \cdots \times x^{z_1 2^0} y (\mathrm{mod}\ N)\rangle \\ &= |z\rangle|x^z y (\mathrm{mod}\ N)\rangle. \end{aligned}$$

Thus the sequence of controlled-$U^{2j}$ operations used in phase estimation is equivalent to multiplying the contents of the second register by the modular exponential $x^z$(mod N), where z is the contents of the first register.

This operation may be accomplished classically using $O(L^3)$ gates. The classical circuit can be transformed into a reversible circuit, which can be translated into a quantum circuit of similar complexity, computing the transformation $|z\rangle|y\rangle \rightarrow |z\rangle|x^z y \ (\mathrm{mod}\ N)\rangle$. *The book of Nakahara & Ohimi (p. 156) explains in detail how to do it.*

**Prepare an eigenstate $|u_s\rangle$.** Preparing $|u_s\rangle$ requires that we know r, so this is out of the question. Fortunately, there is a clever observation which allows us to circumvent the problem of preparing $|u_s\rangle$, which is that

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

This means that if we prepare the second register in the state |1>, just before measurement the state of the two registers will be:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\varphi_s\rangle|u_s\rangle$$

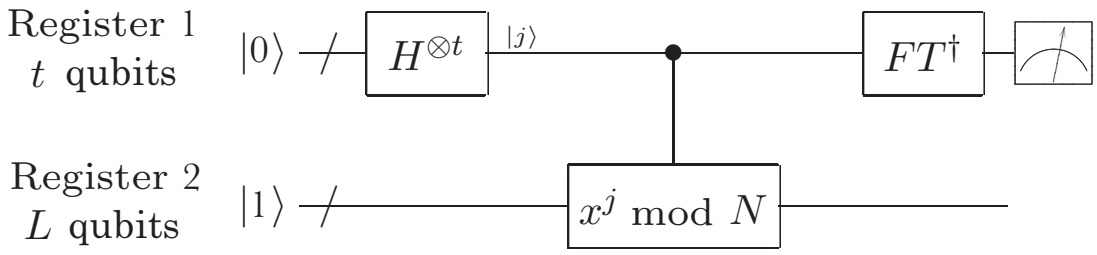
A measurement of the first register will collapse the state of the second register to the eigenstate $|u_s\rangle$, and the first register will end up in the state $|\phi_s\rangle$, from which the phase $\phi \approx s/r$ can be read.

Therefore, if we use

$$t = 2L + 1 + \left\lceil \log\left(2 + \frac{1}{2\epsilon}\right)\right\rceil$$

qubits in the first register and prepare the second register in the state $|1\rangle$, which is trivial to construct, it follows that for each s in the range 0 through r − 1, we will obtain an estimate of the phase $\phi \approx s/r$ accurate to 2L + 1 bits, with probability at least $(1 - \varepsilon)/r$.

The order finding algorithm the is:




**How to extract r from $\phi \approx s/r$.** We only know $\phi$ to 2L + 1 bits, but we also know a priori that it is a rational number – the ratio of two integers – and if we could compute the nearest such fraction to $\phi$ we might obtain r.

There is an algorithm which accomplishes this task efficiently, known as the continued fractions algorithm: given $\phi$ the continued fractions algorithm efficiently produces numbers s′ and r′ with no common factor, such that $s'/r' = s/r$. The number r′ is our candidate for the order. We can check to see whether it is the order by calculating $x^{r'} \bmod N$, and seeing if the result is 1. If so, then r′ is the order of x modulo N, and we are done.

**Performance.** How can the order-finding algorithm fail? There are two possibilities.

First, the phase estimation procedure might produce a bad estimate to s/r. This occurs with probability at most ε, and can be made small with a negligible increase in the size of the circuit.

More seriously, it might be that s and r have a common factor, in which case the number r′ returned by the continued fractions algorithm be a factor of r, and not r itself. Fortunately, there are at least three ways around this problem. Perhaps the most straightforward way is to note that for randomly chosen s in the range 0 through r − 1, it's actually pretty likely that s and r are co-prime, in which case the continued fractions algorithm must return r. Specifically, one can show that by repeating the algorithm 2log(N) times we will, with high probability, observe a phase s/r such that s and r are co-prime, and therefore the continued fractions algorithm produces r, as desired.

See Nielsen and Chuang for further details.

**Note.** The quantum state produced in the order-finding algorithm, before the inverse Fourier transform, is

$$|\psi\rangle = \sum_{j=0}^{2^t-1} |j\rangle U^j |1\rangle = \sum_{j=0}^{2^t-1} |j\rangle |x^j \bmod N\rangle$$

if we initialize the second register as $|1\rangle$. The same state is obtained if we replace $U^j$ with a different unitary transform V, which computes

$$V|j\rangle|k\rangle = |j\rangle|k + x^j \bmod N\rangle$$

and start the second register in the state $|0\rangle$. Moreover, V can be constructed also using $O(L^3)$ gates.

The order finding algorithm then is

**Inputs:** (1) A black box $U_{x,N}$ which performs the transformation $|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, for $x$ co-prime to the $L$-bit number $N$, (2) $t = 2L + 1 + \lceil \log\left(2 + \frac{1}{2\epsilon}\right) \rceil$ qubits initialized to $|0\rangle$, and (3) $L$ qubits initialized to the state $|1\rangle$.

**Outputs:** The least integer $r > 0$ such that $x^r = 1 \pmod N$.

**Runtime:** $O(L^3)$ operations. Succeeds with probability $O(1)$.

**Procedure:**

| | | |
|---|---|---|
| **1.** | $\|0\rangle\|1\rangle$ | initial state |
| **2.** | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|1\rangle$ | create superposition |
| **3.** | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|x^j \bmod N\rangle$ | apply $U_{x,N}$ |
| | $\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} \|j\rangle\|u_s\rangle$ | |
| **4.** | $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \|\widetilde{s/r}\rangle\|u_s\rangle$ | apply inverse Fourier transform to first register |
| **5.** | $\rightarrow \widetilde{s/r}$ | measure first register |
| **6.** | $\rightarrow r$ | apply continued fractions algorithm |

**Example:** Find the order r of x = 7 mod N = 15. We use L = 8 qubits. The initial state is:

$$|\mathbf{00...0}\rangle_8 \, |\mathbf{00...0}\rangle_8 = |0\rangle \, |0\rangle$$

Next, we apply the Hadamard transformation to the first register

$$\frac{1}{\sqrt{256}} \Big( |\mathbf{00...0}\rangle_8 + \ldots |\mathbf{11...1}\rangle_8 \Big) \, |\mathbf{00...0}\rangle_8$$

=0 =255

The next step is to perform the modular exponentiation. One gets

$$\frac{1}{\sqrt{256}}\Big( |0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle \ldots |255\rangle|13\rangle \Big)$$

which can be rewritten as

$$\begin{aligned} &= \frac{1}{\sqrt{4}}\left(\frac{|0\rangle + |4\rangle + \ldots + |252\rangle}{\sqrt{64}}\right)|1\rangle + \frac{1}{\sqrt{4}}\left(\frac{|1\rangle + |5\rangle + \ldots + |253\rangle}{\sqrt{64}}\right)|7\rangle \\ &+ \frac{1}{\sqrt{4}}\left(\frac{|2\rangle + |6\rangle + \ldots + |254\rangle}{\sqrt{64}}\right)|4\rangle + \frac{1}{\sqrt{4}}\left(\frac{|3\rangle + |7\rangle + \ldots + |255\rangle}{\sqrt{64}}\right)|13\rangle \end{aligned}$$

DA CONTINUARE SU QISKIT

# Factoring

The factoring problem turns out to be equivalent to the order-finding problem we just studied, in the sense that a fast algorithm for order-finding can easily be turned into a fast algorithm for factoring. The reduction of factoring to order-finding proceeds in two basic steps.

The **first step** is to show that we can compute a **factor of N** if we can find a non-trivial solution x neq ± 1(mod N) to the equation **$x^2 = 1 (\text{mod } N)$.**

The **second step** is to show that a **randomly chosen y co-prime to N** is quite likely to have an order r which is even, and such that $y^{r/2}$ neq ± 1(mod N). **Thus $x \equiv y^{r/2} (\text{mod } N)$ is a non-trivial solution to $x^2 = 1 (\text{mod } N)$.**

The algorithm runs as follows.

**Inputs:** A composite number $N$

**Outputs:** A non-trivial factor of $N$.

**Runtime:** $O((\log N)^3)$ operations. Succeeds with probability $O(1)$.

**Procedure:**

1. If $N$ is even, return the factor 2.
2. Determine whether $N = a^b$ for integers $a \geq 1$ and $b \geq 2$, and if so return the factor $a$ (uses the classical algorithm of Exercise 5.17).
3. Randomly choose $x$ in the range 1 to $N-1$. If $\gcd(x, N) > 1$ then return the factor $\gcd(x, N)$.
4. Use the order-finding subroutine to find the order $r$ of $x$ modulo $N$.
5. If $r$ is even and $x^{r/2} \neq -1 (\text{mod } N)$ then compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$, and test to see if one of these is a non-trivial factor, returning that factor if so. Otherwise, the algorithm fails.

**Steps 1 and 2** of the algorithm either return a factor, or else ensure that **N is an odd integer with more than one prime factor**. These steps may be performed using O(1) and O($L^3$) operations, respectively.

**Step 3** either returns a factor, or produces a **randomly chosen element x** of {0, 1, 2, . . . , N − 1}, **co-prime to N**.

**Step 4 calls the order-finding subroutine, computing the order r of x modulo N**.

**Step 5** completes the algorithm, since Theorem 5.3 of Nielsen & Chuang guarantees that with probability at least one-half, r will be even and $x^{r/2}$ neq − 1(mod N ), and then Theorem 5.2 of Nielsen & Chuang guarantees that either gcd($x^{r/2}$ − 1, N ) or gcd($x^{r/2}$ + 1,N) is a non-trivial factor of N.

**Example**: Factoring **N = 15**.

15 is neither even, nor of the form $a^b$ with a greater or equal to 1 and b greater or equal to 2. Therefore steps 1 and 2 do not return anything.

Step 3 requires to pick randomly a number between 1 and d 14. Following the previous example, suppose we choose **x = 7**. It is co-prime to 15.

Step 4 makes use of the order finding algorithm to find the order r of x = 7 mod N = 15. We saw that the output is **r = 4**.

By chance, 4 is even, and more over $x^{r/2}$ mod 15 = 4 differ from -1 mod 15, so the algorithm works. Computing the greatest common divisor gcd($x^2$ − 1, 15) = 3 and gcd($x^2$ + 1, 15) = 5 tells us that **15 = 3×5**.