Miriam Backens¹ (they/them)

m.backens@cs.bham.ac.uk

School of Computer Science, University of Birmingham

Quantum Hackathon, 13th November 2019

¹With thanks to Ashley Montanaro, whose slides parts of this talk are based on.
Outline

1. Introduction to quantum physics
2. What quantum computers are useful for
3. How to program a quantum computer
4. Building quantum computers
5. Conclusions
Outline

1. Introduction to quantum physics
2. What quantum computers are useful for
3. How to program a quantum computer
4. Building quantum computers
5. Conclusions
What is quantum physics?

The system of physical laws that govern very small things.

Pic: Wikipedia/Caffeine
What is quantum physics?

The system of physical laws that govern very small things.

- Developed in **early 20th century** (and ongoing).

Pic: Wikipedia/Caffeine
What is quantum physics?

The system of physical laws that govern very small things.

Developed in early 20th century (and ongoing).
Early applications include lasers, LEDs and transistors.
What is quantum physics?

The system of physical laws that govern very small things.

- Developed in **early 20th century** (and ongoing).
- Early applications include lasers, LEDs and transistors.
- There are many other **quantum phenomena whose technological exploitation is only beginning**.
Key properties of quantum mechanics

Very small things show behaviours that do not appear in the realm of everyday experience

1. **Superposition**: If a system can be in state A or state B, it can also be in a ‘mixture’ of the two.

2. **Measurements**: If we measure a system that is in a superposition of states A and B, we see either A or B probabilistically. Repeated measurements (without resetting) will yield the same result as the first measurement.

3. **Uncertainty**: There are pairs of measurements where greater certainty of the outcome of one measurement implies greater uncertainty of the outcome of the other measurement.

4. **Entanglement**: There exist states of multipartite systems which cannot be described in terms of states of the constituent systems.
Very small things show behaviours that do not appear in the realm of everyday experience, such as:

1. **Superposition**: If a system can be in state A or state B, it can also be in a ‘mixture’ of the two.
Key properties of quantum mechanics

Very small things show **behaviours that do not appear in the realm of everyday experience**, such as:

1. **Superposition**: If a system can be in state A or state B, it can also be in a ‘*mixture*’ of the two.

2. **Measurements**: If we measure a system that is in a superposition of states A and B, we see either A or B **probabilistically**. Repeated measurements (without resetting) will yield **the same result as the first measurement**.
Key properties of quantum mechanics

Very small things show behaviours that do not appear in the realm of everyday experience, such as:

1. **Superposition**: If a system can be in state A or state B, it can also be in a ‘mixture’ of the two.

2. **Measurements**: If we measure a system that is in a superposition of states A and B, we see either A or B probabilistically. Repeated measurements (without resetting) will yield the same result as the first measurement.

3. **Uncertainty**: There are pairs of measurements where greater certainty of the outcome of one measurement implies greater uncertainty of the outcome of the other measurement.
Key properties of quantum mechanics

Very small things show behaviours that do not appear in the realm of everyday experience, such as:

1. **Superposition**: If a system can be in state A or state B, it can also be in a ‘mixture’ of the two.

2. **Measurements**: If we measure a system that is in a superposition of states A and B, we see either A or B probabilistically. Repeated measurements (without resetting) will yield the same result as the first measurement.

3. **Uncertainty**: There are pairs of measurements where greater certainty of the outcome of one measurement implies greater uncertainty of the outcome of the other measurement.

4. **Entanglement**: There exist states of multipartite systems which cannot be described in terms of states of the constituent systems.
Superposition and measurement: Schrödinger’s cat

Pic: Wikipedia/Schrodinger's_cat
Uncertainty (e.g. of position and momentum)

‘Do you know how fast you were going?’

‘No, but I know where I am.’

‘You were doing 90 miles an hour.’

‘Great, now I’m lost.’
A quantum system with two distinct states is a qubit.

For example, a photon – a particle of light – has a property called polarisation which can be vertical or horizontal ($\uparrow$ or $\rightarrow$):

![Diagram of qubit states](image-url)
A quantum system with two distinct states is a **qubit**.

For example, a photon – a particle of light – has a property called polarisation which can be *vertical* or *horizontal* (↑ or →):

![Diagram of qubit states](image)

Just as a classical computer operates on bits, a quantum computer operates on qubits.
Imagine we have a pair of entangled qubits:

Entanglement

Imagine we have a pair of entangled qubits:

Even if we move one of the qubits to the Moon, the global state of the two qubits cannot be described solely in terms of the individual state of each of them!

In particular, if we measure one of the qubits, this apparently instantaneously affects the other one.
Outline

1. Introduction to quantum physics
2. What quantum computers are useful for
3. How to program a quantum computer
4. Building quantum computers
5. Conclusions
Quantum simulation

There is no efficient general-purpose method known to simulate quantum physics on a standard computer.
Quantum simulation

There is no efficient general-purpose method known to simulate quantum physics on a standard computer.

1982: Nobel Laureate Richard Feynman asks whether quantum physics could be simulated efficiently using a quantum computer.

Pic: WP/Richard Feynman

Simulating quantum physics has applications to drug design, materials science, high-energy physics, ...

Pic: WP/Seth Lloyd

Miriam Backens
Quantum Computing 101
There is no efficient general-purpose method known to simulate quantum physics on a standard computer.

1982: Nobel Laureate Richard Feynman asks whether quantum physics could be simulated efficiently using a quantum computer. 1996: Seth Lloyd proposes a quantum algorithm which can simulate quantum-mechanical systems.
Quantum simulation

There is no efficient general-purpose method known to simulate quantum physics on a standard computer.

1982: Nobel Laureate Richard Feynman asks whether quantum physics could be simulated efficiently using a quantum computer. 1996: Seth Lloyd proposes a quantum algorithm which can simulate quantum-mechanical systems.

Simulating quantum physics has applications to drug design, materials science, high-energy physics, ...
Shor’s algorithm for factoring

1994: Peter Shor shows that quantum computers can factorise large integers efficiently.

Given an integer $N = p \times q$ for prime numbers $p$ and $q$, Shor’s algorithm outputs $p$ and $q$.

No efficient classical algorithm for this task is known.

Pic: WP/Peter Shor
1994: Peter Shor shows that quantum computers can factorise large integers efficiently.

Given an integer $N = p \times q$ for prime numbers $p$ and $q$, Shor’s algorithm outputs $p$ and $q$.

No efficient classical algorithm for this task is known.

The quantum part of the algorithm uses period-finding: given a function $f : \mathbb{Z} \rightarrow \mathbb{Z}$ and the promise that there exists a number $a$ such that $f(x + a) = f(x)$ for all $x$, find $a$. 

Pic: WP/Peter Shor
1994: Peter Shor shows that quantum computers can factorise large integers efficiently.

Given an integer $N = p \times q$ for prime numbers $p$ and $q$, Shor’s algorithm outputs $p$ and $q$.

No efficient classical algorithm for this task is known.

The quantum part of the algorithm uses period-finding: given a function $f : \mathbb{Z} \rightarrow \mathbb{Z}$ and the promise that there exists a number $a$ such that $f(x + a) = f(x)$ for all $x$, find $a$.

Shor’s algorithm breaks the RSA public-key cryptosystem on which Internet security is based.
Grover’s algorithm for unstructured search

**Unstructured search** is one of the most basic problems in computer science:

- Imagine we have \( n \) boxes, each containing a 0 or a 1. We can look inside a box at a cost of one *query*.

  ![Box with numbers](image)

  0 0 1 0 0 0 1 0

- We want to find a box containing a 1. On a classical computer, this task could require \( n \) queries in the worst case.
Grover’s algorithm for unstructured search

Unstructured search is one of the most basic problems in computer science:

- Imagine we have \( n \) boxes, each containing a 0 or a 1. We can look inside a box at a cost of one query.

\[
\begin{array}{cccccccc}
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

- We want to find a box containing a 1. On a classical computer, this task could require \( n \) queries in the worst case.

1996: Lov Grover gives a quantum algorithm which solves this problem using about \( \sqrt{n} \) queries.

Pic: www.dcs.warwick.ac.uk/~tim/quantumcomputing/when/slide5.html
Unstructured search is one of the most basic problems in computer science:

- Imagine we have $n$ boxes, each containing a 0 or a 1. We can look inside a box at a cost of one query.

  0 0 1 0 0 0 1 0

- We want to find a box containing a 1. On a classical computer, this task could require $n$ queries in the worst case.

1996: Lov Grover gives a quantum algorithm which solves this problem using about $\sqrt{n}$ queries.

The square-root speedup of Grover’s algorithm finds many applications to search and optimisation problems, including in quantum machine learning.

Pic: www.dcs.warwick.ac.uk/~tim/quantumcomputing/when/slide5.html
The HHL algorithm for systems of linear equations

Solving a system of linear equations: Given a $N \times N$ matrix $A$ and a unit vector $b$, find the vector $x$ satisfying $Ax = b$. 

Aram Harrow, Avinatan Hassidim, Seth Lloyd (2008): Given $A$ and $b$, make a measurement on the quantum state described by the vector $x$ satisfying $Ax = b$. 

We don't get the solution $x$ itself. The matrix $A$ needs to be sparse. Running time is $O(\log(N)\kappa^2)$ vs $O(N\kappa)$ on a standard computer, where $\kappa$ is the 'condition number' of $A$ (roughly, the absolute value of the ratio between the biggest and smallest eigenvalue). Applications in science, engineering, machine learning and big data.
The HHL algorithm for systems of linear equations

Solving a system of linear equations: Given a $N \times N$ matrix $A$ and a unit vector $b$, find the vector $x$ satisfying $Ax = b$.

Aram Harrow, Avinatan Hassidim, Seth Lloyd (2008): Given $A$ and $b$, make a measurement on the quantum state described by the vector $x$ satisfying $Ax = b$.
The HHL algorithm for systems of linear equations

Solving a system of linear equations: Given a $N \times N$ matrix $A$ and a unit vector $b$, find the vector $x$ satisfying $Ax = b$.

Aram Harrow, Avinatan Hassidim, Seth Lloyd (2008): Given $A$ and $b$, make a measurement on the quantum state described by the vector $x$ satisfying $Ax = b$.

- We don’t get the solution $x$ itself.
The HHL algorithm for systems of linear equations

Solving a system of linear equations: Given a $N \times N$ matrix $A$ and a unit vector $b$, find the vector $x$ satisfying $Ax = b$.

Aram Harrow, Avinatan Hassidim, Seth Lloyd (2008): Given $A$ and $b$, make a measurement on the quantum state described by the vector $x$ satisfying $Ax = b$.

- We don’t get the solution $x$ itself.
- The matrix $A$ needs to be sparse.
The HHL algorithm for systems of linear equations

Solving a system of linear equations: Given a $N \times N$ matrix $A$ and a unit vector $b$, find the vector $x$ satisfying $Ax = b$.

Aram Harrow, Avinatan Hassidim, Seth Lloyd (2008): Given $A$ and $b$, make a measurement on the quantum state described by the vector $x$ satisfying $Ax = b$.

- We don’t get the solution $x$ itself.
- The matrix $A$ needs to be sparse.
- Running time is $O(\log(N)\kappa^2)$ vs $O(N\kappa)$ on a standard computer, where $\kappa$ is the ‘condition number’ of $A$ (roughly, the absolute value of the ratio between the biggest and smallest eigenvalue).
The HHL algorithm for systems of linear equations

Solving a system of linear equations: Given a $N \times N$ matrix $A$ and a unit vector $b$, find the vector $x$ satisfying $Ax = b$.

Aram Harrow, Avinatan Hassidim, Seth Lloyd (2008): Given $A$ and $b$, make a measurement on the quantum state described by the vector $x$ satisfying $Ax = b$.

- We don’t get the solution $x$ itself.
- The matrix $A$ needs to be sparse.
- Running time is $O(\log(N)\kappa^2)$ vs $O(N\kappa)$ on a standard computer, where $\kappa$ is the ‘condition number’ of $A$ (roughly, the absolute value of the ratio between the biggest and smallest eigenvalue).
- Applications in science, engineering, machine learning and big data.
Secure quantum computing in the cloud

Anne Broadbent, Joseph Fitzsimons and Elham Kashefi (2009) introduce the ‘blind quantum computing’ protocol.

Pic: mysite.science.uottawa.ca/abroadbe/
Pic: jfitzsimons.org/
Pic: www.cs.ox.ac.uk/people/elham.kashefi/
Secure quantum computing in the cloud

Anne Broadbent, Joseph Fitzsimons and Elham Kashefi (2009) introduce the ‘blind quantum computing’ protocol.

The protocol allows the secure delegation of quantum computations to a quantum server. The client does not need to perform any quantum computation (only certain state preparations and measurements).
Secure quantum computing in the cloud

Anne Broadbent, Joseph Fitzsimons and Elham Kashefi (2009) introduce the ‘blind quantum computing’ protocol.

The protocol allows the secure delegation of quantum computations to a quantum server. The client does not need to perform any quantum computation (only certain state preparations and measurements).

The server learns nothing about the data or the type of computation.
Outline

1. Introduction to quantum physics
2. What quantum computers are useful for
3. How to program a quantum computer
4. Building quantum computers
5. Conclusions
### What quantum computations consist of

<table>
<thead>
<tr>
<th>Measurements</th>
<th>Unitary operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>probabilistic</td>
<td>deterministic</td>
</tr>
<tr>
<td>irreversible</td>
<td>reversible</td>
</tr>
<tr>
<td>lose ‘quantumness’</td>
<td>keep ‘quantumness’</td>
</tr>
</tbody>
</table>
What quantum computations consist of

<table>
<thead>
<tr>
<th>Measurements</th>
<th>Unitary operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>probabilistic</td>
<td>deterministic</td>
</tr>
<tr>
<td>irreversible</td>
<td>reversible</td>
</tr>
<tr>
<td>lose ‘quantumness’</td>
<td>keep ‘quantumness’</td>
</tr>
</tbody>
</table>

Unitary operations usually make up the **bulk of a quantum computation**.
What quantum computations consist of

Measurements
- probabilistic
- irreversible
- lose ‘quantumness’

Unitary operations
- deterministic
- reversible
- keep ‘quantumness’

Unitary operations usually make up the bulk of a quantum computation. They are written down as quantum circuits:

```
  H   H
  |   |   |   |
  +---+---+---+
  X   Z
  |   |   |   |
  +---+---+---+
```

Each horizontal wire represents a qubit, each gate represents an operation on one or more qubits.
What quantum computations consist of

Measurements
- probabilistic
- irreversible
- lose ‘quantumness’

Unitary operations
- deterministic
- reversible
- keep ‘quantumness’

Unitary operations usually make up the bulk of a quantum computation. They are written down as quantum circuits:

Each horizontal wire represents a qubit, each gate represents an operation on one or more qubits.
Qubit states as vectors

A qubit state is described by a unit vector \((a, b)\) where \(a\) and \(b\) are complex numbers satisfying:

\[|a|^2 + |b|^2 = 1\]
A qubit state is described by a unit vector \((a b)\) where \(a\) and \(b\) are complex numbers satisfying:

\[
|a|^2 + |b|^2 = 1
\]

Given a single qubit, we can’t learn the values of \(a\) and \(b\): a measurement will give the outcome ‘0’ with probability \(|a|^2\) and the outcome ‘1’ with probability \(|b|^2\).
A qubit state is described by a unit vector \((a b)\) where \(a\) and \(b\) are complex numbers satisfying:

\[ |a|^2 + |b|^2 = 1 \]

Given a single qubit, we can’t learn the values of \(a\) and \(b\): a measurement will give the outcome ‘0’ with probability \(|a|^2\) and the outcome ‘1’ with probability \(|b|^2\).

So the vector \(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\) corresponds to the bit value 0 and the vector \(\begin{pmatrix} 0 \\ 1 \end{pmatrix}\) corresponds to the bit value 1, each with certainty.
**Qubit states as vectors**

A qubit state is described by a unit vector \( \begin{pmatrix} a \\ b \end{pmatrix} \) where \( a \) and \( b \) are complex numbers satisfying:

\[
|a|^2 + |b|^2 = 1
\]

Given a single qubit, we can’t learn the values of \( a \) and \( b \): a measurement will give the outcome ‘0’ with probability \( |a|^2 \) and the outcome ‘1’ with probability \( |b|^2 \).

So the vector \( \begin{pmatrix} 1 \\ 0 \end{pmatrix} \) corresponds to the bit value 0 and the vector \( \begin{pmatrix} 0 \\ 1 \end{pmatrix} \) corresponds to the bit value 1, each with certainty.

Complex numbers matter: \( \frac{1}{\sqrt{5}} \begin{pmatrix} 1 \\ 2 \end{pmatrix} \) and \( \frac{1}{\sqrt{5}} \begin{pmatrix} -1 \\ 2 \end{pmatrix} \) give the same probabilities but they are different states.
A state of two qubits is described by a vector of length 4, whose components determine the probabilities of finding the two qubits in the states 00, 01, 10, and 11, respectively.

For example, \((1, 0, 0, 0)\) means both qubits are 0 and \((\frac{3}{5}, 0, 0, \frac{4}{5})\) means either both qubits are 0 or both are 1 (this state is entangled).

A state of \(n\) qubits is described by a vector of length \(2^n\) whose components determine the probabilities for all the different \(n\)-bit strings.

For example, the three-qubit state \((0, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, 0, \frac{1}{\sqrt{3}}, 0, 0)\) has equal probabilities of giving the bit strings 001, 010, or 100 when all qubits are measured.
States of multiple qubits

A state of two qubits is described by a vector of length 4, whose components determine the probabilities of finding the two qubits in the states 00, 01, 10, and 11, respectively.

For example, \((1, 0, 0, 0)\) means both qubits are 0 and \((\frac{3}{5}, 0, 0, \frac{4i}{5})\) means either both qubits are 0 or both are 1 (this state is entangled).
A state of two qubits is described by a vector of length 4, whose components determine the probabilities of finding the two qubits in the states 00, 01, 10, and 11, respectively.

For example, \((1, 0, 0, 0)\) means both qubits are 0 and \((\frac{3}{5}, 0, 0, \frac{4i}{5})\) means either both qubits are 0 or both are 1 (this state is entangled).

A state of \(n\) qubits is described by a vector of length \(2^n\) whose components determine the probabilities for all the different \(n\)-bit strings.
States of multiple qubits

A state of two qubits is described by a vector of length 4, whose components determine the probabilities of finding the two qubits in the states 00, 01, 10, and 11, respectively.

For example, \((1, 0, 0, 0)\) means both qubits are 0 and \((\frac{3}{5}, 0, 0, \frac{4i}{5})\) means either both qubits are 0 or both are 1 (this state is entangled).

A state of \(n\) qubits is described by a vector of length \(2^n\) whose components determine the probabilities for all the different \(n\)-bit strings.

For example, the three-qubit state \((0, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, 0, \frac{1}{\sqrt{3}}, 0, 0, 0)\) has equal probabilities of giving the bit strings 001, 010, or 100 when all qubits are measured.
Reversible logic gates as unitary operations

The **NOT gate** \( \overline{X} \) corresponds to the matrix \( \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \):

\[
\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}
\]

i.e. \( \{0 \mapsto 1, 1 \mapsto 0\} \)

This is a reversible version of \( \text{XOR} \), acting on bits as \( (x, y) \mapsto (x, y \oplus x) \).
Reversible logic gates as unitary operations

The NOT gate \( \overline{X} \) corresponds to the matrix \(
\begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
\): 

\[
\begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}
\]

i.e. \[
\begin{array}{ccc}
0 & \mapsto & 1 \\
1 & \mapsto & 0
\end{array}
\]

The controlled-NOT gate \( \overline{\otimes} \) corresponds to \[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}
\]:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ d \\ c \end{pmatrix}
\]

i.e. \[
\begin{array}{ccc}
00 & \mapsto & 00 \\
01 & \mapsto & 01 \\
10 & \mapsto & 11 \\
11 & \mapsto & 10
\end{array}
\]
Reversible logic gates as unitary operations

The **NOT gate** \( \chi \) corresponds to the matrix \( \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \):

\[
\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}
\]

* i.e. \( \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 0 \end{cases} \)

The **controlled-NOT gate** corresponds to the matrix:

\[
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}
\]

\[
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ d \\ c \end{pmatrix}
\]

* i.e. \( \begin{cases} 00 \mapsto 00 \\ 01 \mapsto 01 \\ 10 \mapsto 11 \\ 11 \mapsto 10 \end{cases} \)

This is a reversible version of XOR, acting on bits as \( (x, y) \mapsto (x, y \oplus x) \)
The Pauli-Z gate corresponds to the matrix $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$:

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ -b \end{pmatrix}$$
Quantum gates with no classical counterpart

The Pauli-Z gate \( Z \) corresponds to the matrix \( \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \):

\[
\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ -b \end{pmatrix}
\]

The Hadamard gate \( H \) corresponds to \( \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \):

\[
\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} a + b \\ a - b \end{pmatrix}
\]
Quantum gates with no classical counterpart

The Pauli-Z gate $\begin{bmatrix} Z \end{bmatrix}$ corresponds to the matrix $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$:

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

The Hadamard gate $\begin{bmatrix} H \end{bmatrix}$ corresponds to $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} a + b \\ a - b \end{pmatrix}$$

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ -b \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} a - b \\ a + b \end{pmatrix}$$
Combining gates into circuits

Connect gates by (arbitrarily long) wires:
Combining gates into circuits

Connect gates by (arbitrarily long) wires:

Besides the gates introduced on the previous slides, there are many other gates that are commonly used in quantum circuits in different combinations.
Translating circuits to matrices

Two gates on the same wire correspond to the matrix product:

\[
\begin{pmatrix}
Z & H
\end{pmatrix}
\text{ is }
\frac{1}{\sqrt{2}} \begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}
= \frac{1}{\sqrt{2}} \begin{pmatrix}
1 & -1 \\
1 & 1
\end{pmatrix}
\]

- Careful about the reversed order!
Translating circuits to matrices

Two gates on the same wire correspond to the **matrix product**:

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

- Careful about the **reversed order**!

Two gates on parallel wires correspond to the **Kronecker product** (also called tensor product):

\[
\begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\
0 & -1
\end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\
0 & -1 & 0 & -1 \\
1 & 0 & -1 & 0 \\
0 & -1 & 0 & 1
\end{pmatrix}
\]

- This is **not commutative**.
The basic gates, $H$, and $R_{Z,\theta}$, corresponding to the matrices
\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix},
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix},
\frac{1}{\sqrt{2}}\begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix},
\begin{pmatrix}
1 & 0 \\
0 & e^{i\theta}
\end{pmatrix},
\]
are enough to write down a circuit for any unitary operation on a quantum computer.

Here, $\theta$ is an arbitrary real number, making $e^{i\theta}$ a complex number of absolute value 1.
Outline

1. Introduction to quantum physics
2. What quantum computers are useful for
3. How to program a quantum computer
4. Building quantum computers
5. Conclusions
Some approaches to quantum computing

Photonics, Bristol

Superconducting electronics, UCSB

Ion trap, Oxford
Quantum error correction

Building a large-scale quantum computer is extremely challenging because of decoherence.

If a quantum computer interacts with the outside world and is subject to noise, it can lose its ‘quantumness’ and behave like a classical computer.
Building a large-scale quantum computer is extremely challenging because of decoherence.

If a quantum computer interacts with the outside world and is subject to noise, it can lose its ‘quantumness’ and behave like a classical computer.

- **Quantum error-correcting codes** can be used to fight decoherence.
Quantum error correction

Building a large-scale quantum computer is extremely challenging because of decoherence.

If a quantum computer interacts with the outside world and is subject to noise, it can lose its ‘quantumness’ and behave like a classical computer.

- **Quantum error-correcting codes** can be used to fight decoherence.
- Optimistic estimates say error rates of up to 1% should be ok.

Pic: DOI:10.1126/science.1253742
Quantum error correction

Building a large-scale quantum computer is extremely challenging because of decoherence.

If a quantum computer interacts with the outside world and is subject to noise, it can lose its ‘quantumness’ and behave like a classical computer.

- Quantum error-correcting codes can be used to fight decoherence.
- Optimistic estimates say error rates of up to 1% should be ok.
- Error-correction will massively increase the number of physical qubits needed to implement a given computation (by a factor of 1,000 or more).
Noisy Intermediate-Scale Quantum Computation

Often abbreviated to **NISQ**.

- **Noisy**: does not use error correction.
- **Intermediate-scale**: about 50-100 qubits.

Computations are **kept short** to avoid errors accumulating, but are expected to **outperform standard computers** on certain tasks.

Pic: WP/John Preskill

October 2019: Google announces they have performed a computation in 600 seconds on their chip of 53 superconducting 'transmon' qubits, which would take 10,000 years on standard computers, or 2.5 days on IBM's Oak Ridge Summit Supercomputer.
Noisy Intermediate-Scale Quantum Computation

Often abbreviated to NISQ.

- Noisy: does not use error correction.
- Intermediate-scale: about 50-100 qubits.

Computations are kept short to avoid errors accumulating, but are expected to outperform standard computers on certain tasks.

October 2019: Google announces they have performed a computation in 600 seconds on their chip of 53 superconducting ‘transmon’ qubits, which would take 10,000 years on standard computers, or 2.5 days on IBM’s Oak Ridge Summit Supercomputer.
Outline

1. Introduction to quantum physics
2. What quantum computers are useful for
3. How to program a quantum computer
4. Building quantum computers
5. Conclusions
Quantum physics has strange effects such as superposition of states, entanglement, and measurement affecting the state.
Quantum physics has *strange effects* such as superposition of states, entanglement, and measurement affecting the state.

Quantum computers *use these effects* to solve certain problems better than standard computers can.
Quantum physics has **strange effects** such as superposition of states, entanglement, and measurement affecting the state.

Quantum computers **use these effects** to solve certain problems better than standard computers can.

Quantum algorithms are written down as **quantum circuits**.
Quantum physics has **strange effects** such as superposition of states, entanglement, and measurement affecting the state.

Quantum computers **use these effects** to solve certain problems better than standard computers can.

Quantum algorithms are written down as **quantum circuits**.

Theory and implementation of quantum computers for the NISQ era and beyond are being **actively developed**.
Quantum physics has **strange effects** such as superposition of states, entanglement, and measurement affecting the state.

Quantum computers **use these effects** to solve certain problems better than standard computers can.

Quantum algorithms are written down as **quantum circuits**.

Theory and implementation of quantum computers for the NISQ era and beyond are being **actively developed**.

There are still many interesting **open questions** about the power and potential of quantum computing to be explored.
Further reading

- Quantum Computing Since Democritus  
  Scott Aaronson  
  http://www.scottaaronson.com/democritus/

- Introduction to Quantum Computing  
  John Watrous  
  https://cs.uwaterloo.ca/~watrous/LectureNotes.html

- Quantum Computer Science  
  N. David Mermin, Cambridge University Press

- Quantum Computation and Quantum Information  
  Michael Nielsen and Isaac Chuang, Cambridge University Press

- Why Google’s Quantum Supremacy Milestone Matters  
  Scott Aaronson  
  https://www.nytimes.com/2019/10/30/opinion/google-quantum-computer-sycamore.html