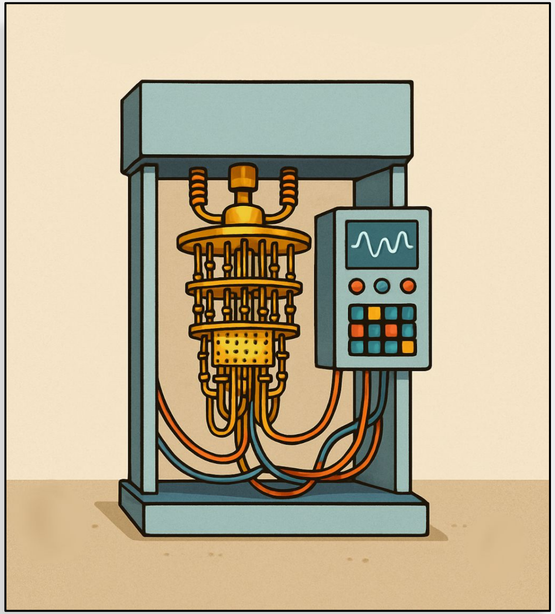
(Fall 2025) CS-599-P1:

Introduction to Quantum Computation



Instructor: Alexander Poremba

E-mail: poremba@bu.edu

Course website:

Myth #1

You need to know **quantum mechanics** to understand quantum computing.

Myth #1

You need to know **quantum mechanics** to understand quantum computing.

FALSE

Myth #2

Quantum computers can instantly solve **NP-complete** problems by trying all possible solutions at once.

Myth #2

Quantum computers can instantly solve **NP-complete** problems by trying all possible solutions at once.

Myth #3

Quantum computers can solve certain problems faster using **quantum effects** like superposition and entanglement.

Myth #3

Quantum computers can solve certain problems faster using **quantum effects** like superposition and entanglement.

Myth #4

Quantum computers can break **almost all** of the public-key cryptography we use on the internet today.

Myth #4

Quantum computers can break **almost all** of the public-key cryptography we use on the internet today.

Myth #5

Quantum computers **do not yet exist**.

Myth #5

Quantum computers **do not yet exist**.

FALSE

Myth #6

Quantum computers can
help us create new **drugs** and
cheaper **fertilizer**.

Myth #6

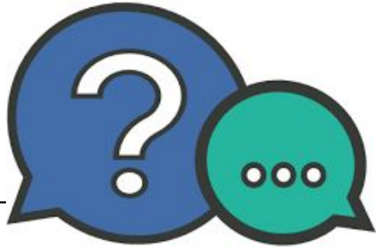
Quantum computers can
help us create new **drugs** and
cheaper **fertilizer**.

Myth #7

Quantum computers will solve **climate change**.

Myth #7

Quantum computers will
solve **climate change**.



Who is this class for?

This class is for you, if

- you’re an *advanced undergrad / grad student* who is curious about quantum computing
- you have some *familiarity* with complex numbers and linear algebra
- you’re especially drawn to computer science aspects of quantum computation

This class is for you, if

- you're an *advanced undergrad / grad student* who is curious about quantum computing
- you have some *familiarity* with complex numbers and linear algebra
- you're especially drawn to computer science aspects of quantum computation

This class *might not* be for you, if

- you want to *properly* learn quantum mechanics
- you want to learn how to *build* a quantum computer

This class is for you, if

- you're an *advanced undergrad / grad student* who is curious about quantum computing
- you have some *familiarity* with complex numbers and linear algebra
- you're especially drawn to computer science aspects of quantum computation

This class *might not* be for you, if

- you want to *properly* learn quantum mechanics
- you want to learn how to *build* a quantum computer

Parallel class at BU Physics:

PY 536: Quantum Computing

Instructor: Anushya Chandran

Time: Tue/Thu *11am – 12.15pm*

Location: SCI B58

What you will get out of this class

- **You will learn** the basics of quantum information and computation
- **You will learn** important quantum algorithms and protocols
- **You will get an idea** where the field is at today, and where it's headed
- **You will know enough** material to get involved in research

Logistics

- **Time:** Tuesday and Thursday (**5pm – 6.15pm**)
  from **09/02** to **12/10**. *The lectures will not be recorded.*
- **Location:** **CDS 265** (*665 Comm Ave, Center for Computing & Data Sciences*)
- **Office hours:** By appointment (**CDS 1037**)
- **Important sites:** Piazza and Gradescope (please sign up!)
- **Course website:** http://scc1.bu.edu/poremba/courses/index.html

If you haven't registered yet, please do!

Or, e-mail me at poremba@bu.edu if you want to receive updates.

Evaluation

- **25% Homework:**

  4 problem sets in total (bi-weekly)

  Posted *every other Tuesday* on **Piazza**.

  Due Tuesday, <u>two weeks later</u>, **11.59pm** on **Gradescope**

  (One token for a 48h extension, no questions asked)

- **25% Midterm Exam:**

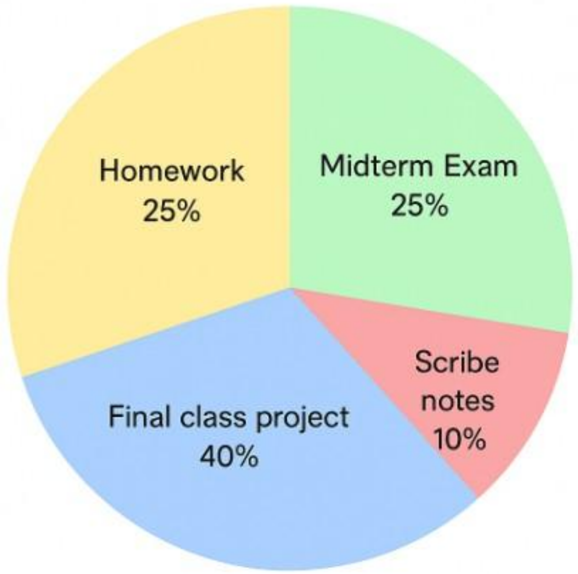
  In-class (closed book) on Thursday, <u>October 23rd</u>.

- **10% Scribe Notes:**

  Type up LaTeX notes for one of the classes (due **11.59pm** the day before the next class)

- **40% Final Class Project:**

  Prepare *a 15 min* presentation & written report (*5-10 pages*) on a selected topic.



Course policy

- Collaboration on homework problems is permitted and encouraged, but limited to groups of at most <u>two</u> students.

- You are free to work out the answers together. But you must write up the solutions <u>in your own words</u>.

- <u>Do not</u> copy solutions and <u>do not</u> use AI tools to solve your homework sets

- You have to acknowledge any resources you used for your assignments, especially for your final class project.

Check the website for academic policy at BU.

Worksheets

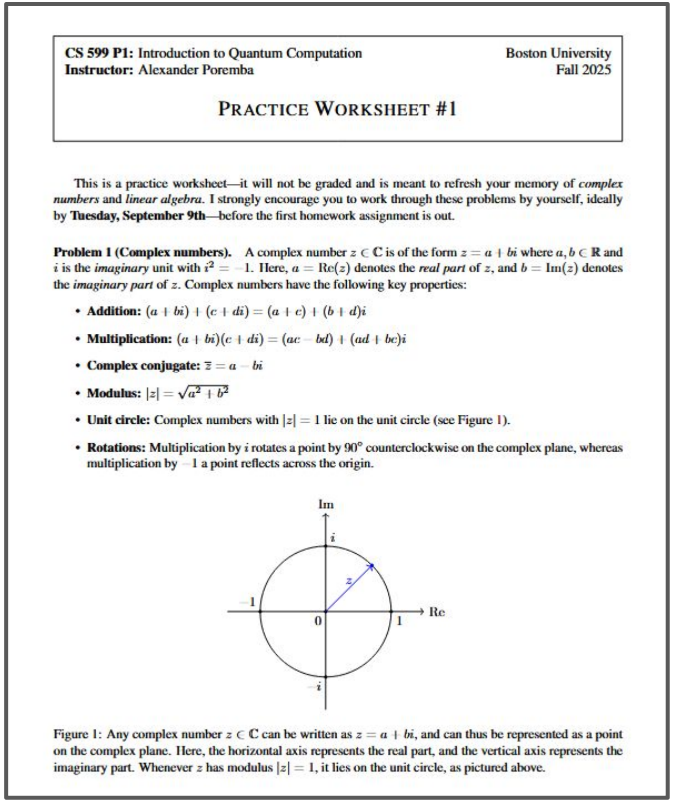
- 4 practice worksheets
  (bi-weekly, during off-weeks)

- These <u>will not be graded</u> and are meant to help you practice/review material.

- Posted *every other Tuesday* on the website.

Worksheets

- 4 practice worksheets
  (bi-weekly, during off-weeks)
- These <u>will not be graded</u> and are meant to help you practice/review material.
- Posted *every other Tuesday* on the website.
- First worksheet will be posted <u>today!</u>



CS 599 P1: Introduction to Quantum Computation — Boston University
Instructor: Alexander Poremba — Fall 2025

PRACTICE WORKSHEET #1

This is a practice worksheet—it will not be graded and is meant to refresh your memory of *complex numbers* and *linear algebra*. I strongly encourage you to work through these problems by yourself, ideally by **Tuesday, September 9th**—before the first homework assignment is out.

Problem 1 (Complex numbers). A complex number $z \in \mathbb{C}$ is of the form $z = a + bi$ where $a, b \in \mathbb{R}$ and i is the *imaginary* unit with $i^2 = -1$. Here, $a = \mathrm{Re}(z)$ denotes the *real part* of z, and $b = \mathrm{Im}(z)$ denotes the *imaginary part* of z. Complex numbers have the following key properties:

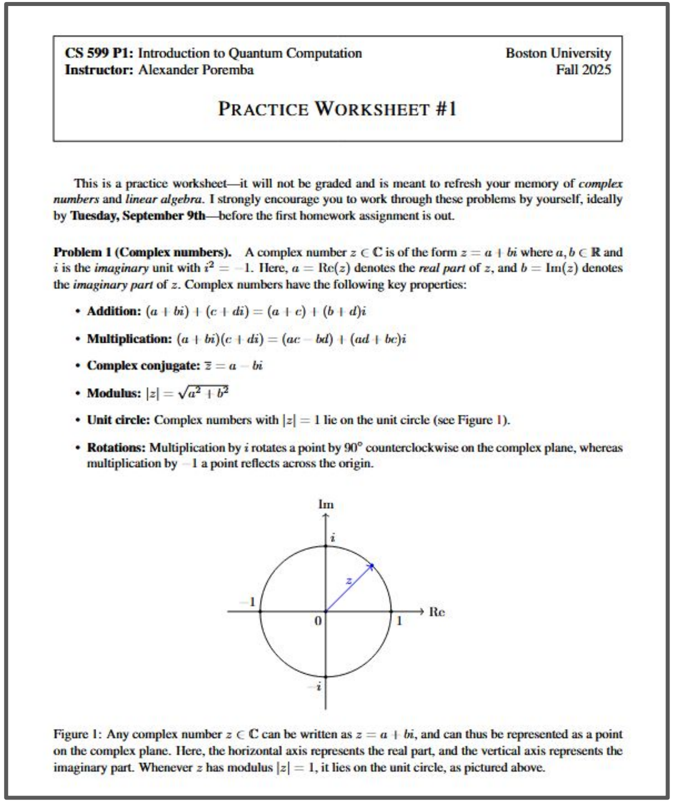
- **Addition:** $(a + bi) + (c + di) = (a + c) + (b + d)i$
- **Multiplication:** $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$
- **Complex conjugate:** $\bar{z} = a - bi$
- **Modulus:** $|z| = \sqrt{a^2 + b^2}$
- **Unit circle:** Complex numbers with $|z| = 1$ lie on the unit circle (see Figure 1).
- **Rotations:** Multiplication by i rotates a point by $90°$ counterclockwise on the complex plane, whereas multiplication by -1 a point reflects across the origin.

Figure 1: Any complex number $z \in \mathbb{C}$ can be written as $z = a + bi$, and can thus be represented as a point on the complex plane. Here, the horizontal axis represents the real part, and the vertical axis represents the imaginary part. Whenever z has modulus $|z| = 1$, it lies on the unit circle, as pictured above.

Worksheets

- 4 practice worksheets
  (bi-weekly, during off-weeks)
- These <u>will not be graded</u> and are meant to help you practice/review material.
- Posted *every other Tuesday* on the website.
- First worksheet will be posted <u>today!</u>

This is an important refresher for complex numbers & linear algebra



CS 599 P1: Introduction to Quantum Computation — Boston University
Instructor: Alexander Poremba — Fall 2025

PRACTICE WORKSHEET #1

This is a practice worksheet—it will not be graded and is meant to refresh your memory of *complex numbers* and *linear algebra*. I strongly encourage you to work through these problems by yourself, ideally by **Tuesday, September 9th**—before the first homework assignment is out.

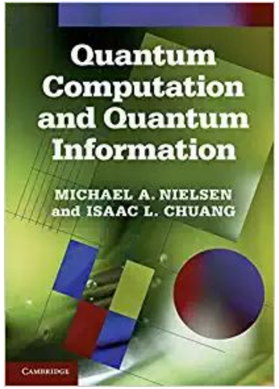
Problem 1 (Complex numbers). A complex number $z \in \mathbb{C}$ is of the form $z = a + bi$ where $a, b \in \mathbb{R}$ and i is the *imaginary* unit with $i^2 = -1$. Here, $a = \mathrm{Re}(z)$ denotes the *real part* of z, and $b = \mathrm{Im}(z)$ denotes the *imaginary part* of z. Complex numbers have the following key properties:

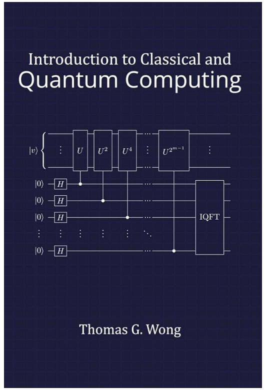
- **Addition:** $(a + bi) + (c + di) = (a + c) + (b + d)i$
- **Multiplication:** $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$
- **Complex conjugate:** $\overline{z} = a - bi$
- **Modulus:** $|z| = \sqrt{a^2 + b^2}$
- **Unit circle:** Complex numbers with $|z| = 1$ lie on the unit circle (see Figure 1).
- **Rotations:** Multiplication by i rotates a point by $90°$ counterclockwise on the complex plane, whereas multiplication by -1 a point reflects across the origin.

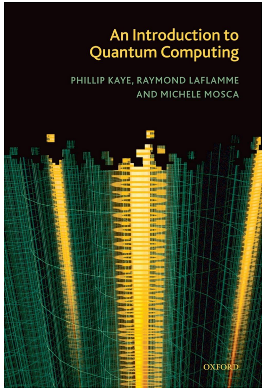
Figure 1: Any complex number $z \in \mathbb{C}$ can be written as $z = a + bi$, and can thus be represented as a point on the complex plane. Here, the horizontal axis represents the real part, and the vertical axis represents the imaginary part. Whenever z has modulus $|z| = 1$, it lies on the unit circle, as pictured above.

Useful resources

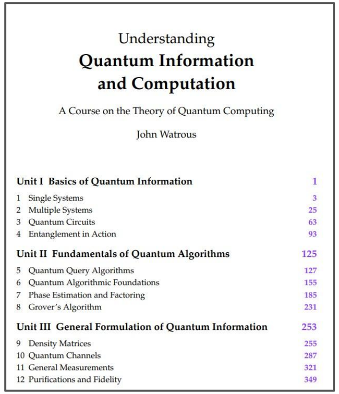
There is *no official textbook*, but you might find these books useful!







I also recommend these *lecture notes* by John Watrous:



Understanding
Quantum Information and Computation
A Course on the Theory of Quantum Computing
John Watrous

You can find links to all of these resources on the course website!

Why learn quantum computing?

Why learn quantum computing?

- Model of computation, based on our most most successful and complete theory of physics—**quantum mechanics**.

Why learn quantum computing?

- Model of computation, based on our most most successful and complete theory of physics—**quantum mechanics**.


NEWS QUANTUM PHYSICS

Quantum mechanics was born 100 years ago. Physicists are celebrating

The International Year of Quantum marks a century of scientific developments

Why learn quantum computing?

- Model of computation, based on our most most successful and complete theory of physics—**quantum mechanics**.

- It's here to stay! There is a lot of recent exciting **experimental progress** in building actual quantum hardware.

NEWS QUANTUM PHYSICS

Quantum mechanics was born 100 years ago. Physicists are celebrating

The International Year of Quantum marks a century of scientific developments

Why learn quantum computing?

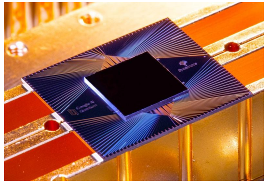
- Model of computation, based on our most most successful and complete theory of physics—**quantum mechanics**.

- It's here to stay! There is a lot of recent exciting **experimental progress** in building actual quantum hardware.

NEWS QUANTUM PHYSICS

Quantum mechanics was born 100 years ago. Physicists are celebrating

The International Year of Quantum marks a century of scientific developments



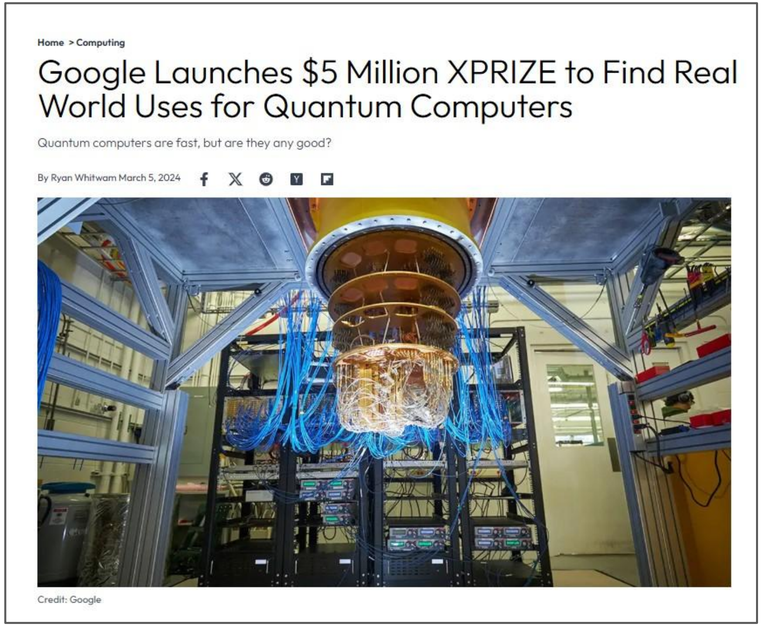
QUANTINUUM Google IBM

XANADU IONQ

PsiQuantum QuEra Computing Inc.

Why learn quantum computing?

- Model of computation, based on our most most successful and complete theory of physics—**quantum mechanics**.
- It's here to stay! There is a lot of recent exciting **experimental progress** in building actual quantum hardware.
- There is a lot of **opportunity** to get involved in cutting-edge research, especially for *computer scientists.*



Where did it all begin?

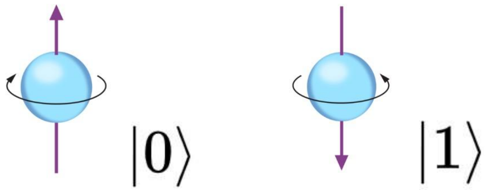
The origins of quantum computing

The *simplest* possible quantum mechanical system has two degrees of freedom.

The origins of quantum computing

The *simplest* possible quantum mechanical system has two degrees of freedom.

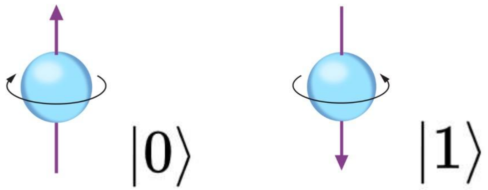
Electron spin



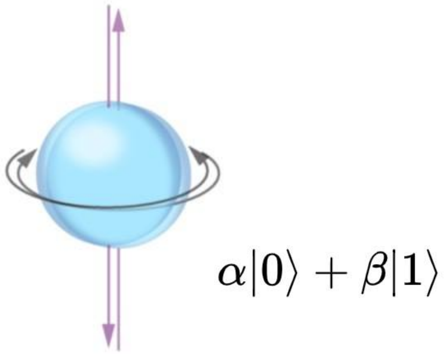
The origins of quantum computing

The *simplest* possible quantum mechanical system has two degrees of freedom.

Electron spin



Until we *observe* the electron using a measurement, we don't know whether it is spin up or spin down.

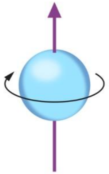


We call this a superposition.

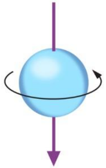
The origins of quantum computing

The *simplest* possible quantum mechanical system has two degrees of freedom.

Electron spin

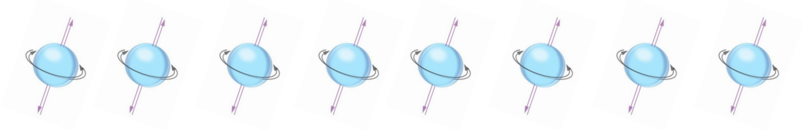


$|0\rangle$



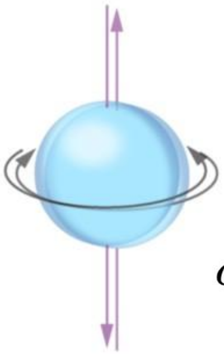
$|1\rangle$

What about a system of *N* spins?



This system is described by 2^N possible states!

Until we *observe* the electron using a measurement, we don't know whether it is spin up or spin down.



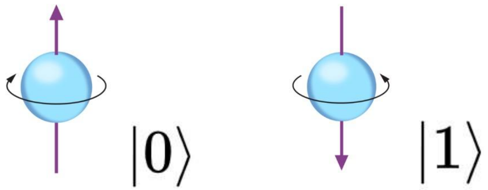
$\alpha|0\rangle + \beta|1\rangle$

We call this a superposition.

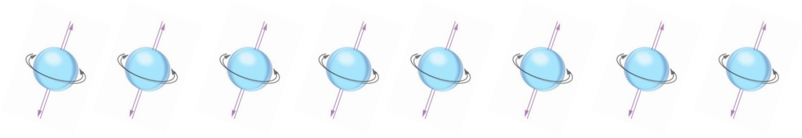
The origins of quantum computing

The *simplest* possible quantum mechanical system has two degrees of freedom.

Electron spin



What about a system of N spins?



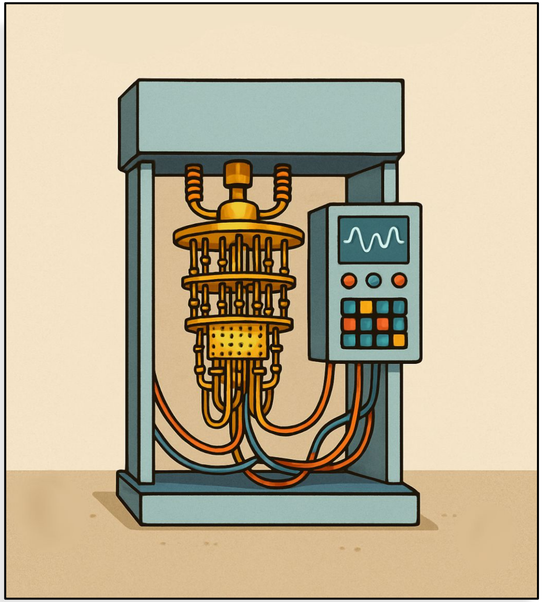
This system is described by 2^N possible states!

Richard Feynman (1981)

"...Nature isn't classical, damnit, and if you want to make a simulation of nature, you'd better make it quantum mechanical."

Quantum computing: the early days

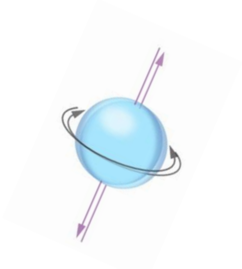
- **1980s:** Feynman (along with David Deutsch, Paul Benioff, Yuri Manin) proposed the notion of a "quantum computer".



Quantum computing: the early days

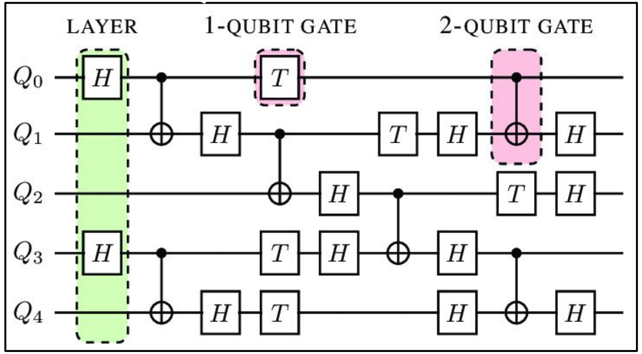
- **1980s:** Feynman (along with David Deutsch, Paul Benioff, Yuri Manin) proposed the notion of a "quantum computer".

Quantum information processing



Qubit

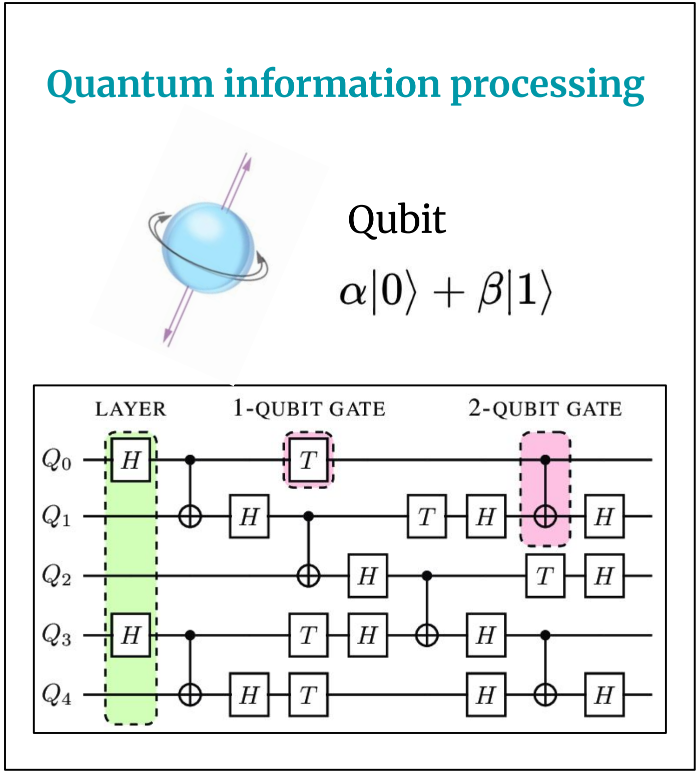
$$\alpha|0\rangle + \beta|1\rangle$$



Quantum computing: the early days

- **1980s:** Feynman (along with David Deutsch, Paul Benioff, Yuri Manin) proposed the notion of a "**quantum computer**".

- **1985:** *David Deutsch* found a **non-physics problem** that could be solved faster on a quantum computer than on any classical computer (by a constant factor)



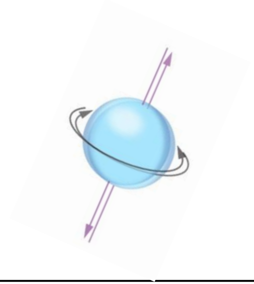
Quantum computing: the early days

- **1980s:** Feynman (along with David Deutsch, Paul Benioff, Yuri Manin) proposed the notion of a "**quantum computer**".

- **1985:** *David Deutsch* found a **non-physics problem** that could be solved faster on a quantum computer than on any classical computer (by a constant factor)

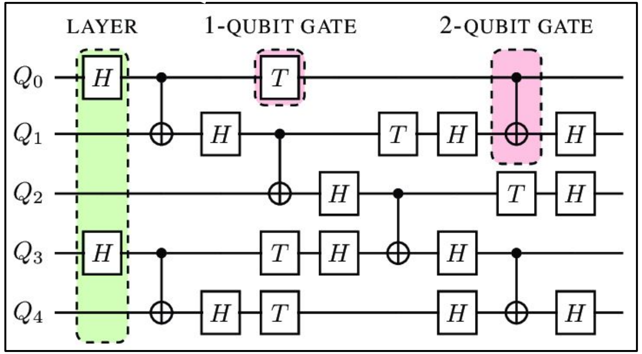
- **1992–1994:** *Bernstein and Vazirani* (and later *Dan Simon*) discovered the first problems that could be solved **exponentially faster**!

Quantum information processing



Qubit

$$\alpha|0\rangle + \beta|1\rangle$$

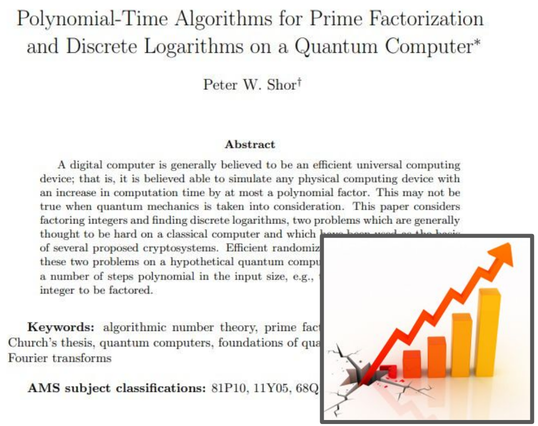


Quantum computing: the early days

- **1980s:** Feynman (along with David Deutsch, Paul Benioff, Yuri Manin) proposed the notion of a "**quantum computer**".

- **1985:** *David Deutsch* found a **non-physics problem** that could be solved faster on a quantum computer than on any classical computer (by a constant factor)

- **1992–1994:** *Bernstein and Vazirani* (and later *Dan Simon*) discovered the first problems that could be solved **exponentially faster**!



Polynomial-Time Algorithms for Prime Factorization
and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which
of several proposed cryptosystems. Efficient randomiz
these two problems on a hypothetical quantum compu
a number of steps polynomial in the input size, e.g.,
integer to be factored.

Keywords: algorithmic number theory, prime fac
Church's thesis, quantum computers, foundations of qua
Fourier transforms

AMS subject classifications: 81P10, 11Y05, 68Q

***1994:** Peter Shor* discovered a **quantum factoring algorithm**.

"These algorithms take a number of steps polynomial in the input size, for example, the number of digits of the integer to be factored."

Why Shor’s algorithm matters

Why Shor's algorithm matters

Can you *factor* this integer?

51 =

Why Shor's algorithm matters

Can you *factor* this integer?

51 = 3 ✖ *17*

Why Shor's algorithm matters

Can you *factor* this integer?

51 = 3 ✖ 17

How about this one?

1487342080 =

Why Shor’s algorithm matters

Can you *factor* this integer?

51 = 3 ✖ *17*

How about this one?

1487342080 = 15904 ✖ *93520*

Why Shor's algorithm matters

Can you *factor* this integer?

$51 = 3 \times 17$

How about this one?

$1487342080 = 15904 \times 93520$

For an n-digit number, this problem is believed to computationally intractable for current computers.

Why Shor's algorithm matters

Can you *factor* this integer?

$51 = 3 \times 17$

How about this one?

$1487342080 = 15904 \times 93520$

For an n-digit number, this problem is believed to computationally intractable for current computers.

The best known algorithms require a runtime which is *superpolynomial* in n.

Why Shor's algorithm matters

Can you *factor* this integer?

51 = 3 ✖ 17

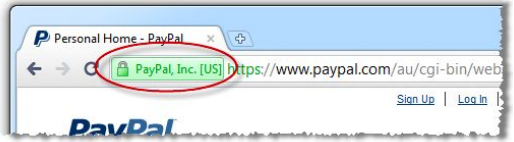
How about this one?

1487342080 = 15904 ✖ 93520

For an *n*-digit number, this problem is believed to computationally intractable for current computers.

The best known algorithms require a runtime which is *superpolynomial* in *n*.

The security of the **RSA cryptosystem** relies on the hardness of factoring!



It is the **most widely used** *public-key encryption scheme* that we use on the internet today!

So, what’s stopping us?

Building a *large-scale* quantum computer is an enormous scientific challenge.

So, what's stopping us?

Building a *large-scale* quantum computer is an enormous scientific challenge.

To harness **quantum effects**, these devices have to operate on extremely small scales, which makes them:

- susceptible to **thermal noise**,
- difficult to **control**, and
- challenging to **scale**.

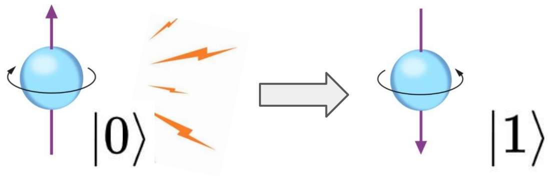
So, what’s stopping us?

Building a *large-scale* quantum computer is an enormous scientific challenge.

To harness **quantum effects**, these devices have to operate on extremely small scales, which makes them:

- susceptible to **thermal noise**,
- difficult to **control**, and
- challenging to **scale**.

Noise in quantum systems



So, what's stopping us?

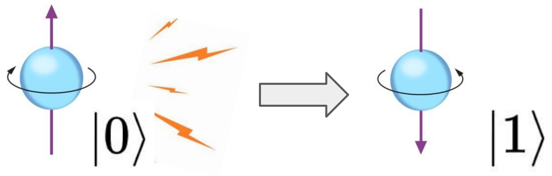
Building a *large-scale* quantum computer is an enormous scientific challenge.

To harness **quantum effects**, these devices have to operate on extremely small scales, which makes them:

- susceptible to **thermal noise**,
- difficult to **control**, and
- challenging to **scale**.

Is quantum computing *doomed to fail*?

Noise in quantum systems



So, what’s stopping us?

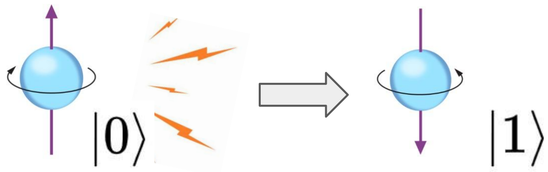
Building a *large-scale* quantum computer is an enormous scientific challenge.

To harness **quantum effects**, these devices have to operate on extremely small scales, which makes them:

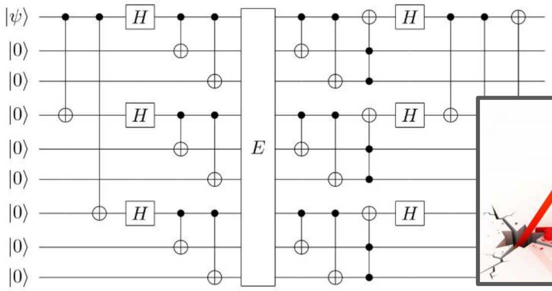
- susceptible to **thermal noise**,
- difficult to **control**, and
- challenging to **scale**.

Is quantum computing *doomed to fail*?

Noise in quantum systems



1995: Peter Shor comes up with a **quantum error-correcting code**!



Quantum fault-tolerance

Since the discovery of the **Shor code** in 1995, many more *quantum error-correcting codes* have been discovered.

Quantum fault-tolerance

Since the discovery of the **Shor code** in 1995, many more *quantum error-correcting codes* have been discovered.

The existence of these codes tells us that **quantum fault-tolerance** is *actually* possible—at least in principle!

Quantum fault-tolerance

Since the discovery of the **Shor code** in 1995, many more *quantum error-correcting codes* have been discovered.

The existence of these codes tells us that **quantum fault-tolerance** is *actually* possible—at least in principle!

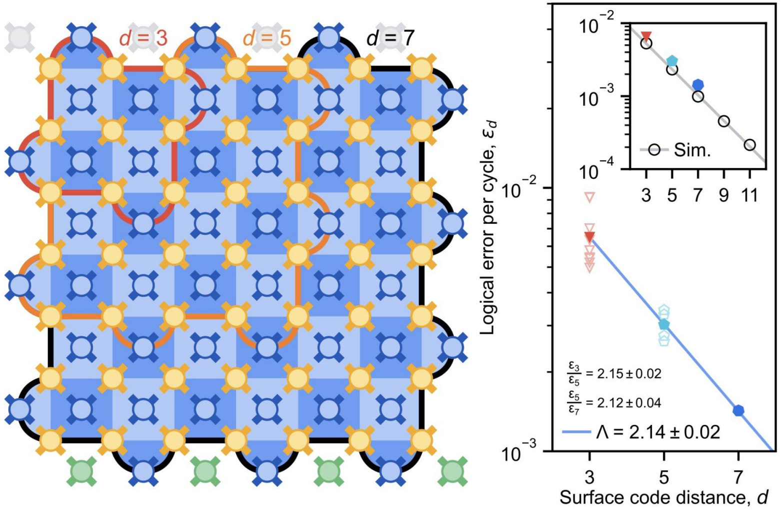


Image (Google): *Quantum error correction below the surface code threshold*, **Nature (2025).**

Quantum fault-tolerance

Since the discovery of the Shor code in 1995, many more *quantum error-correcting codes* have been discovered.

The existence of these codes tells us that **quantum fault-tolerance** is *actually* possible—at least in principle!

Quantum error-correction, today, is a major research area with lots of recent progress.

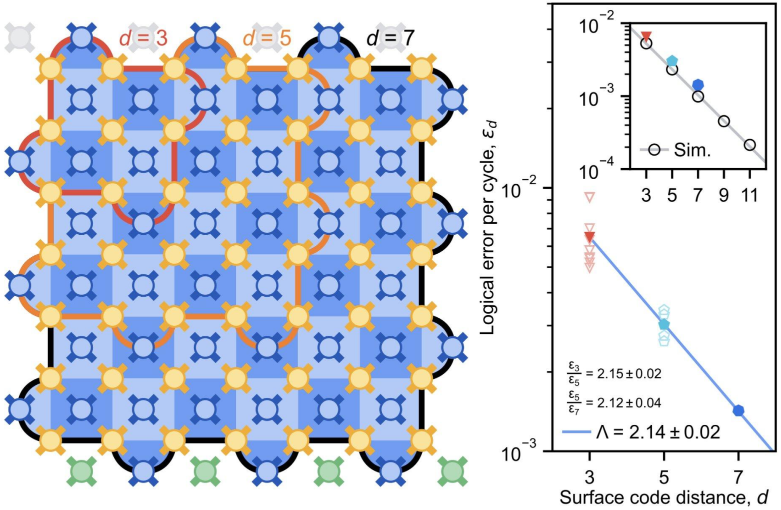


Image (Google): *Quantum error correction below the surface code threshold*, **Nature (2025).**

The road ahead

Our understanding of what quantum computers can do is still very *limited*.

Image: Google

What are quantum computers good for?

So far we found several interesting examples of *quantum advantage*:

- Quantum algorithms for **algebraic problems**, e.g. *factoring, discrete logarithm, hidden subgroup problem*

$N = p \times q$ $\qquad g^a \to a$

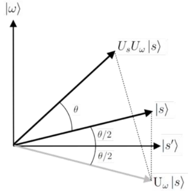
What are quantum computers good for?

So far we found several interesting examples of *quantum advantage*:

- Quantum algorithms for **algebraic problems**, e.g.
  factoring, discrete logarithm, hidden subgroup problem

$N = p \times q$ $\quad\quad g^a \rightarrow a$

- Quantum algorithms for **unstructured search**, e.g.
  Grover search, quantum walks



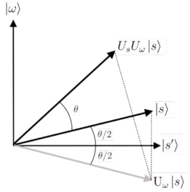
What are quantum computers good for?

So far we found several interesting examples of *quantum advantage*:

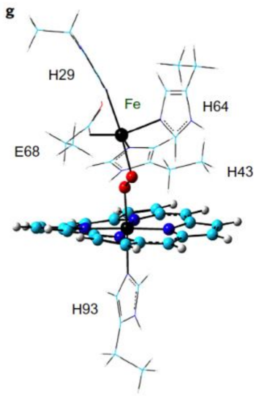
- Quantum algorithms for **algebraic problems**, e.g.
  factoring, discrete logarithm, hidden subgroup problem

$$N = p \times q \qquad\qquad g^a \to a$$

- Quantum algorithms for **unstructured search**, e.g.
  Grover search, quantum walks



- Quantum algorithms for **simulating physics and chemistry**
  (i.e., Feynman's original dream)



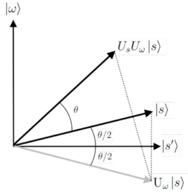
What are quantum computers good for?

So far we found several interesting examples of *quantum advantage*:

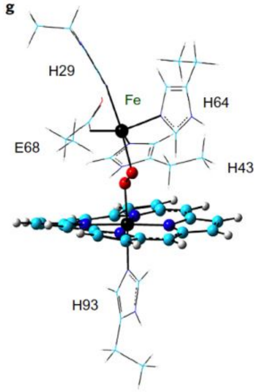
- Quantum algorithms for **algebraic problems**, e.g.
  factoring, discrete logarithm, hidden subgroup problem

$N = p \times q \qquad g^a \to a$

- Quantum algorithms for **unstructured search**, e.g.
  Grover search, quantum walks



- Quantum algorithms for **simulating physics and chemistry**
  (i.e., Feynman's original dream)



- Quantum algorithms in **optimization and machine learning**, e.g.
  for constrained satisfaction problems, SDPs, topological data analysis

Overview of the class

- **(Weeks 1-3)** Basics of quantum information and quantum computation
- **(Weeks 4-8)** Quantum gates & quantum circuits, fundamental quantum algorithms, and quantum complexity theory
- **(Weeks 9-10)** Mixed-state formalism, noisy quantum systems, quantum error correction and quantum fault-tolerance
- **(Weeks 11-14)** Quantum cryptography & selected advanced topics
- **(Week 15)** Final student presentations

Overview of the class

- **(Weeks 1-3)** Basics of quantum information and quantum computation
- **(Weeks 4-8)** Quantum gates & quantum circuits, fundamental quantum algorithms, and quantum complexity theory

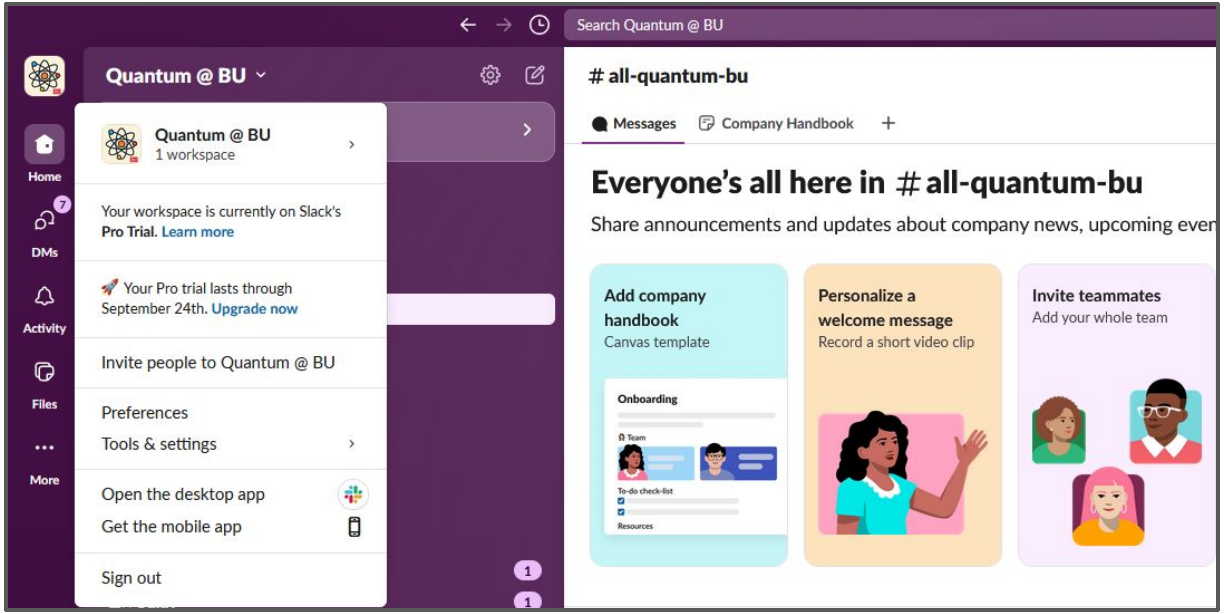
Covered in the midterm exam!

- **(Weeks 9-10)** Mixed-state formalism, noisy quantum systems, quantum error correction and quantum fault-tolerance
- **(Weeks 11-14)** Quantum cryptography & selected advanced topics
- **(Week 15)** Final student presentations

Quantum @ BU slack channel

Join to stay updated on **quantum-related events at BU**!

To sign up, you can find an invitation link on my website.



Next time: The Qubit