

CPEN 400Q Lecture 01

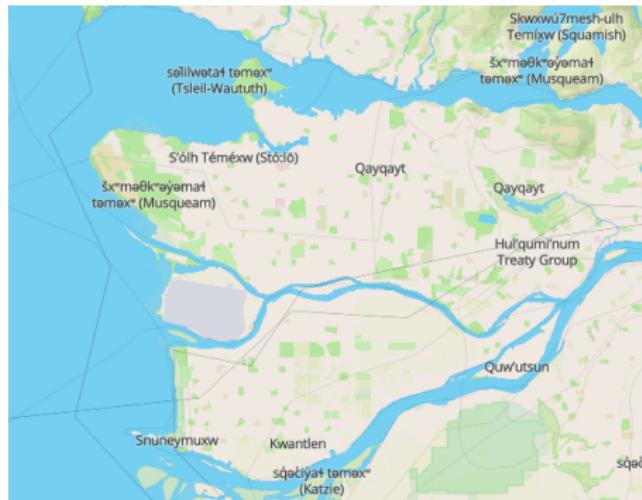
Overview and intro to gate model

quantum computing

Monday 8 January 2024

Land acknowledgement

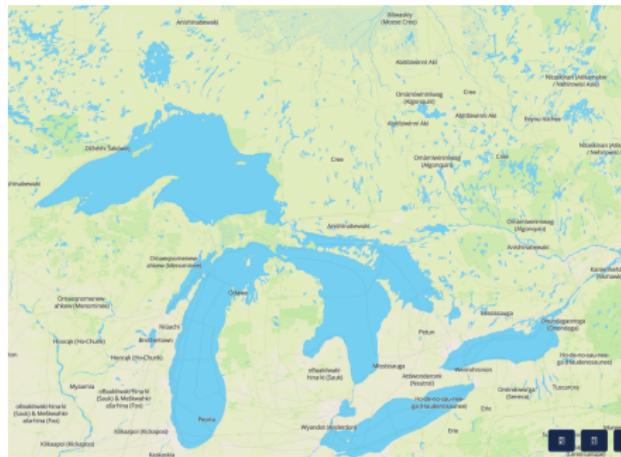
The land on which we gather today is the traditional, ancestral, and unceded territory of the Musqueam People.



Explore more:

- <https://www.musqueam.bc.ca/>
 - <https://native-land.ca/>

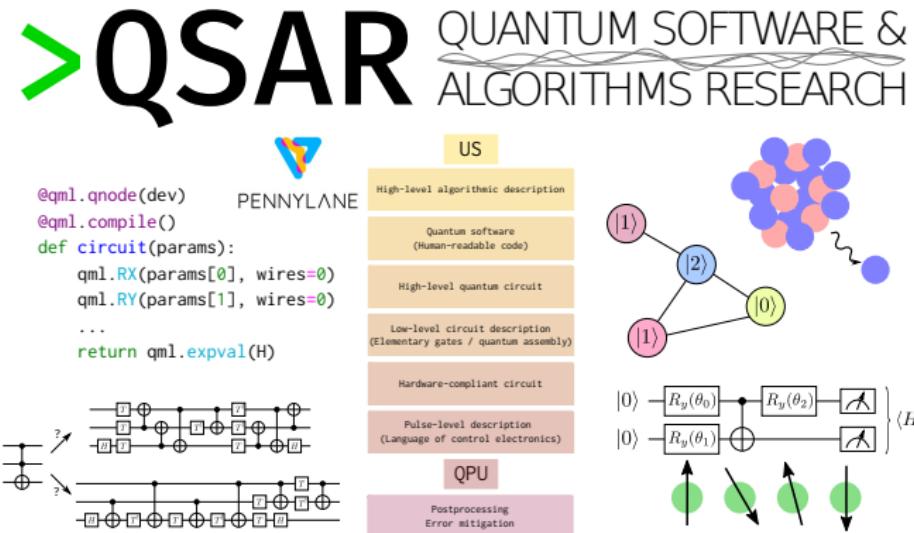
Thunder Bay is located on the traditional lands of the Fort William First Nation, Signatory to the Robinson-Superior Treaty of 1850. Waterloo is in the traditional territory of the Neutral, Anishinaabeg, and Haudenosaunee peoples.



Intros

Olivia Di Matteo – olivia@ece.ubc.ca

I have a physics background my group works on open-source quantum software and algorithms.



Your perceptions about quantum computing

Go to <https://www.menti.com> and type in the code.



Why quantum computing?

How do we make computers more powerful?

Why quantum computing?

How do we make computers more powerful?

- make smaller transistors
- put more transistors onto a single chip
- use multiple processors in parallel



Frontier (ORNL): \sim 8.7 million cores, \sim 1.2 exaFLOPS

Image credit:

<https://www.olcf.ornl.gov/wp-content/themes/olcf-edition-child/frontier-assets/systems/frontier.jpg>

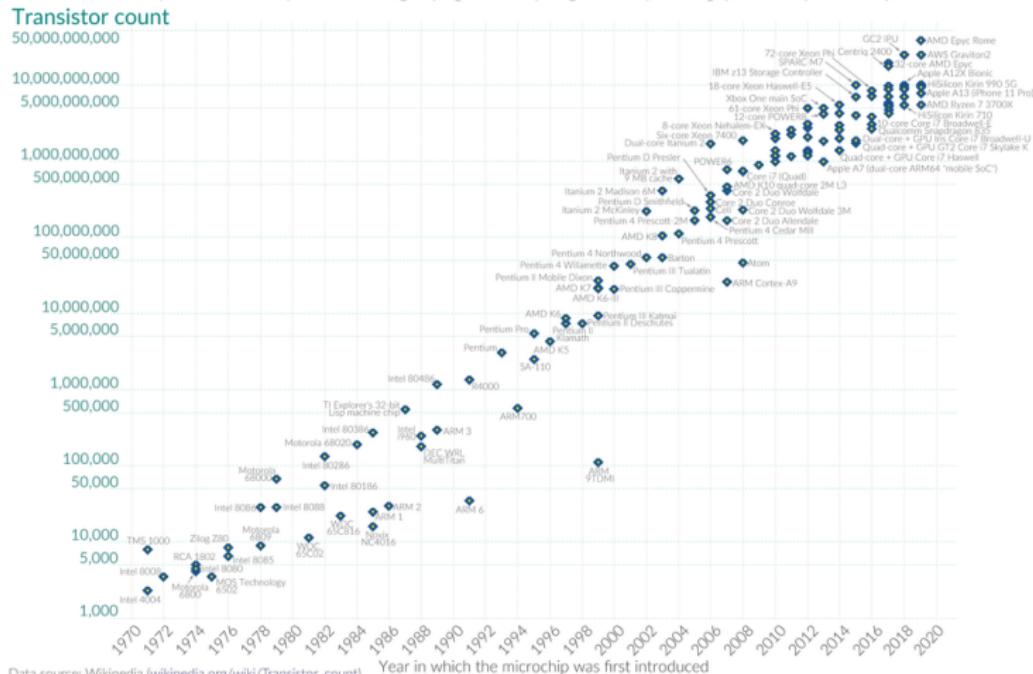
Why quantum computing?

This will only get us so far.

Moore's Law: The number of transistors on microchips doubles every two years

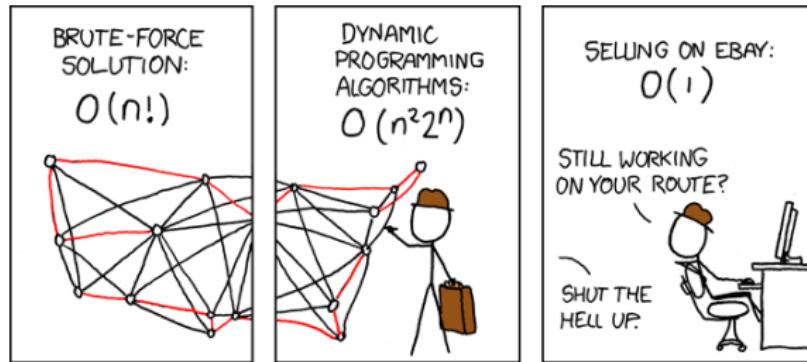
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World
in Data



Why quantum computing?

Some problems will still remain intractable.



Sometimes that's a good thing:

- our cryptographic infrastructure relies on some mathematical problems being *computationally hard*

Why quantum computing?

But usually it just prevents us from doing interesting things:

- solving complex optimization problems
- simulation of molecules and quantum systems
- searching large spaces
- machine learning with large amounts of data

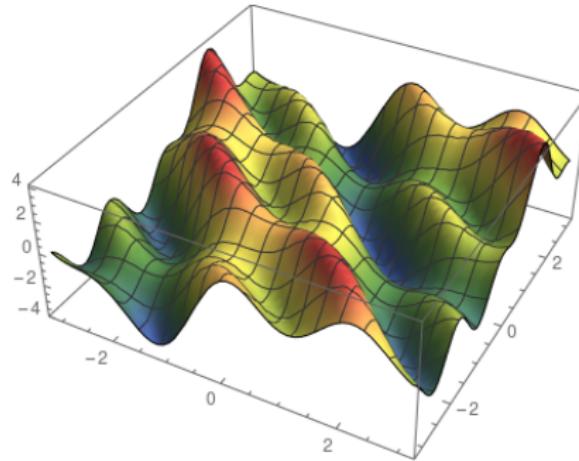


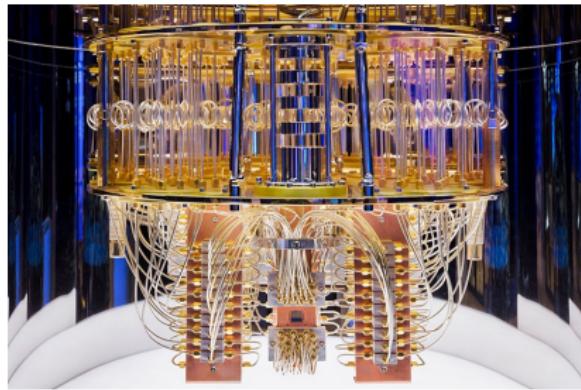
Image credit:

<https://towardsdatascience.com/the-mathematics-of-optimization-for-deep-learning-11af2b1fda30>

Quantum computing

“Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems.”

— Nielsen & Chuang



There exist quantum algorithms that have an *exponential speedup* over regular computers (and some that don't, but are still good).

Quantum computing

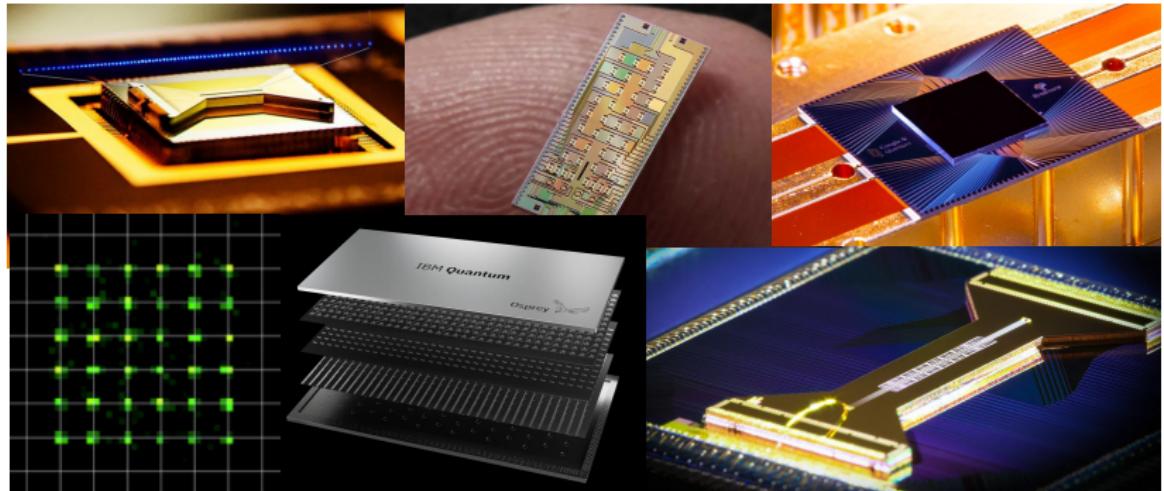
Lots of potential applications:

- Simulating and calculating properties of physical systems
- Searching large spaces
- Discrete mathematics, (breaking) cryptography
- Optimization
- Machine learning
- Things we haven't thought of yet!

We are already capable of doing small, proof-of-concept implementations of some of these algorithms...

Quantum computers are here

There are lots of different ones.

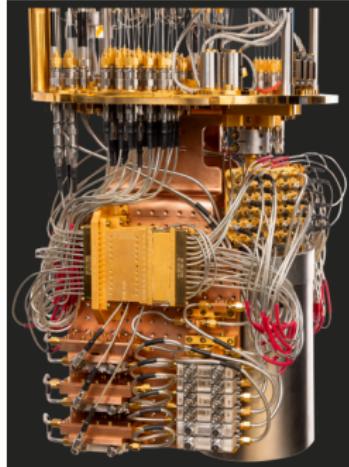


Most qubits: IBM Q Condor, 1121 superconducting qubits

Commercial quantum processors (images clockwise from top-left): IonQ (trapped-ion), Xanadu (photonic), Google (superconducting), Quantinuum (trapped-ion), IBM (superconducting), Pasqal (neutral atoms)

Quantum computers are here

You can literally buy one off-the-shelf*



Advance the science.
Advance your work.

The Novera QPU is available to ship immediately. Allow 4-6 weeks for delivery once your order has been confirmed and shipping logistics are finalized.

ORDER YOURS

FROM \$900K USD

NOVERA

*dilution refrigerator not included

Image source: <https://www.rigetti.com/novera>

Quantum computers are here

IBM Quantum roadmap (2023)

Development Roadmap

	2016–2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2033+	IBM Quantum
Data Scientist	Run quantum circuits on the IBM Quantum Platform	Beta test multi-dimensional roadmap publicly with initial aim focused on scaling	Enhancing quantum execution speed by 100x with Qiskit Runtime	Bring dynamic circuits to unlock more computations	Enhancing quantum execution speed by 5x with quantum serverless and Execution modes	Improving quantum circuit quality and parallelization with partitioning and quantum modularity	Enhancing quantum execution speed and parallelization with partitioning and quantum modularity	Improving quantum circuit quality to allow 7.5K gates	Improving quantum circuit quality to allow 10K gates	Improving quantum circuit quality to allow 15K gates	Improving quantum circuit quality to allow 100M gates	Beyond 2033, quantum-centric supercomputers will include 1000's of logical qubits unlocking the full power of quantum computing	
Researchers													
Quantum Physical	IBM Quantum Experience	Qiskit	Qiskit Runtime	Quantum Services	Transpile Service	Resource Management	Gauge Keeping & P.	Intelligent Orchestration					General purpose QC libraries
	Early	Early	Qiskit	Dynamic circuits	Execution Modes	Heron (5K)	Flamingo (5K)	Flamingo (7.5K)	Flamingo (10K)	Flamingo (15K)	Starling (100M)	Blue Jay (1B)	
	Carey 5 qubits Altairia 16 qubits Pegasus 20 qubits Prototyp 53 qubits	Falcon Benchmarking 27 qubits	Eagle Benchmarking 127 qubits			Error Mitigation 5k qubits 150 qubits Classical modular 1333x + 199 qubits	Error Mitigation 5k qubits 150 qubits Quantum modular 150x7x + 1092 qubits	Error Mitigation 7.5k qubits 250 qubits Quantum modular 151x7x + 1092 qubits	Error Mitigation 10k qubits 250 qubits Quantum modular 151x7x + 1092 qubits	Error Mitigation 15k qubits 350 qubits Quantum modular 151x7x + 1092 qubits	Error Mitigation 100M qubits 2000 qubits Cross-ported modularity	Error correction 100 qubits 2000 qubits Cross-ported modularity	

Innovation Roadmap

Software Innovation	IBM Quantum Experience	Qiskit	Application modules	Qiskit Runtime	Serverless	AI enhanced quantum	Resource management	Scalable circuit knitting	Error correction decoder				
Hardware Innovation	Early	Falcon	Hummingbird	Eagle	Osprey	Condor	Flamingo	Kookaburra	Cockatoo	Starling			
	Carey 4 qubits Fengate 16 qubits Altairia 16 qubits Prototyp 53 qubits	Demonstrate scaling with high density with bump bonds	Demonstrate scaling with high density with interconnecting resistors	Demonstrate scaling with high density with PCB	Demonstrate scaling with high density signal delivery	Demonstrate scaling with high density interconnects	Demonstrate scaling with high density signal delivery	Demonstrate scaling with high density interconnects	Demonstrate path to high performance logical memory	Demonstrate path to high performance logical memory			

● Executed by IBM
⌚ On target

IBM Quantum / © 2023 IBM Corporation

Image source: https://www.flickr.com/photos/ibm_research_zurich/53347055153/

Quantum computers are here

Today's QCs are termed **noisy, intermediate scale quantum** (NISQ) devices.

Quantum Computing in the NISQ era and beyond

John Preskill

Institute for Quantum Information and Matter and Walter Burke Institute for Theoretical Physics, California Institute of Technology, Pasadena CA 91125, USA

Published: 2018-08-06, [volume 2](#), page 79

Eprint: [arXiv:1801.00862v3](https://arxiv.org/abs/1801.00862)

Doi: <https://doi.org/10.22331/q-2018-08-06-79>

Citation: [Quantum 2, 79 \(2018\)](#).

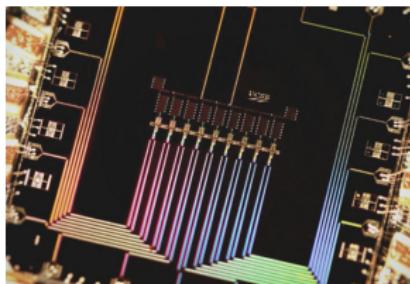
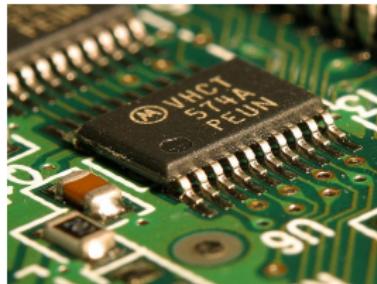
- Short coherence times
- High gate error rates (single-qubit, two-qubit operations)
- Limited qubit connectivity
- Challenging to scale up the hardware

Quantum computers are here

Many competing technologies; still too early to know which (combination?) of them, if any, will win out long term.



VS.



VS.



Image credits:

<https://hubpages.com/business/What-Is-a-Transistor-and-Why-is-it-Important>

https://en.wikipedia.org/wiki/Solid-state_electronics

<https://physicsworld.com/a/google-gains-new-ground-on-universal-quantum-computer/>

Quantum computers are here

We can still use NISQ devices to do interesting things; but to solve problems “at scale” we need **fault-tolerant quantum computing**.

We are starting to see error-corrected qubits in the wild...

nature

Explore content ▾ About the journal ▾ Publish with us ▾

nature > articles > article

Article | Published: 06 December 2023

Logical quantum processor based on reconfigurable atom arrays

Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataldeas, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić & Mikhail D. Lukin 

quantum computing. Here we report the realization of a programmable quantum processor based on encoded logical qubits operating with up to 280 physical qubits. Utilizing logical-level control and a zoned architecture in reconfigurable neutral atom arrays⁷, our system combines high two-qubit gate fidelities⁸, arbitrary connectivity^{7,9}, as well as fully programmable single-qubit rotations and mid-circuit readout^{10,11,12,13,14,15}. Operating this logical processor with various types of encodings, we demonstrate improvement of a two-qubit logic gate by scaling surface code⁶ distance from $d = 3$ to $d = 7$, preparation of color code qubits with break-even fidelities⁵, fault-tolerant creation of logical GHZ states and feedforward entanglement teleportation, as well as operation of 40 color code qubits. Finally, using three-dimensional [[8,3,2]] code blocks^{16,17}, we realize computationally complex sampling circuits¹⁸ with up to 48 logical qubits entangled with hypercube connectivity¹⁹ with 228 logical two-qubit gates and 48 logical CCZ gates²⁰. We find that this logical encoding

(Neutral atom processor, US-based startup QuEra)

Quantum computers are here

Some applications will require *millions* of qubits to run an algorithm on noisy qubits with full error correction.

RSA-2048				Old estimates		Current estimates			
p_g	n_ℓ	n_p	quantum resources	time	n_ℓ	n_p	quantum resources	time	
10^{-3}	6190	19.20		1.17	1.46	8194	22.27	0.27	0.34
10^{-5}	6190	9.66		0.34	0.84	8194	8.70	0.06	0.15

Table 2. RSA-2048 security estimates. Here n_ℓ denotes the number of logical qubits, n_p denotes the number of physical qubits (in millions), time denotes the expected time (in hours) to break the scheme, and quantum resources (quantum resources) are expressed in units of megaqubitdays. The corresponding classical security parameter is 112 bits.

Image: V. Gheorghiu and M. Mosca, *A resource estimation framework for quantum attacks against cryptographic functions - recent developments*. Feb. 15 2021.

Quantum computers are here

You can go use them right now!



XANADU CLOUD
Welcome to Xanadu Cloud
Join a community of learners, researchers, and developers to build the next generation of quantum technology.

D:wave Leap

Amazon Braket Quantum Computers
Amazon Braket provides AWS customers access to quantum computing capabilities from four quantum providers, including D-Wave Systems, IonQ, OQC, and Rigetti. Learn more about these quantum computing providers.

IONQ
IonQ's quantum computers are universal gate-based machines using ion-trap quantum optics. The internal state of these optical atoms make up the qubits. The evolution of computation is accomplished by programming the sequence of laser pulses needed to prepare and question gate operations.

OQC
OQC's Quantum Computer is based on Rydberg atom optics, which utilizes ionization lasers of individual rubidium atoms to request an entanglement between two qubits. Qubit operations can be done via the detection of atoms.

IQuera> COMPUTING INC.
IQuera's quantum computer is based on Rydberg atom optics, which utilizes ionization lasers of individual rubidium atoms to request an entanglement between two qubits. Qubit operations can be done via the detection of atoms.

rigetti
Rigetti's quantum processor is a universal gate-based machine based on superconducting qubits made of niobium tin. Rigetti's Frequency and Variable control architecture allows for the creation of complex quantum circuits.

WELCOME TO CLASSIQ
Whether you need 10 or 11,000 qubits, Classiq lets you model, simulate and analyse quantum circuits in a fraction of the time previously required.

Quantum Virtual Machine
Availability of Quantum Virtual Machines
Develop your own quantum code with Classiq, or learn more about the QVM how-to-build your own solution (QVM how-to) to create your private quantum virtual machine.

Access Harmony And Aria On The IonQ Quantum Cloud Today
Get Your Private Cloud Today | Get Access →

But what would you do with them?

Course learning outcomes

Core goal: **learn how to program quantum computers** in a hands-on, software-focused setting.

- Describe the societal importance and implications of quantum computing
- Explain the theory and principles behind gate-model quantum computing
- Outline and describe the operation of core quantum algorithms
- Implement basic and research-level quantum algorithms using Python and PennyLane

In this course you will implement everything you learn!

See syllabus and GitHub repo for detailed list of learning outcomes.

Covered in this course

Divided into 5 modules:

1. Basic elements of quantum computation
2. Oracle-based algorithms, complexity, and quantum resources
3. Quantum Fourier transform-based algorithms
4. Simulating physical systems (Hamiltonian simulation)
5. Characterizing noise in quantum systems

Covered in this course

Divided into 5 modules:

1. Basic elements of quantum computation
2. Oracle-based algorithms, complexity, and quantum resources
3. Quantum Fourier transform-based algorithms
4. Simulating physical systems (Hamiltonian simulation)
5. Characterizing noise in quantum systems

Differences from previous years:

- Each module ends with a mostly hands-on lecture
- Less emphasis on variational algorithms
- New content in modules 4 and 5

Not covered in this course

- How the hardware actually works
- Non-gate-model quantum computing
 - adiabatic quantum computing
 - quantum annealing
 - measurement-based quantum computing
 - continuous-variable quantum computing
- Advanced theoretical topics (quantum information, complexity theory, quantum error correction)

Logistics

My info:

Office:	KAIS 3043
Office hrs:	M 13:00-14:00, Th 14:30-15:30, or by appointment. Open-door policy after 12:00 weekdays
Email:	olivia@ece.ubc.ca
GitHub:	glassnotes

TA info: see syllabus and Piazza

All lecture slides/demos will be posted on the course GitHub:

<https://github.com/glassnotes/CPEN-400Q>

(Find previous years' materials under "Releases" section)

Textbook

Primary resource is the **free, online** Xanadu Quantum Codebook.
Use to learn about the content, PennyLane, and do practice
programming exercises to supplement class work.



XANADU QUANTUM CODEBOOK

codebook.xanadu.ai

Secondary (optional) resource: *Introduction to Quantum Computation and Quantum Information*, Nielsen and Chuang.

Assignments and grading

Five components:

- 20% technical assignments
- 10% “quantum literacy” assignments
- 10% in-class quizzes
- 20% in-class midterm exam
- 40% final group project

Assignments and grading, cntd.

20% technical assignments (~10):

- Done on PrairieLearn
- Mix of pen-and-paper and programming problems
- Includes hands-on group activities from class
- May discuss with classmates but what you submit must be your own work (and cite your collaborators!)

Purpose is to gain deeper understanding of content learned in class, and apply it to new situations.

Assignment 0 available today (review of linear algebra, Python, NumPy, and introduction to PrairieLearn).

Assignments and grading, cntd.

10% quantum literacy assignments (3-4):

- Distributed on PrairieLearn
- Done individually, no collaboration unless otherwise stated
- Activities may include:
 - critical analysis of media articles or videos
 - small creative writing projects
 - group discussions or debates

Purpose is to explore the ethical and societal implications around construction and use of quantum computers, and help dispel hype.

You may submit one assignment (of either type) up to 24hrs late, with advance notice.

Assignments and grading, cntd.

10% quizzes (10):

- Starting next Monday; done at start of class in PrairieLearn
- Done individually, but open book/notes/docs
- Includes math and/or programming problems and conceptual questions; based on previous week's lecture material

Purpose is for reviewing content.

Your lowest quiz grade will be automatically dropped.

Assignments and grading, cntd.

20% midterm:

- Wednesday 31 January during lecture
- closed-book and closed-notes
- pen-and-paper only; no programming

Purpose is to test knowledge of core content before studying advanced algorithms.

Assignments and grading, cntd.

40% final project:

- Implement the algorithms in a research paper and reproduce the results.
- Work in groups of 4
- Two components:
 - Fully-documented software implementation
 - Companion report
- Use a different quantum programming framework if you like

More details (rubric, grading, papers, etc.) after midterm.

Module 1: basic elements of quantum computation

Module 1 learning outcomes

Learning outcomes:

- perform quantum computations using Dirac notation and matrix algebra
- express quantum computations using quantum circuits
- program quantum circuits in PennyLane
- use the Bloch sphere to represent states and the action of quantum operations
- list and define the core set of elementary quantum operations
- define and give examples of entangled states
- compute the result of measurements on one or more qubits in multiple bases
- use Bell basis measurements to implement superdense coding and teleportation
- describe the structure of variational quantum algorithms

Today

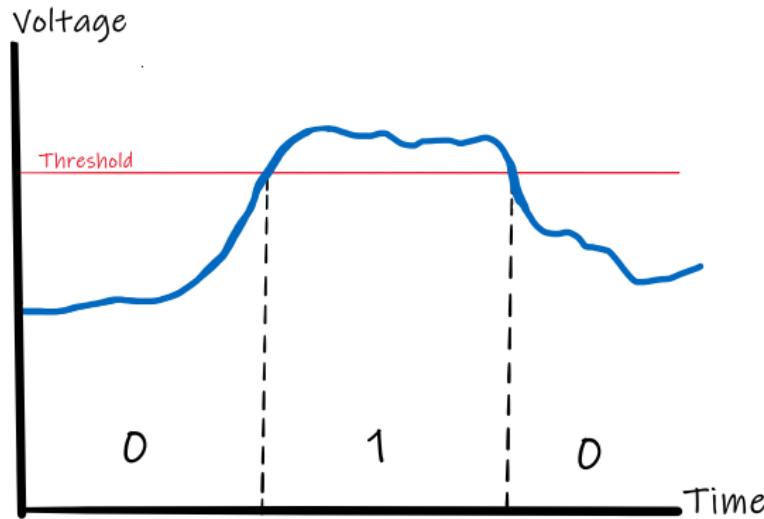
Learning outcomes:

1. Define quantum computing, and explain its societal importance
2. Define a *qubit* both conceptually and mathematically
3. Simulate quantum computing on a single qubit

Bits

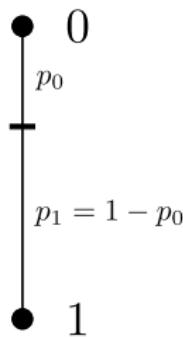
All computation in our computers today is done with bits.

The state of a bit is *either* 0 or 1.



Probabilistic bits

The value of a bit can be represented by a probability distribution.



Example: A coin flip results in Heads (1) or Tails (0).

Example: Is it raining? Yes (1) or No (0)

Probabilistic bits

These probability distributions can evolve over time.

Exercise:

- If rain today, 70% probability it will rain tomorrow.
- If no rain today, 20% probability it will rain tomorrow.

Suppose it is raining today. What is the % chance it will rain two days from now?

Probabilistic bits

Represent the system as a 2-dimensional real-valued vector (\mathbb{R}^2)

Elements necessarily sum to 1. More compactly:

We call \vec{r} and \vec{n} basis vectors.

Probabilistic bits

- Rain today, 70% probability it will rain tomorrow.
- No rain today, 20% probability it will rain tomorrow.

How does this affect our basis vectors?

More compactly,

Probabilistic bits

To solve our problem, check what happens tomorrow:

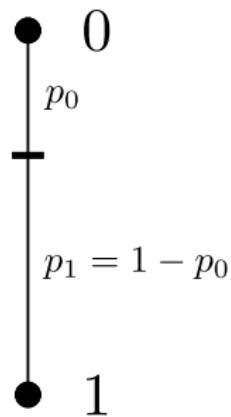
Then the day after:

Note that these operations are linear, and we can solve without explicit matrix multiplication:

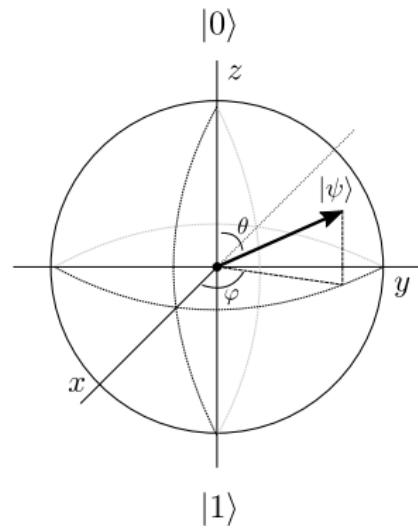
Qubits

Extension of probabilistic bits to 2-level quantum systems:

- 0

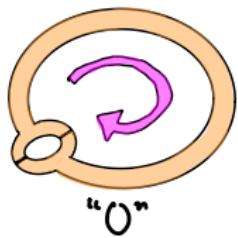


- 1

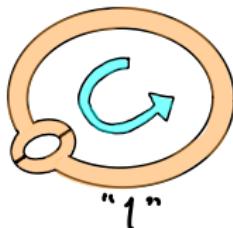


Qubits

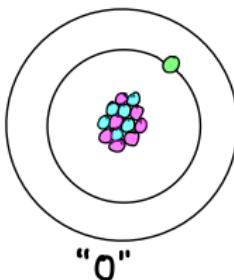
Quantum computers work by manipulating **qubits**.



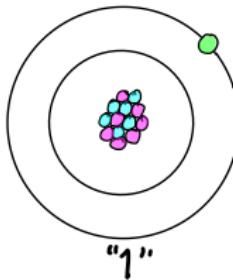
"0"



"1"



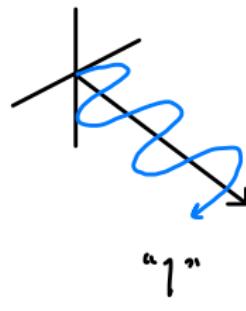
"0"



"1"



"0"



"1"



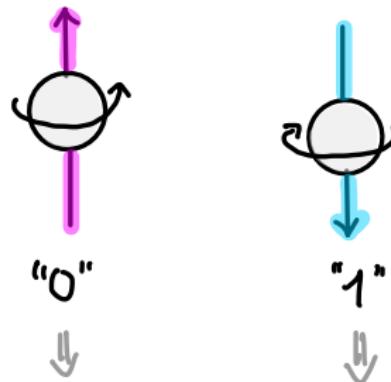
"0"



"1"

Mathematical representation of qubits

The vector space where qubits live is called **Hilbert space**.



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

These two vectors together form the **computational basis**.

The notation $|\cdot\rangle$ is called Dirac, or **bra-ket notation**. $|\cdot\rangle$ is a ket.

Mathematical representation of qubits

Qubit states are a linear combination of these basis states; coefficients can be *complex*.

Diagram illustrating the mathematical representation of a qubit state. A 2D coordinate system is shown with two basis vectors originating from the origin:

- The horizontal axis is labeled $(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}) = \hat{e}_0$ in red.
- The vertical axis is labeled $(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}) = \hat{e}_1$ in blue.

A vector v is shown originating from the origin, representing a linear combination of the basis states:

$$v = c_0 \hat{e}_0 + c_1 \hat{e}_1, \quad c_0, c_1 \in \mathbb{R}$$

⇒

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \alpha, \beta \in \mathbb{C}$$

The coefficients α and β are called **amplitudes**. “Valid” qubit state vectors have unit length:

Such states are called **normalized**.

Mathematical representation of qubits

Exercise: is $|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}e^{0.2i}|1\rangle$ a valid quantum state?

Solution: check that $|\alpha|^2 + |\beta|^2 = 1$.

Mathematical representation of qubits

A qubit in a linear combination

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

is in a **superposition** of the two basis states.

Superposition is one of the unique features that makes quantum computing very powerful.

Note: a qubit in superposition is not "*in both states at the same time*"!

Using qubits for computation

1. Prepare qubits in a **superposition**
2. Apply **operations** that **entangle** the qubits and manipulate the amplitudes
3. **Measure** qubits to extract an answer
4. Profit

Using qubits for computation

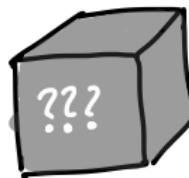
1. Prepare qubits in a **superposition**
2. Apply **operations** that **entangle** the qubits and manipulate the amplitudes
3. **Measure** qubits to extract an answer
4. Profit

...how do we do this?

Manipulating qubits

Manipulating a qubit changes its state:

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



$$|\Psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$$

We need a mechanism for sending valid quantum state vectors to other valid quantum state vectors.

Manipulating qubits: unitary operations

Single-qubit states are manipulated by 2×2 unitary matrices. A matrix U is unitary if

The \dagger means “conjugate transpose”:

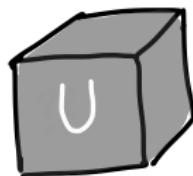
Unitary operations, or **gates**, are **reversible**: if we apply U , we can undo it by applying U^\dagger .

Manipulating qubits: unitary operations

Unitary operations preserve the length of vectors, i.e., *maintain the normalization of qubit states.*

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

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



$$UU^\dagger = U^\dagger U = 1L$$



$$|\Psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$$

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

Manipulating qubits: unitary operations

Unitary operations act *linearly* on superpositions:

This is a key contributor to the power of quantum computing!

Example: X

The matrix

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

is unitary.

Exercise: Determine what it does by evaluating its action on the basis states.

Example: X

Exercise: What does X do to the state

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

Example: H

Perhaps the most important unitary in quantum computing is the Hadamard gate (H):

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

This gate creates a *uniform superposition*:

Example: H

The Hadamard is also self-inverse:

Similarly, $H|-\rangle = |1\rangle$.

Example: Z

The gate

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

does something a little different.

Apply to basis states:

Just changes the sign. (This will be important later!)

Single-qubit gates: applying sequences of gates

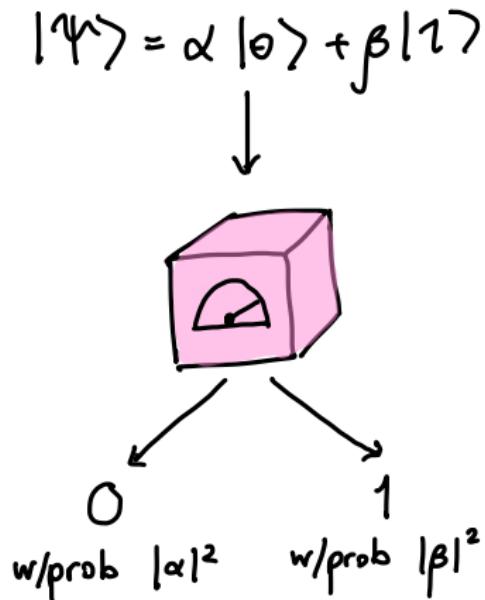
We apply *products* of single-qubit operations to represent performing gates in sequence.

Exercise: Apply H , then Z , then X to $|0\rangle$ without doing any matrix multiplication.

Measuring qubits

Manipulating qubits changes their amplitudes. These determine probability of observing the qubit in $|0\rangle$ or $|1\rangle$ when we measure (it's **probabilistic!**).

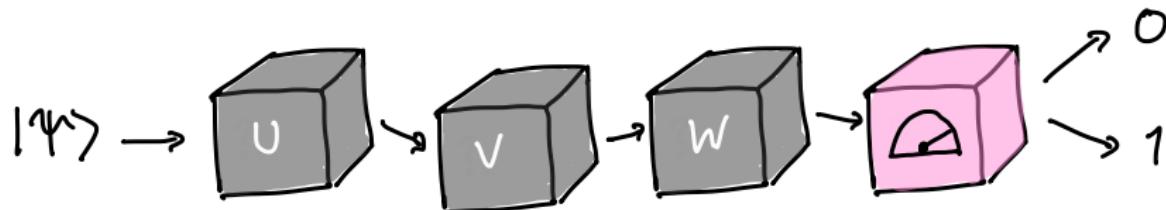
Running an algorithm and measuring gives us a single bit of information (0 or 1). To get more, we need to run the algorithm multiple times (multiple **shots**).



Quantum computing

Quantum computing is the act of manipulating the state of qubits in a way that represents solution of a computational problem:

1. Prepare qubits in a **superposition**
2. Apply **operations** that **entangle** the qubits and manipulate the amplitudes
3. **Measure** qubits to extract an answer
4. Profit



Let's simulate this using NumPy.

Programming quantum computers

That was tedious—let's never do that again.

Let's use some real quantum software instead: there is a fantastic ecosystem of open-source tools.



Cirq



Qiskit



TensorFlow Quantum



```
import pennylane as qml

H = qml.Hamiltonian(...)

dev = qml.device('default.qubit', wires=2)

@qml.qnode(dev)
def quantum_circuit(params):
    qml.RY(params[0], wires=0)
    qml.RY(params[1], wires=1)
    qml.CNOT(wires=[0, 1])
    qml.RY(params[2], wires=0)
    return qml.expval(H)

quantum_circuit([0.1, 0.2, 0.3])
```

PennyLane

We will use PennyLane, a Python framework developed by **Xanadu** (a Toronto-based quantum startup).

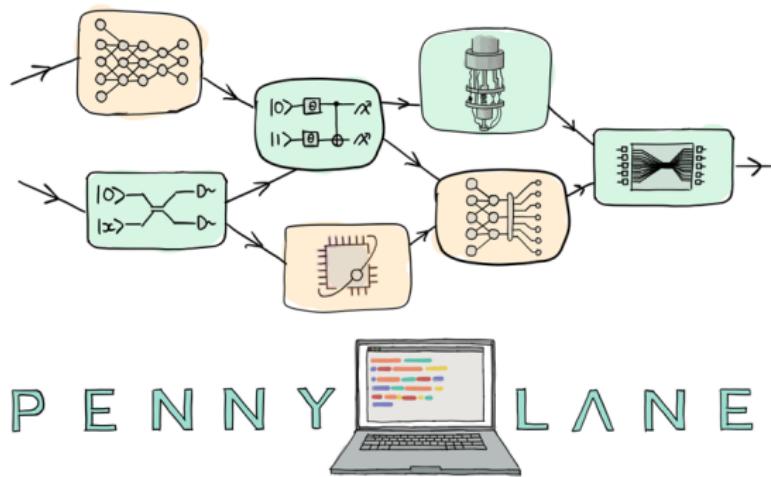


Image credit: <https://pennylane.ai/>

GitHub: <https://github.com/PennyLaneAI/PennyLane>

Documentation: <https://pennylane.readthedocs.io/en/stable/>

Demonstrations: <https://pennylane.ai/qml/demonstrations.html>

Discussion Forum: <https://discuss.pennylane.ai/>

Its key use case is **differentiable quantum programming** and quantum machine learning; it is also a valuable tool for quantum computing algorithms and applications.

PennyLane

```
def ket_0():
    return np.array([1.0, 0.0])

def apply_ops(ops, state):
    for op in ops:
        state = np.dot(op, state)
    return state

def measure(state, num_samples=100):
    prob_0 = state[0] * state[0].conj()
    prob_1 = state[1] * state[1].conj()

    samples = np.random.choice(
        [0, 1], size=num_samples, p=[prob_0, prob_1]
    )
    return samples

H = (1/np.sqrt(2)) * np.array([[1, 1], [1, -1]])
X = np.array([[0, 1], [1, 0]])
Z = np.array([[1, 0], [0, -1]])

input_state = ket_0()
output_state = apply_ops([H, X, Z], input_state)
results = measure(output_state, num_samples=10)

print(results)
[1 0 0 1 0 0 0 1 1 0]
```

Sample NumPy

```
dev = qml.device('default.qubit', wires=1, shots=10)

@qml.qnode(dev)
def my_circuit():
    qml.Hadamard(wires=0)
    qml.PauliX(wires=0)
    qml.PauliZ(wires=0)
    return qml.sample()

print(my_circuit())
[1 0 1 0 1 1 0 1 0 0]
```

PennyLane

Recap

Today's learning outcomes were:

1. Define quantum computing, and explain its societal importance
2. Define a *qubit* both conceptually and mathematically
3. Simulate quantum computing on a single qubit

For next time

Content:

- Getting started with PennyLane
- Quantum circuits
- More unitary operations

Action items:

1. Sign up for Xanadu Quantum Cloud, or configure your programming environment with PennyLane v0.34 (come to office hours if you need help) **NOTE: 0.34 being released tomorrow**
2. Follow instructions on Canvas to sign up for PrairieLearn and Piazza
3. Work on Assignment 0

Recommended reading: Codebook nodes I.1-I.8.

Closing survey

Go to <https://www.menti.com> and type in the code.

