Teaching quantum computing through quantum software

Olivia Di Matteo

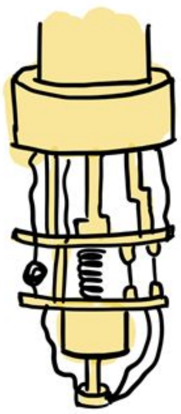
CAP Congress, 8 June 2022

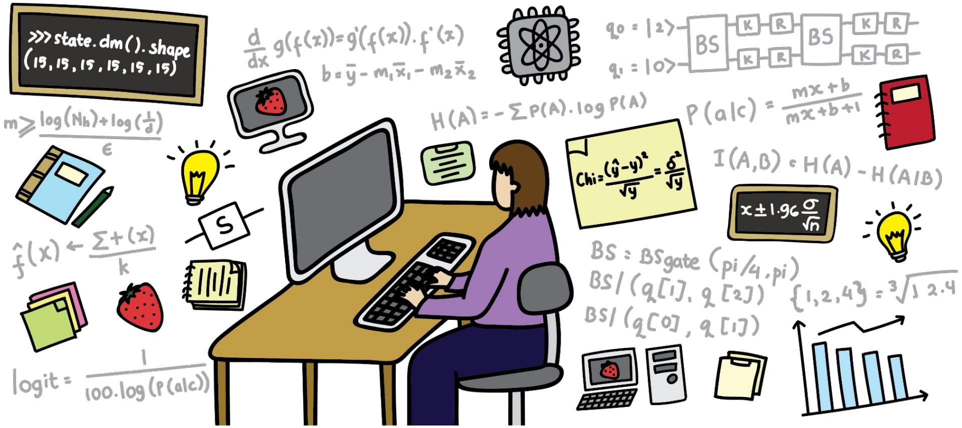
THE UNIVERSITY OF BRITISH COLUMBIA
Electrical and Computer Engineering
Faculty of Applied Science

Quantum computing education

Traditionally, quantum computing courses have:

- Targeted primarily **graduate students** (often physicists)
- Focused mostly on the **underlying theory**
- Not actually taught how to *program* a quantum computer





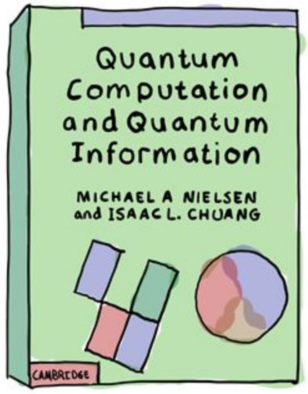


Image credits: https://residency.xanadu.ai/, https://pennylane.ai/qml/whatisqml.html

Quantum computing education

The QC industry is growing. We need more courses that:

- Target undergraduates
- Target non-physicists
- Focus on *actually writing software and algorithms for quantum computers*, using industry-relevant tools

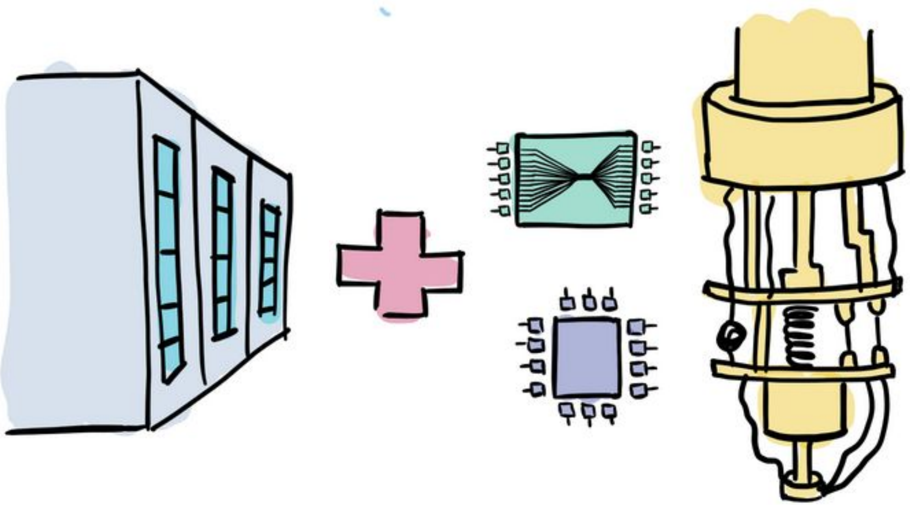


Image credit: https://pennylane.ai/qml/whatisqml.html

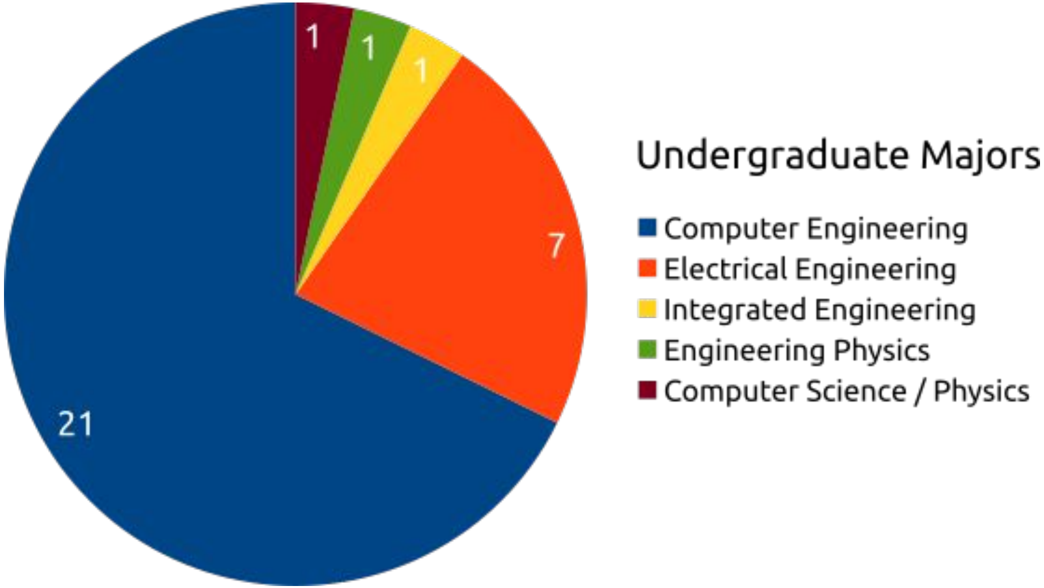
Teaching quantum computing through quantum software

Overview of CPEN 400Q: *Gate-model quantum computing*

- Course content
- ✦ Teaching demo ✦
- Assessment
- Exploring background-specific content design

CPEN 400Q

- ECE department at UBC, Jan-Apr 2022
- First dedicated undergraduate quantum computing course at UBC
- 31 undergrads, and 1 physics grad student
- First month on Zoom, then (mostly) in-person



Slides & demos available open source: https://github.com/glassnotes/CPEN-400Q

CPEN 400Q

Course learning outcomes

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

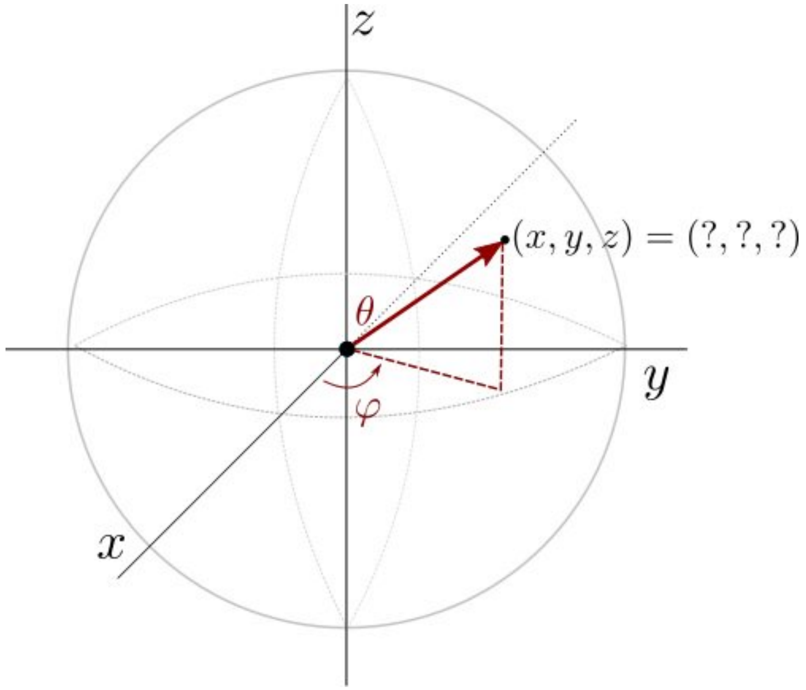
- Describe the societal importance and implications of quantum computing
- Explain the theory and principles behind gate-model quantum computing
- Describe the operation of key quantum algorithms
- Implement basic and research-level quantum algorithms using Python and PennyLane

In this course you will implement everything you learn!

Slides & demos available open source: https://github.com/glassnotes/CPEN-400Q

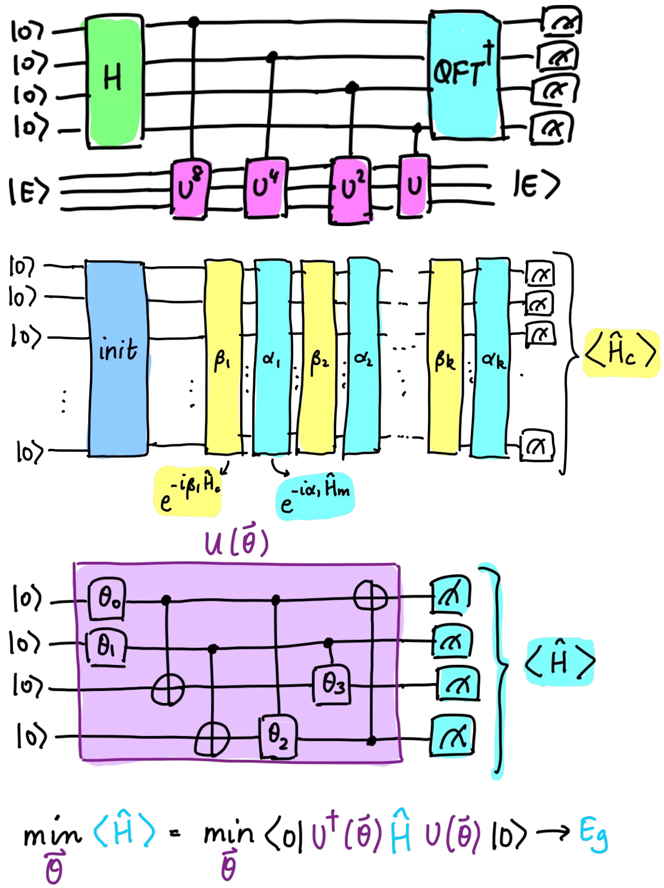
Content: define and compute basic concepts

- Write quantum states, compute action of gates and results of measurements
- Define superposition and entanglement
- Describe which algorithms give quantum speedups and which don't
- Express quantum computations in the quantum circuit model
- Describe the operation and structure of key quantum algorithms...



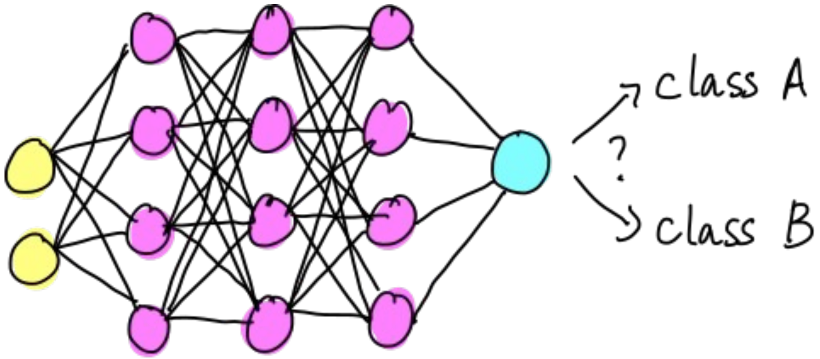
Content: implement core algorithms in software

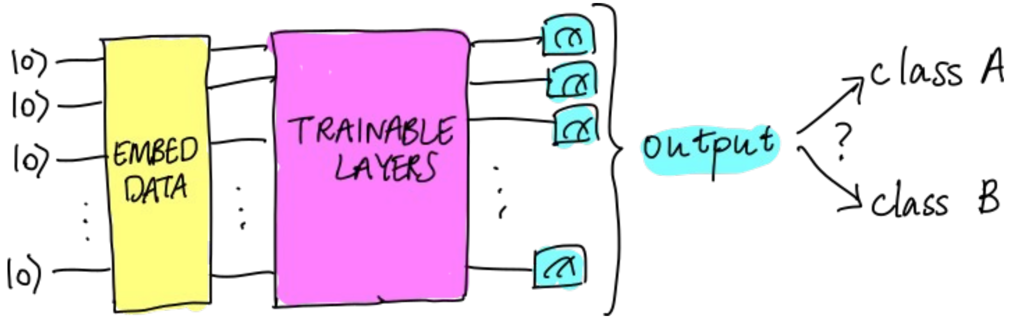
- Quantum teleportation
- Deutsch’s algorithm
- Grover’s search algorithm
- Quantum Fourier transform
- Quantum phase estimation
- Shor’s algorithm
- Variational quantum classifier
- Variational quantum eigensolver
- Quantum approximate optimization algorithm
- Basic Hamiltonian simulation



Content: teaching methods

- Half annotation of slides on iPad, half live coding
- *Lots* of pictures
- Tried to leverage concepts and algorithms they were familiar with before showing analogous quantum algorithms





Content: the first weeks

- Students did not know quantum mechanics, but they *did* know how to program
- Manually programmed the components of simple quantum computation before jumping to quantum software
- All coding afterwards was using **PennyLane**

Demo 3: measurement

```python
def measure(state, shots=50):
    prob_0 = np.abs(state[0]) ** 2
    # prob_0 = state[0] * state[0].conj()

    prob_1 = np.abs(np.vdot(ket_1(), state)) ** 2

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

```python
some_state = apply_multiple_U([H, Z, X], ket_0())
```

```python
measure(some_state, shots=20)
```

```
array([1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1])
```

```python
def quantum_algorithm():
    state = superposition(1j/2, np.sqrt(3)/2)
    state = apply_multiple_U([Z, H, X, H, X, H], state)
    return measure(state)
```

```python
quantum_algorithm()
```

```
array([1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1,
       0, 1, 0, 0, 0, 1])
```

Content: the first weeks

- PennyLane is an open-source quantum software framework; development led by startup Xanadu in Toronto
- Write and run quantum circuits just like Python functions!
- Teaching/learning through software allowed students to apply a familiar language and tools to new concepts

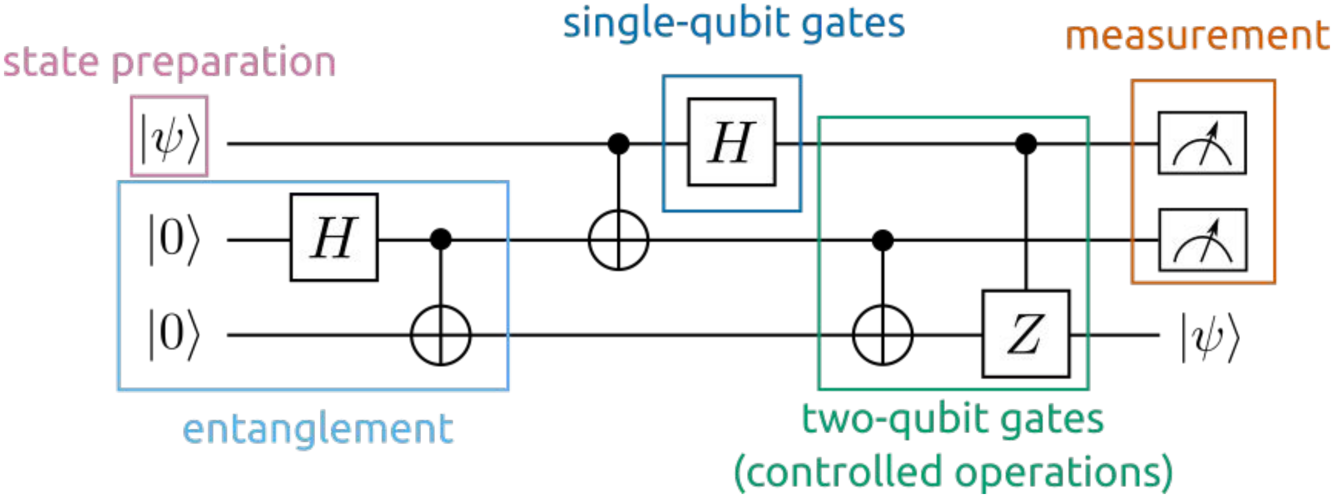
Demo 2: basis rotation

```
dev = qml.device('default.qubit', wires=1, shots=20)

def basis_rotation():
    qml.Hadamard(wires=0)
    qml.S(wires=0)

@qml.qnode(dev)
def circuit(x, y, z):
    qml.RX(x, wires=0)
    qml.RY(y, wires=0)
    qml.RZ(z, wires=0)

    # Rotate back to computational basis
    qml.adjoint(basis_rotation)()

    return qml.sample()
```

```
circuit(0.1, 0.2, 0.3)
```

Content: the first weeks

Teleportation is a great quantum algorithm with which to end the first few weeks:



Access to quantum software helps us express this in terms of smaller algorithmic building blocks that can be combined and reused.

Image credits: https://codebook.xanadu.ai/I.15

Hands-on demo: quantum teleportation

To code along go to:

https://bit.ly/38PVFwO

Assessment

Three components for grading:

- Computational assignments (30%)
- Weekly quizzes (20%)
- Final project (50%)

(No exams!)

Assessment: assignments

- Consisted of solving programming problems
- Distributed and submitted through GitHub
- Grading scripts with test cases were usually provided
- Points for comments, formatting, and source/collaborator citation

Issues:

- Convoluted submission instructions
- Tricky to balance autograding and manual providing of feedback
- Tried to sneak in extra concepts: some liked this, some didn't.

Next time: GitHub classroom?

Assessment: quizzes

- Distributed and submitted through GitHub
- Individual, but can consult documentation and notes
- 4 hour time window to complete

... this didn’t really work. Setup too convoluted for such a short time window.

Next time: in-class quizzes in pairs.

```python
import pennylane as qml


def quiz_1(x, y):
    """Write and execute a QNode that implements the following circuit:

     0: --H-------┌C---|
     1: --RX(x)---|C---|
     2: --RY(y)---└X---| ⟨X⟩

    Args:
        x (float): Angular parameter for X rotation
        y (float): Angular parameter for Y rotation.

    Returns:
        float: The analytical expectation value of X on the final qubit
            obtained after executing your QNode.
    """

    # YOUR CODE HERE

    return
```

Assessment: final project

Replicate the results of a recent research paper in software.

Three (equally-weighted) components:

- Software implementation
- Class presentation (on Zoom) + live coding demo
- Companion report detailed their process and issues

Very challenging, but they did really well!

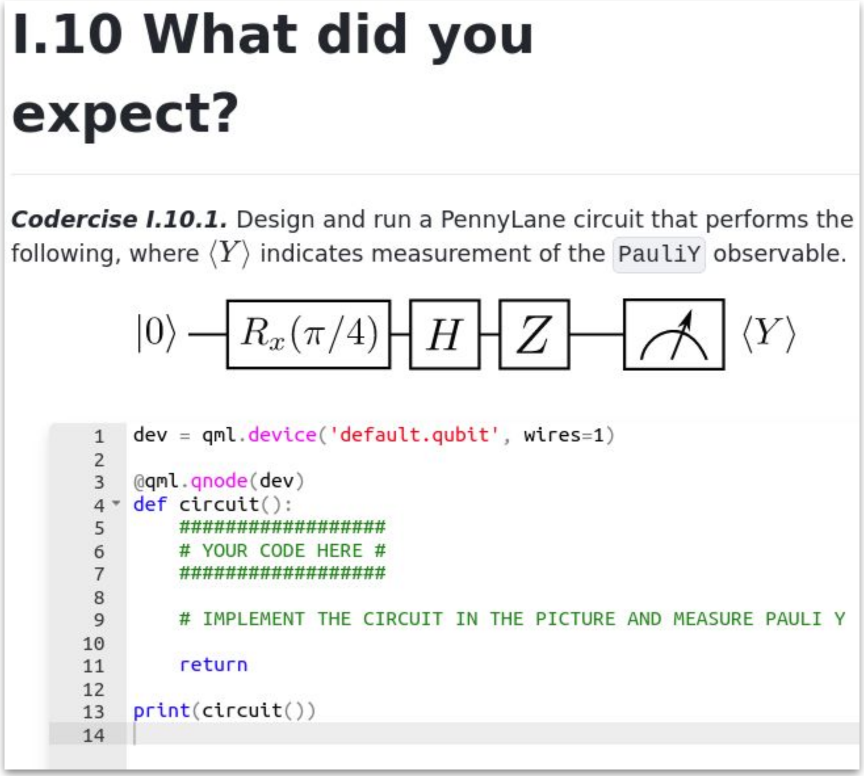
Next time: more checkpoints; permute order of course content...

The textbook

(codebook.xanadu.ai)

Xanadu Quantum Codebook:

- **Free**, **self-paced**, and **hands-on** resource that teaches quantum computing
- Target audience is software developers who know **Python**, and a little bit of **linear algebra**
- Readers **learn by doing** by solving programming exercises.



I.10 What did you expect?

Codercise I.10.1. Design and run a PennyLane circuit that performs the following, where $\langle Y \rangle$ indicates measurement of the `PauliY` observable.

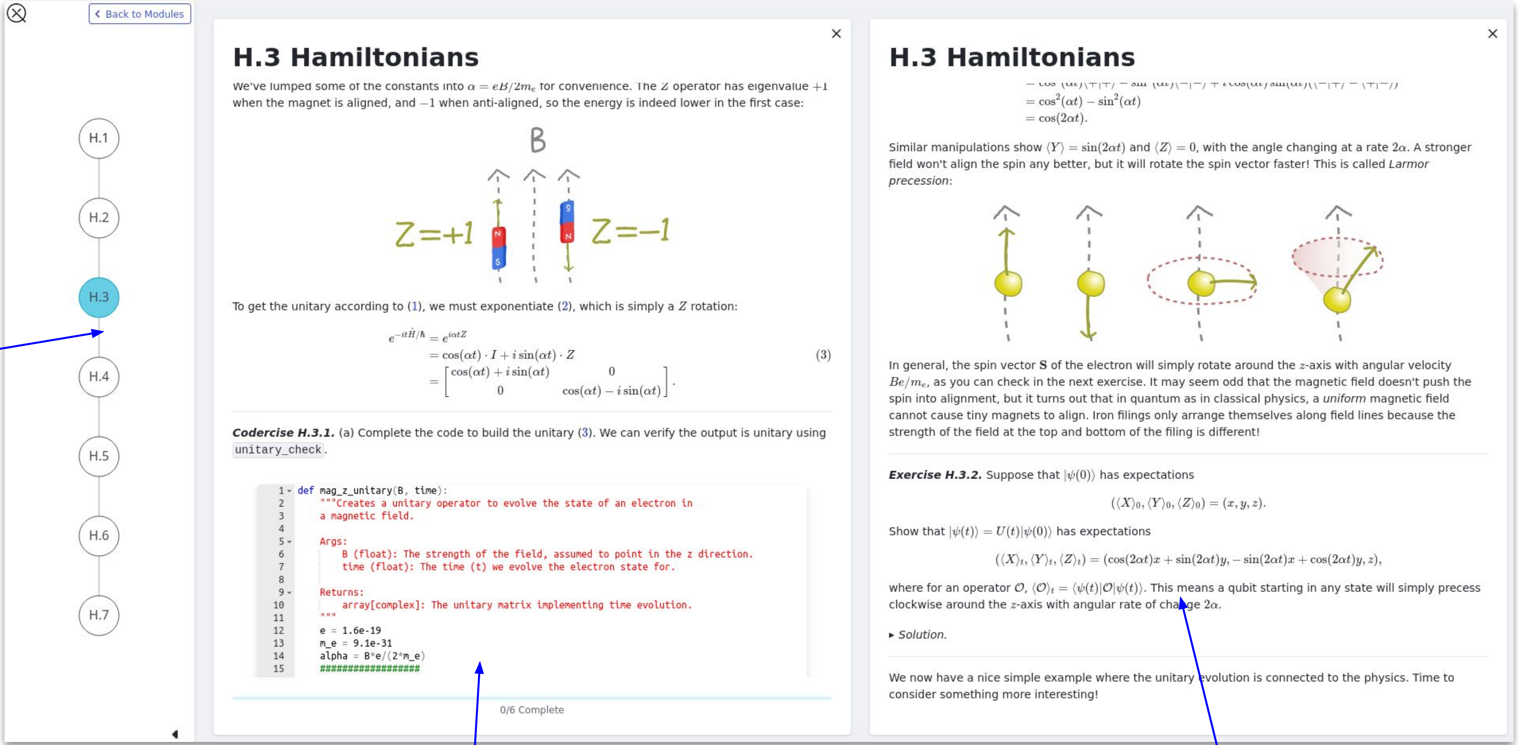
$$|0\rangle - \boxed{R_x(\pi/4)} - \boxed{H} - \boxed{Z} - \boxed{\text{measure}} \; \langle Y \rangle$$

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

@qml.qnode(dev)
def circuit():
    ##################
    # YOUR CODE HERE #
    ##################

    # IMPLEMENT THE CIRCUIT IN THE PICTURE AND MEASURE PAULI Y

    return

print(circuit())

```

The textbook

(codebook.xanadu.ai)

Graph to navigate content

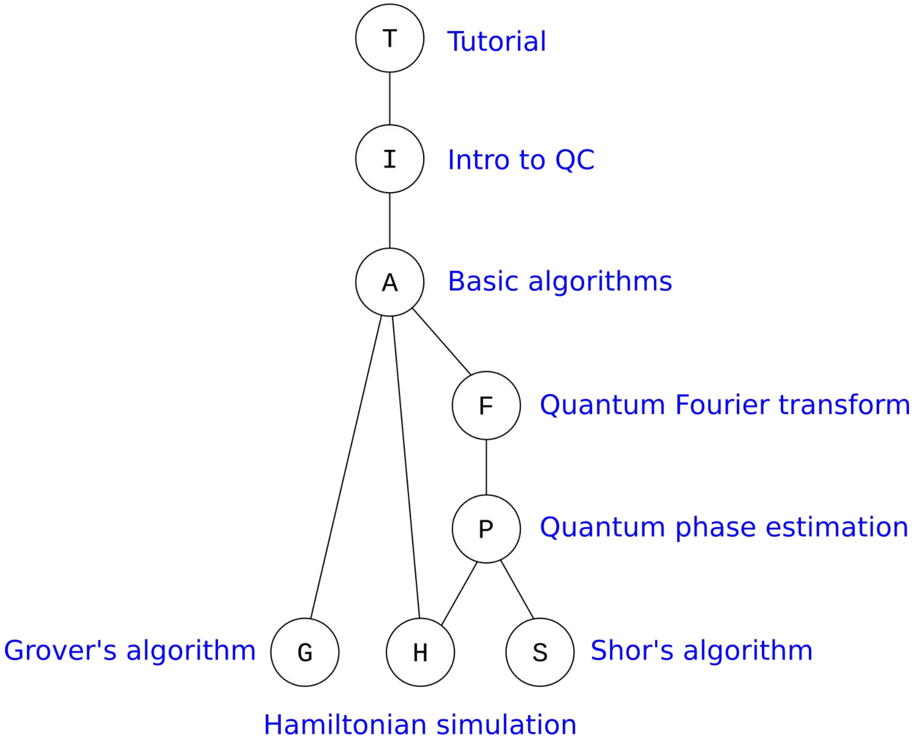


Autograded coding exercises

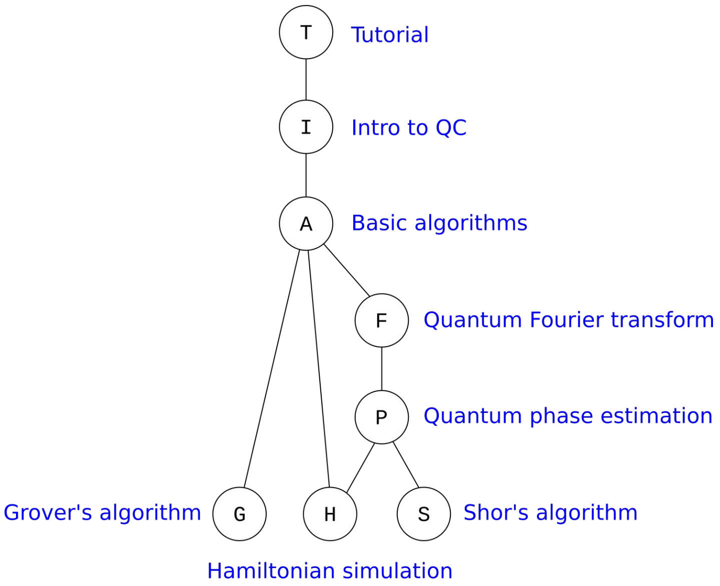
Companion textbook content

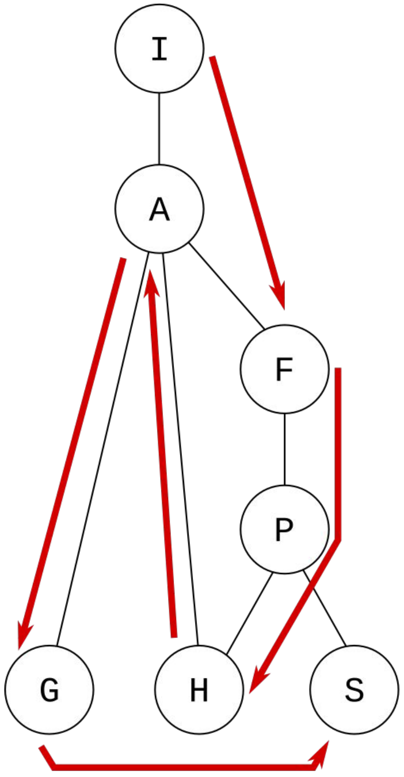
Ordering the material

Codebook content is divided into modules in a non-linear way: choose a path based on your own interests and background.

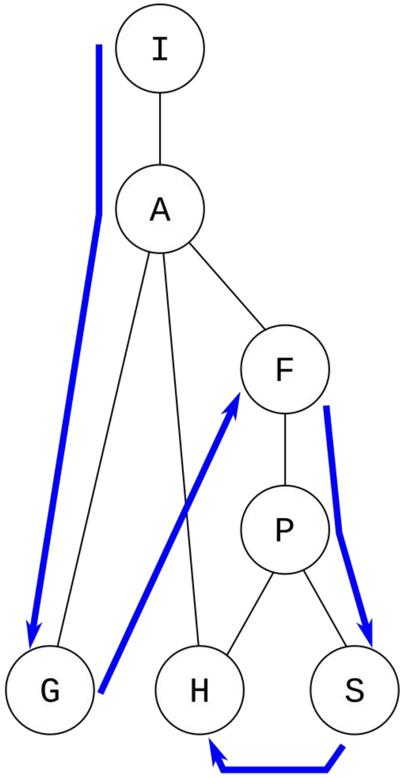


Ordering the material



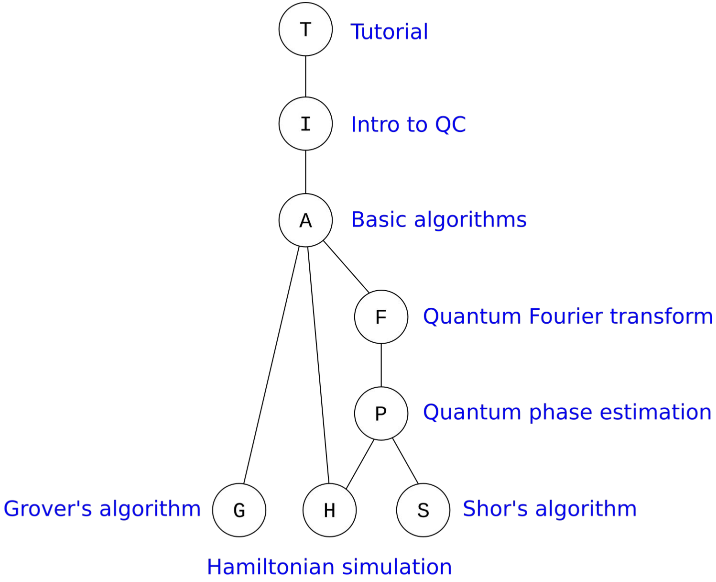


Physics-focused background

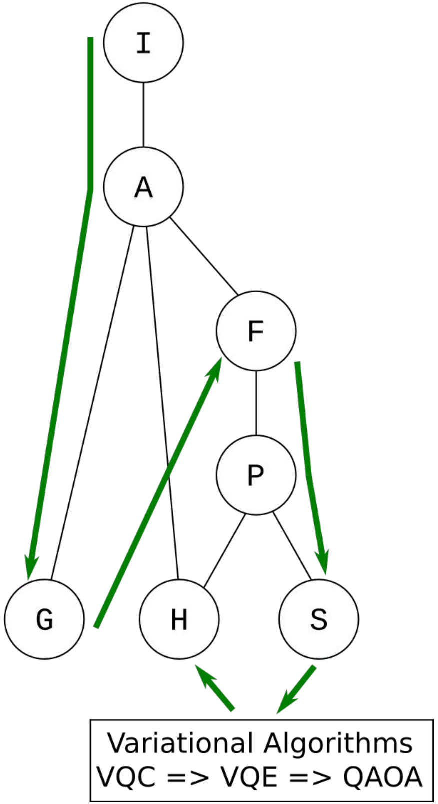


CS-focused background

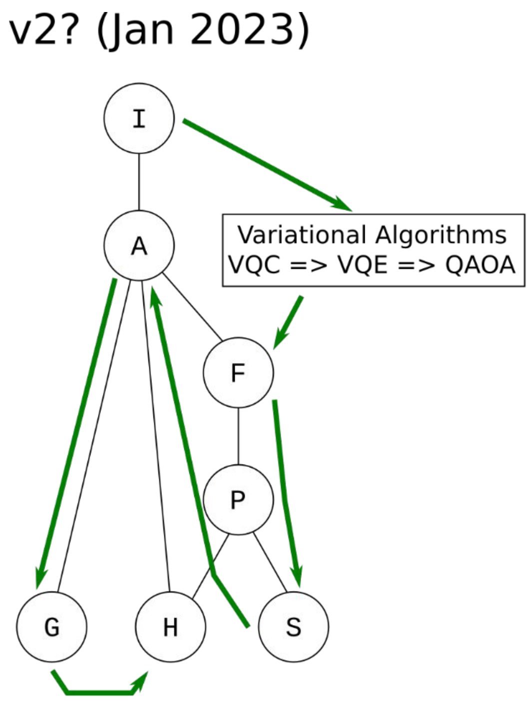
Ordering the material: CPEN 400Q

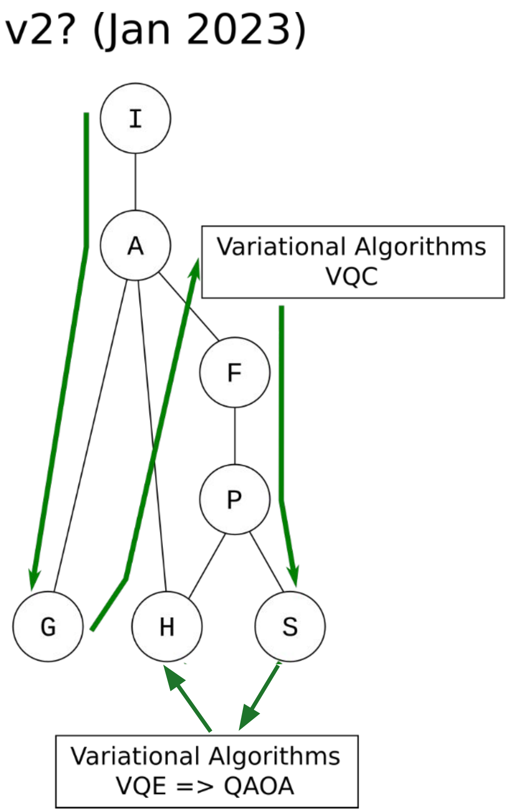


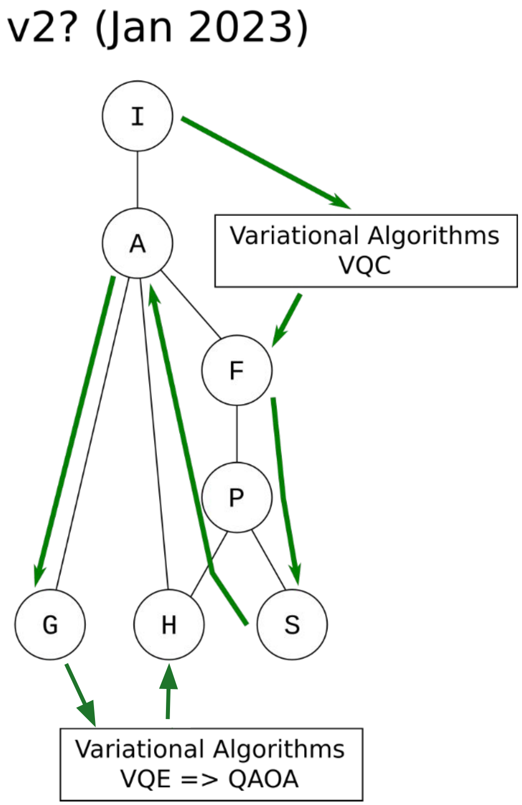
v1 (Jan 2022)



Ordering the material: CPEN 400Q

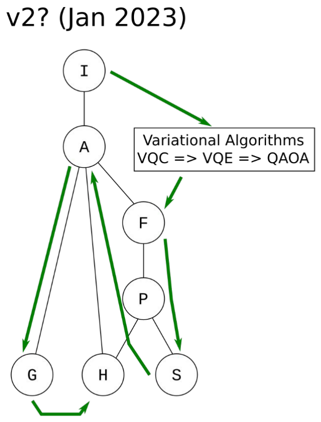


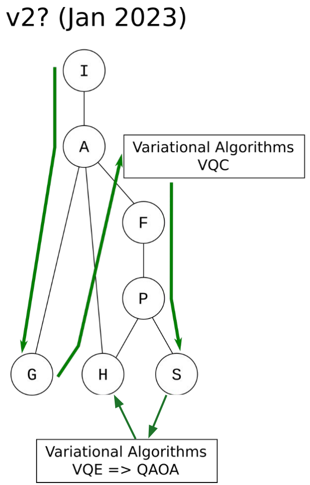


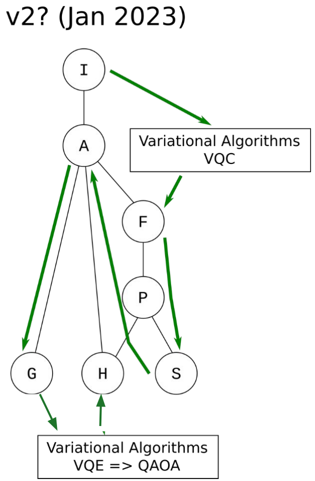


How to learn which order works the best?

- All students are different
- Varying class composition but still mostly CPEN students
- Small class size
- Long time horizon





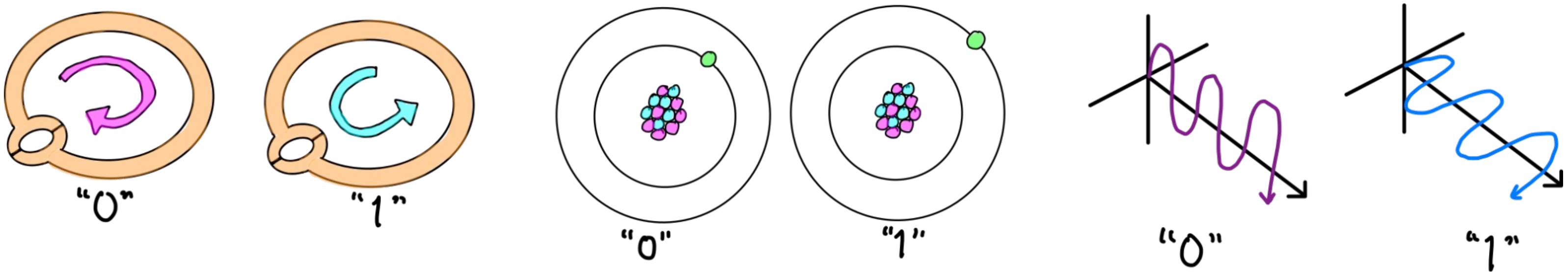


Other content questions:

- What is the right amount of quantum mechanics to teach?
- When to introduce Hamiltonians?
- When/how to facilitate a discussion about ethics?

Takeaways

- Students *love* live coding (even if you make mistakes!)
- Autograded quizzes/assignments are powerful tools but must be wielded wisely
- The order of material must be chosen to suit student backgrounds, but not clear what is the best way to choose it



Thanks & acknowledgments

- To my TA, Gideon Uchehara, for reviewing lectures, testing assignments, and helping out the students
- To the Xanadu team, for providing computing resources for in-class demos, and for fostering a work environment that enabled development of the Codebook
- To all the CPEN 400Q students for their patience and their valuable feedback!