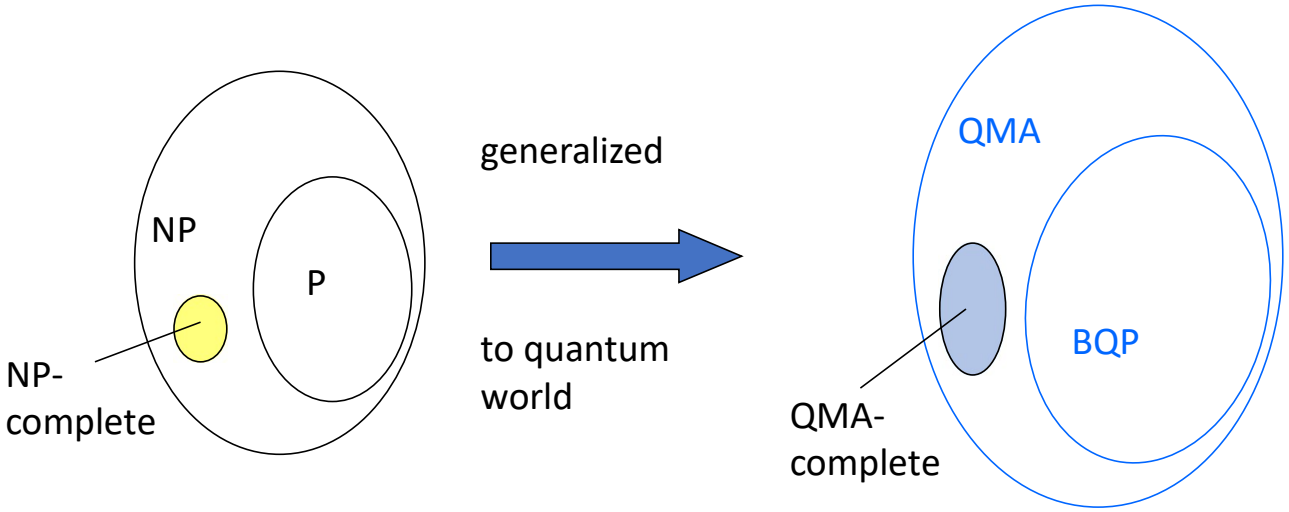
Week 5: Programming through quantum clouds: Computational complexity, Quantum programming on IBM's superconducting quantum computers, including the use of the variational quantum eigensolver (VQE) on quantum chemistry of molecules, quantum approximate optimization algorithm (QAOA) for optimization, hybrid classical-quantum neural network.

Generalizing classical complexity to quantum

- Now if one allows quantum mechanics as part of rules for computation, i.e., suppose we are equipped with a "quantum computer":



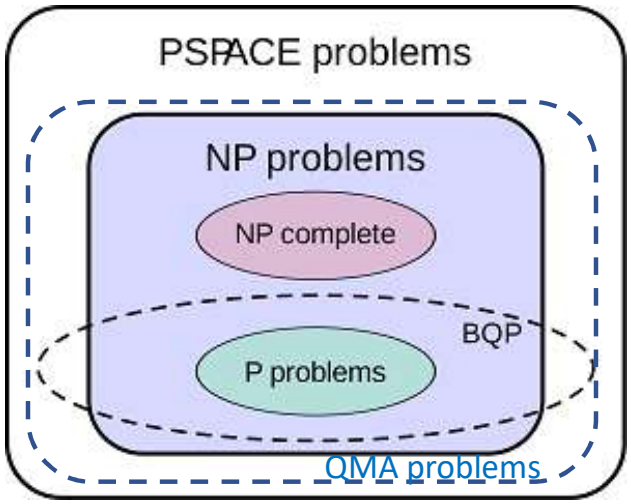
- BQP: class of problems that can be solved in polynomial time of input size by quantum computers
- QMA: class of problems that the solution can be verified in polynomial time by Q computers

Computational complexity

- BQP: Bounded-error Quantum Polynomial

  → BQP is a class of languages $L \subseteq (0, 1)*$, decidable with bounded error probability (say 1/3) by a uniform family of polynomial-size quantum circuit over some universal family of gates

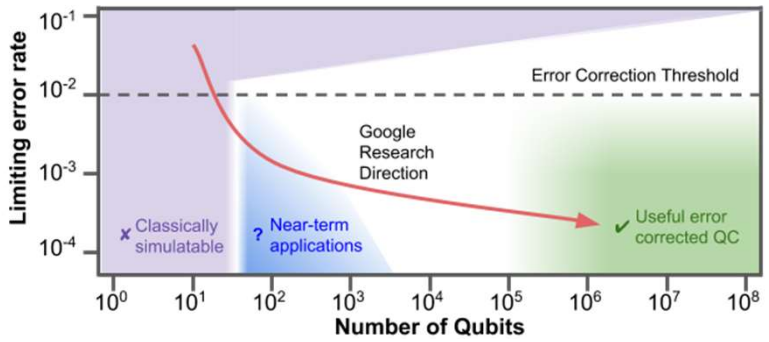
  ➔ BQP is a class of problems Quantum Computers can solve efficiently



Quantum Supremacy

- Demonstrate quantum operation that classical computer cannot simulate efficiently [Preskill…]
  - ⇒ quest for large-scale quantum computing may push physics into a new regime never explored before
  - ⇒ About 50 qubits were the threshold
  - ⇒ Reason why IBM, Intel & Google want to build QC over 50 qubits

- Noisy Intermediate-Scale Quantum (NISQ) devices
  - ⇒ Quantum error correction cannot yet help
  - ⇒ What are the near-term applications?



Variational quantum eigensolver (VQE)

- Variational principle: for a Hamiltonian (i.e. a Hermitian matrix related to the energy of a system) the lowest energy can be approached from above by minimizing over some ansatz

$$E_{\min} = \min_{\substack{\Psi \\ \text{unconstrained}}} \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle} \leq \min_{\vec{\theta}} \frac{\langle\Psi(\vec{\theta})|H|\Psi(\vec{\theta})\rangle}{\langle\Psi(\vec{\theta})|\Psi(\vec{\theta})\rangle}$$

subclass of vectors

- VQE uses certain ansatz for the wavefunction Ψ parametrized by θ's i.e. constructed by a quantum circuit with gates with parameters θ's (thus Ψ normalized, i.e. |Ψ|=1)
  - Use quantum computer and measurement to evaluate the expectation (by repeatedly creating the state Ψ and measuring H)

expectation value → $\langle\Psi(\vec{\theta})|H|\Psi(\vec{\theta})\rangle$

if we can prepare $|\Psi(\vec{\theta})\rangle$ ⇒ perform measurement to estimate H

  - Apply classical optimizer to find the next set of parameters θ'

→ Classical optimization for $\vec{\theta}$

$\vec{\theta}^0$ → $\langle\Psi(\vec{\theta}^0)|H|\Psi(\vec{\theta}^0)\rangle$ → $\vec{\theta}^1$ → ... → $\vec{\theta}^0$ close to minimum

  - Repeat until the expectation value converges within prescribed accuracy

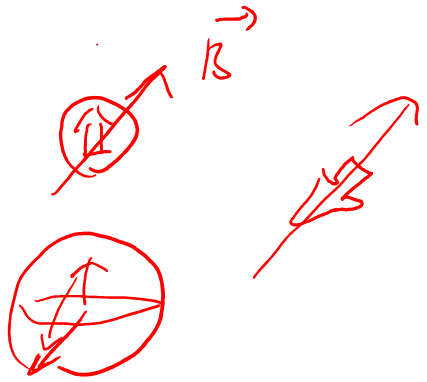
VQE: simple example

- Suppose the Hamiltonian H is

$$H = -B_x\sigma_x - B_z\sigma_z = -\begin{pmatrix} B_z & B_x \\ B_x & -B_z \end{pmatrix}$$

and the wavefunction ansatz is

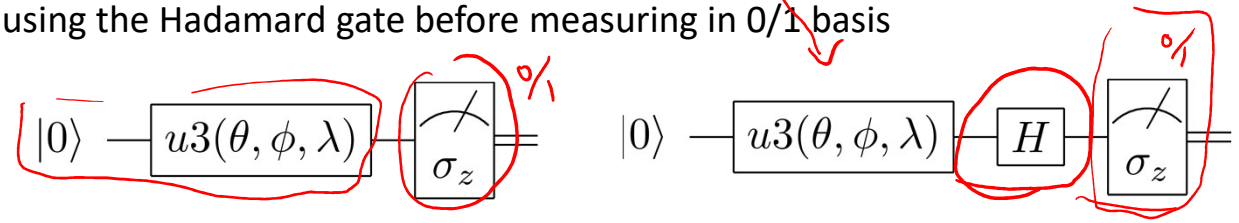
$$|\Psi(\theta,\phi,\lambda)\rangle = u3(\theta,\phi,\lambda)|0\rangle \qquad |0\rangle \text{ —} \boxed{u3(\theta,\phi,\lambda)} \text{—}$$



- To obtain the expectation value, we need to measure separately

$$\langle\Psi|\sigma_z|\Psi\rangle = P_0 - P_1 \qquad \langle\Psi|\sigma_x|\Psi\rangle = P_+ - P_- \quad \text{in +/− basis}$$

The first one is given by the difference between the probabilities of obtaining 0 and 1. The second is similar, but in the +/- basis; can use first rotate to 0/1 using the Hadamard gate before measuring in 0/1 basis



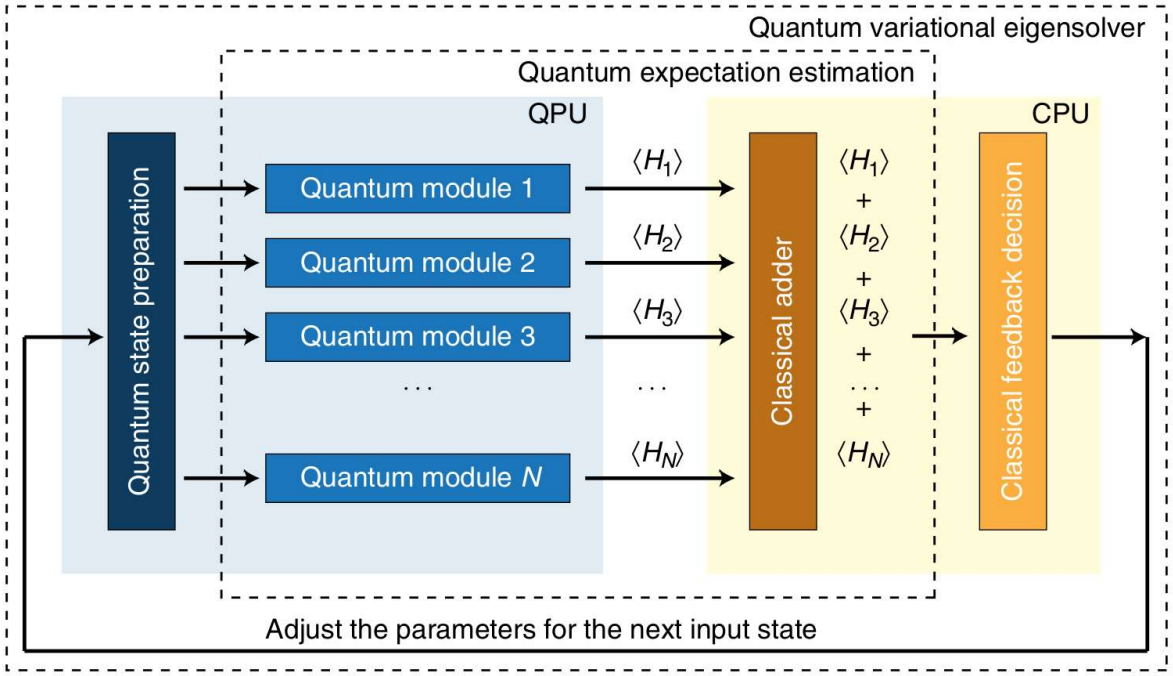
Qiskit implementation

Do Notebook

Summary of VQE

[Peruzzo et al. Nat. Comm. 5, 4213 (2014)]

$$H = \sum_{i=1}^{N} c_i H_i \qquad \text{e.g. } H = -B_x\sigma_x - B_z\sigma_z$$



VQE: multi-qubit and more complicated ansatz

- The RY trial wave function: layers of y rotations with entanglements
- The RYRZ trial wave function: layers of y plus z rotations with entanglements.
- SwapRZ Variational Form: layers of swap plus z rotations with entanglements.

(*note these were in Qiskit 0.19 version but will be deprecated in later versions)