Introduction to the Qiskit Runtime, A Platform for Efficient Quantum-Classical Computation

Jennifer Glick
Jessie Yu

July 11, 2022
8-12 pm CDT
Room 201

github.com/jenglick/scipy22-qiskit-runtime-tutorial

Agenda

Tutorial split into three sections

10-minute breaks between each

Part 1

Quantum Computing and Qiskit

45 min intro

30 min exercise

Part 2

Qiskit Runtime: Primitives & Sessions

35 min intro

40 min exercise

Part 3

Building applications with Qiskit Runtime

15 min intro

45 min exercise

`github.com/jenglick/scipy22-qiskit-runtime-tutorial`

Part 0

Installation and set up

Part I

Intro to quantum computing and Qiskit

Why use a quantum computer?

Are Quantum Computers “Faster”?

$$p * q = N$$

How long does it take to **multiply** 2048-bit integers?

Classical Cost of multiplication: ~ 0.0025s

A. Emerencia, Multiplying huge integers using Fourier transforms (2007).

Quantum Cost of multiplication: ~ 75.0000s

C. Gidney and M. Ekerå, arXiv:1905.09749 (2019).

Are Quantum Computers “Faster”?

$$N = p * q$$

How long does it take to **factor** 2048-bit integers?

Classical Cost of factoring: ~ 4.7 billion CPU years

(2010 RSA-768 bit: approx. 1500 CPU years)

Kleinjung, Thorsten, et. al.; Factorization of a 768-bit RSA modulus; Annual Cryptology Conference; Springer, Berlin, Heidelberg (2010).

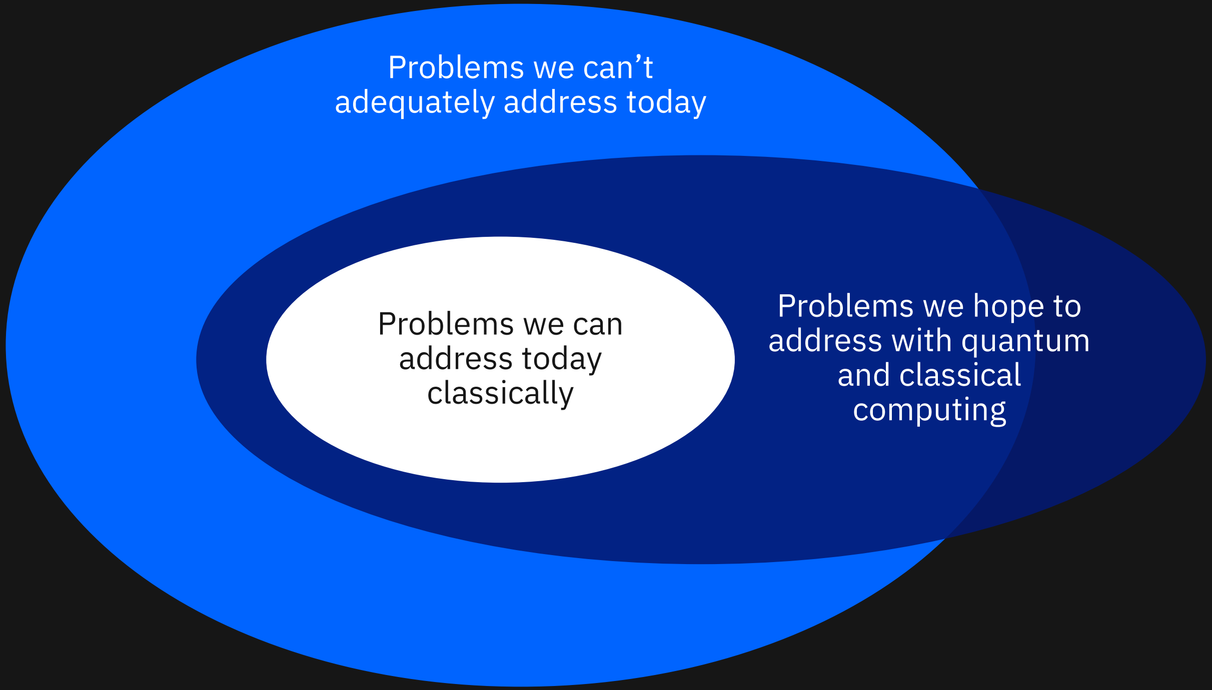
(2019 RSA-795 bit: approx. 4000 CPU years)

www.newscientist.com/article/2226458-number-crunchers-set-new-record-for-cracking-online-encryption-keys/

Quantum Cost of factoring: ~ **8 hours**

C. Gidney and M. Ekerå, arXiv:1905.09749 (2019).

Despite how sophisticated digital “classical ” computing has become, there are many scientific and business problems for which we’ve barely scratched the surface.



Problems suitable for a quantum computer

BQP

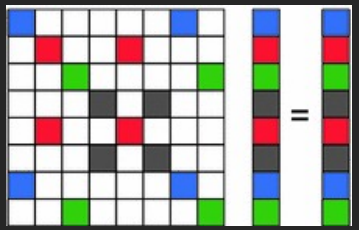
bounded-error quantum polynomial time

Class of problems that admit efficient (polynomial time) quantum algorithms

Large Linear Systems

$$\boldsymbol{Ax = b}$$

2^n



2^n

Linear system solvers

Classification (Machine Learning)

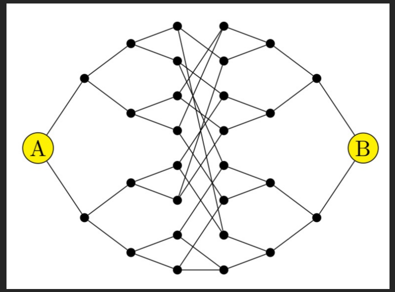
Quantum Simulations

Physics

Chemistry

Materials discovery

Quantum Walks



Graph properties (network flows, electrical resistance)

Search

Problems suitable for a quantum computer

BQP

bounded-error quantum polynomial time

Class of problems that admit efficient (polynomial time) quantum algorithms

Think of problems where the answer depends strongly on details of exponentially many *entangled* degrees of freedom *with structure such that* quantum mechanics evolves to a solution without having to go through all paths, e.g., structure that captures some non-trivial global property.

The complexity theoretic hardness of quantum circuits is the only reason quantum applications will have a quantum advantage.

e.g., IBM used this fact to demonstrate potential *quantum advantage in AI*.

What Makes a Circuit Hard to Simulate?

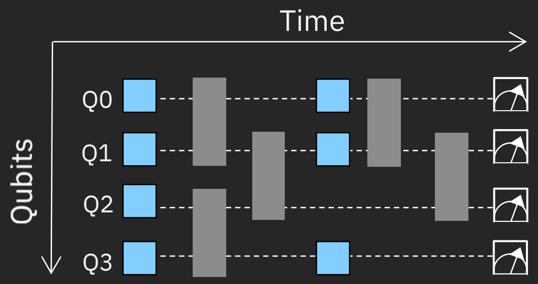
A hard circuit needs:

- a lot of entanglement
- the ability to interfere
- to be algebraically complicated

Near-term quantum hardware is noisy.

Circuits are programmable.

Novel algorithms focused on using short depth circuits that make best use of coherence window.



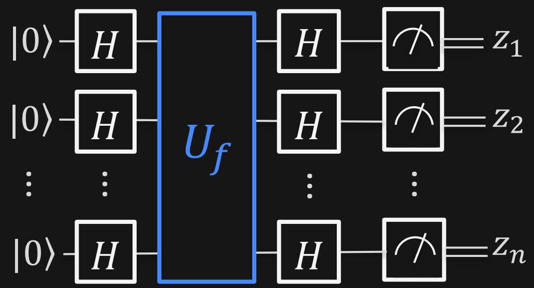
It has been shown there are quantum circuits that are hard to simulate and have short depth.

Bravyi et al. 1704.00690, 1904.01502

What types of circuits are hard?

Sampling from probability distributions that cannot be classically sampled efficiently and accurately.

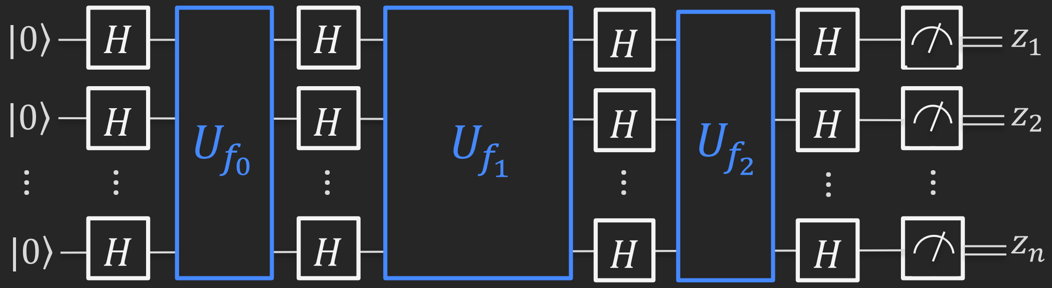
Instantaneous Quantum Polynomial (IQP): quantum circuits constructed from commuting gates.



Extracting hidden features of certain algebraic structures.

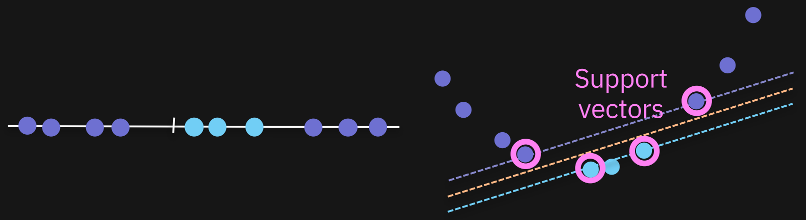
Forrelation: Are two Boolean functions completely independent or is one highly correlated with the Fourier transform of the other?

Exponential query complexity separation between quantum and classical.

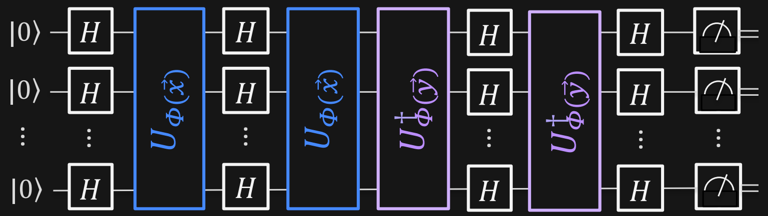


Can we use hard circuits for something valuable?

From Forrelation to Quantum Advantage in AI:



1 Computing kernels in quantum enhanced feature spaces using computationally hard quantum circuits



Havlíček, Córcoles, Temme, et al.
Nature **567**, 209 – 212 (2019)

2 Formal proof quantum kernels ***exist*** with super-polynomial quantum advantage in classification of classical data.

Liu, Arunachalam, Temme
Nature Physics 17, 1013 (2021)

3 Generalized framework of covariant quantum kernels for data with group structure.

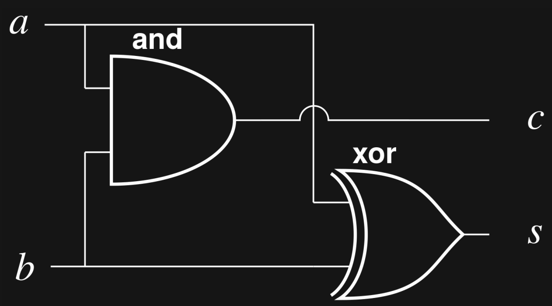
Glick, Gujarati, et al.
arXiv:2105.03406 (2020)

Basic concepts

Bits and classical logic circuits

0

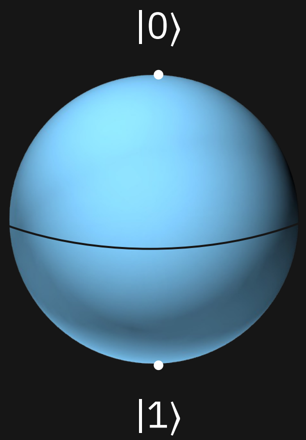
1

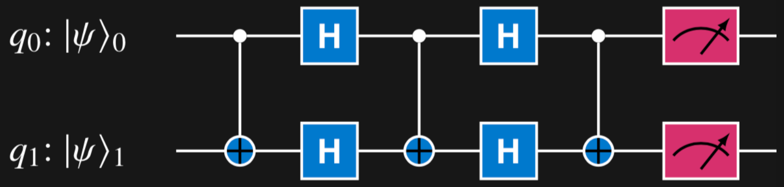


A bit is a controllable classical object that is the unit of information

A classical logic circuit is a set of gate operations on bits and is the unit of computation

Quantum bits (qubits) and quantum circuits





A quantum bit or qubit is a controllable quantum object that is the unit of information

A quantum circuit is a set of quantum gate operations on qubits and is the unit of computation

Quantum computing uses essential ideas from quantum mechanics

Superposition

$|0\rangle$ and $|1\rangle$ are vectors in the two-dimensional complex vector space $\mathbf{C}^2$:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

We can write any vector in $\mathbf{C}^2$ as

$$a\,|0\rangle + b\,|1\rangle$$

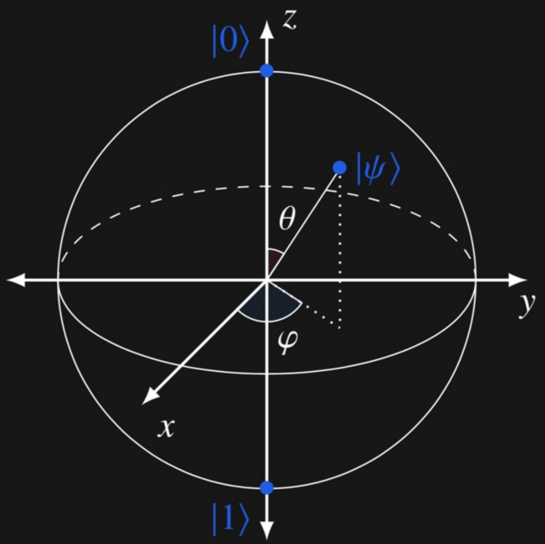
where a, b are complex numbers such that

$$|a|^2 + |b|^2 = 1$$

$|0\rangle$ & $|1\rangle$ are called the *computational basis*.

Quantum computing uses essential ideas from quantum mechanics

Measurement



A qubit state can be mapped onto the *Bloch Sphere*.

When we measure the qubit state

$$a\,|0\rangle + b\,|1\rangle$$

We observe the outcome

"0" with probability $|a|^2$

"1" with probability $|b|^2$

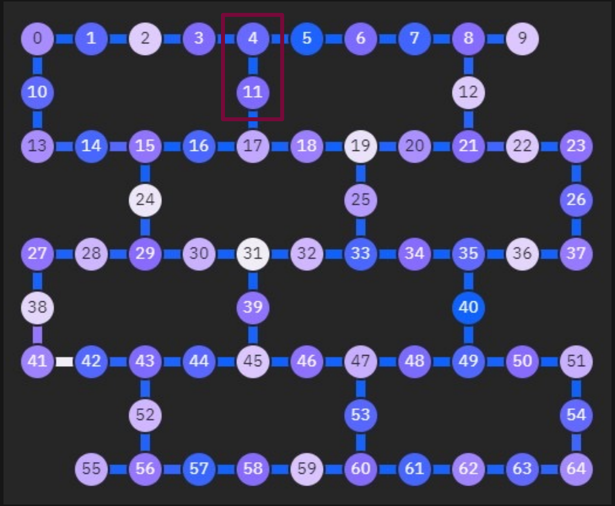
For example,

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

has equal probability of measuring "0" or "1"

Quantum computing uses essential ideas from quantum mechanics

Entanglement



With two qubits we get combinations like

$$a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle$$

where

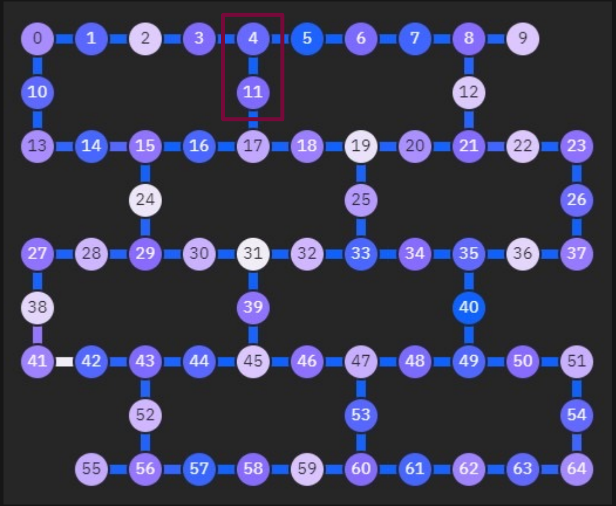
a, b, c, and d are complex numbers and

$$|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$$

Entanglement can occur when two or more of the a, b, c, and d are non-zero

Quantum computing uses essential ideas from quantum mechanics

Entanglement



$\frac{\sqrt{2}}{2}\,|00\rangle + \frac{\sqrt{2}}{2}|01\rangle$ ... not entangled

Because this state is *separable* into

$$|0\rangle\left(\frac{\sqrt{2}}{2}\,|0\rangle + \frac{\sqrt{2}}{2}\,|1\rangle\right)$$

$\frac{\sqrt{2}}{2}\,|01\rangle - \frac{\sqrt{2}}{2}|10\rangle$ ... entangled

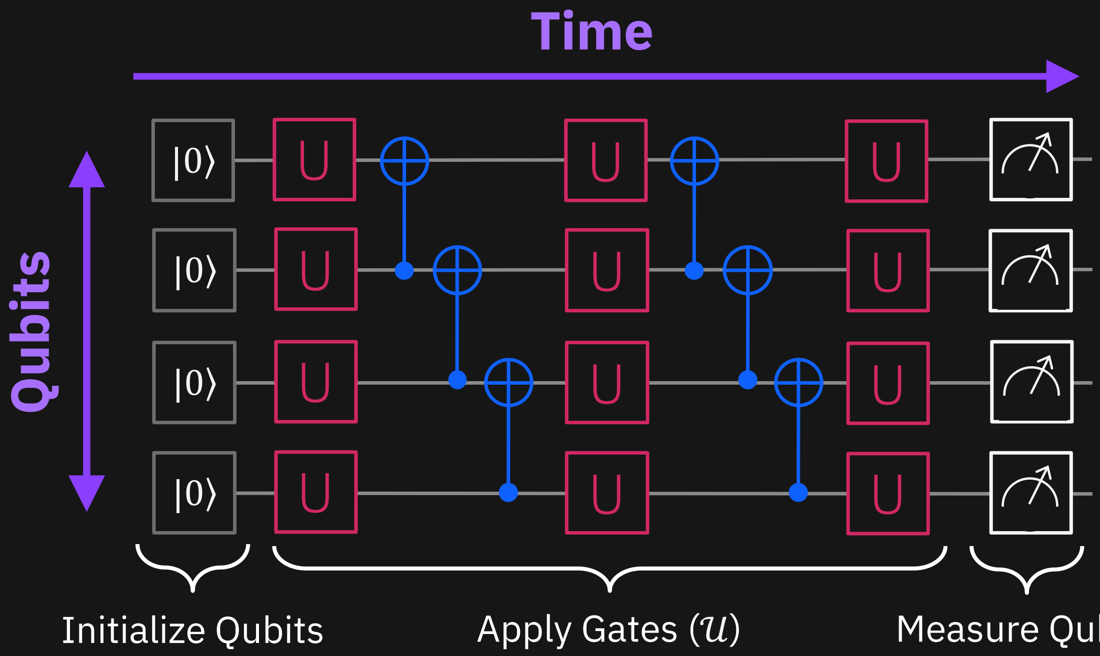
$\frac{\sqrt{2}}{2}\,|00\rangle + \frac{\sqrt{2}}{2}|11\rangle$ ... entangled

Now, if you measure the first qubit, the second is uniquely determined.

Quantum computing uses essential ideas from quantum mechanics

IBM **Quantum**

Circuits

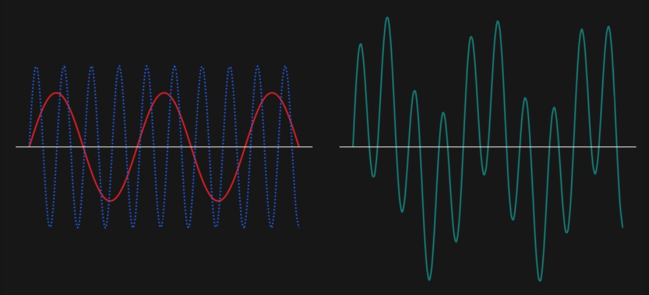


Probability to measure bitstring b:

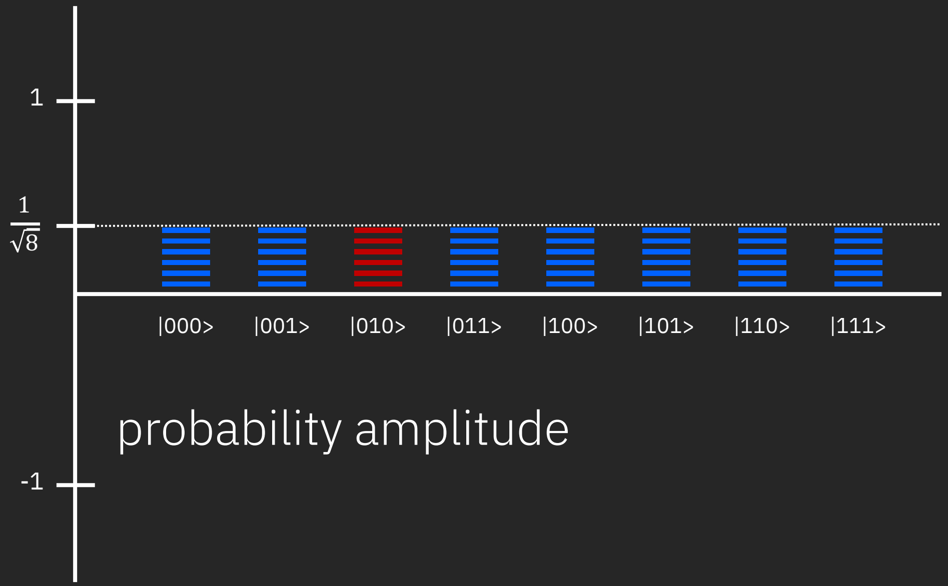
$$p(b) = |\langle b|\mathcal{U}|0\rangle|^2$$

Quantum computing uses essential ideas from quantum mechanics

Interference

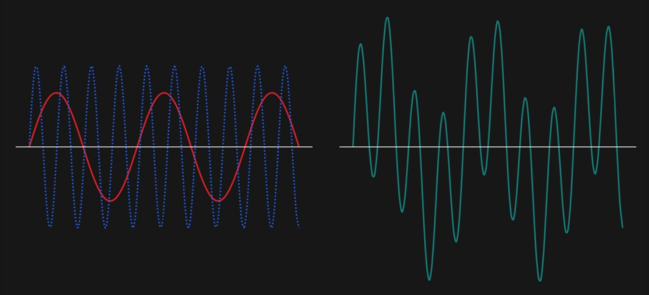


Interference allows us to increase the probability of getting the right answer and decrease the chance of getting the wrong one.

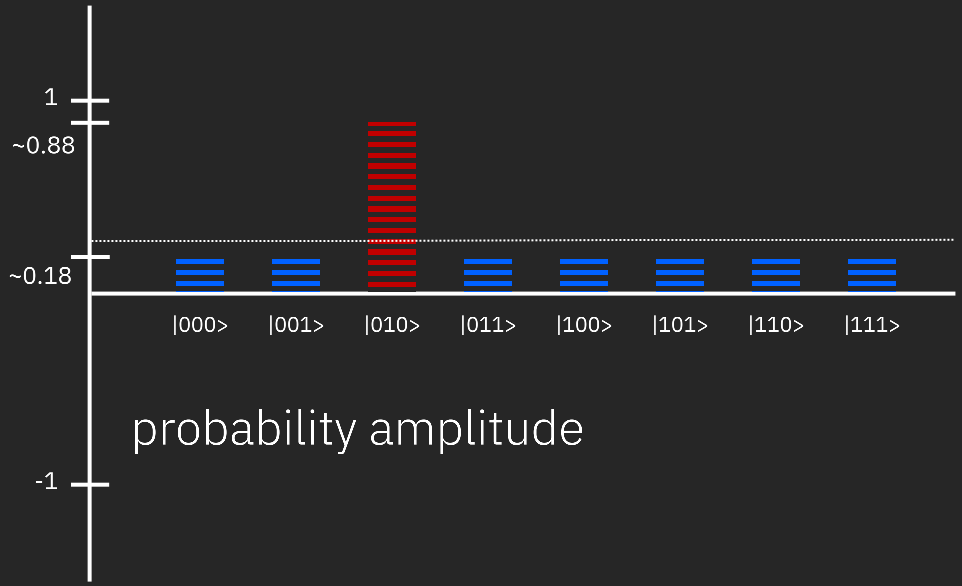


Quantum computing uses essential ideas from quantum mechanics

Interference



Interference allows us to increase the probability of getting the right answer and decrease the chance of getting the wrong one.



Foundational quantum tasks

Foundational quantum tasks

There are certain tasks we want to perform with a quantum computer that form the basis of quantum algorithms and applications

Sampling

Draw samples by measuring circuits

Examples where the solution is obtained from samples drawn:

Quantum phase estimation, Grover's algorithm, Recommendation system

Expectation Values

Measure expectation value of observables

Examples where solution is obtained from expvals:

Molecule energy (VQE), Kernel functions in ML, Combinatorial optimization (QAOA)

Qiskit Runtime

Makes these tasks available as Primitives

1. `Sampler`
2. `Estimator`

How do we calculate expectation values on a quantum computer?

Let's look at computing the expectation value of the Pauli-Z operator.

1. Expectation value is defined as:

$$\langle Z \rangle_\psi = \langle \psi | Z | \psi \rangle$$

2. We can rewrite this operator in the computational basis as:

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|$$

3. Plugging this in yields:

$$\langle Z \rangle_\psi = \langle \psi | 0 \rangle\langle 0 | \psi \rangle - \langle \psi | 1 \rangle\langle 1 | \psi \rangle$$

$$= |\langle 0 | \psi \rangle|^2 - |\langle 1 | \psi \rangle|^2$$

$$= p(0) - p(1)$$

In other words, we sample bitstrings from the circuit, compute probability of observing 0 and 1, and then subtract them!

How do we calculate expectation values on a quantum computer?

IBM **Quantum**

How about for other Pauli operators?

This is the Part 1 exercise!

Preview: quantum applications

Sampling from quantum circuits and/or computing expectation values of observables form the basis of quantum applications.

A prominent example is the Variational Quantum Algorithm.

This includes:

- VQE (e.g., chemistry problems)
- QAOA (combinatorial optimization)
- Quantum Kernel Estimation (ML)

More on all of this in Part II&III!

Qiskit SDK Architecture

IBM **Quantum**

Qiskit

High level applications

Qiskit Nature

For applications relating to simulating quantum mechanical systems and natural phenomena.

Qiskit Finance

For applications relating to financial modeling.

Qiskit Optimization

For applications relating to optimization problems.

Qiskit Machine Learning

For applications relating to machine learning.

Low level applications

Qiskit Metal

For designing quantum hardware and processors.

Qiskit Dynamics

For building, transforming, and solving time-dependent models of quantum systems.

Qiskit Experiments

For running quantum experiments with a library of characterization, calibration, and verification experiments.

Core Capabilities

Qiskit Terra

For building and transforming quantum circuits and operators at the level of gates or pulses.

Simulator

Qiskit Aer

For simulating quantum circuits on classical hardware.

Hardware providers

IBM

IBM Quantum systems

AQT

AQT systems

IonQ

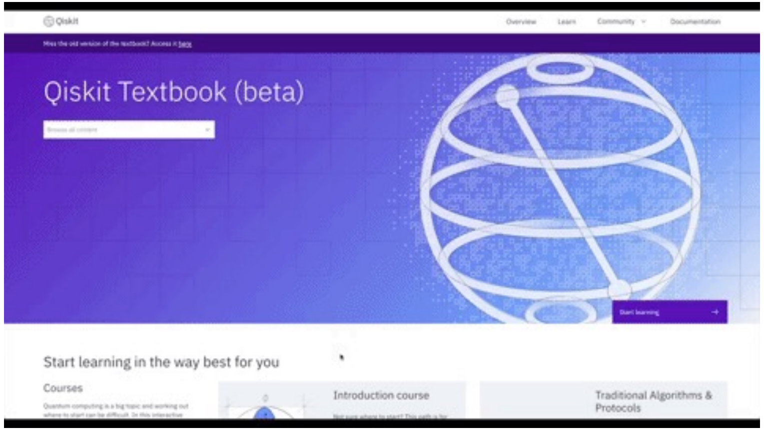
IonQ systems

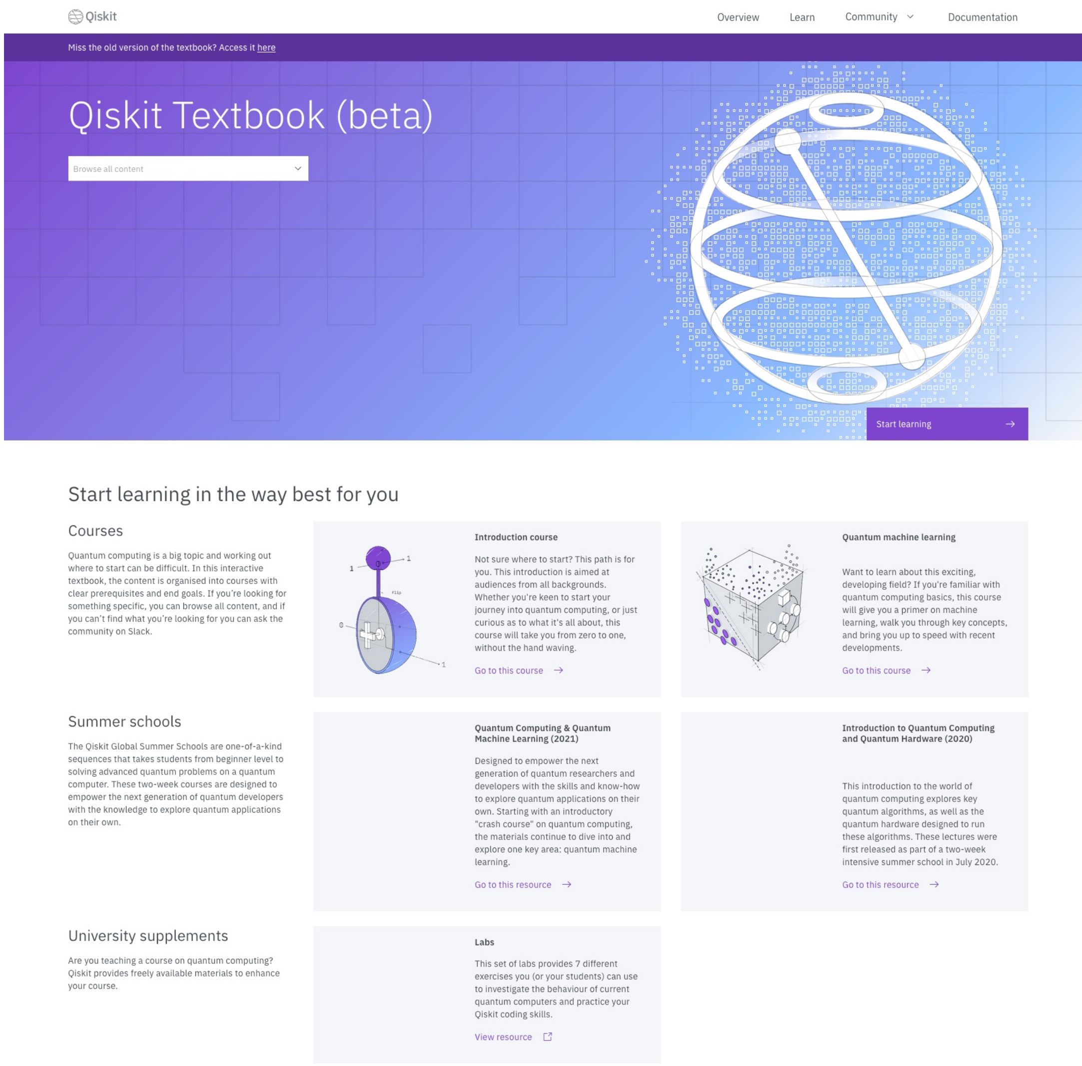
Qiskit can connect to many other systems

Qiskit textbook

The Qiskit textbook (qiskit.org/textbook-beta) is an open-source, university-level quantum algorithms / computation course with Qiskit code implementations and interactive features

Top source of QC knowledge on the web (2nd only to Wikipedia)



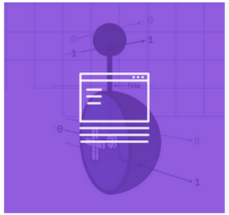


Other learning tools and resources

Qiskit Textbook

The Qiskit Textbook is a free digital open source textbook that will teach the concepts of quantum computing while you learn to use Qiskit.

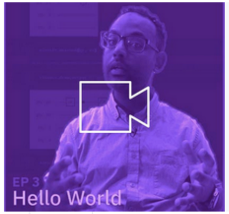
Read the textbook →



Introduction Course

This introduction course is around 3 hours long and will take individuals from all backgrounds through a linear path of content that begins at the atoms of computation and ends at Grover's algorithm.

Take the course →



Coding with Qiskit

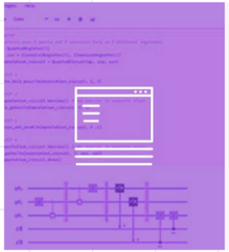
This video series starts at learning how to install Qiskit locally, understanding what gates to do quantum states and explores quantum algorithms and the latest research topics.

Watch the series

Qiskit Medium

This blog provides a nice overview of Qiskit and its direction as we explore what applications can be done on today's quantum devices.

Read the blog



Qiskit Tutorial

Try out this hands on Qiskit tutorial that will provide an overview of working with Qiskit, building circuits, visualizing results and exploring more advanced features in the SDK.

Go to tutorials

Introduction to Quantum Computing and Quantum Hardware

An introduction to the world of quantum computing, with an exploration of some of the key quantum algorithms and their implementations, as well as the quantum hardware that is designed to run these algorithms.

Join the lecture →

Qiskit Learn (qiskit.org/learn)

- Qiskit Textbook
- Qiskit Tutorials
- Qiskit YouTube
- Qiskit Medium
- QGSS recordings and materials

Qiskit Global Summer Schools

IBM **Quantum**

2-week intensive summer school (equivalent to one-semester course)

Largest quantum summer school (4000+ students for both 2020 and 2021)

QGSS 2022 will focus on quantum simulations. Early bird ran out in just a few hours! Registration will open again on June 2, 6pm ET. (Follow Qiskit Twitter for the latest announcement).

About the event:

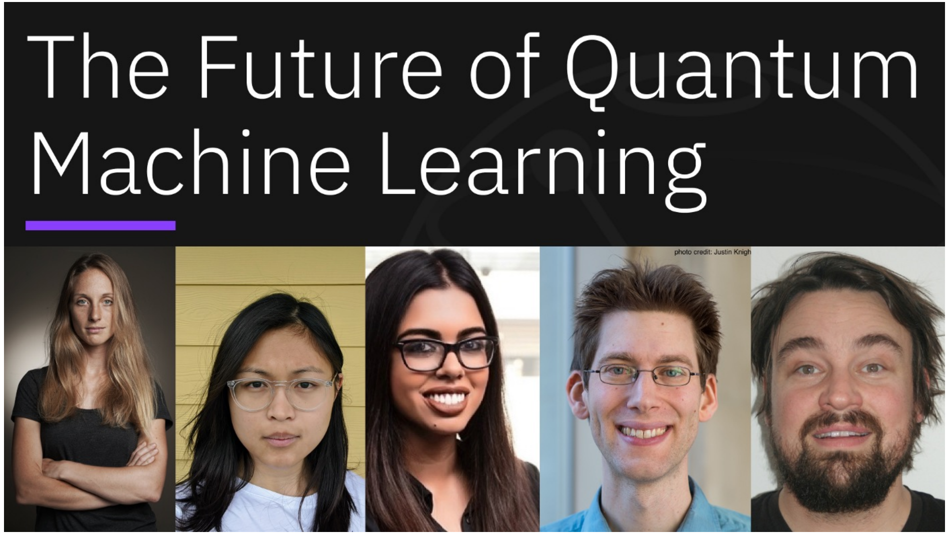
Qiskit Global Summer School 2021: Quantum Machine Learning

The Qiskit Global Summer School returns as a two-week intensive course focused on Quantum Machine Learning and more!

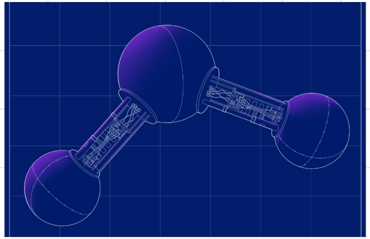
Online

July 12 - 23, 2021

Learn more



About the event:



Qiskit Global Summer School 2022: Quantum Simulations

The Qiskit Global Summer School returns as a two-week intensive course focused on Quantum Simulations and more!

Online

July 18 - 29, 2022

Learn more

Exercise I

Sampling & Expectation Values

`Part1_Sampling_ExpectationValues_EXERCISE.ipynb`

Part II

Introduction to Qiskit Runtime

Variational Quantum Algorithms

Variational Quantum Algorithms (VQA) use a classical optimizer to train a parameterized quantum circuit to approximate solutions for a given problem.

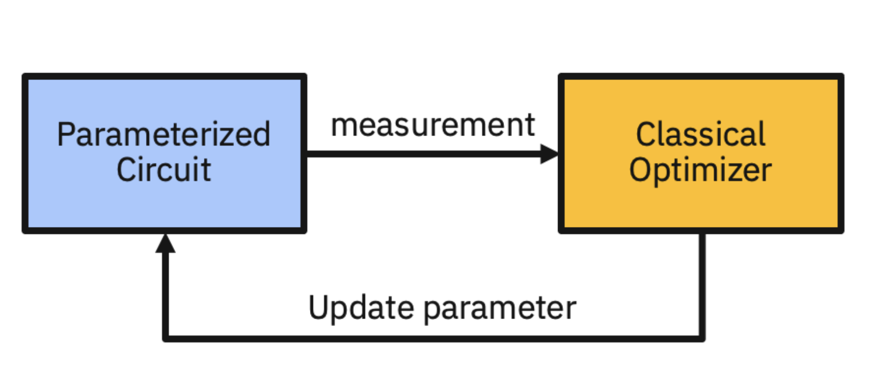
VQA's typically need fewer gates and qubits. In turn, they are more resistant to noise.

Therefore, they are well suited to handle near-term quantum computer constraints.

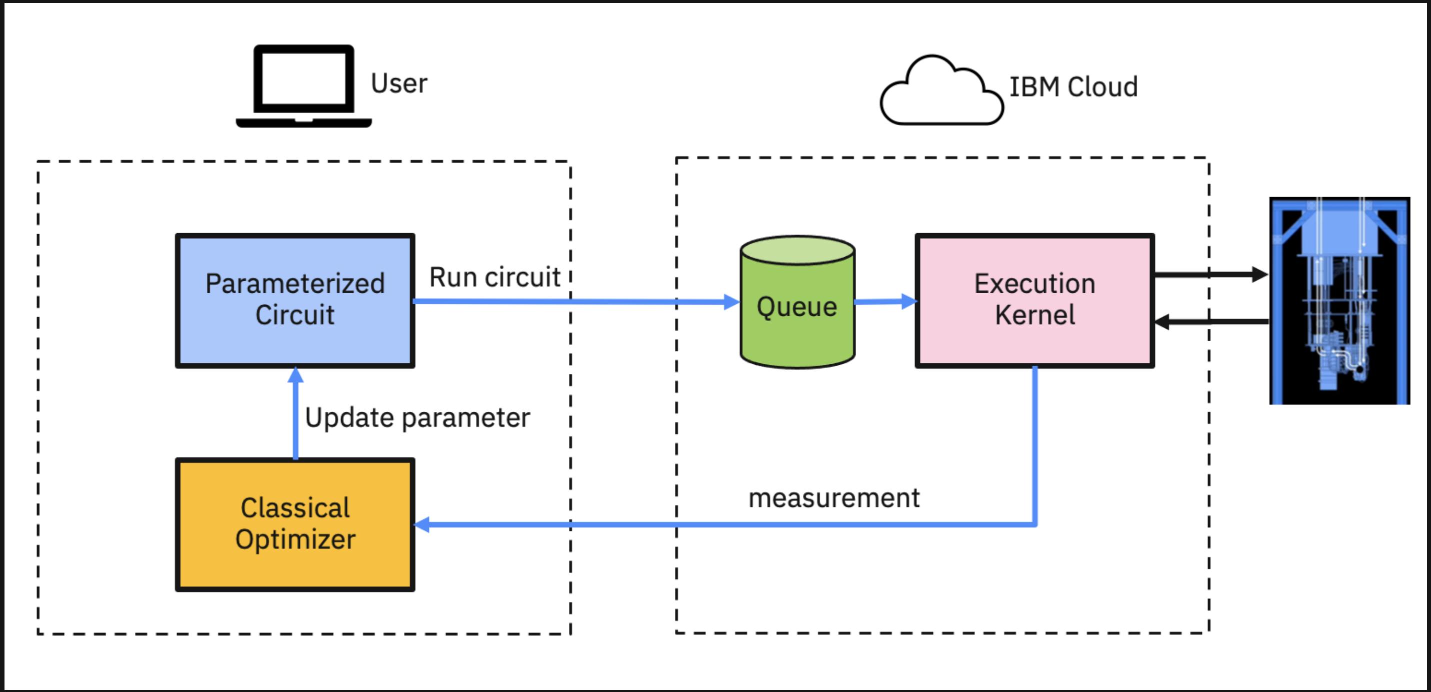
VQAs are iterative

VQA's are typically iterative. Each iteration involves both quantum and classical processing.

Output (a measurement) from one iteration is sent to the classical optimizer which generates input (a parameter) for the next iteration:



Running a VQA prior to the Qiskit Runtime



Qiskit Runtime

A quantum computing service and programming model to efficiently execute optimized workloads at scale.

Qiskit Runtime sessions reduce I/O overhead for applications, such as VQA's, that need many iterations that use both quantum and classical processing.

Qiskit Runtime primitives provides simplified interfaces to perform foundational quantum computing tasks with built-in optimization.

Qiskit Runtime session

An interactive communication that minimizes artificial latency for iterative VQAs.

Jobs within a session are prioritized by the scheduler.

Data used in a session, such as parameterized circuits, is cached on the server.

Coupled with classical serverless technology enables scalable classical capabilities.

Qiskit Runtime primitives

Predefined programs with simplified interface to perform essential quantum computing tasks.

Constantly updated to use the latest quantum software and hardware capabilities, such as readout error mitigation.

Qiskit Runtime primitives

Sampler

takes circuit(s) as an input

outputs quasi-probability distribution

useful for search algorithms like Grover's

Estimator

takes circuit(s) and observable(s) as inputs

outputs expectation values

useful for encoding a variety of things, such as electronic structure of a molecule and cost function for an optimization problem

Show me the code!

Exercise II

IBM **Quantum**

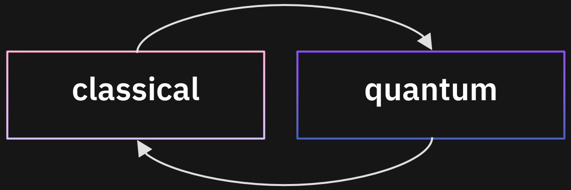
Writing a program using the Estimator primitive

`Part2_Qiskit_Runtime_EXERCISE.ipynb`

Part III

Building applications with Qiskit Runtime

Structure of a typical workload



Real workloads are not purely quantum, but rather require interaction between quantum and classical compute resources.

A prominent example of this workload is the "Variational Quantum Algorithm"

Many applications can be framed as Variational Quantum Algorithms

Examples:

Chemistry (VQE)

Find the ground state energy of a molecule

Optimization (QAOA)

Find the maximum cut of a graph

Machine Learning (VQC, QKA)

Train a quantum circuit to learn from data

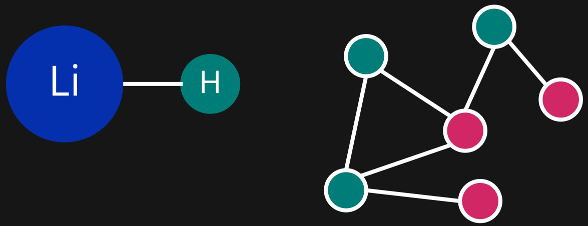
Often, these applications require minimizing the expectation value of an observable (e.g., energy)

Why are VQAs useful?

- Amenable to today's noisy hw (fault tolerant algorithms require error correction; aren't possible to run at scale today)
- Short depth circuits

Variational Quantum Algorithms (VQAs)

Goal Find groundstate of target system $\hat{H}$

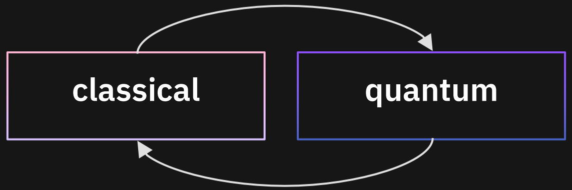


Idea Use ansatz $|\psi(\theta)\rangle$ and variational principle

$$E(\theta^*) = \langle\psi(\theta^*)|\hat{H}|\psi(\theta^*)\rangle \geq E_{\text{exact}}$$

Optimization routine

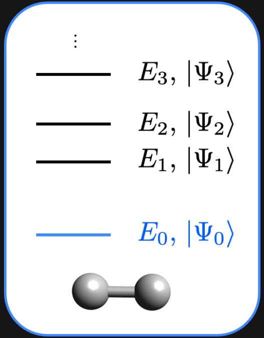
$$\theta^{(k)} \rightarrow \theta^{(k+1)}$$

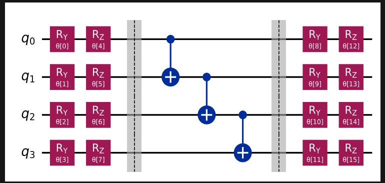


Expectation value calculation

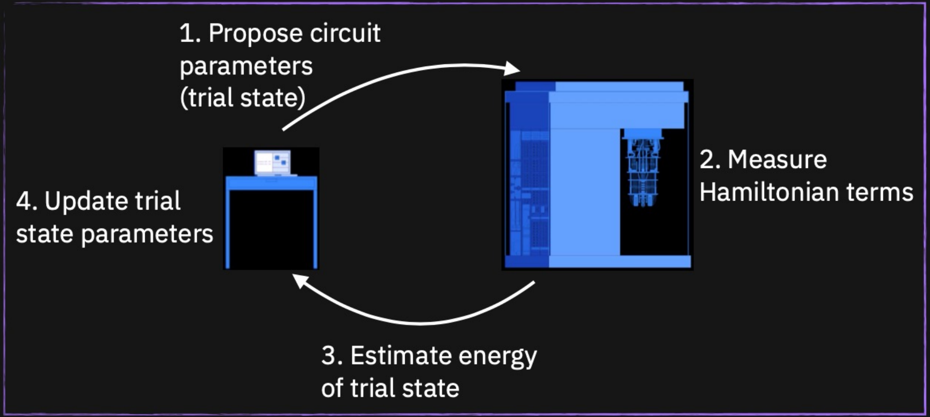
$$E(\theta^{(k)}) = \langle\psi(\theta^{(k)})|\hat{H}|\psi(\theta^{(k)})\rangle$$

VQA: Chemistry





Molecular Hamiltonian (physics) → Map to qubits (circuit) → Variational quantum eigensolver (VQE) → Molecule properties



VQA: Combinatorial optimization

IBM **Quantum**

bits → **qubits**

$$\{0,1\} \rightarrow |0\rangle, |1\rangle$$

cost → **Ising Hamiltonian**

$$C(\mathbf{z}) \rightarrow H_c$$

cost penalty → energy of **Hamiltonian**

$$E(|\psi\rangle) = \langle\psi|H_c|\psi\rangle$$

Find good solution by optimizing energy

Optimization routine

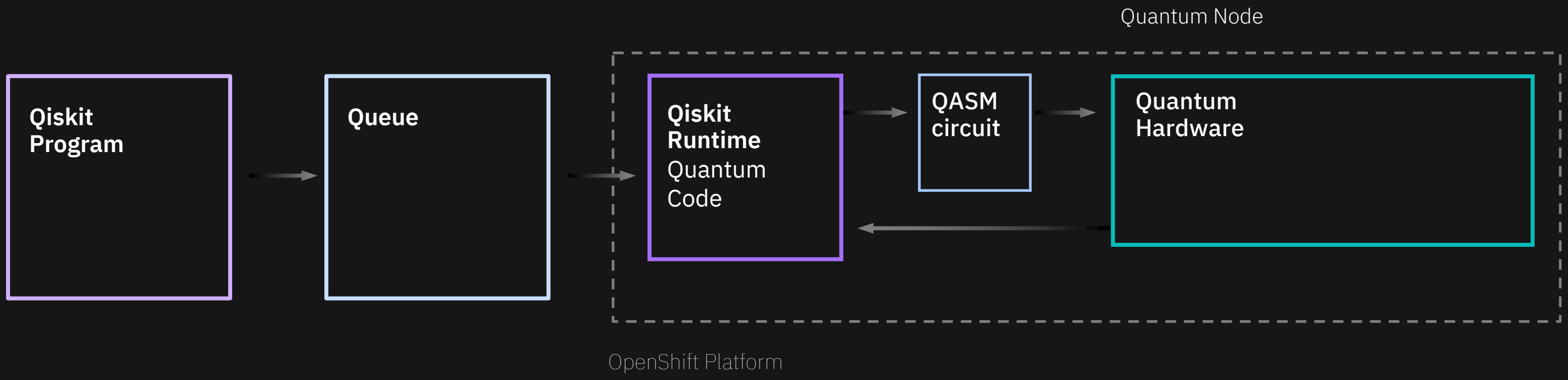
$$\theta^{(k)} \rightarrow \theta^{(k+1)}$$

classical ⇄ **quantum**

Expectation value calculation

$$E(\theta^{(k)}) = \langle\psi(\theta^{(k)})|\hat{H}|\psi(\theta^{(k)})\rangle$$

VQAs are well suited for the Qiskit Runtime framework



Qiskit Runtime is a quantum computing service and programming model that allows users to optimize workloads and efficiently execute them on quantum systems at scale. The programming model extends the existing interface in Qiskit with a set of new primitive programs.

Other Runtime Programs

Qiskit Applications modules include pre-built runtime programs

1

VQE Runtime

Pre-built version of VQE

Ex: Ground state of a molecule

2

QAOA Runtime

Seamlessly fits in the Qiskit Optimization workflow

Ex: Maxcut, TSP

Exercise III

Use Qiskit Runtime to solve a chemistry or optimization problem

`Part3_VQA_Chemistry_EXERCISE.ipynb`
`Part3_VQA_Optimization_EXERCISE.ipynb`