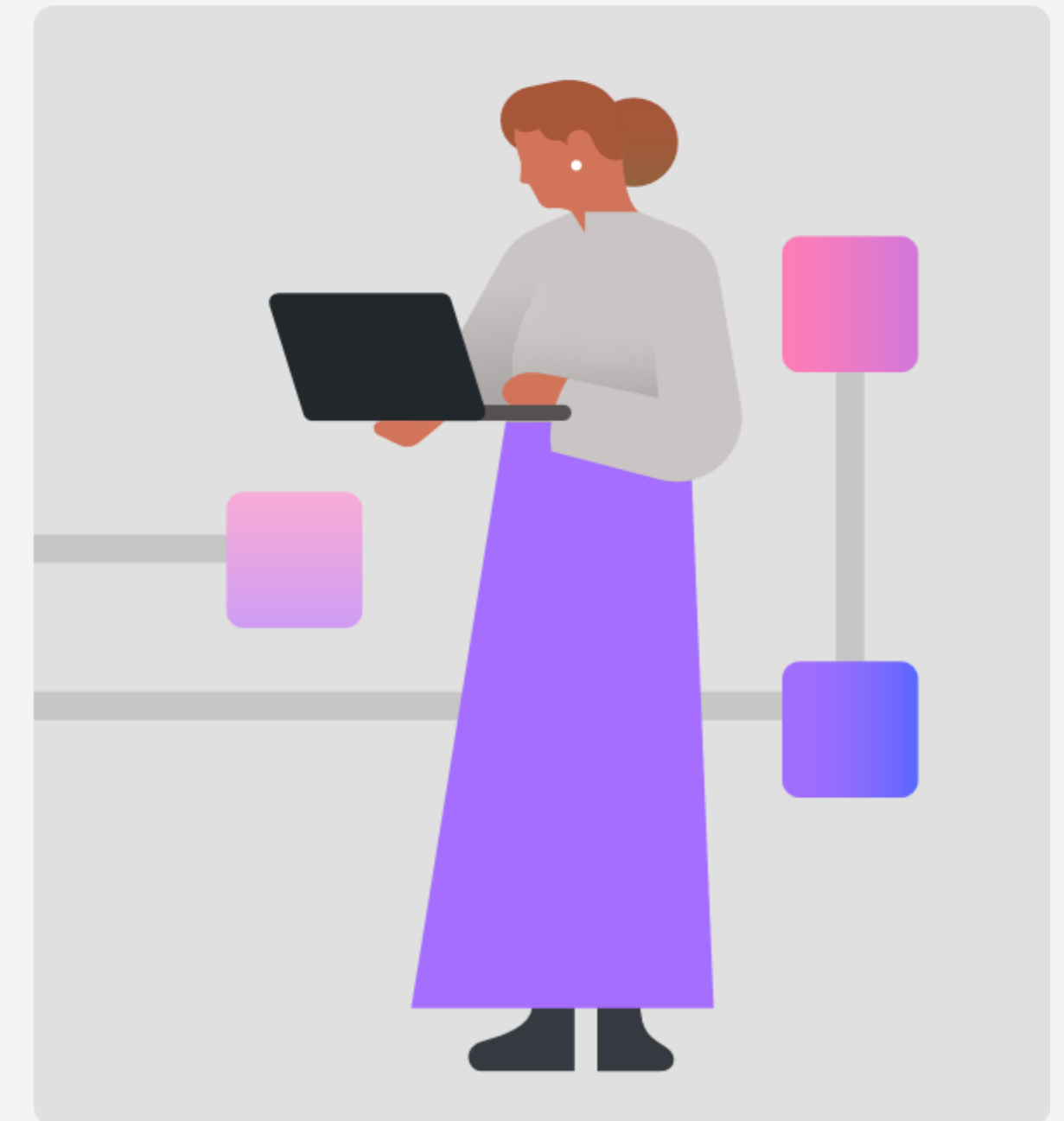


Qiskit Fall Fest 2025

Qiskit Fall Fest 2025: Running a Quantum and Qiskit 101 session

Vishal Sharathchandra Bajpe
Quantum Algorithms Engineer
IBM



Audience takeaway – Aim to:

- Spark an inspiration to embark on a quantum journey



Audience takeaway – Aim to:



- Spark an inspiration to embark on a quantum journey
- Provide a hands-on starting point.

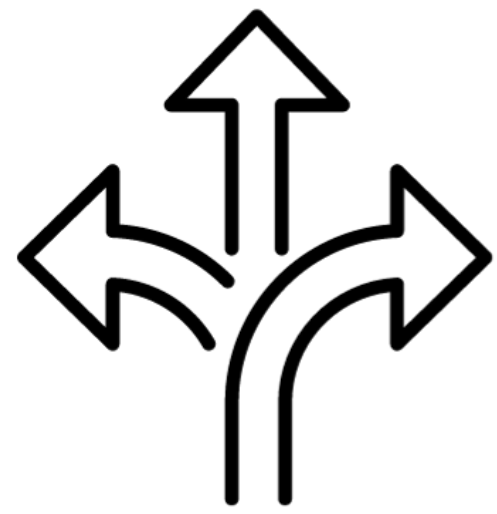
Not a
Qiskit/Quantum
101 lecture!

Audience takeaway – Aim to:



- Spark an inspiration to embark on a quantum journey
- Provide a hands-on starting point.

What will we look at today?



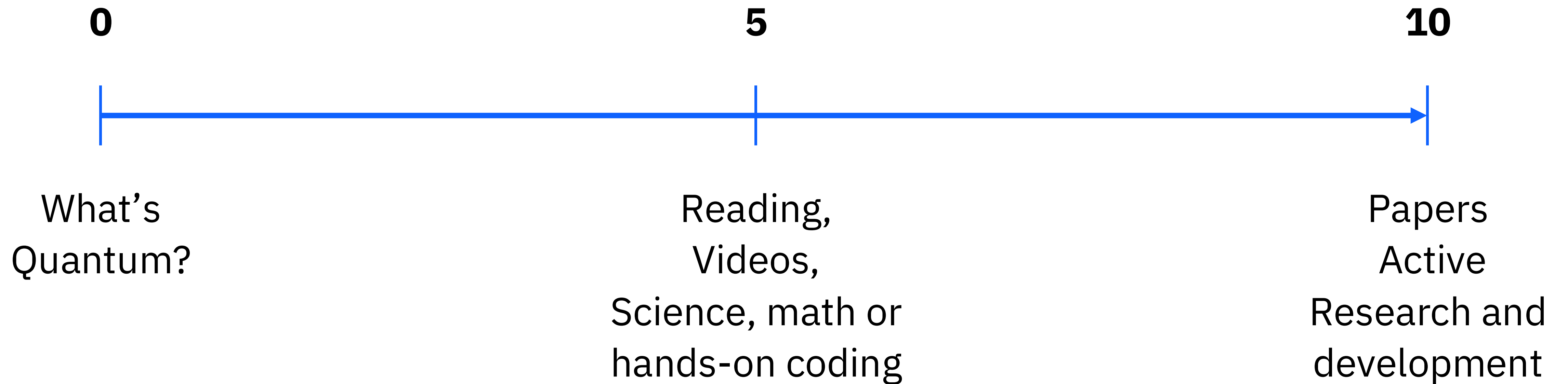
Some **pathways**
to consider



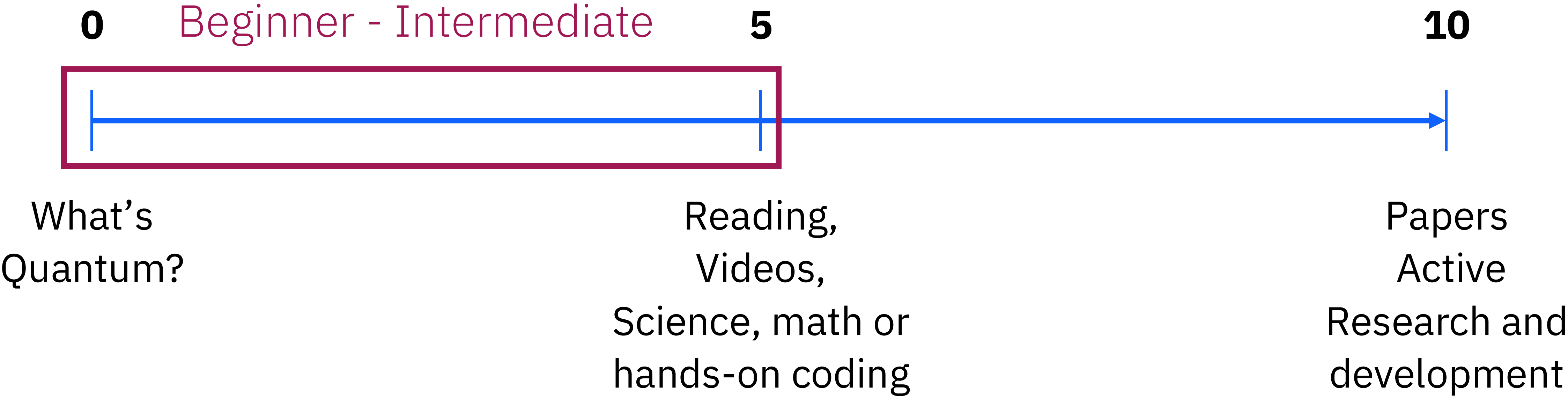
Accompanying resources
to save time

Let's Begin!

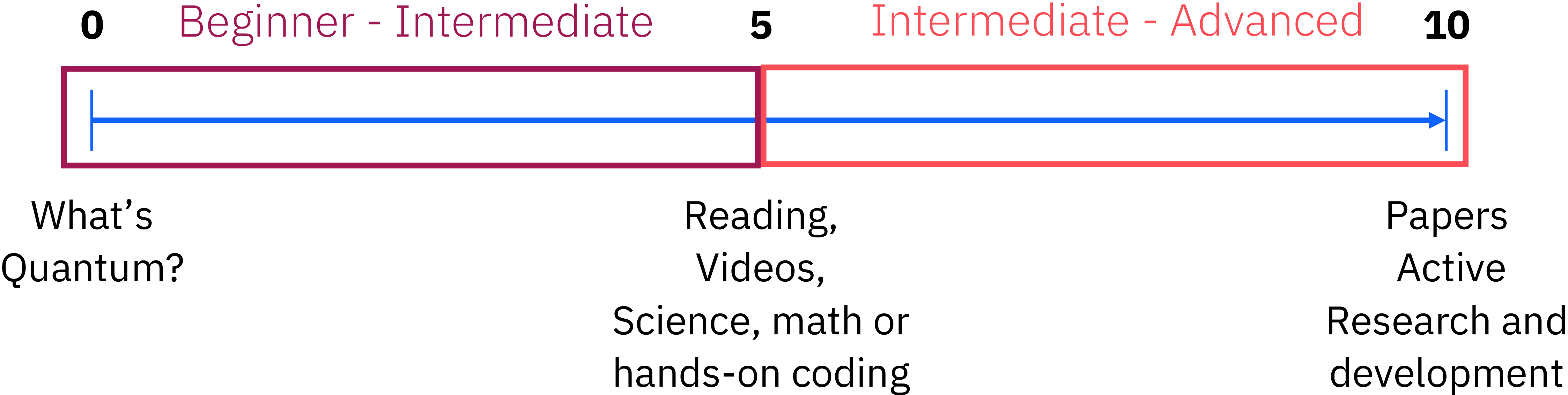
Where are your participants on their Quantum journey?



Where are your participants on their Quantum journey?



Where are your participants on their Quantum journey?



What are we aiming for?



What are we aiming for?

Build confidence and curiosity

Help participants feel comfortable with quantum concepts and tools and try to spark genuine interest to explore further.

Hands-on first steps

Enable attendees to experience Qiskit SDK, run circuits and see results in action

Set a launchpad

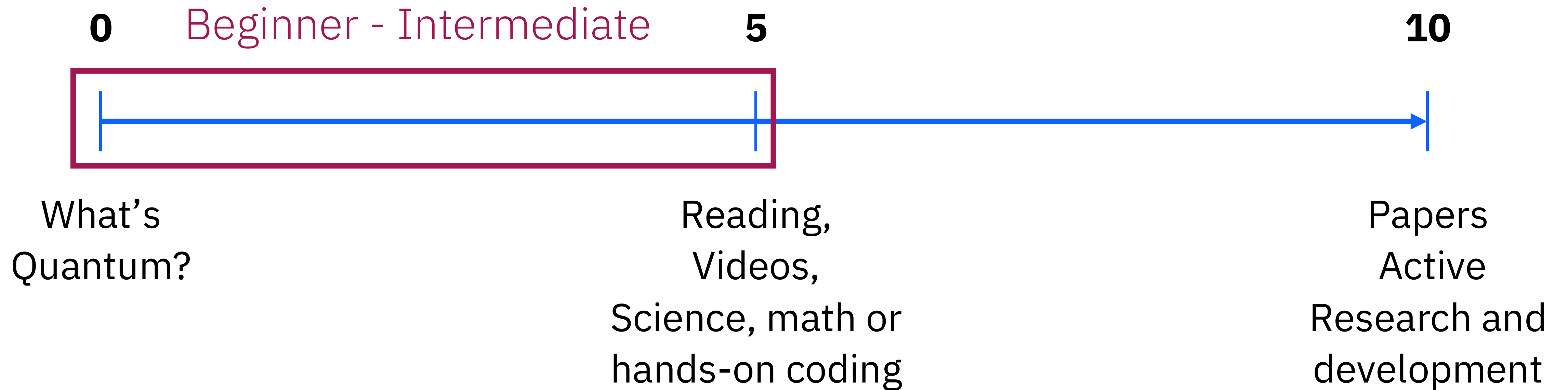
Provide a pathways for participants to continue their journey beyond the session.



Qiskit



Where are your participants on their Quantum journey?



Qiskit Pattern:

The anatomy of a quantum algorithm



01
Map problem instance
to quantum circuits and
operators

Q⁺
Map

02
Optimize for target
hardware execution

↔
Optimize

03
Execute via
Qiskit Runtime

🖨️
Execute

04
Result processing

📈
Post-Process

Qiskit SDK sets the foundation

Qiskit SDK gives us a base layer of building blocks for building and running quantum algorithms



Qiskit Circuit Library	Transpiler	Primitives	Quantum Info
Input: Domain inputs	Input: Circuits, observable	Input: ISA circuit, observable	Input: Expectation value/samples
Output: Circuits, observable	Output: ISA circuit, observable	Output: Expectation value/samples	Output: Data objects/visualizations
<hr/> Map	<hr/> Optimize	<hr/> Execute	<hr/> Post-Process

IBM Quantum Platform

Hello World: A quick hands on introduction to running circuits on Qiskit and IBM Quantum platform

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/hello-world>



IBM Quantum Platform

DocumentationGuidesTutorialsAPI referenceAdditional resources

TutorialsGet startedOverviewHello worldVariational quantum eigensolverCHSH inequalityExplore workflows toward advantageVerifiable sampling algorithmsObservable estimationFault-tolerant algorithmsLeverage Qiskit capabilitiesWorkload optimizationQiskit FunctionsQiskit addonsError mitigationError detection

Hello world

> Package versions

This example contains two parts. You will first create a simple quantum program and run it on a quantum processing unit (QPU). Because actual quantum research requires much more robust programs, in the second section ([Scale to large numbers of qubits](#)), you will scale the simple program up to utility level.

Before you begin

Follow the [Install and set up](#) instructions if you haven't already, including the steps to [Set up to use IBM Cloud®](#).

The code examples found in the IBM Quantum Platform documentation were developed by using [Jupyter](#) notebooks. To follow along with the examples, it is recommended that you set up an environment to run Jupyter notebooks [locally](#) or [online](#). Be sure to install the recommended extra visualization support (`'qiskit[visualization]'`). You'll also need the `matplotlib` package for the second part of this example.

To learn about quantum computing in general, visit the [Basics of quantum information course](#) in IBM Quantum Learning.

IBM® is committed to the responsible development of quantum computing. Learn more about responsible quantum at IBM and review our responsible quantum principles in the [Responsible quantum computing and inclusive tech](#) topic.

Create and run a simple quantum program

On this page

Before you begin

Create and run a simple quantum program

Step 1. Map the problem to a quantum-native format

Step 2. Optimize the circuits and operators

Step 3. Execute using the quantum primitives

Step 4. Analyze the results

Scale to large numbers of qubits

Step 1. Map the problem

Step 2. Optimize the problem for execution on quantum hardware

Step 3. Execute on hardware

Step 4. Post-process results

Next steps

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on [GitHub](#)



The Battle of the Install



The Battle of the Install

What we want to avoid?



The Battle of the Install

What we want to avoid?

`pip install ... and pray` 🙏

Common pitfalls:

Installation Delays

Setting up Python, Qiskit, and dependencies can eat up lot of valuable hack time if not done beforehand.

Environment Issues

Conflicts between OS, IDEs, or package versions often derail momentum.

Wi-Fi & Access

Large downloads or cloud account sign-ups (IBM Quantum) can slow teams down if left to the last minute.



IBM Quantum Platform

Hello World: A quick hands on introduction to running circuits on Qiskit and IBM Quantum platform

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/hello-world>



IBM Quantum Platform

DocumentationGuidesTutorialsAPI referenceAdditional resources

TutorialsGet startedOverviewHello worldVariational quantum eigensolverCHSH inequalityExplore workflows toward advantageVerifiable sampling algorithmsObservable estimationFault-tolerant algorithmsLeverage Qiskit capabilitiesWorkload optimizationQiskit FunctionsQiskit addonsError mitigationError detection

Hello world

> Package versions

This example contains two parts. You will first create a simple quantum program and run it on a quantum processing unit (QPU). Because actual quantum research requires much more robust programs, in the second section ([Scale to large numbers of qubits](#)), you will scale the simple program up to utility level.

Before you begin

Follow the [Install and set up](#) instructions if you haven't already, including the steps to [Set up to use IBM Cloud®](#).

The code examples found in the IBM Quantum Platform documentation were developed by using [Jupyter](#) notebooks. To follow along with the examples, it is recommended that you set up an environment to run Jupyter notebooks [locally](#) or [online](#). Be sure to install the recommended extra visualization support (`'qiskit[visualization]'`). You'll also need the `matplotlib` package for the second part of this example.

To learn about quantum computing in general, visit the [Basics of quantum information course](#) in IBM Quantum Learning.

IBM® is committed to the responsible development of quantum computing. Learn more about responsible quantum at IBM and review our responsible quantum principles in the [Responsible quantum computing and inclusive tech](#) topic.

Create and run a simple quantum program

On this page

Before you begin

Create and run a simple quantum program

Step 1. Map the problem to a quantum-native format

Step 2. Optimize the circuits and operators

Step 3. Execute using the quantum primitives

Step 4. Analyze the results

Scale to large numbers of qubits

Step 1. Map the problem

Step 2. Optimize the problem for execution on quantum hardware

Step 3. Execute on hardware

Step 4. Post-process results

Next steps

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on GitHub

IBM Quantum Platform

Hello World: A quick hands on introduction to running circuits on Qiskit and IBM Quantum platform

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/hello-world>



IBM Quantum Platform

DocumentationGuidesTutorialsAPI referenceAdditional resources

Tutorials

Get started

Overview

Hello world

Variational quantum eigensolver

CHSH inequality

Explore workflows toward advantage

Verifiable sampling algorithms

Observable estimation

Fault-tolerant algorithms

Leverage Qiskit capabilities

Workload optimization

Qiskit Functions

Qiskit addons

Error mitigation

Error detection

Hello world

> Package versions

This example contains two parts. You will first create a simple quantum program and run it on a quantum processing unit (QPU). Because actual quantum research requires much more robust programs, in the second section ([Scale to large numbers of qubits](#)), you will scale the simple program up to utility level.

Before you begin

Follow the [Install and set up](#) instructions if you haven't already, including the steps to [Set up to use IBM Cloud®](#).

The code examples found in the IBM Quantum Platform documentation were developed by using [Jupyter](#) notebooks. To follow along with the examples, it is recommended that you set up an environment to run Jupyter notebooks [locally](#) or [online](#). Be sure to install the recommended extra visualization support (`'qiskit[visualization]'`). You'll also need the `matplotlib` package for the second part of this example.

To learn about quantum computing in general, visit the [Basics of quantum information course](#) in IBM Quantum Learning.

IBM® is committed to the responsible development of quantum computing. Learn more about responsible quantum at IBM and review our responsible quantum principles in the [Responsible quantum computing and inclusive tech](#) topic.

Create and run a simple quantum program

On this page

Before you begin

Create and run a simple quantum program

Step 1. Map the problem to a quantum-native format

Step 2. Optimize the circuits and operators

Step 3. Execute using the quantum primitives

Step 4. Analyze the results

Scale to large numbers of qubits

Step 1. Map the problem

Step 2. Optimize the problem for execution on quantum hardware

Step 3. Execute on hardware

Step 4. Post-process results

Next steps

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on GitHub

Qiskit 101: First steps into quantum computing for a hands-on visual introduction to Qiskit and Quantum gates

Notebook: Will be shared to you



First Step into Quantum Computing

- Difficulties: Beginner
- QPU time usage: 11s

Welcome, hackers! We're thrilled to have you for the workshop. The main goal of this introductory hands-on is to make you ready for your quantum journey by 1) guiding you how to install qiskit 2) how to create IBM Cloud account and prepare `api_key` and `crn` to use a real quantum computer and 3) make your first quantum circuit, 4) solve a quantum state quiz and 5) Run you circuits on the real quantum computer and plot the result.

1. First things first: Qiskit

What is Qiskit



IBM Quantum Platform

Running Quantum Circuits: A descriptive introduction to running circuits on Qiskit and IBM Quantum platform

Link:
<https://quantum.cloud.ibm.com/learning/en/courses/quantum-computing-in-practice/running-quantum-circuits>



IBM Quantum Platform

LearningComposer ↗

←

Quantum computing in practice

Overview

Lessons

Course introduction

Running quantum circuits

Utility-scale QAOA

Which problems are quantum computers good for?

Mapping New

Running quantum circuits

Watch the video on quantum circuits and primitives from Olivia Lanes, or open the video in a separate window on [YouTube](#). ↗

Quantum Computing in practice

Lesson 02
Quantum Circuits

Lesson overview

This lesson will be a high-level look at the basics of running a utility-scale quantum computation, from the quantum hardware used to principles to consider when designing a quantum circuit. Ideally, by the end of this lesson, you'll know:

On this page

Lesson overview

Hardware – IBM quantum processors

Software: Qiskit and Qiskit Runtime

Quantum Circuits

Designing a quantum circuit: Qiskit patterns

Map

Optimize

Execute

Post-process

Conclusion

Download notebook ⬇

Was this page helpful?

Yes👍No👎

Report a bug or request content on [GitHub](#) ↗.

Example workflows



Quantum Approximate optimization Algorithm (QAOA) Workflow

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/quantum-approximate-optimization-algorithm>



IBM Quantum Platform

DocumentationGuidesTutorialsAPI referenceAdditional resources

Tutorials

Get started

Overview

Hello world

Variational quantum eigensolver

CHSH inequality

Explore workflows toward advantage

Verifiable sampling algorithms

Sample-based quantum diagonalization of a chemistry Hamiltonian

Sample-based Krylov quantum diagonalization of a fermionic lattice model

Quantum approximate optimization algorithm

Advanced techniques for QAOA

Pauli correlation encoding to reduce Maxcut requirements

Observable estimation

Fault-tolerant algorithms

Leverage Qiskit capabilities

Workload optimization

Qiskit Functions

Qiskit addons

Error mitigation

Error detection

Quantum approximate optimization algorithm

Usage estimate: 8 minutes on IBM Sherbrooke (NOTE: This is an estimate only. Your runtime may vary.)

Background

This tutorial demonstrates how to implement the **Quantum Approximate Optimization Algorithm (QAOA)** – a hybrid (quantum-classical) iterative method – within the context of Qiskit patterns. You will first solve the **Maximum-Cut** (or **Max-Cut**) problem for a small graph and then learn how to execute it at utility scale. All the hardware executions in the notebook should run within the time limit for the freely-accessible Open Plan.

The Max-Cut problem is an optimization problem that is hard to solve (more specifically, it is an *NP-hard* problem) with a number of different applications in clustering, network science, and statistical physics. This tutorial considers a graph of nodes connected by edges, and aims to partition it into separate graphs such that they are both as large as possible. Put another way, the goal of this problem is to partition the nodes of a graph into two sets such that the number of edges traversed by this cut is maximized.

On this page

Background

Requirements

Setup

Part I. Small-scale QAOA

Step 1: Map classical inputs to a quantum problem

Optimization problem → Hamiltonian

Step 2: Optimize problem for quantum hardware execution

Step 3: Execute using Qiskit primitives

Step 4: Post-process and return result in desired classical format

Part II. scale it up!

Step 1: Map classical inputs to a quantum problem

Step 2: Optimize problem for quantum execution

Step 3: Execute using Qiskit primitives

Step 4: Post-process and return result in desired classical format

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on GitHub

IBM Quantum Platform

Quantum Approximate optimization Algorithm (QAOA) Workflow walkthrough

Link:
<https://quantum.cloud.ibm.com/learning/en/courses/quantum-computing-in-practice/utility-scale-qaoa>



IBM Quantum Platform

LearningComposer ↗

←

Quantum computing in practice

Overview

Lessons

Course introduction

Running quantum circuits

Utility-scale QAOA

Which problems are quantum computers good for?

Mapping New

Utility-scale QAOA

Watch the video on utility-scale QAOA from Olivia Lanes, or open the video in a separate window on [YouTube](#). ↗

Lesson 03

Utility-scale QAOA

Quantum Computing in practice

Lesson overview:

So far in this course, we hope we've given you a solid foundation of the framework and tools needed to solve utility-scale problems on a quantum computer. Now, we're finally going to see these tools in action.

In this lesson, we'll get our hands dirty with a utility-scale example of the Max-Cut problem, which is a famous problem in graph theory involving how to best partition a graph into two. We'll start with a simple, five-node

On this page

Lesson overview:

The Problem

What is Max-Cut?

The Solution

Map

Optimize

Execute

Post-process

At utility-scale

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on [GitHub](#) ↗

Error Mitigation (optional)



Notebook: Combine error mitigation options with the Estimator Primitive

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/combine-error-mitigation-techniques>



IBM Quantum Platform

DocumentationGuidesTutorialsAPI referenceAdditional resources

Explore workflows toward advantage

Verifiable sampling algorithms

Sample-based quantum diagonalization of a chemistry Hamiltonian

Sample-based Krylov quantum diagonalization of a fermionic lattice model

Quantum approximate optimization algorithm

Advanced techniques for QAOA

Pauli correlation encoding to reduce Maxcut requirements

Observable estimation

Fault-tolerant algorithms

Leverage Qiskit capabilities

Workload optimization

Qiskit Functions

Qiskit addons

Error mitigation

Utility-scale error mitigation with probabilistic error amplification

Combine error mitigation options with the Estimator primitive

Real-time benchmarking for qubit selection

Error detection

Combine error mitigation options with the Estimator primitive

Usage estimate: 8 minutes on ibm_fez (NOTE: This is an estimate only. Your runtime may vary.)

Background

In this tutorial, you'll explore the error suppression and error mitigation options available with the Estimator primitive from Qiskit Runtime. You will construct a circuit and observable and submit jobs using the Estimator primitive using different combinations of error mitigation settings. Then, you will plot the results to observe the effects of the various settings. Most of the tutorial uses a 10-qubit circuit to make visualizations easier, and at the end, you can scale up the workflow to 50 qubits.

These are the error suppression and mitigation options you will use:

- Dynamical decoupling
- Measurement error mitigation
- Gate twirling
- Zero-noise extrapolation (ZNE)

Requirements

On this page

Background

Requirements

Setup

Step 1: Map classical inputs to a quantum problem

Step 2: Optimize problem for quantum hardware execution

Step 3: Execute using Qiskit primitives

Step 4: Post-process and return result in desired classical format

Scale the experiment up

Conclusion

Tutorial survey

Download notebook

Was this page helpful?

Yes

No

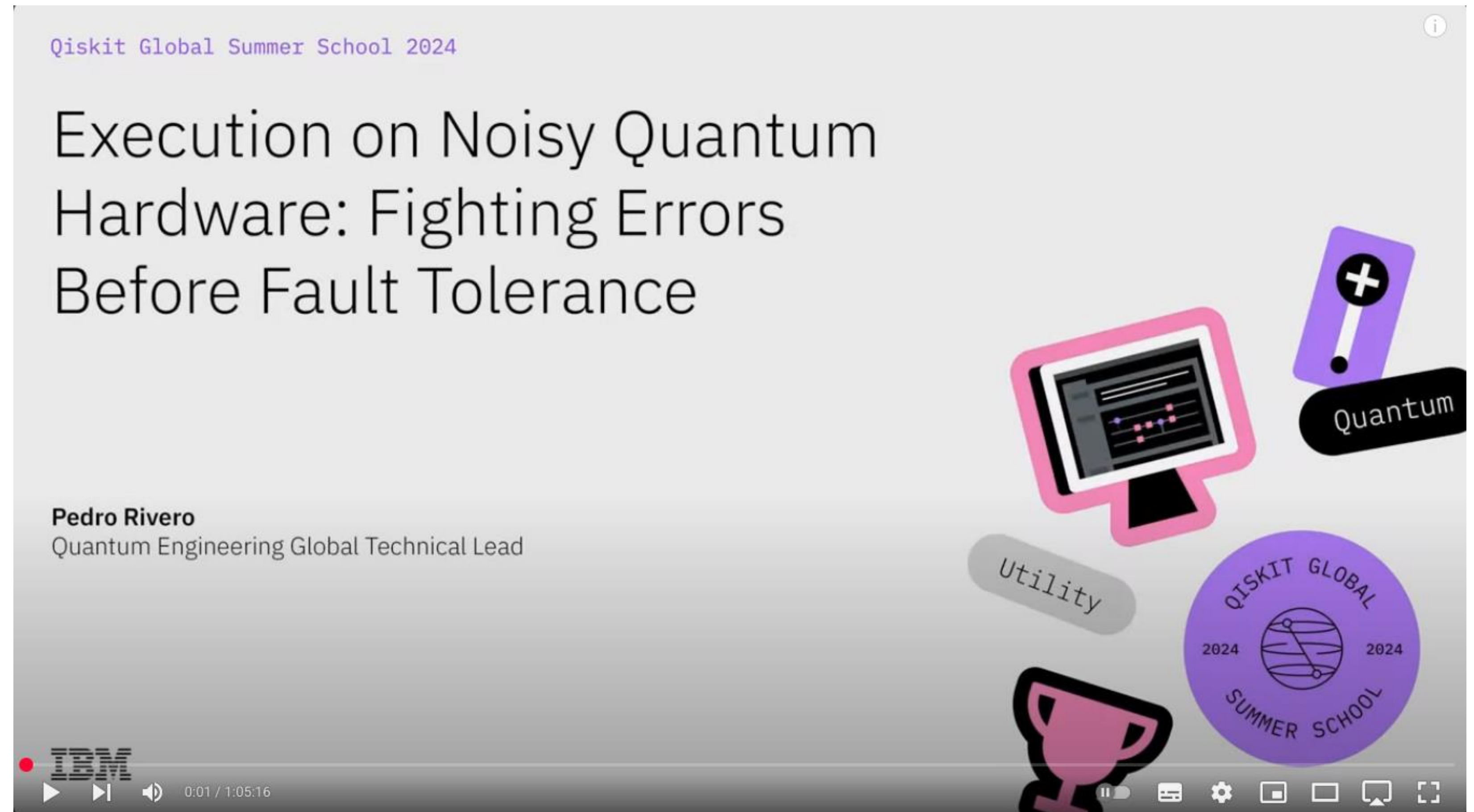
Report a bug or request content on GitHub

IBM Quantum Platform

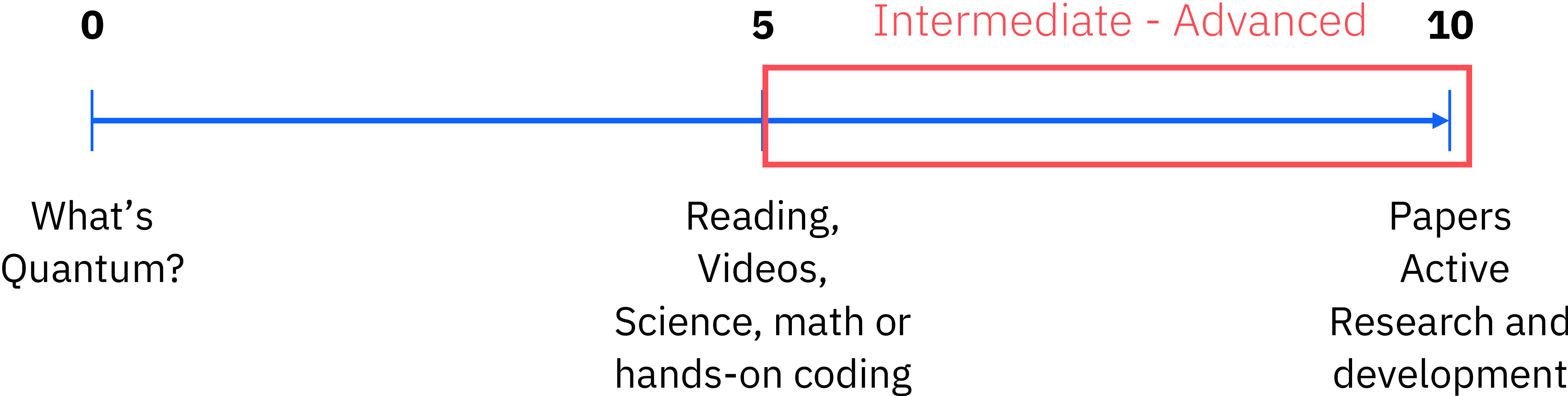
Video: Execution on Noisy Quantum Hardware: Fighting errors before Fault Tolerance

Link:

https://www.youtube.com/watch?v=l5V8J_yMEis



Where are your participants on their Quantum journey?



Qiskit Pattern:
The anatomy of a quantum algorithm

SAME!



01
Map problem instance
to quantum circuits and
operators

Q⁺
Map

02
Optimize for target
hardware execution

↔
Optimize

03
Execute via
Qiskit Runtime

📄
Execute

04
Result processing

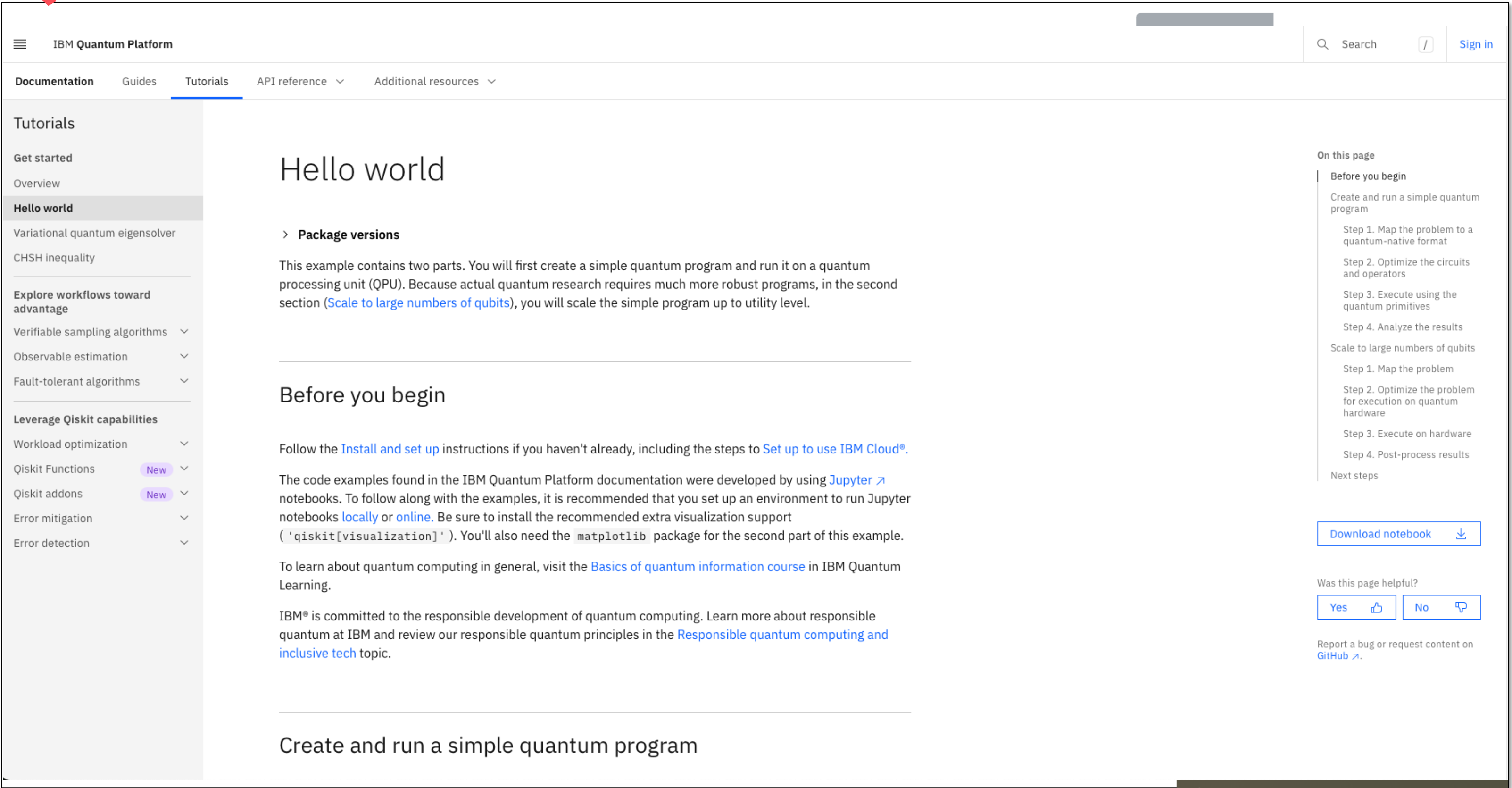
📈
Post-Process

IBM Quantum Platform

Hello World: A quick hands on introduction to running circuits on Qiskit and IBM Quantum platform

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/hello-world>

SAME!



IBM Quantum Platform

Running Quantum Circuits: A descriptive introduction to running circuits on Qiskit and IBM Quantum platform

Link:
<https://quantum.cloud.ibm.com/learning/en/courses/quantum-computing-in-practice/running-quantum-circuits>



SAME!

IBM Quantum Platform

LearningComposer ↗

←

Quantum computing in practice

Overview

Lessons

Course introduction

Running quantum circuits

Utility-scale QAOA

Which problems are quantum computers good for?

Mapping

New

Running quantum circuits

Watch the video on quantum circuits and primitives from Olivia Lanes, or open the video in a separate window on [YouTube](#). ↗

Quantum Computing in practice

Lesson 02

Quantum Circuits

Lesson overview

This lesson will be a high-level look at the basics of running a utility-scale quantum computation, from the quantum hardware used to principles to consider when designing a quantum circuit. Ideally, by the end of this lesson, you'll know:

On this page

Lesson overview

Hardware – IBM quantum processors

Software: Qiskit and Qiskit Runtime

Quantum Circuits

Designing a quantum circuit: Qiskit patterns

Map

Optimize

Execute

Post-process

Conclusion

Download notebook ↕

Was this page helpful?

Yes👍No👎

Report a bug or request content on [GitHub](#) ↗.





Example workflows



Qiskit addons build on the Qiskit SDK

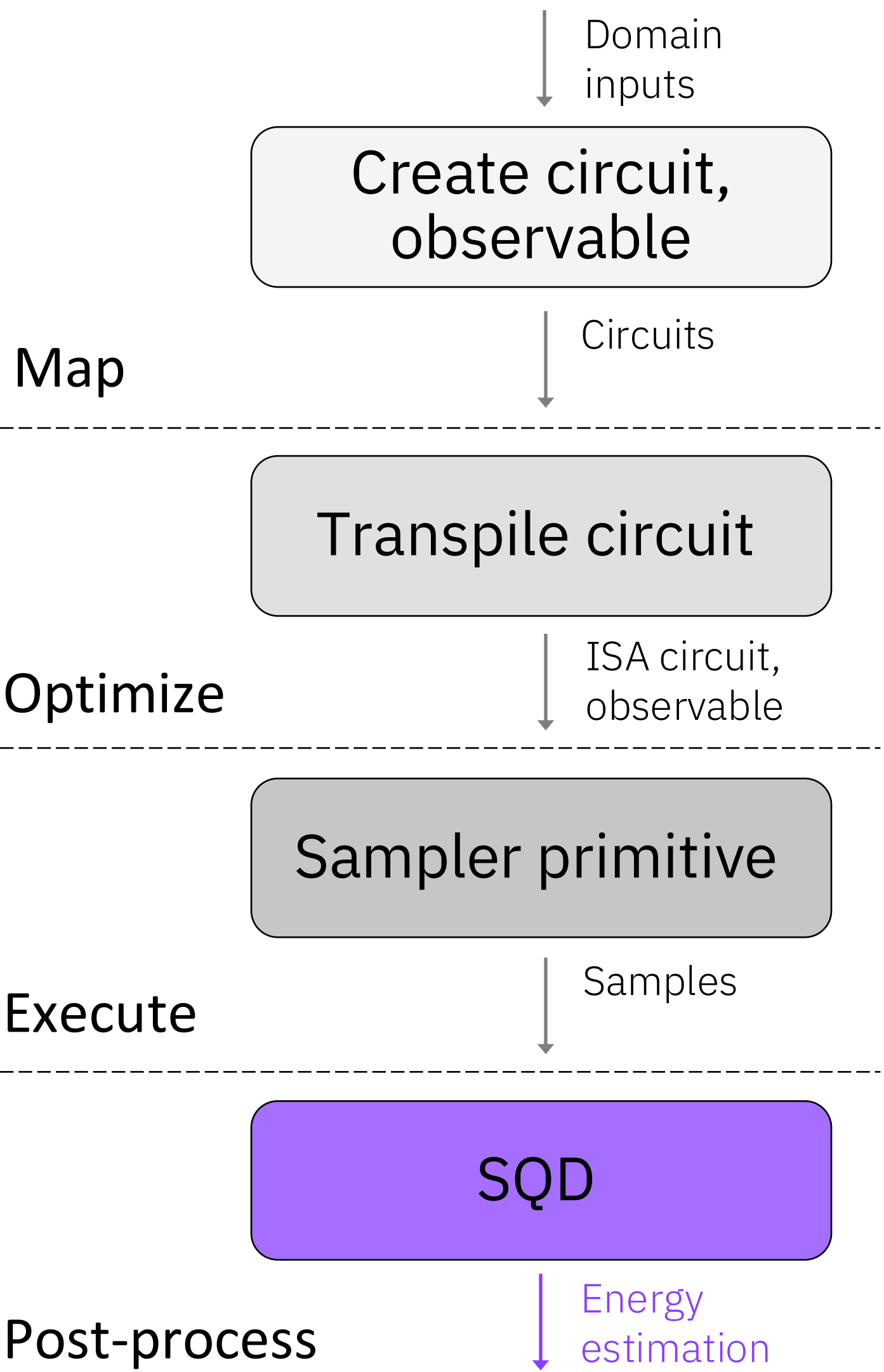
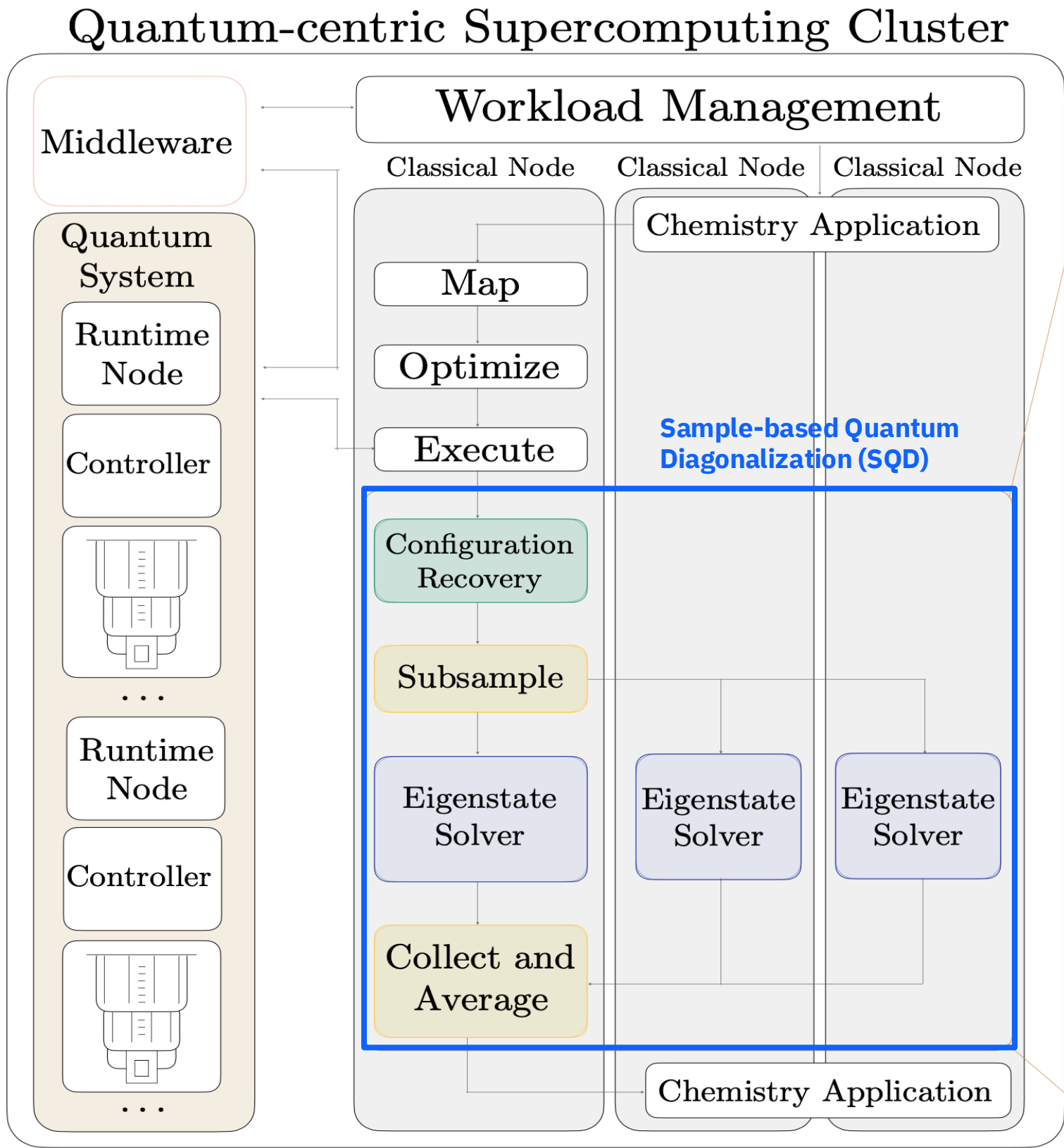
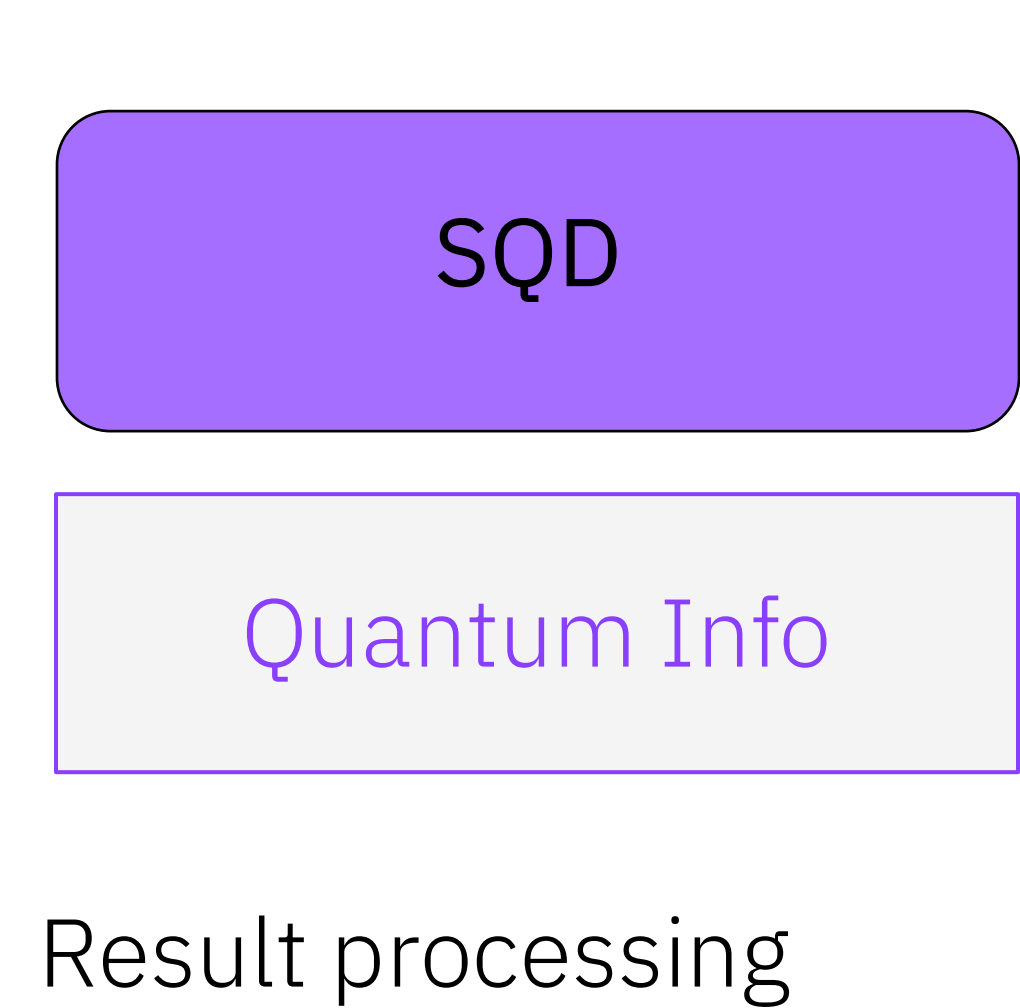
A collection of research capabilities developed as modular tools that can plug into a workflow to design new algorithms at the utility scale

Starting with multi-product formulas (MPF), approximate quantum compilation (AQC-Tensor), operator backpropagation (OBP), and sample-based quantum diagonalization (SQD).

AQC-Tensor	OBP		SQD
MPF	Circuit cutting		M3
Qiskit Circuit Library	Transpiler	Primitives	Quantum Info
Input: Domain inputs	Input: Circuits, observable	Input: ISA circuit, observable	Input: Expectation value/samples
Output: Circuits, observable	Output: ISA circuit, observable	Output: Expectation value/samples	Output: Data objects/visualizations
 Map	 Optimize	 Execute	 Post-Process

Qiskit Addon: Sampling-based quantum diagonalization

SQD classically post-processes noisy samples from a quantum processor to produce more accurate energy estimations



RIKEN
IBM Quantum
University of Colorado Boulder

Research
Science Advances vol. 11, no. 25 (2025) ➡ Development
<https://github.com/Qiskit/qiskit-addon-sqd>

IBM Quantum Platform

Qiskit Addons workflow: Sample based quantum diagonalization of a chemistry Hamiltonian

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/sample-based-quantum-diagonalization>

Lecture notes from QGSS:
<https://github.com/qiskit-community/qgss-2025-lecture-notes/blob/main/Day%205%20-%20Practical%20Quantum%20Algorithms%20by%20Joana%20Fraxanet%20Morales.pdf>

IBM Quantum Platform

DocumentationGuidesTutorialsAPI referenceAdditional resources

Tutorials

Get started

Overview

Hello world

Variational quantum eigensolver

CHSH inequality

Explore workflows toward advantage

Verifiable sampling algorithms

Sample-based quantum diagonalization of a chemistry Hamiltonian

Sample-based Krylov quantum diagonalization of a fermionic lattice model

Quantum approximate optimization algorithm

Advanced techniques for QAOA

Pauli correlation encoding to reduce Maxcut requirements

Observable estimation

Fault-tolerant algorithms

Leverage Qiskit capabilities

Workload optimization

Qiskit AI-powered transpiler service introduction

Transpilation optimizations with SABRE

Sample-based quantum diagonalization of a chemistry Hamiltonian

Usage estimate: under 1 minute on ibm_fez (NOTE: This is an estimate only. Your runtime may vary.)

Background

In this tutorial, we show how to post-process noisy quantum samples to find an approximation to the ground state of the nitrogen molecule N_2 at equilibrium bond length, using the [sample-based quantum diagonalization \(SQD\) algorithm](#), for samples taken from a 59-qubit quantum circuit (52 system qubits + 7 ancilla qubits). A Qiskit-based implementation is available in the [SQD Qiskit addons](#), more details can be found in the corresponding [docs](#) with a [simple example](#) to get started. SQD is a technique for finding eigenvalues and eigenvectors of quantum operators, such as a quantum system Hamiltonian, using quantum and distributed classical computing together. Classical distributed computing is used to process samples obtained from a quantum processor, and to project and diagonalize a target Hamiltonian in a subspace spanned by them. This allows SQD to be robust to samples corrupted by quantum noise and deal with large Hamiltonians, such as chemistry Hamiltonians with millions of interaction terms, beyond the reach of any exact diagonalization methods. An SQD-based workflow has the following steps:

1. Choose a circuit ansatz and apply it on a quantum computer to a reference state (in this case, the [Hartree-Fock](#) state).
2. Sample bitstrings from the resulting quantum state.
3. Run the *self-consistent configuration recovery* procedure on the bitstrings to obtain the approximation to the ground state.

SQD is known to work well when the target eigenstate is sparse: the wave function is supported in a set of basis states $S = \{i_1, i_2, \dots, i_N\}$ whose size decreases exponentially with the size of the problem.

On this page

Background

Quantum chemistry

Local unitary cluster Jastrow (LUCJ) ansatz

Self-consistent configuration recovery

SQD workflow diagram

Requirements

Setup

Step 1: Map classical inputs to a quantum problem

Step 2: Optimize problem for quantum hardware execution

Step 3: Execute using Qiskit primitives

Step 4: Post-process and return result in desired classical format

Visualize the results

Tutorial survey

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on GitHub

IBM Quantum Platform

Qiskit Addons workflow: Sample based quantum diagonalization of a chemistry Hamiltonian walkthrough

Link:
https://www.youtube.com/watch?v=0eTmqj5nf7c&list=PLOFEBZVS-VvoIfbpOb_geVnwFmbW6ij0m&index=5

IBM Quantum Platform

LearningComposer ↗

←

Quantum computing in practice

Overview

Lessons

Course introduction

Running quantum circuits

Utility-scale QAOA

Which problems are quantum computers good for?

Mapping

New

Running quantum circuits

Watch the video on quantum circuits and primitives from Olivia Lanes, or open the video in a separate window on [YouTube](#). ↗

Quantum Computing in practice

Lesson 02

Quantum Circuits

Lesson overview

This lesson will be a high-level look at the basics of running a utility-scale quantum computation, from the quantum hardware used to principles to consider when designing a quantum circuit. Ideally, by the end of this lesson, you'll know:

On this page

Lesson overview

Hardware – IBM quantum processors

Software: Qiskit and Qiskit Runtime

Quantum Circuits

Designing a quantum circuit: Qiskit patterns

Map

Optimize

Execute

Post-process

Conclusion

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on [GitHub](#). ↗

Advanced Techniques for QAOA

Link:
<https://quantum.cloud.ibm.com/docs/en/tutorials/advanced-techniques-for-qaoa>



IBM Quantum Platform

DocumentationGuidesTutorialsAPI referenceAdditional resources

Tutorials

Get started

Overview

Hello world

Variational quantum eigensolver

CHSH inequality

Explore workflows toward advantage

Verifiable sampling algorithms

Sample-based quantum diagonalization of a chemistry Hamiltonian

Sample-based Krylov quantum diagonalization of a fermionic lattice model

Quantum approximate optimization algorithm

Advanced techniques for QAOA

Pauli correlation encoding to reduce Maxcut requirements

Observable estimation

Krylov quantum diagonalization of lattice Hamiltonians

Nishimori phase transition

Ground-state energy estimation of the Heisenberg chain with VQE

Quantum kernel training

Advanced techniques for QAOA

Usage estimate: 25 minutes on IBM Sherbrooke (NOTE: This is an estimate only. Your runtime may vary.)

Background

This notebook introduces advanced techniques to improve the performance of the **Quantum Approximate Optimization Algorithm (QAOA)** with a large number of qubits. Please see [Solve utility-scale quantum optimization problems](#) for an introduction to QAOA.

The advanced techniques in this notebook include:

- SWAP strategy with SAT initial mapping:** This is a specifically designed transpiler pass for QAOA that uses a SWAP strategy and a SAT solver together to improve the selection of which physical qubits on the QPU to use. The SWAP strategy exploits the commutativity of the QAOA operators to reorder gates so that layers of SWAP gates can be simultaneously executed, thus reducing the depth of the circuit [1]. The SAT solver is used to find an initial mapping that minimizes the number of SWAP operations needed to map the qubits in the circuit to the physical qubits on the device [2].
- CVaR cost function:** Typically the expected value of the cost Hamiltonian is used as the cost function for QAOA, but as was shown in [3], focusing on the tail of the distribution, rather than the expected value, can improve the performance of QAOA for combinatorial optimization problems. The CVaR accomplishes this. For a given set of shots with corresponding objective values of the considered optimization problem, the Conditional Value at Risk (CVaR) with confidence level $\alpha \in [0, 1]$ is defined as the average of the α best shots [3]. Thus, $\alpha = 1$ corresponds to the standard expected value, while $\alpha = 0$ corresponds to the minimum of the given shots, and $\alpha \in (0, 1)$ is a tradeoff between focusing on better shots, while still applying some averaging to smooth out the optimization landscape. Additionally, the CVaR can be used as an error mitigation technique to improve the quality of the objective value estimation [4].

On this page

Background

Requirements

Setup

Step 1: Map classical inputs to a quantum problem

Max-Cut Problem

Graph \rightarrow Hamiltonian

Hamiltonian \rightarrow quantum circuit

Step 2: Optimize problem for quantum hardware execution

SWAP strategy with the SAT initial mapping

Step 3: Execute using Qiskit primitives

Define a CVaR cost function

Step 4: Post-process and return result in desired classical format

References

Tutorial survey

Download notebook

Was this page helpful?

Yes

No

Report a bug or request content on GitHub

IBM Quantum Learning


Learn the basics of quantum computing and how to solve real-world problems with IBM Quantum services and systems

Courses, tutorials, and educational resources by leading quantum experts.



Quantum learning

Kickstart your quantum learning journey with a selection of courses designed to help you learn the basics or explore more focused topics. If you're an instructor, explore content specifically tailored to incorporating quantum in the classroom.



Foundations

Courses to learn about quantum information and how quantum computing works, from the basics onward.

Quantum information and computation I

Basics of quantum information

Learn about quantum information, from states and measurements to quantum circuits and entanglement.

Course

Quantum information and computation II

Fundamentals of quantum algorithms

Learn how quantum algorithms beat classical algorithms for problems including integer factoring and search.

Course

Quantum information and computation III

General formulation of quantum information

Dive deeper into quantum information, including density matrices, channels, and general measurements.

Course

Quantum information and computation IV

Foundations of quantum error correction

Learn how quantum computations can be protected against noise through quantum error correcting codes and fault tolerance.

Course New lesson

Quantum computing in practice

Learn potential use cases and best practices for experimenting with quantum processors having 100+ qubits.

Course New lesson

Focused topics

Continue your learning journey by diving into more focused topics related to quantum computing.

Quantum machine learning

Learn to leverage the power of quantum computing in machine learning methods.

Course New

Variational algorithm design

An overview of variational algorithms: hybrid classical quantum algorithms.

Course

Quantum chemistry with VQE

An introduction to VQE that covers basic building blocks and applications.

Course

Quantum diagonalization algorithms

Multiple quantum approaches to matrix diagonalization are explored, including VQE, QKD, SKD, and variations of these.

Course New

Utility-scale quantum computing

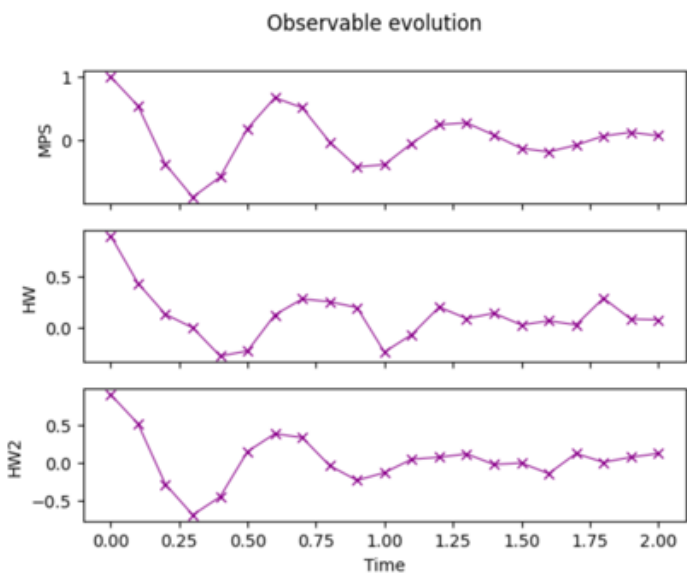
A collection of learning assets from a 14-lesson course on utility-scale quantum computing.

Course

Utility Scale Modules

1D Transverse Ising Model (70 qubits x 80 ent. gates)

Utility I



This module explores Quantum simulation of a 1D transverse Ising model on linear lattices.

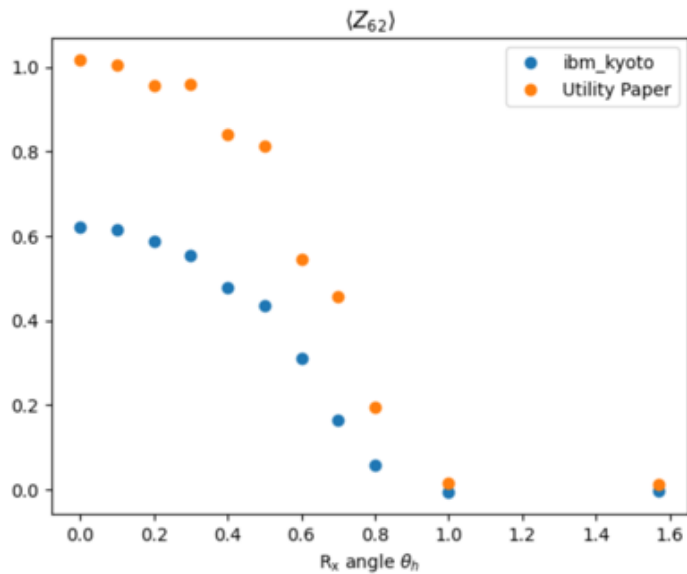
Performs time-evolution simulation using Statevector and MPS simulators first with N=20, then scales to N=70 on a real hardware to compare with the N=20 simulator results.

Focuses on teaching the workflow (Map>Optimize>Run>Post-process) in the process.

.

Nature Paper Simulation (127 qubits x 60 ent. gates)

Utility II



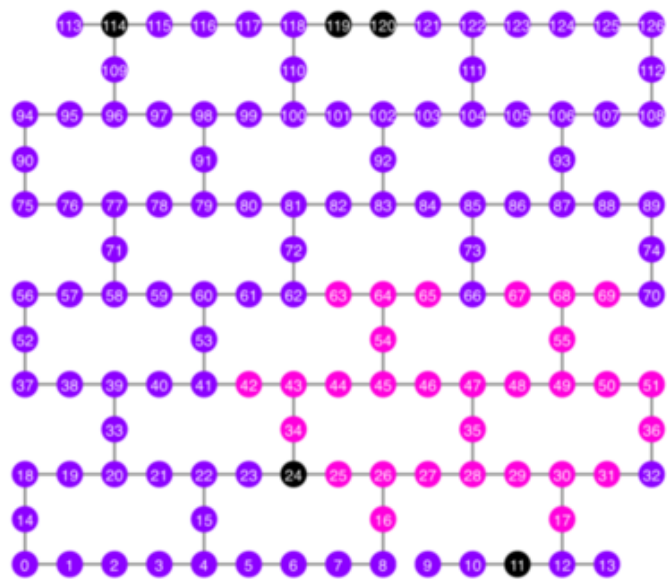
This module walks learners through reproducing the Nature paper with a focus on implementation of the problem without any error mitigation applied at first.

Learners will compare results to that of the Nature paper that applied error mitigation techniques.

Finally, students can explore an option to try PEA (Probabilistic Error Amplification) to run their circuits and apply error mitigation in post-processing.

Long-entanglement with GHZ State (127 qubits)

Utility III



This module can be utilized as a group project or an in-class challenge. The goal is to build a GHZ circuit with 20 qubits or more that meets a certain fidelity criteria (i.e., GHZ state fidelity > 0.5)

Several ideas are presented as an example such as finding the best qubit chain based on 2 gate errors, finding a balanced tree of qubits, applying error mitigation options in Sampler V2.

The module encourages students to produce their own ideas while exploring the suggested options first.




IBM Quantum Learning

Learn the basics of quantum computing and how to solve real-world problems with IBM Quantum services and systems

Courses, tutorials, and educational resources by leading quantum experts.

Quantum learning

Kickstart your quantum learning journey with a selection of courses designed to help you learn the basics or explore more focused topics. If you're an instructor, explore content specifically tailored to incorporating quantum in the classroom.



Foundations

Courses to learn about quantum information and how quantum computing works, from the basics onward.

Quantum information and computation I

Basics of quantum information

Learn about quantum information, from states and measurements to quantum circuits and entanglement.

Course

Quantum information and computation II

Fundamentals of quantum algorithms

Learn how quantum algorithms beat classical algorithms for problems including integer factoring and search.

Course

Quantum information and computation III

General formulation of quantum information

Dive deeper into quantum information, including density matrices, channels, and general measurements.

Course

Quantum information and computation IV

Foundations of quantum error correction

Learn how quantum computations can be protected against noise through quantum error correcting codes and fault tolerance.

Course New lesson

Quantum computing in practice

Learn potential use cases and best practices for experimenting with quantum processors having 100+ qubits.

Course New lesson

Focused topics

Continue your learning journey by diving into more focused topics related to quantum computing.

Quantum machine learning

Learn to leverage the power of quantum computing in machine learning methods.

Course New

Variational algorithm design

An overview of variational algorithms: hybrid classical quantum algorithms.

Course

Quantum chemistry with VQE

An introduction to VQE that covers basic building blocks and applications.

Course

Quantum diagonalization algorithms

Multiple quantum approaches to matrix diagonalization are explored, including VQE, QKD, SKD, and variations of these.

Course New

Utility-scale quantum computing

A collection of learning assets from a 14-lesson course on utility-scale quantum computing.

Course



Qiskit Quantum Seminar

Qiskit
153 videos 29,763 views Updated today



▶ Play all **↻ Shuffle**

Stay up to date with the latest academic and research topics in the quantum community by joining our live discussions every Friday at 12PM EDT. Tune in to gain insights from experts and engage with a community of quantum enthusiasts!

- 1

Qiskit Quantum Seminar
Anushya Chandran

1:05:40

Efficient classical shadow tomography with number conservation with Anushya Chandran
Qiskit • 116 views • Streamed 1 hour ago
- 2

Qiskit Quantum Seminar
Elisa Bäumer

1:04:34

Efficient Long-Range Entanglement using Dynamic Circuits with Elisa Bäumer
Qiskit • 2.4K views • Streamed 3 weeks ago
- 3

Qiskit Quantum Seminar
Amira Abbas

56:03

On quantum backpropagation and information reuse | Qiskit Quantum Seminar with Amira Abbas
Qiskit • 2.6K views • Streamed 4 weeks ago
- 4

Qiskit Quantum Seminar
Ashley Montanaro

1:00:58

Near-Term Quantum Algorithms for Optimization with Ashley Montanaro
Qiskit • 2.4K views • Streamed 1 month ago
- 5

Qiskit Quantum Seminar
Matthew Fisher

1:05:07

Quantum Many-body theory in the Quantum Information era with Matthew Fisher |Qiskit Quantum Seminar
Qiskit • 2.8K views • Streamed 1 month ago
- 6

Qiskit Quantum Seminar
Frank Pollmann

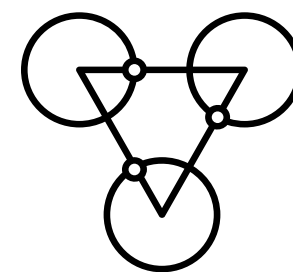
1:03:10

Realization and Characterization of Topological Phases on Quantum Processors
Qiskit • 2.9K views • Streamed 2 months ago

Some overarching suggestions

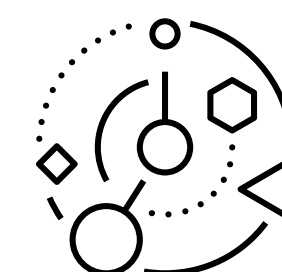
Multiple entry ramps

Offer a blend of engaging channels: live events, self-paced tutorials, and peer support so learners can start where they feel comfortable.



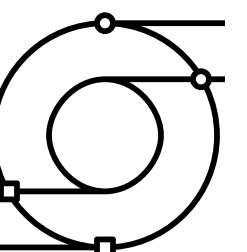
Reusable artifacts

Recordings, notebooks, and FAQs to turn one-off sessions into long-term resources that participants can pick up later.



Small, continuous feedback channels

Open communications channels to preventing drift between what we teach and what participants need.



Thank you

© 2024 International Business Machines Corporation

IBM, IBM logo, and IBM QUANTUM are trademarks of IBM Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on ibm.com/trademark.

THIS DOCUMENT IS DISTRIBUTED “AS IS” WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT, SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY.

Client examples are presented as illustrations of how those clients have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

Not all offerings are available in every country in which IBM operates.

Any statements regarding IBM’s future direction, intent or product plans are subject to change or withdrawal without notice.

IBM