



IBM Quantum Experience Overview : Portal, API key, Docs, Tutos, Examples, and Composer

IBM Client Center Montpellier

JM Torres | torresjm@fr.ibm.com | 2-4 décembre 2019

Tous

Images

Actualités

Vidéos

Shopping

Plus

Paramètres

Outils

Environ 195 000 000 résultats (0,52 secondes)

IBM Q - Quantum Computing - IBM Research

www.research.ibm.com ▾ Traduire cette page

IBM Q is an industry first initiative to build universal quantum computers for business and science. Our cross-disciplinary team is developing scalable quantum ...

[IBM Q Experience](#) · [What is quantum](#) [Learn the ...](#) · [IBM Q - Network](#) · [Jobs at IBM](#)

Experience - IBM Q

www.ibm.com ▾ Traduire cette page

IBM Q Experience is quantum on the cloud. Accelerate your research and applications with the next generation of the leading quantum cloud services and ...

IBM Q Experience

quantumexperience.ng.bluemix.net ▾ Traduire cette page

Run algorithms and experiments on IBM's quantum processor via IBM Cloud.

CES 2019 : Est-ce qu'IBM vient de dévoiler le premier ordinateur ...

www.zdnet.fr ▾

9 janv. 2019 - Le système Q System One a une conception modulaire et des composants conçus pour minimiser les interférences. Le plan d'IBM est tout ...

IBM Q Experience - Wikipedia

en.wikipedia.org ▾ Traduire cette page

IBM Q Experience



Traduit de l'anglais - IBM Q Experience est une plate-forme en ligne qui donne aux utilisateurs du grand public accès à un ensemble de prototypes de processeurs quantiques d'IBM via le cloud, un forum en ligne sur Internet ... [Wikipédia \(anglais\)](#)

[Afficher la description d'origine](#) ▾[Commentaires](#)

Afficher les résultats pour

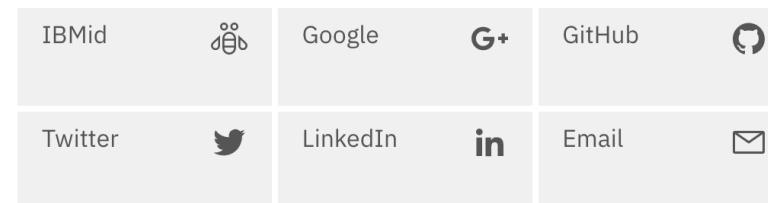
IBM Q System One (Ordinateur)

IBM Q System One est le tout premier ordinateur quantique commercial au ...



Sign in to IBM Q Experience

What is IBM Q Experience? [Learn more](#)





Welcome JEAN-MICHEL TORRES

Your accounts

IBM Q Hub France

IBM Q Network member

[See more](#)



Circuit Composer

Explore the graphical interface
for creating and testing circuits

[Create a circuit →](#)

Qiskit Notebooks

Create your first notebook and
start using Qiskit

[Create a notebook →](#)

Pending results (1)

Status

Id

Account

Backend

Your backends (7)

These are the quantum systems and
simulators that you have access to.

[Got it!](#)

online

ibmq_poughkeepsie (20 qubits)



Queue: 0 runs

online

ibmq_20_tokyo (20 qubits)



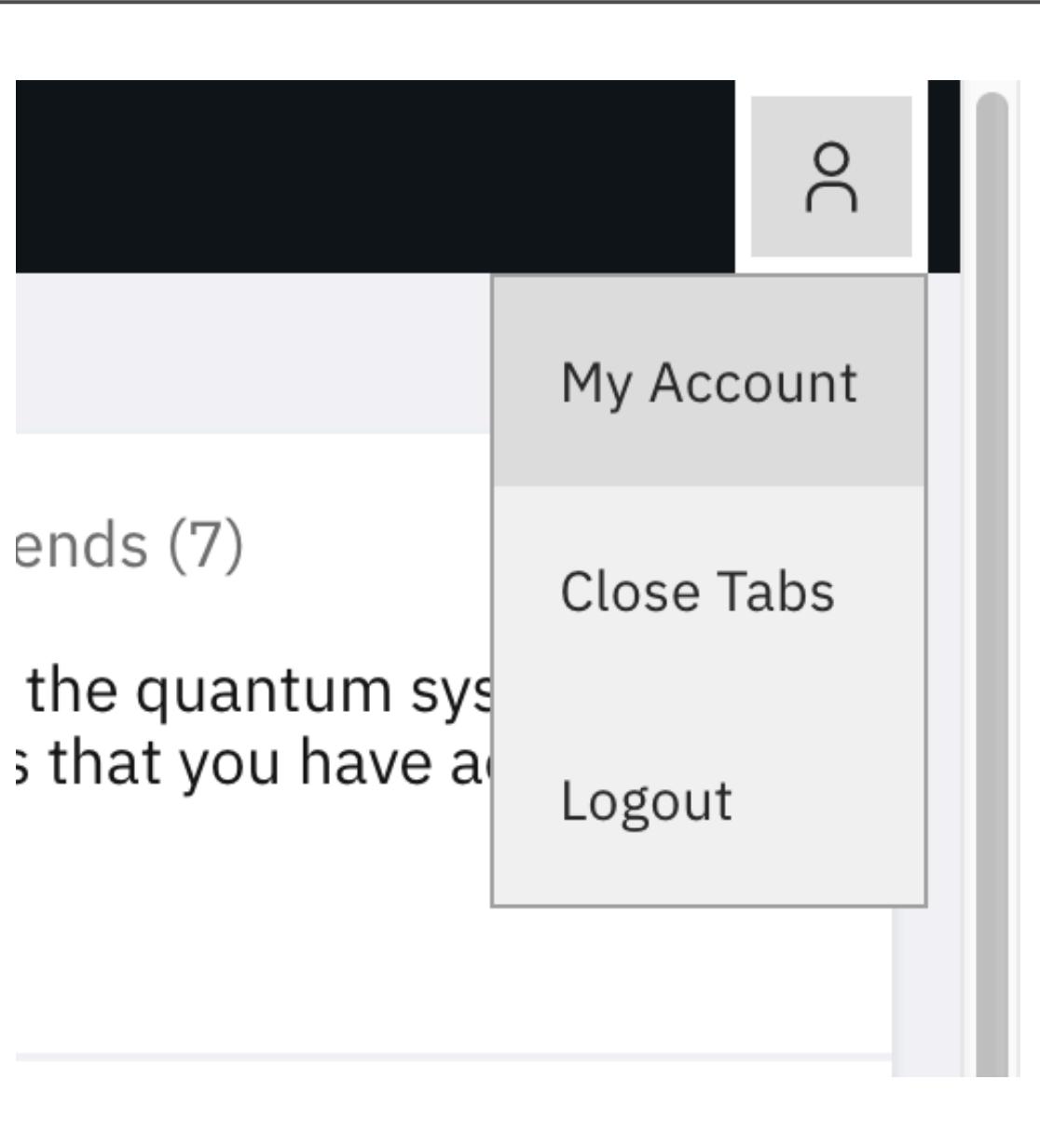
Queue: 0 runs

online

ibmq_16_melbourne (14 qubits)



Queue: 132 runs

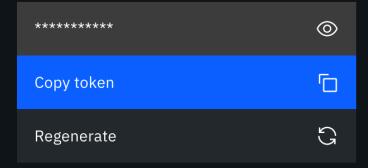


Qiskit in IBM Q Experience

- No setup required
- Create Qiskit notebook [here](#)

Qiskit in local environment

1. Install [Qiskit](#)
2. Follow the instructions to [access the IBM Q Devices from Qiskit](#), this is your API Token:



A screenshot of a dark-themed application window titled "Qiskit in local environment". It displays two sections: "Qiskit in IBM Q Experience" and "Qiskit in local environment". The "Qiskit in local environment" section contains two numbered steps. Step 1 is "Install Qiskit". Step 2 is "Follow the instructions to access the IBM Q Devices from Qiskit, this is your API Token:". Below this text is a screenshot of a token management interface. The interface shows a token value consisting of several asterisks ("*****") followed by an eye icon. A blue rectangular button labeled "Copy token" is highlighted. To the right of the token are two more icons: a refresh symbol and a circular arrow symbol.



Welcome to Quantum.

Qiskit is an open-source quantum computing software development kit for leveraging today's quantum processors in research, education, and business

 Join the Slack community

Qiskit 0.12

Featuring faster circuit transpilation times, a wider variety of classical simulation methods, more device characterization routines, and a host of new quantum applications, the premiere quantum computing software framework just got even better.

[Learn more](#)

Coding with Qiskit

Would you like to learn how to code a quantum computer? Take a look at the Coding with Qiskit Video Series, where Abraham Asfaw explains everything you need to know. Starting with installing Qiskit, to investigating the latest algorithms and research topics.

[Watch it](#)

Qiskit

English Search docs... Installing Qiskit Getting Started with Qiskit The Qiskit Elements Development Strategy Contributing to Qiskit Release Notes Frequently Asked Questions API References 

Qiskit API documentation

Qiskit is an open-source framework for working with quantum computers at the level of circuits, pulses, and algorithms.

A central goal of Qiskit is to build a software stack that makes it easy for anyone to use quantum computers. However, Qiskit also aims to facilitate research on the most important open issues facing quantum computation today.

You can use Qiskit to easily design experiments and run them on simulators and real quantum computers.

Qiskit consists of four foundational elements:

- [Qiskit Terra](#): Composing quantum programs at the level of circuits and pulses with the code foundation.
- [Qiskit Aer](#): Accelerating development via simulators, emulators, and debuggers
- [Qiskit Ignis](#): Addressing noise and errors
- [Qiskit Aqua](#): Building algorithms and applications

 Note

This is the Qiskit API documentation. For detailed workflow examples, please sign in to the IBM Q Experience to [explore Qiskit tutorials](#).

Qiskit

English ▾

Search docs...



Installing Qiskit

Requirements

Install

Access IBM Q Systems

Checking Which Version is Installed

Getting Started with Qiskit

The Qiskit Elements

Development Strategy

Contributing to Qiskit

Release Notes

Frequently Asked Questions

Installing Qiskit

Requirements

Qiskit supports Python 3.5 or later.

We recommend installing [Anaconda](#), a cross-platform Python distribution for scientific computing. Jupyter, included in Anaconda, is recommended for interacting with Qiskit.

Qiskit is tested and supported on the following 64-bit systems:

- Ubuntu 16.04 or later
- macOS 10.12.6 or later
- Windows 7 or later

Using Qiskit on Windows requires VC++ runtime components. We recommend one of the following:

- [Microsoft Visual C++ Redistributable for Visual Studio 2017](#)
- [Microsoft Visual C++ Redistributable for Visual Studio 2015](#)

Install

We recommend using Python virtual environments to cleanly separate Qiskit from other applications and improve your experience.



Welcome JEAN-MICHEL TORRES

Your accounts

IBM Q Hub France

IBM Q Network member

[See more](#)



Circuit Composer

Explore the graphical interface
for creating and testing circuits

[Create a circuit →](#)

Qiskit Notebooks

Create your first notebook and
start using Qiskit

[Create a notebook →](#)

Pending results (1)

Status

Id

Account

Backend

Your backends (7)

These are the quantum system
simulators that you have acces

[Got it!](#)

online

ibmq_poughkeepsie (20 qubits)



Queue: 0 runs

online

ibmq_20_tokyo (20 qubits)



Queue: 0 runs

online

ibmq_16_melbourne (14 qubits)



Queue: 132 runs

ibmq_poughkeepsie v1.2.0

X



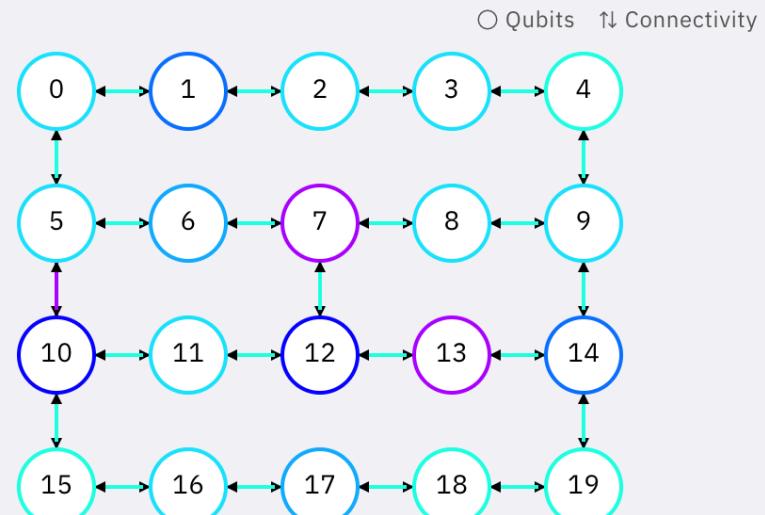
Queue: 0 runs

Accounts:

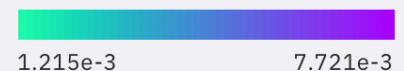
Hub: ibm-q-france

Group: quantum

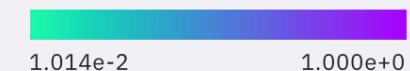
Project: onboarding-test



Single-qubit U3 error rate



CNOT error rate



Download Calibrations ↓

Qubits

20

Online since

2018-08-29

Basis gates

u1, u2, u3, cx, id



Welcome JEAN-MICHEL TORRES

Your accounts

IBM Q Hub France

IBM Q Network member

[See more](#)



Circuit Composer

Explore the graphical interface
for creating and testing circuits

[Create a circuit →](#)

Qiskit Notebooks

Create your first notebook and
start using Qiskit

[Create a notebook →](#)

Pending results (1)

Status

Id

Account

Backend

Your backends (7)

These are the quantum systems and
simulators that you have access to.

[Got it!](#)



ibmq_poughkeepsie (20 qubits)



Queue: 0 runs



ibmq_20_tokyo (20 qubits)



Queue: 0 runs



ibmq_16_melbourne (14 qubits)



Queue: 132 runs

[Dashboard](#)

TOOLS

[Results](#)[Circuit Composer](#)[Qiskit Notebooks](#)

RESOURCES

[Documentation & Support](#)e
ICHEL

New here? Get started with the IBM Q Experience!



Circuit Composer

Explore the graphical interface for creating and testing circuits

[Create a circuit →](#)

Qiskit Notebooks

Create your first notebook and start using Qiskit

[Create a notebook →](#)

Your backends (7)

These are the quantum systems and simulators that you have access to.

[Got it!](#) [online](#)**ibmq_poughkeepsie** (20 qubits)

Queue: 0 runs

 [online](#)**ibmq_20_tokyo** (20 qubits)

Queue: 0 runs

 [online](#)**ibmq_16_melbourne** (14 qubits)

Queue: 132 runs



Save Clear Help

Untitled Experiment

Unsaved changes 📁

Save your experiment before running it

Run



Composer help



The circuit composer is a tool that allows you to visually learn how to create quantum circuits. Here are some resources to get you started.

[Composer guide](#)[Gates overview](#)

Circuit composer

[Gates overview](#)

Gates



Operations



Subroutines

[+ Add](#)

q[0] |0⟩

q[1] |0⟩

q[2] |0⟩

q[3] |0⟩

q[4] |0⟩

c5



Overview of Quantum Gates

Overview of Quantum Gates

H gate

The H or Hadamard gate rotates the states $|0\rangle$ and $|1\rangle$ to $|+\rangle$ and $|-\rangle$, respectively. It is useful for making superpositions. As a Clifford gate, it is useful for moving information between the x and z bases.

Composer reference



Qasm reference

```
h q[0];
```

Learn more

[Quantum gates: Hadamard and S](#)

Contents

H gate

CX gate

Id gate

U3 gate

U2 gate

U1 gate

Rx gate

Ry gate

Rz gate

X gate

CX gate

The controlled-X gate is also known as the controlled-NOT. It acts on a pair of qubits, with one acting as ‘control’ and the other as ‘target’. It performs an X on the target whenever the control is in state $|1\rangle$. If the control qubit is in a superposition, this gate creates entanglement.

Composer reference



Qasm reference

```
cx q[0], q[1];
```

Learn more



Introduction to Quantum Circuits

Pauli matrices and the Bloch sphere

States for many qubits

Quantum gates

Qubits as physical systems

Putting gates to use

Basic circuit identities

Entanglement and Bell Tests

GHZ States

Universality of quantum computation

Fun with matrices

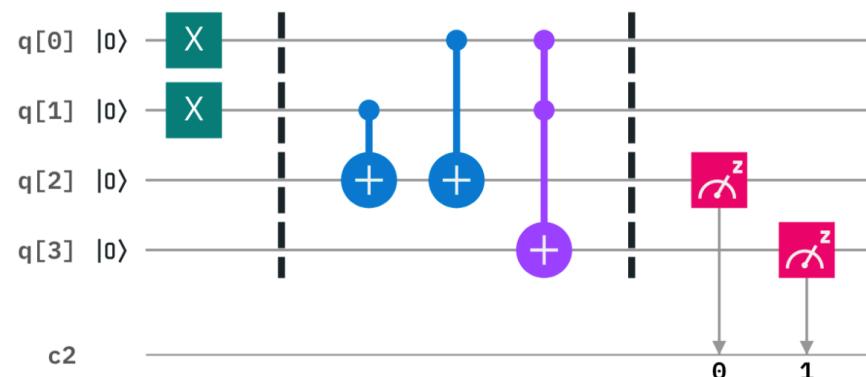
The standard gate set

The meaning of universality

Proving universality

Getting started with quantum circuits

If you think quantum mechanics sounds challenging, you are not alone. All of our intuitions are based on day-to-day experiences, and so are better at understanding the behavior of balls and bananas than atoms or electrons. Though quantum objects can seem random and chaotic at first, they just follow a different set of rules. Once we know what those rules are, we can use them to create new and powerful technology. Quantum computing will be the most revolutionary example of this.



A quantum circuit to calculate 1+1

Contents

Contributors



Save Clear Help

Untitled Experiment

Unused changes

Save your experiment before running it

Run



Composer help

The circuit composer is a tool that allows you to visually learn how to create quantum circuits. Here are some resources to get you started.

[Composer guide](#)[Gates overview](#)

Circuit composer

[Gates overview](#)

Gates



Operations



Subroutines

[+ Add](#)

q[0] |0>

q[1] |0>

q[2] |0>

q[3] |0>

q[4] |0>

c5

Circuit composer

[Gates overview](#)

Gates



Barrier



Operations



Subroutines

[+ Add](#)

Composer help X

The circuit composer is a tool that allows you to visually learn how to create quantum circuits. Here are some resources to get you started.

[Composer guide](#) ≡

[Gates overview](#)

Circuit composer

Gates



Barrier

Operations

[Gates overview](#)

Subroutines

+ Add



Visualizations

Statevector

Statevector

The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.

```
[ 0.707+0j, 0.707+0j, 0+0j, 0+0j ]
```

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates

Operations

Subroutines

+ Add

q[0] |0> — H

q[1] |0>

q[2] |0>

q[3] |0>

q[4] |0>

c5

Visualizations

Density Matrix

Re[ρ]

■ [-1, 0] ■ 0 ■ [0, 1]

00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Circuit composer

Gates

Operations

Subroutines

Barrier

Gates overview

+

|0>

IF

α^z

+ Add

q[0] |0> — H

q[1] |0>

q[2] |0>

q[3] |0>

q[4] |0>

c5

Visualizations

Measurement Probabilities

Re[ρ]

State

State 00000

probabilities: 50%

50%

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates



Gates overview

Barrier

Operations



Subroutines

+ Add

q[0] |0> — H

q[1] |0>

q[2] |0>

q[3] |0>

q[4] |0>

c5

Circuit editor

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[5];
creg c[5];
h q[0];
```

Circuit composer

Gates

The circuit diagram shows five horizontal lines representing qubits q[0] through q[4]. A single horizontal blue bar representing an H gate is positioned above q[0]. The other qubits are shown with empty horizontal lines below them.

Operations

Subroutines

Barrier

Gates overview

+ Add

☰

Visualizations

Statevector

The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.

```
[ 0.5+0j, 0.5+0j, 0.5+0j, 0.5+0j, 0+0j, 0+0j, 0+0j,
 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j,
 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j ]
```

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates

Operations

Subroutines

q[0] |0> — **H**

q[1] |0> — **H**

q[2] |0>

q[3] |0>

q[4] |0>

c5

Visualizations

Density Matrix

Re[p]

■ [-1, 0] ■ 0 ■ [0, 1]

00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Im[p]

Circuit composer

Gates

Operations

Subroutines

q[0] |0> — H

q[1] |0> — H

q[2] |0>

q[3] |0>

q[4] |0>

c5

Gates overview

Barrier

Visualizations

Measurement Probabilities

Re[p]
 $\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}$

State

25%

25%

25%

25%

The qubit 0 is the one that is furthest to the right on the state.

Gates overview

Gates

Barrier

Operations

Subroutines

q[0] $|0\rangle$ — H

q[1] $|0\rangle$ — H

q[2] $|0\rangle$ —

q[3] $|0\rangle$ —

q[4] $|0\rangle$ —

c5 —

Visualizations

Statevector

The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.

```
[ 0.177+0j, 0.177+0j, 0.177+0j, 0.177+0j, 0.177+0j,
0.177+0j, 0.177+0j ]
```

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates

H	$+$	ID	$U3$	$U2$	$U1$	Rx	Ry	Rz	X	Y	Z	S	S^\dagger	T	T^\dagger	cH
cY	cZ	cRz	$cU1$	$cU3$	$+$	\otimes										

Operations Subroutines

$|0\rangle$ IF α^z $nG0$ + Add

q[0] $|0\rangle$ — H
q[1] $|0\rangle$ — H
q[2] $|0\rangle$ — H
q[3] $|0\rangle$ — H
q[4] $|0\rangle$ — H

c5

Visualizations

Density Matrix

Re[ρ]

■ [-1, 0] ■ 0 ■ [0, 1]

00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

Gates

Operations

Subroutines

Gates overview

Barrier

Circuit composer

Gates

Operations

Subroutines

q[0] |0> — H

q[1] |0> — H

q[2] |0> — H

q[3] |0> — H

q[4] |0> — H

c5

Circuit editor

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[2];
creg c[2];
h q[0];
h q[1];
```

Circuit composer

Gates

Operations

Subroutines

+ Add

Pending results

No results available

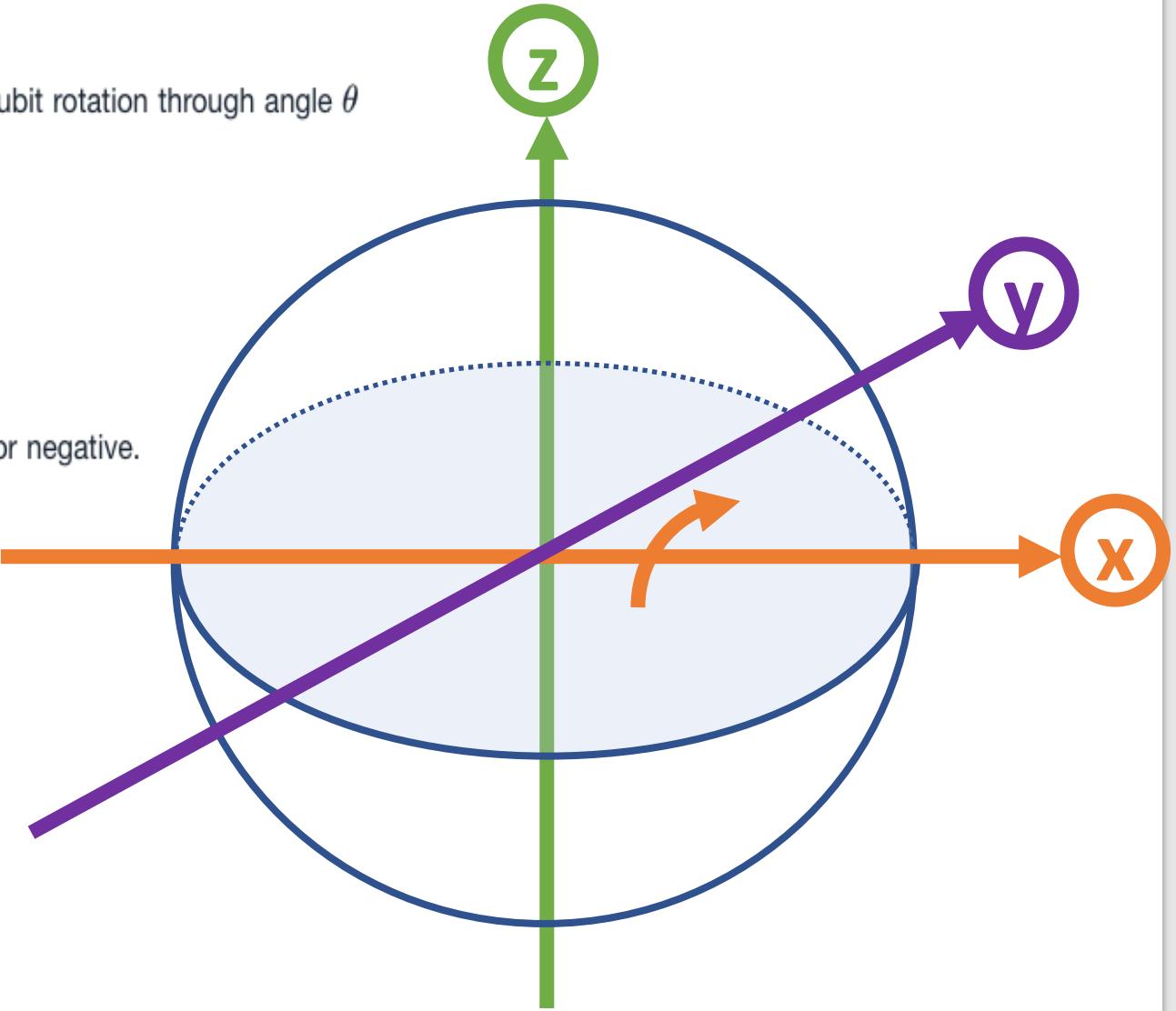
Rx Gate

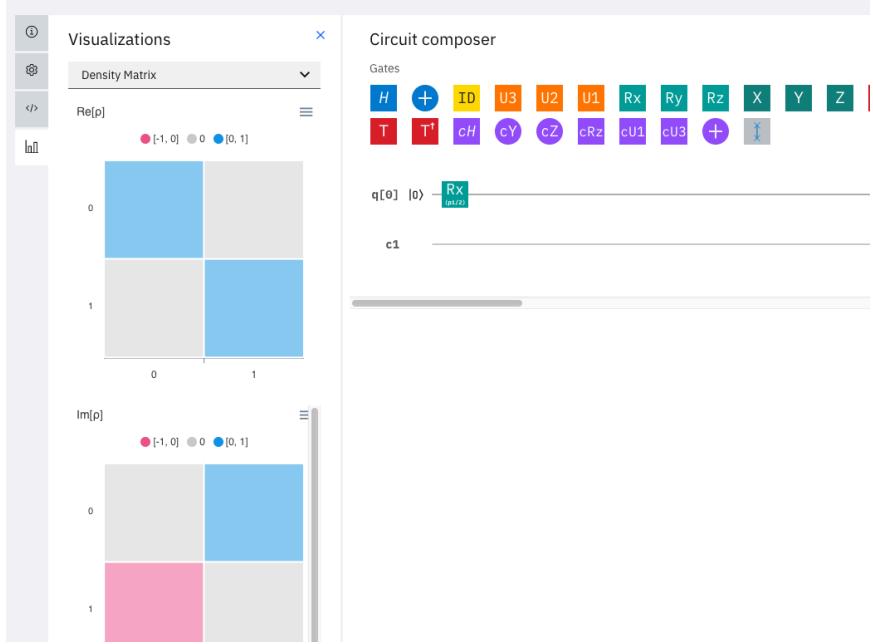
The Rx gate is one of the [Rotation operators](#). The Rx gate is a single-qubit rotation through angle θ (radians) around the x-axis.

$$R_x(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i \sin\left(\frac{\theta}{2}\right) \\ -i \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

The angle of rotation must be specified in radians and can be positive or negative.

`rx(θ) q[θ];`





$$R_X\left(\frac{\pi}{2}\right) = \begin{pmatrix} \cos\frac{\pi}{4} & -i \sin\frac{\pi}{4} \\ -i \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} \\ \frac{-i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$|\psi\rangle = R_X\left(\frac{\pi}{2}\right)|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} \\ \frac{-i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -i \end{pmatrix}$$

$$|\psi\rangle\langle\psi| = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -i \end{pmatrix} \times \begin{pmatrix} 1 & i \\ -i & \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix}$$



Visualizations

X

Measurement Probabilities



Re[ρ]



The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates



Your circuit > rx

Subroutine params
theta

$\pi/4$



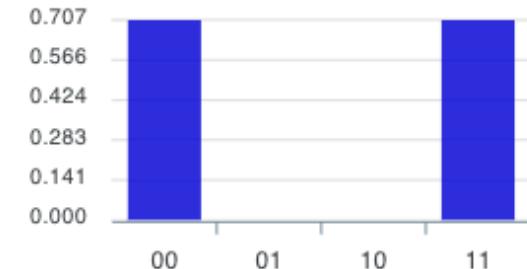
Visualizations



Statevector



Statevector



The height of the bar is the complex modulus.

The color of the bar is based in the complex argument or phase.

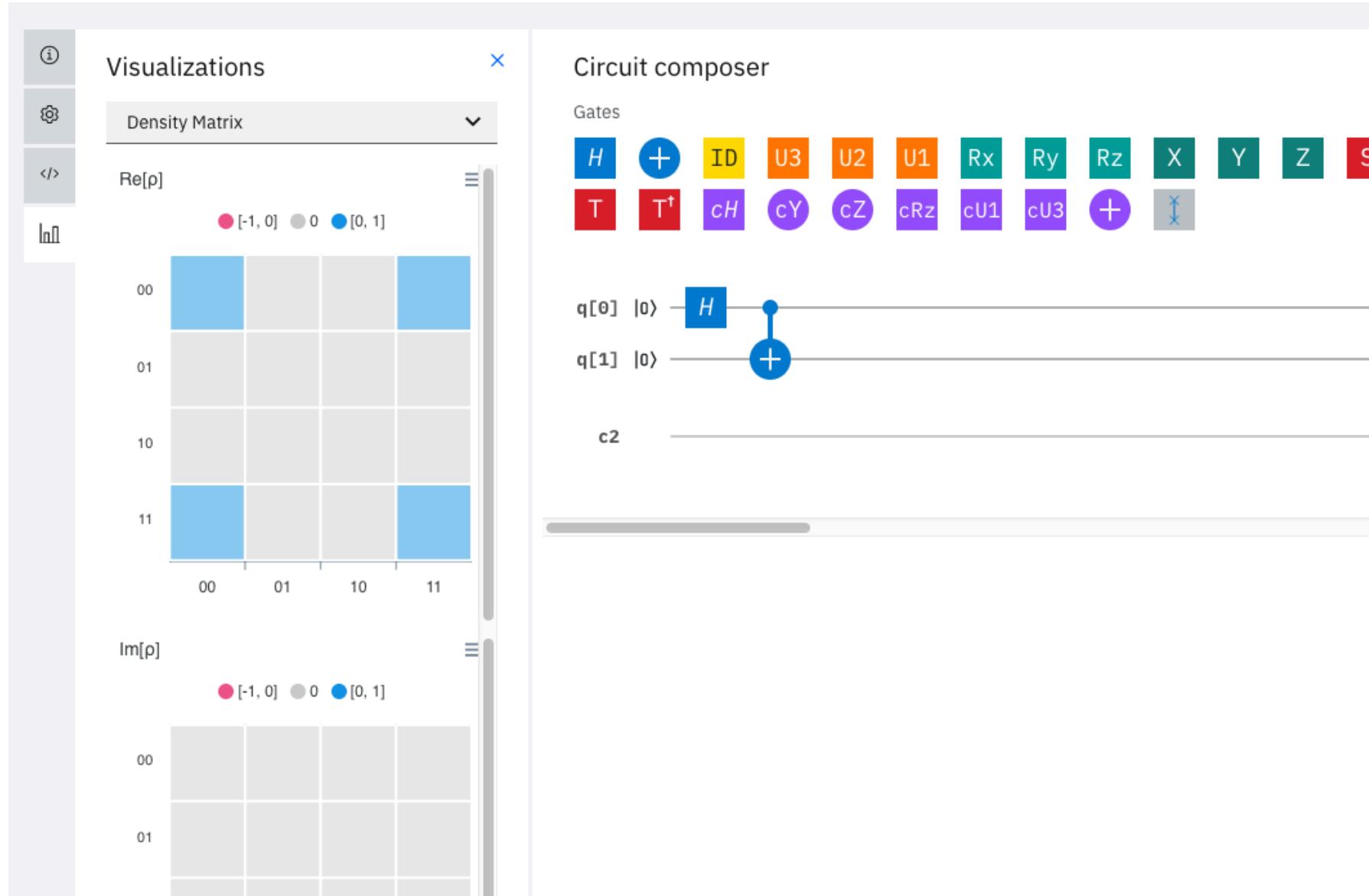
```
[ 0.707+0j, 0+0j, 0+0j, 0.707+0j ]
```

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates







Visualizations



Measurement Probabilities



Ref[0]

State 00



State

probabilities: 50%

50%

50%

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates



$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow{H} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

- Applying CNOT(0,1):

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$DM = \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \times (1 \ 0 \ 0 \ 1) = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

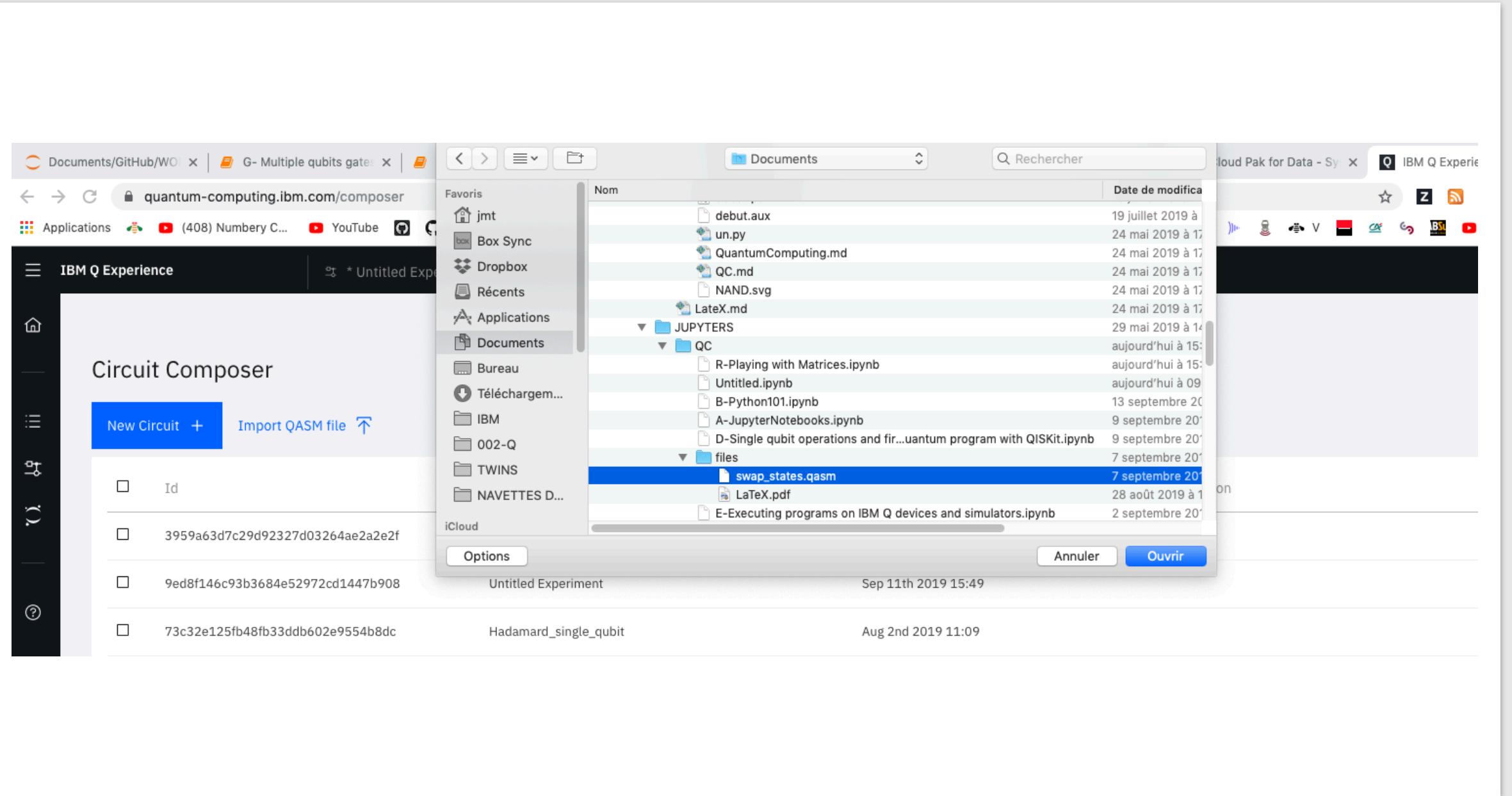


Circuit Composer

New Circuit +

Import QASM file ↴

<input type="checkbox"/>	Id	Name	Updated	Description
<input type="checkbox"/>	3959a63d7c29d92327d03264ae2a2e2f	Untitled Experiment	Sep 12th 2019 16:38	
<input type="checkbox"/>	9ed8f146c93b3684e52972cd1447b908	Untitled Experiment	Sep 11th 2019 15:49	
<input type="checkbox"/>	73c32e125fb48fb33ddb602e9554b8dc	Hadamard_single_qubit	Aug 2nd 2019 11:09	
<input type="checkbox"/>	19f62ac3fa8b327c80ceafb33a9d9985	bell_state_1	Aug 27th 2019 11:05	
<input type="checkbox"/>	c69fbe2ef7254197b79bdbd6e1727a43	Untitled Experiment	Jul 26th 2019 11:35	
<input type="checkbox"/>	5d596b76afe3a8af5820643f445323a5	Untitled Experiment	Jul 26th 2019 10:35	



Circuit Composer

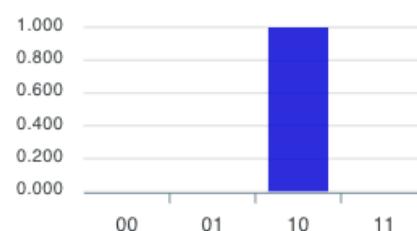
New Circuit +

Import QASM file ↴

<input type="checkbox"/>	Id	Name	Updated	Description
<input type="checkbox"/>	10fd00c545cc38a299373b0d98f5f38a	swap_states	Sep 16th 2019 16:33	
<input type="checkbox"/>	3959a63d7c29d92327d03264ae2a2e2f	Untitled Experiment	Sep 12th 2019 16:38	
<input type="checkbox"/>	9ed8f146c93b3684e52972cd1447b908	Untitled Experiment	Sep 11th 2019 15:49	
<input type="checkbox"/>	73c32e125fb48fb33ddb602e9554b8dc	Hadamard_single_qubit	Aug 2nd 2019 11:09	
<input type="checkbox"/>	19f62ac3fa8b327c80ceafb33a9d9985	bell_state_1	Aug 27th 2019 11:05	
<input type="checkbox"/>	c69fbe2ef7254197b79bdbd6e1727a43	Untitled Experiment	Jul 26th 2019 11:35	
<input type="checkbox"/>	5d596b76afe3a8af5820643f445323a5	Untitled Experiment	Jul 26th 2019 10:35	
<input type="checkbox"/>	23ce5d7c7a072e91402ddfe23632d2c6	Untitled Experiment	Jul 26th 2019 10:28	

Visualizations

Statevector



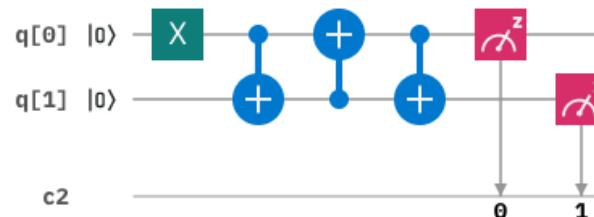
The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.

```
[ 0+0j, 0+0j, 1+0j, 0+0j ]
```

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates





Composer help



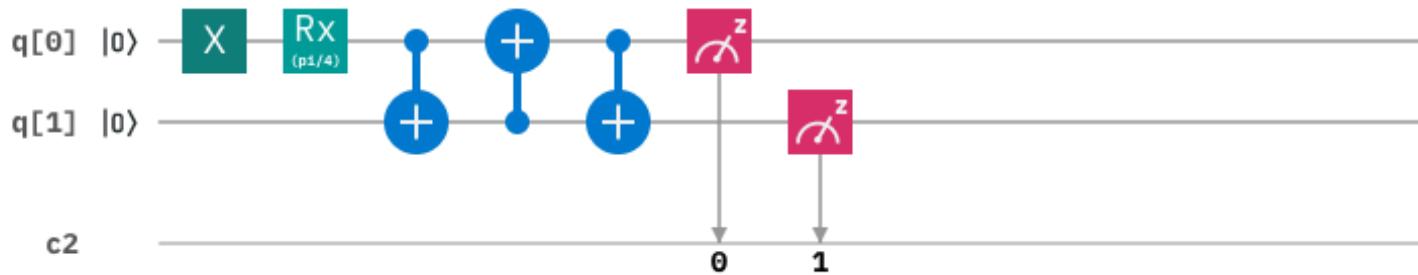
The circuit composer is a tool that allows you to visually learn how to create quantum circuits. Here are some resources to get you started.

[Composer guide](#)

[Gates overview](#)

Circuit composer

Gates



Visualizations

Statevector

State	Modulus
00⟩	0.370
01⟩	0.185
10⟩	0.924
11⟩	0.739

The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.

```
[ 0-0.383j, 0+0j, 0.924+0j, 0+0j ]
```

The qubit 0 is the one that is furthest to the right on the state.

Circuit composer

Gates

Operations

Subroutines

+ Add

X

Circuit composer

Gates

Barrier

Opera

Run your circuit

X

1. Select an available backend

Backends availability and functionality can vary depending on the provider.

ibmq_16_melbourne in ibm-q/open/main ▾

2. Select number of shots

Increase the number of shots to improve statistical accuracy.

1024 ▾

Cancel

Run →

Pending results (1)

ibmq_16_melbourne

1024 shots

a few seconds ago

Pending results (2)

ibmq_qasm_simulator

1024 shots

a few seconds ago

ibmq_16_melbourne

1024 shots

2 minutes ago

Circuit "swap_states" was sent to ibmq_qasm_simulator. 04:50



Pending results (1)

ibmq_16_melbourne

1024 shots

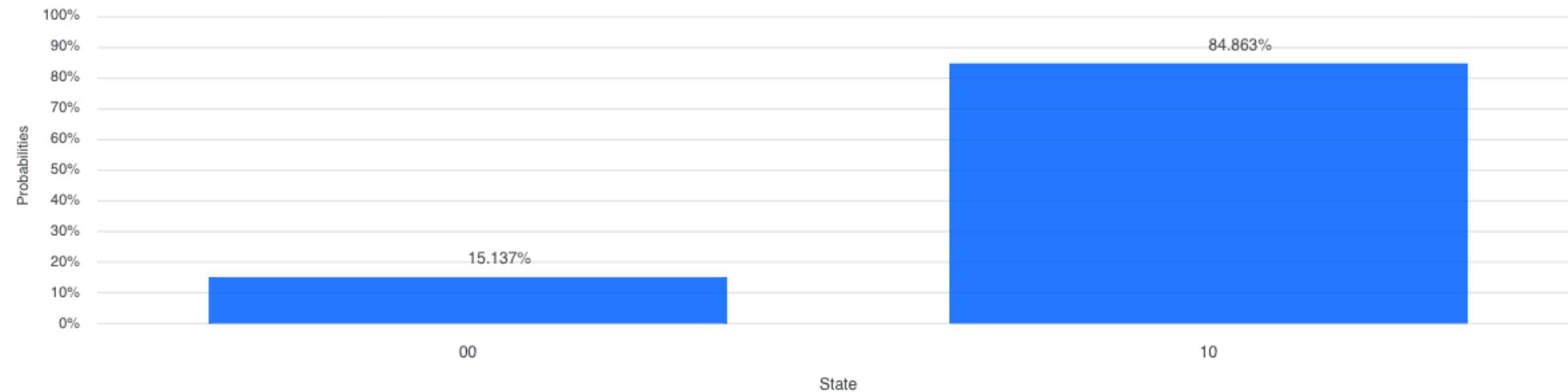
[2 minutes ago](#)

Results (1)

- [ibmq_qasm_simulator - 1024 shots - a few seconds ago.](#) Status: COMPLETED

Result

Histogram



Results

Pending [Refresh](#) 

<input type="checkbox"/>	Status	ID	Service	Provider	Run date
<input type="checkbox"/>	RUNNING	5d7fa0d7edaddd50013174232	Backend: ibmq_16_melbourne	ibm-q/open/main	Monday, September 16th 2019, 4:48:55 pm

Items per page: ▼ | 1 - 1 of 1 items | ◀◀ 1 ▶▶

Completed [Refresh](#) 

<input type="checkbox"/>	Status	ID	Service	Provider	Run date
<input type="checkbox"/>	COMPLETED	5d7fa149234397001196a292	Backend: ibmq_qasm_simulator	ibm-q/open/main	Monday, September 16th 2019, 4:50:49 pm
<input type="checkbox"/>	COMPLETED	5d78fb6f1cad1900183e4137	Backend: ibmq_qasm_simulator	ibm-q/open/main	Wednesday, September 11th 2019, 3:49:35 pm
<input type="checkbox"/>	COMPLETED	5d7375d44da2b200120d9d81	Backend: ibmq_qasm_simulator	ibm-q/open/main	Saturday, September 7th 2019, 11:18:12 am
<input type="checkbox"/>	COMPLETED	5d6d10494e38b1001afc2086	Backend: ibmq_ourense	ibm-q/open/main	Monday, September 2nd 2019, 2:51:21 pm
<input type="checkbox"/>	COMPLETED	5d67f18909ad9800126c1926	Backend: ibmq_ourense	ibm-q/open/main	Thursday, August 29th 2019, 5:38:49 pm

Results (2)

- [ibmq_qasm_simulator](#) - 1024 shots - 20 minutes ago. Status: COMPLETED
- [ibmq_16_melbourne](#) - 1024 shots - 22 minutes ago. Status: COMPLETED

Run details

Backend
ibmq_16_melbourne

Shots

Status
COMPLETED

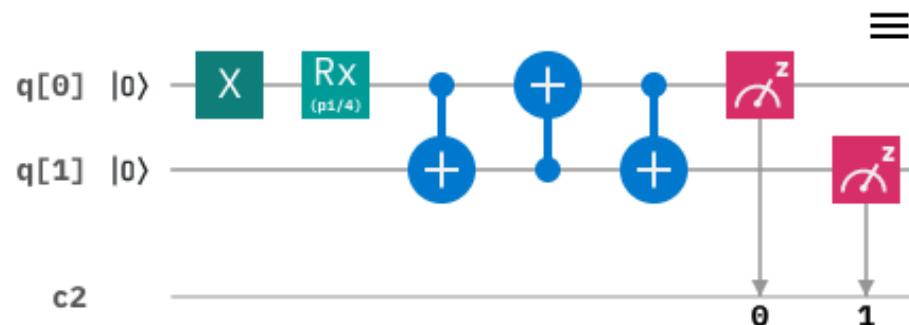
Time taken
-

Last Update
Sep 16th 2019 17:04:56

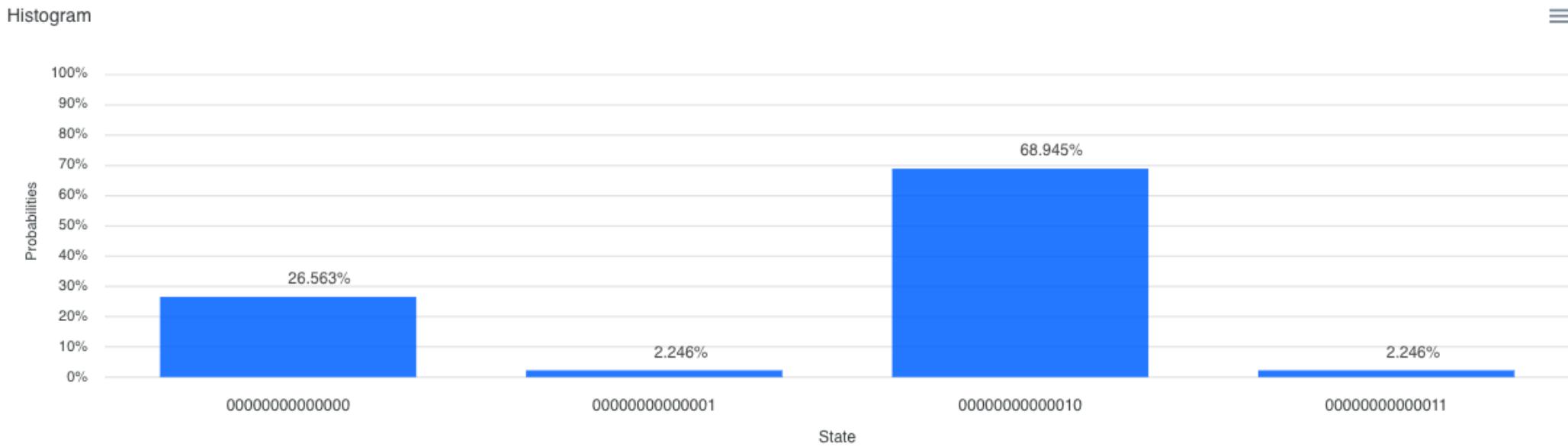
Circuit diagram



Original circuit diagram



Result



New Save Clear Help

swapWithTopoConstraint

Saved

Run



Visualizations

Statevector

State	Probability
00	0.924
01	0.000
10	0.076
11	0.000

The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.

```
[ 0.924+0j, 0+0j, 0-0.383j, 0+0j ]
```

Circuit composer

Gates

Operations

Subroutines

+ Add

File Edit View Insert Cell Kernel Widgets Help

A set of small, light-gray navigation icons typically found in LaTeX editors like Overleaf. From left to right, they include: a document icon, a plus sign, a minus sign, a magnifying glass, a double arrow, an upward arrow, a downward arrow, a right-pointing arrow, the word "Exécuter" (Execute), a black square, a letter "C", a right-pointing arrow, and the word "Markdown".

◀ ▶ 🔍

Basic version for entanglement.

Entrée [3]

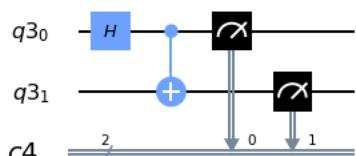
```
%matplotlib inline
# Importing standard Qiskit libraries and configuring account
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, execute, Aer, IBMQ
from qiskit.compiler import transpile, assemble
from qiskit.tools.jupyter import *
from qiskit.visualization import *
# Loading your IBM Q account(s)
provider = IBMQ.load_account()
```

Entrée [8]

```
qr = QuantumRegister(2)
cr = ClassicalRegister(2)
qc = QuantumCircuit(qr,cr)

qc.h(qr[0])
qc.cx(qr[0],qr[1])
qc.measure(qr,cr)

qc.draw(output='mpl')
```



Quantum Algorithms with Qiskit

Building your first quantum algorithm: quantum teleportation

Textbook quantum algorithms

Deutsch-Jozsa algorithm and its Qiskit implementation

Simon's algorithm and its Qiskit implementation

Bernstein-Vazirani algorithm and its Qiskit implementation

Quantum Fourier Transform and its Qiskit Implementation

Quantum phase estimation and its Qiskit implementation

Grover's algorithm and its Qiskit implementation

Advanced usage of Qiskit

Pulse-level quantum control techniques

Realistic simulations using device noise models

Transpiling quantum circuits

Bernstein-Vazirani algorithm and its Qiskit implementation

Introduction

Another algorithm demonstrating a quantum speed-up is the Bernstein-Vazirani algorithm. The algorithm solves the following problem: given a function $f : 0, 1^n \rightarrow 0, 1$ which takes an n bit string as input and outputs a single bit where there is exactly one n bit string $s \in 0, 1^n$ that returns a value $f(s) = 1$, what is s ?

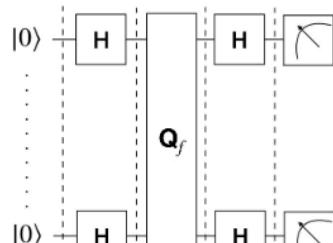
In other words, for each input x , $f(x) = s \cdot x \pmod{2}$. We are expected to find s .

To find s classically, we would have to call the function $f(x)$ n times, each time determining one of the bits of s .

On the other hand, a quantum algorithm can solve the problem in only one step! This gives a polynomial speed-up with respect to the classical algorithm. Let us see the quantum circuit that implements the corresponding algorithm.

Quantum circuit for the Bernstein-Vazirani Algorithm

The quantum circuit that implements the Bernstein-Vazirani algorithm is shown below.



Contents

Introduction

Quantum circuit for the Bernstein-Vazirani Algorithm

Example

Qiskit Implementation

Experiment with Simulators

Experiment with Real Devices

References

And :



Qiskit
qiskit.slack.com



Qiskit ▾

Jean-Michel Torres

Accéder à... < >

Messages non lus

Fils de discussion

Chaînes

aer

aqua

general

ibm-q-systems

meetups

qiskitters

terra

+ Ajouter une chaîne

Messages directs

Slackbot

Jean-Michel Torres (vous)

weihu

+ Inviter des personnes

Applications

+ Ajouter des applications

Nouveaux messages depuis 12 h 22 le 7 septembre

Marquer comme lu(s) x

2 réponses Dernière réponse il y a 12 jours

Syed Ali Hamza 0 h 47

I'm trying to implement this 4-qubit perceptron from <https://arxiv.org/pdf/1811.02266.pdf> and I can't exactly see how I can implement the 4-qubit controlled-Z gates. Can I please get any leads to where I can possibly find a solution? Also, as per one solution here: <https://quantumcomputing.stackexchange.com/questions/4078/how-to-construct-a-multi-qubit-controlled-z-from-elementary-gates>, I can't get how exactly I can implement the [+] and [-] gates.

image.png ▾

Quantum circuit of a $N = 4$ perceptron. An example of a typical quantum circuit for a perceptron model with $N = 4$ qubits (i.e., capable of processing $m = 2^4 = 16$ dimensional input vectors), which employs the algorithm for the generation of hypergraph states, including the HSGS (see main text). In this example, the input vector has elements $i_0 = i_1 = -1$, and $i_j = 1$ for $j = 2, \dots, 15$, while the weight vector has elements $w_2 = w_3 = w_4 = -1$, and 1 in all other entries. Multi-controlled CZ gates are denoted by vertical lines and black dots on the qubits involved. The HSGS is realized inside the U_i block after the initial $H^{\otimes N}$ gate, and in the U_w block before the final $H^{\otimes N}$ and NOT $^{\otimes N}$ operations.

6 réponses Dernière réponse il y a 12 jours

yanwu gu 7 h 49

Affichage des archives : 27 juil. 2019 - 7 sept. 2019

Accéder aux messages récents ▾

Envoyer un message à #general @ ☺

Recherche

@ ⭐ ⋮

Jeudi 29 août

Fil de discussion

general

Syed Ali Hamza 29 août à 0 h 47

I'm trying to implement this 4-qubit perceptron from <https://arxiv.org/pdf/1811.02266.pdf> and I can't exactly see how I can implement the 4-qubit controlled-Z gates. Can I please get any leads to where I can possibly find a solution? Also, as per one solution here: <https://quantumcomputing.stackexchange.com/questions/4078/how-to-construct-a-multi-qubit-controlled-z-from-elementary-gates>, I can't get how exactly I can implement the [+] and [-] gates.

image.png ▾

Quantum circuit of a $N = 4$ perceptron. An example of a typical quantum circuit for a perceptron model with $N = 4$ qubits (i.e., capable of processing $m = 2^4 = 16$ dimensional input vectors), which employs the algorithm for the generation of hypergraph states, including the HSGS (see main text). In this example, the input vector has elements $i_0 = i_1 = -1$, and $i_j = 1$ for $j = 2, \dots, 15$, while the weight vector has elements $w_2 = w_3 = w_4 = -1$, and 1 in all other entries. Multi-controlled CZ gates are denoted by vertical lines and black dots on the qubits involved. The HSGS is realized inside the U_i block after the initial $H^{\otimes N}$ gate, and in the U_w block before the final $H^{\otimes N}$ and NOT $^{\otimes N}$ operations.

6 réponses

Steve Wood il y a 12 jours

Does this help you at all?
<https://github.com/Qiskit/qiskit-aqua/blob/master/qiskit/aqua/circuits/gates/>