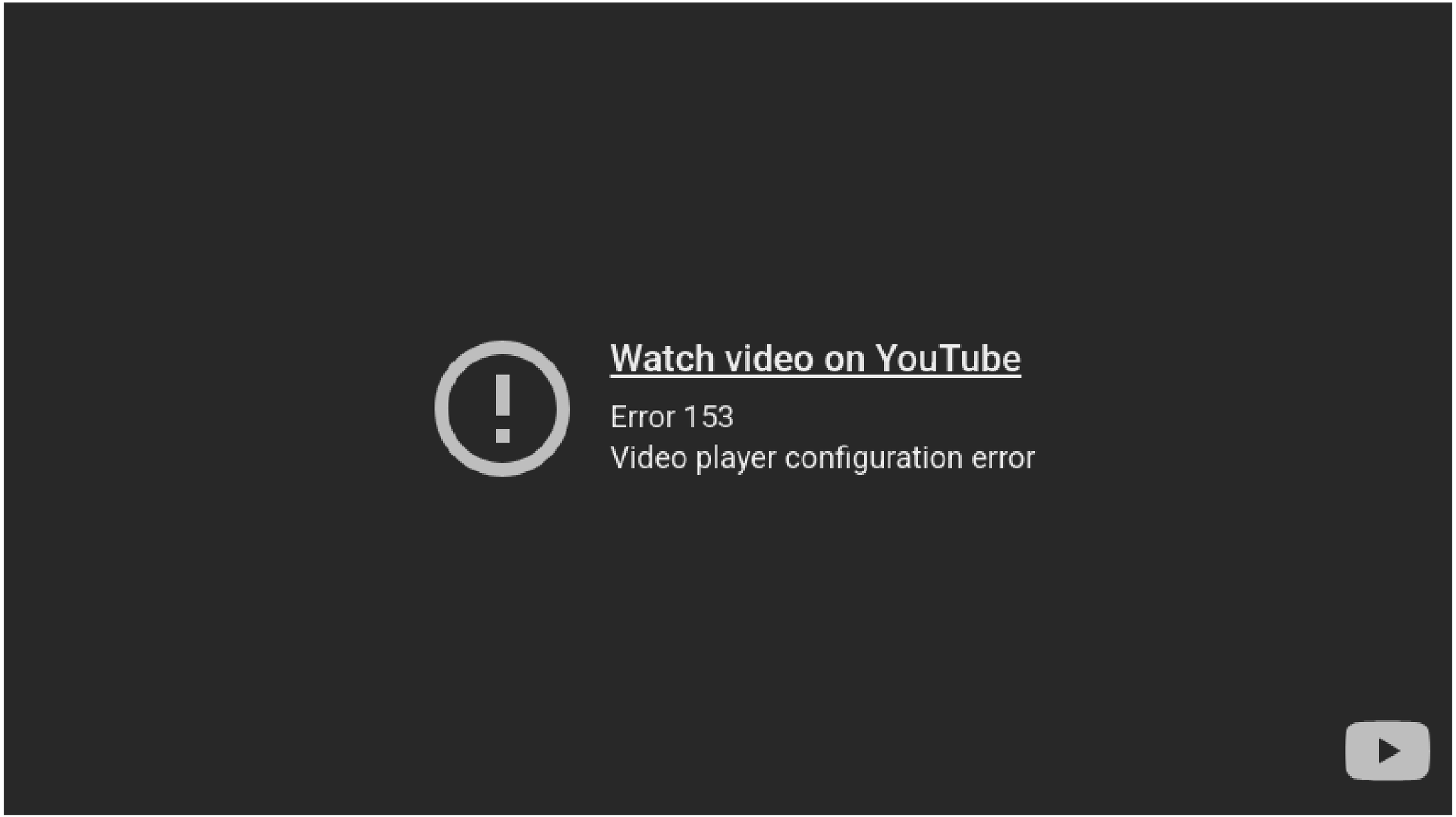
Quantum Computing

Practical Guidelines



Quantum Computers Explained – Limits of Human Technology

Credits: kurzgesagt

Google Colab

- !pip install qiskit qiskit-ibm-runtime

  !pip install qiskit-aer

  ```
  try:
      import cirq
  except ImportError:
      print("installing cirq...")
      !pip install --quiet cirq
      print("installed cirq.")
      import cirq
  ```

QuantumPracticalWithPython.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

1s

```
from qiskit import QuantumCircuit
qc = QuantumCircuit (3,3)
qc.draw()
```

```
q_0:

q_1:

q_2:

c: 3/
```

QuantumPracticalWithPython.ipynb

File Edit View Insert Runtime Tools Help

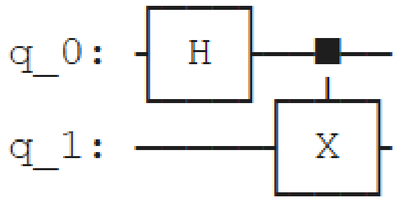
Commands + Code + Text Run all

0s

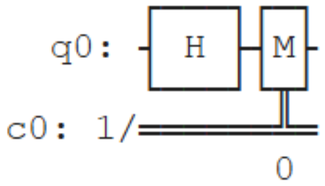
```
q0, q1, q2 = cirq.GridQubit.rect(1,3)
circuit = cirq.Circuit(
    (cirq.X ** 0.5).on_each(q0, q1, q2),
    cirq.CZ(q0, q1),
    cirq.CZ(q1, q2),
    cirq.measure(q2, key='m')
)
print(circuit)
```

```
(0, 0): ---X^0.5---@-----------------
                   |
(0, 1): ---X^0.5---@---@-------------
                       |
(0, 2): ---X^0.5-------@---M('m')---
```

```
qc = QuantumCircuit(2)
qc.h(0)
qc.cx(0, 1)
qc.draw()
```




```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
qr = QuantumRegister(1)
cr = ClassicalRegister(1)
# the quantum circuit takes in the quantum register and classical register
qc = QuantumCircuit(qr, cr)
qc.draw()
#
qc.h(qr[0])
qc.measure(qr, cr)
%matplotlib inline
qc.draw()
```



QuantumPracticalWithPython.ip

colab.research.google.com/drive/1VQiCv2zYirxK7hf1QI1HZL6kFc8RAto5#scrollTo=3nNYRqoJqneM

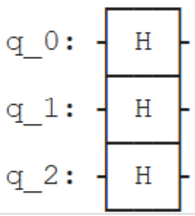
QuantumPracticalWithPython.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

0s

```
from qiskit_aer import AerSimulator
from qiskit import *
backend = AerSimulator()

job = backend.run(qc, shots=1024)

from qiskit.visualization import plot_histogram
result = job.result()
counts = result.get_counts()
plot_histogram(counts)

from qiskit import QuantumCircuit
from qiskit_aer import Aer
import numpy as np
from qiskit.visualization import plot_histogram, plot_bloch_multivector
qc = QuantumCircuit(3)
for qubit in range(3):
  qc.h(qubit)
qc.draw()
```



What can I help you build?

Miniconda

Miniconda is a free, miniature installation of Anaconda Distribution that includes only conda, Python, the packages they both depend on, and a small number of other useful packages.

If you need more packages, use the `conda install` command to install from thousands of packages available by default in Anaconda's public repo, or from other channels, like conda-forge or bioconda.

- Is Miniconda free for me?
- Should I install Miniconda or Anaconda Distribution?

Latest Miniconda installer links

For the latest Miniconda installers for Python 3.13, go to anaconda.com/download. The Miniconda installers are on the same page as the Anaconda Distribution installers, past registration.

For a list of Miniconda hashes and an archive of Miniconda versions, including installers for older versions of Python, see https://repo.anaconda.com/miniconda.

Quick command line install

See our new section on the Installing Miniconda page!

Distribution

FREE DOWNLOAD*

Register to get everything you need to get started on your workstation including Cloud Notebooks, Navigator, AI Assistant, Learning and more.

- Easily search and install thousands of data science, machine learning, and AI packages
- Manage packages and environments from a desktop application or work from the command line
- Deploy across hardware and software platforms
- Distribution installation on Windows, MacOS, or Linux

*Use of Anaconda's Offerings at an organization of more than 200 employees requires a Business or Enterprise license. See Pricing

Provide email to download Distribution

Email Address:
hbpatel1976@gmail.com

Agree to receive communication from Anaconda regarding relevant content, products, and services. I understand that I can revoke this consent here at any time.

By continuing, I agree to Anaconda's Privacy Policy and Terms of Service.

Submit

Skip registration

Manage Trusted Packages and

Download Now

Download Anaconda Distribution or Miniconda by choosing the proper installer for your machine. Learn the difference from our Documentation.

Distribution Installers

Download

For installation assistance, refer to troubleshooting.

Windows

Miniconda Installers

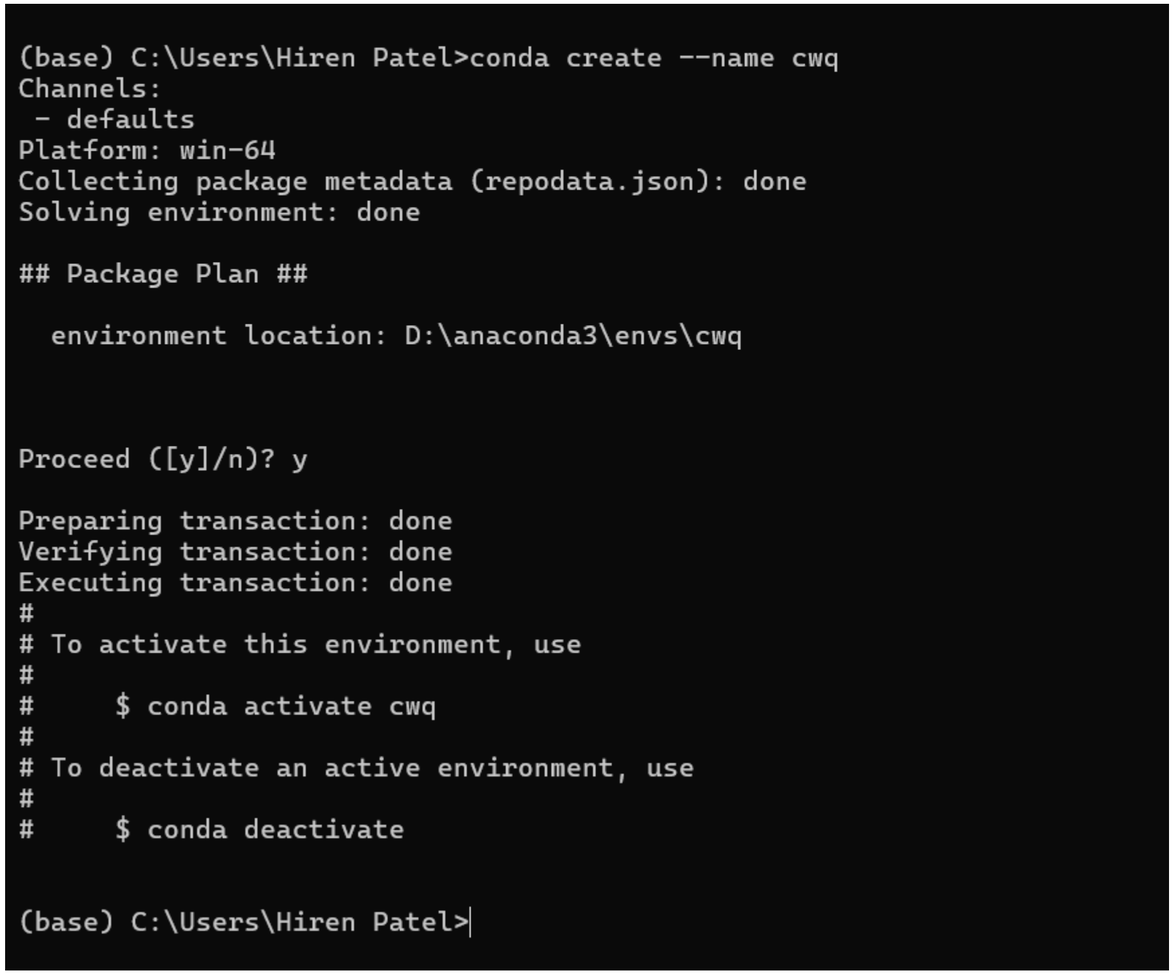
Download

For installation assistance, refer to troubleshooting.

Windows

Now, open Anaconda Powershell prompt, and execute following commands:

```
conda create --name cwq

conda activate cwq

conda install pip

pip install qiskit

pip install jupyter

pip install matplotlib

pip install qiskit_ibm_runtime

pip install pylatexenc
```



Now, open Visual Studio Code
View -> Command Palette -> >create jupyter notebook
Select Kernel -> cwq

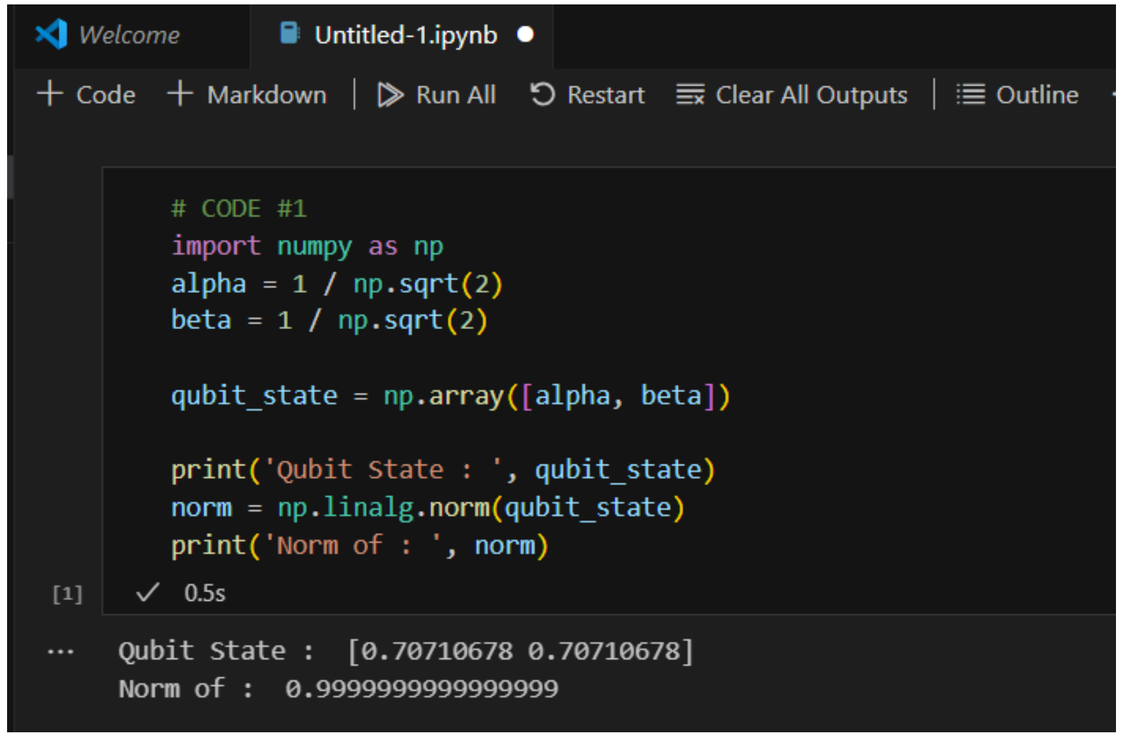
Start coding now:

 # CODE #1

```
import numpy as np
alpha = 1 / np.sqrt(2)
beta = 1 / np.sqrt(2)

qubit_state = np.array([alpha, beta])

print('Qubit State : ', qubit_state)

norm = np.linalg.norm(qubit_state)
print('Norm of : ', norm)
```

OUTPUT:

```
Qubit State :  [0.70710678 0.70710678]
Norm of :  0.9999999999999999
```

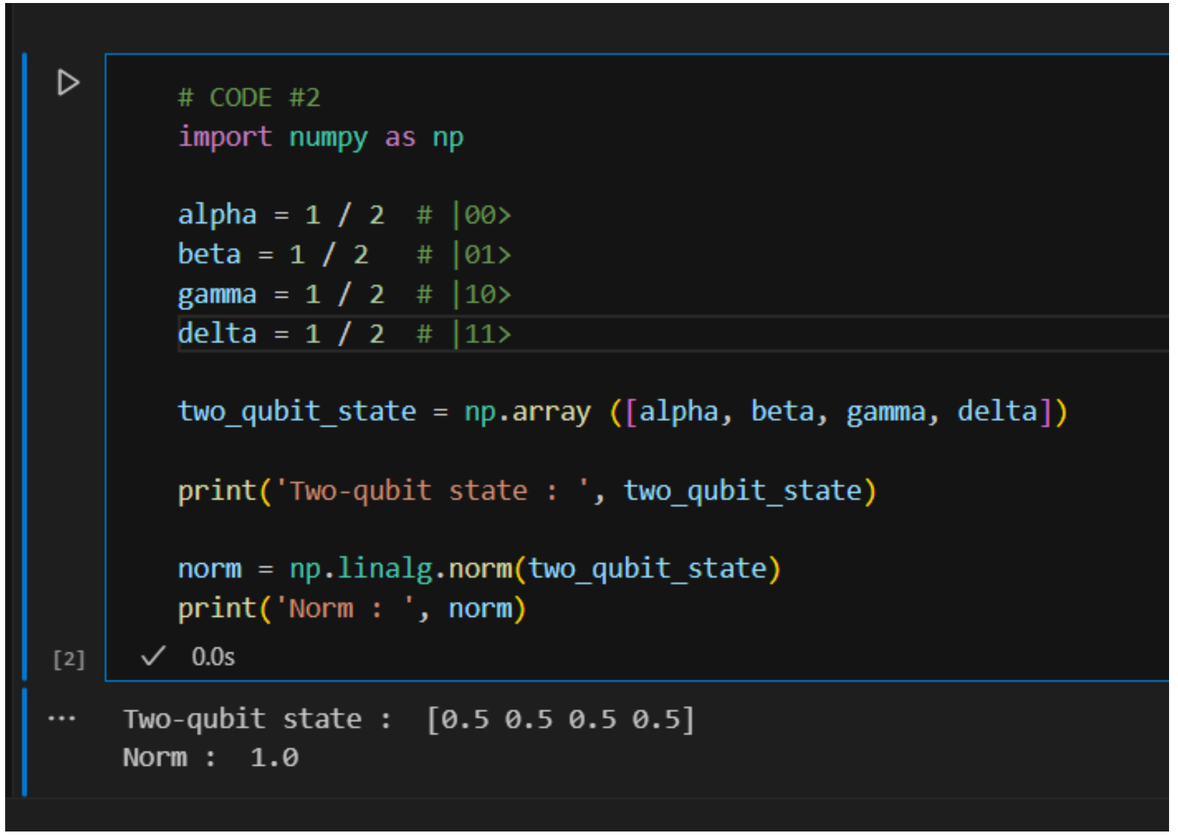


 # CODE #2

```
import numpy as np

alpha = 1 / 2   # |00>
beta = 1 / 2    # |01>
gamma = 1 / 2   # |10>
delta = 1 / 2   # |11>

two_qubit_state = np.array ([alpha, beta,
gamma, delta])

print('Two-qubit state : ', two_qubit_state)

norm = np.linalg.norm(two_qubit_state)
print('Norm : ', norm)
```

OUTPUT:

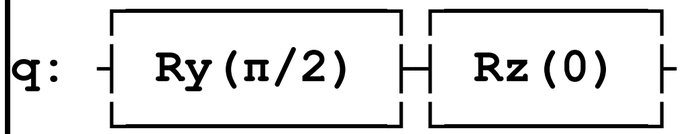
```
Two-qubit state :  [0.5 0.5 0.5 0.5]
Norm :  1.0
```



Code #3

```
from qiskit import QuantumCircuit
import numpy as np

def create_simple_qubit_circuit(alpha, beta):
    if not np.isclose(abs(alpha)**2 + abs(beta)**2, 1.0):
        raise ValueError("Addition of both the Probability
qc = QuantumCircuit(1)
theta = 2 * np.arccos(abs(alpha))
phi = np.angle(beta) - np.angle(alpha)

qc.ry(theta, 0)
qc.rz(phi, 0)

return qc

if __name__== "__main__":
    alpha = 1 / np.sqrt(2)
    beta = 1 / np.sqrt(2)

    circuit = create_simple_qubit_circuit(alpha, beta)

    print(circuit)
    circuit.draw(output="text")
```

OUTPUT:

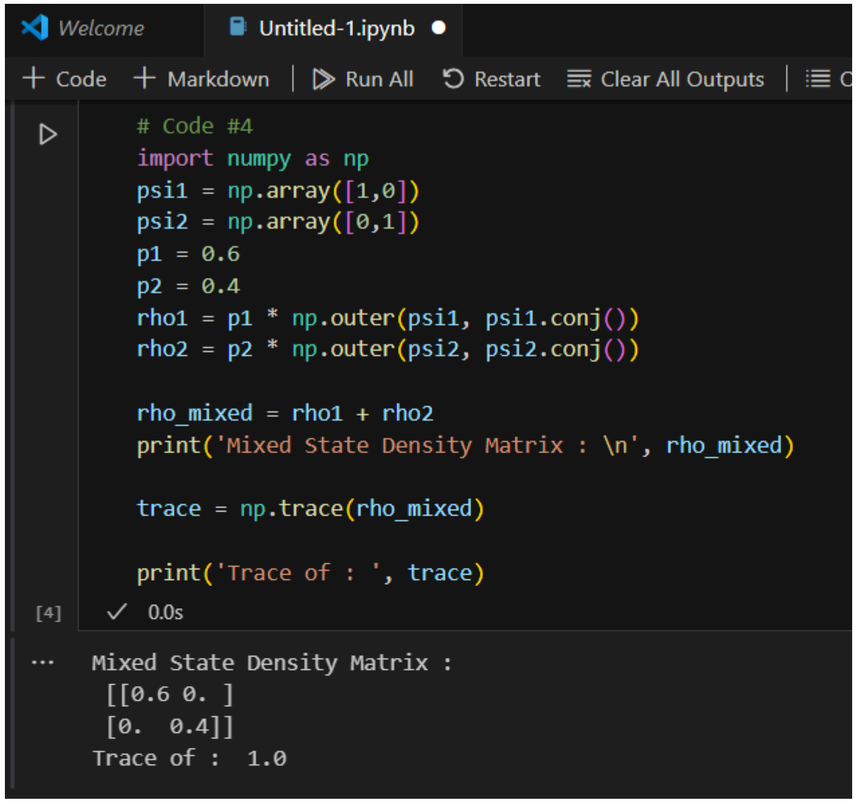


 # Code #4

```
import numpy as np
psi1 = np.array([1,0])
psi2 = np.array([0,1])

p1 = 0.6
p2 = 0.4

rho1 = p1 * np.outer(psi1, psi1.conj())
rho2 = p2 * np.outer(psi2, psi2.conj())

rho_mixed = rho1 + rho2
print('Mixed State Density Matrix : \n', rho_mixed)

trace = np.trace(rho_mixed)

print('Trace of : ', trace)
```

OUTPUT:

```
Mixed State Density Matrix :
 [[0.6 0. ]
 [0.  0.4]]
Trace of :  1.0
```

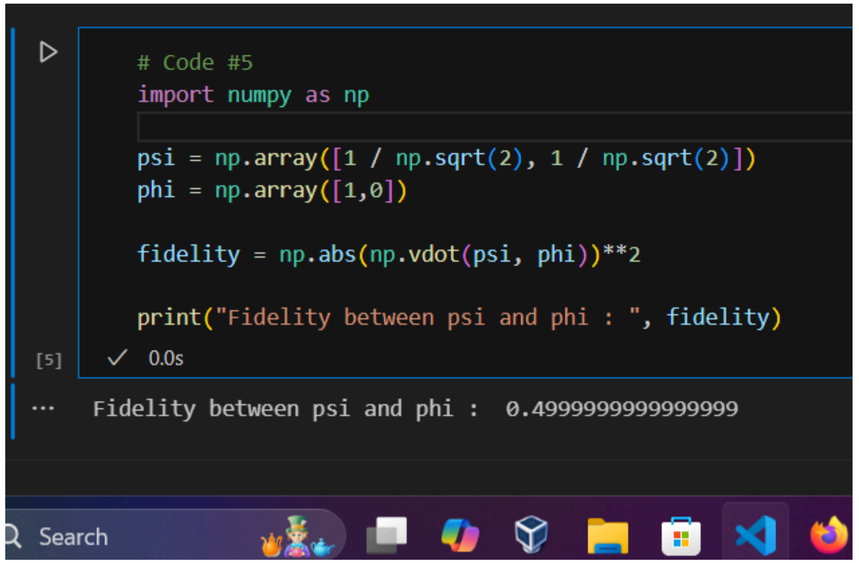


 # Code #5

```
import numpy as np

psi = np.array([1 / np.sqrt(2), 1 / np.sqrt(2)])
phi = np.array([1,0])

fidelity = np.abs(np.vdot(psi, phi))**2

print("Fidelity between psi and phi : ", fidelity)
```

OUTPUT:

```
Fidelity between psi and phi :  0.4999999999999999
```

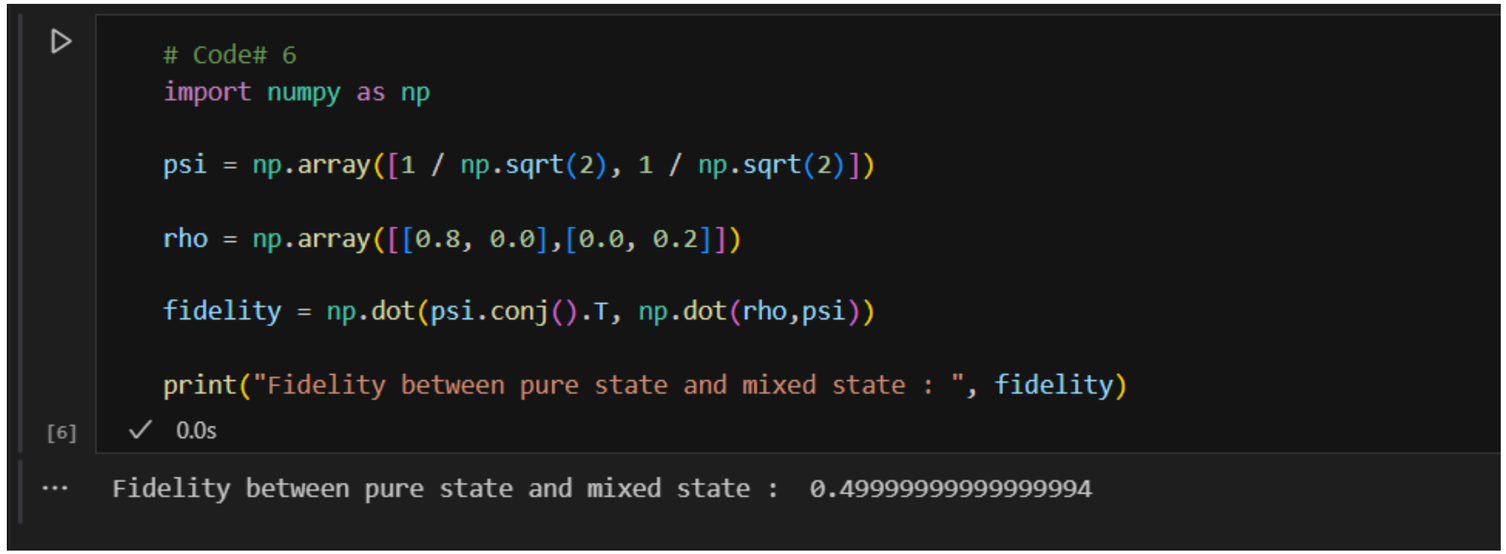


Code# 6

```
import numpy as np

psi = np.array([1 / np.sqrt(2), 1 / np.sqrt(2)])

rho = np.array([[0.8, 0.0],[0.0, 0.2]])

fidelity = np.dot(psi.conj().T, np.dot(rho,psi))

print("Fidelity between pure state and mixed state : ", fidelity)
```

OUTPUT:

```
Fidelity between pure state and mixed state :  0.4999999999999994
```

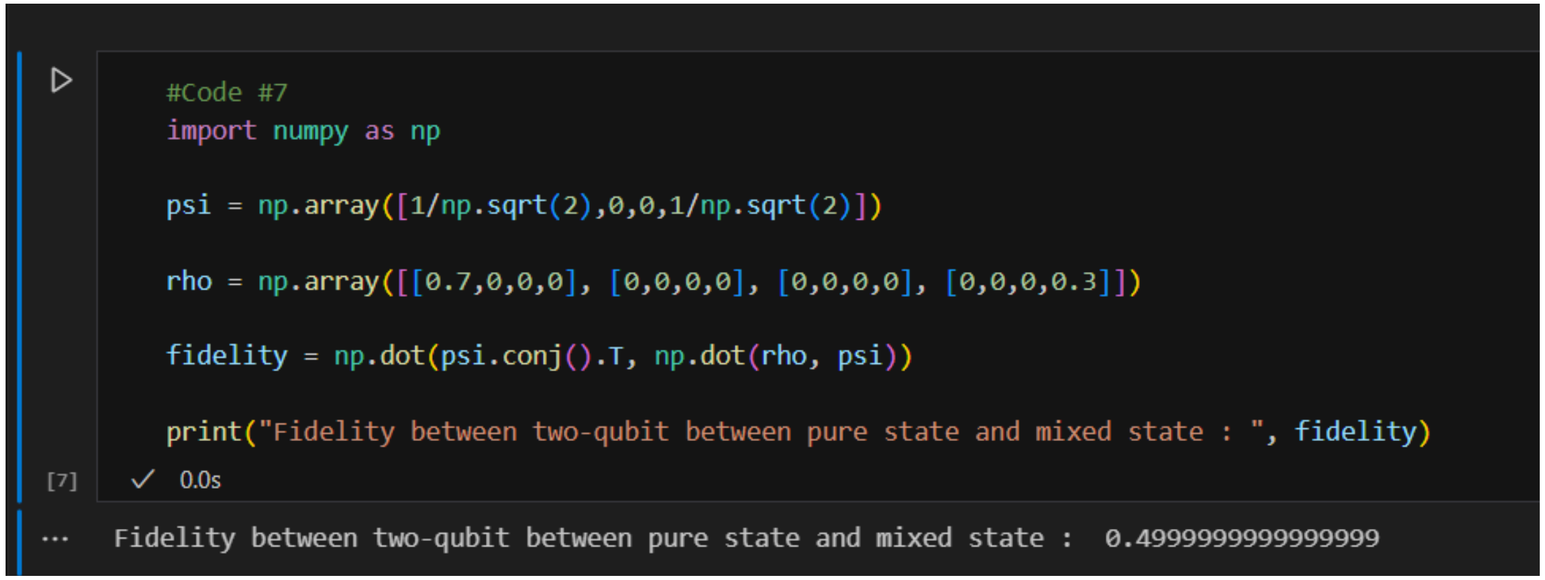


#Code #7

```
import numpy as np

psi = np.array([1/np.sqrt(2),0,0,1/np.sqrt(2)])

rho = np.array([[0.7,0,0,0], [0,0,0,0], [0,0,0,0], [0,0,0,0.3]])

fidelity = np.dot(psi.conj().T, np.dot(rho, psi))

print("Fidelity between two-qubit between pure state and mixed state : ", fidelity)
```

OUTPUT:

```
Fidelity between two-qubit between pure state and mixed state :  0.4999999999999999
```



Hiren Patel

IBM Quantum Platform

Create API key +

Welcome to the upgraded platform! Take a quick tour to learn about the new experience. Get started →

Account Instances View all

There are no instances associated with this account

To start running quantum workloads, switch accounts or contact an account administrator to create an instance.

Featured resources

Documentation
Hit the ground running with Qiskit, our open-source toolkit for quantum computing.
View the docs →

Tutorials
Explore utility-grade examples that leverage IBM Quantum and Qiskit technologies.
Browse tutorials →

Composer
Build, simulate, and run quantum circuits with a drag-and-drop interface.
Start building →

Announcements
Stay up to date with the latest news, service alerts, and product updates.
View announcements →

Inbox - 1976hbpatel@gmail.com | Composer | IBM Quantum Platform

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbQAsAXRnNFK5pp316IADQg6EGNwQgAqnQAuDJ626cizBvObtXIAC%2BQA

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ I T S Z T† S† P RZ |0⟩ if √X √X† Y RX RY RXX RZZ U RCCX RC3X

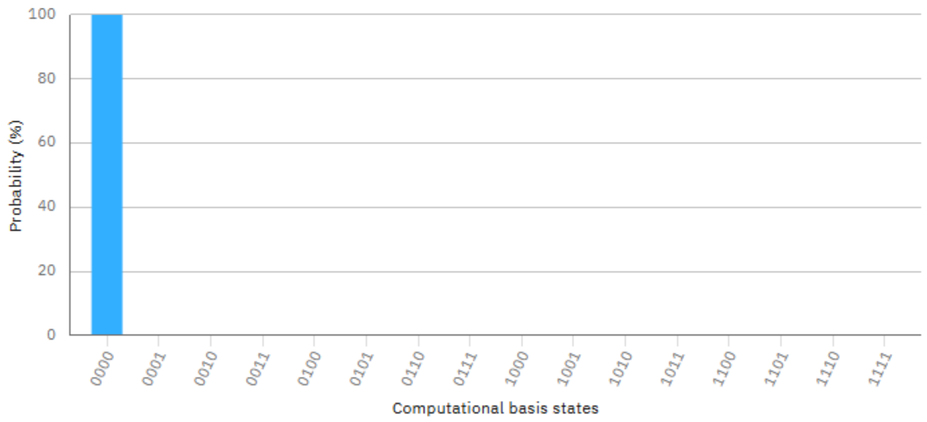
Left alignment | Inspect

q[0]
q[1]
q[2]
q[3]
c4

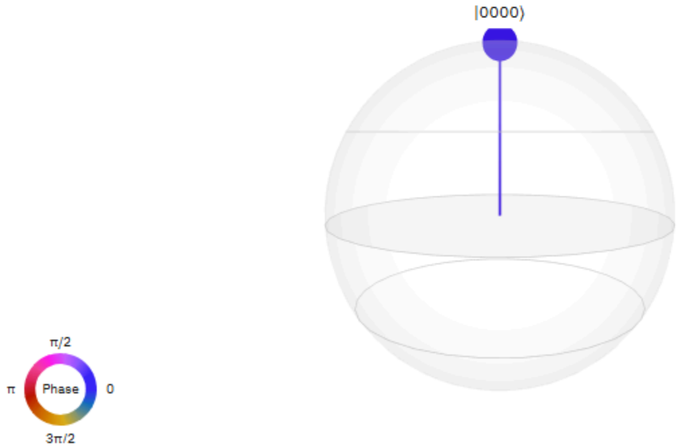
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[4];
5  creg c[4];
6
7
```

Probabilities



Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:18 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbVEBdGc0UrmagCzb6iqNwAuqtRRN0AHtdsyY3aO0VPNRALQA%2BIgN...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ I T S Z T† S† P RZ |0⟩ if √X √X† Y RX RY RXX RZZ U RCCX RC3X

Left alignment | Inspect

q[0] |0⟩ ⊕ 0

c4

OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[1];
5  creg c[4];
6  reset q[0];
7  x q[0];
8  measure q[0] -> c[0];
9
```

Probabilities

Probability (% of 1024 shots)

100 80 60 40 20 0

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

Computational basis states

Q-sphere

π/2 π Phase 0 3π/2

|1⟩

State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:21 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbVEBdGc0UrmagCzb6iqNwAuqtRRN0AntdsyY3aO0VPNRALQA%2BIgNn...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ I

T S Z T† S† P

RZ |0⟩ if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

Left alignment | Inspect

q[0] |0⟩ Y

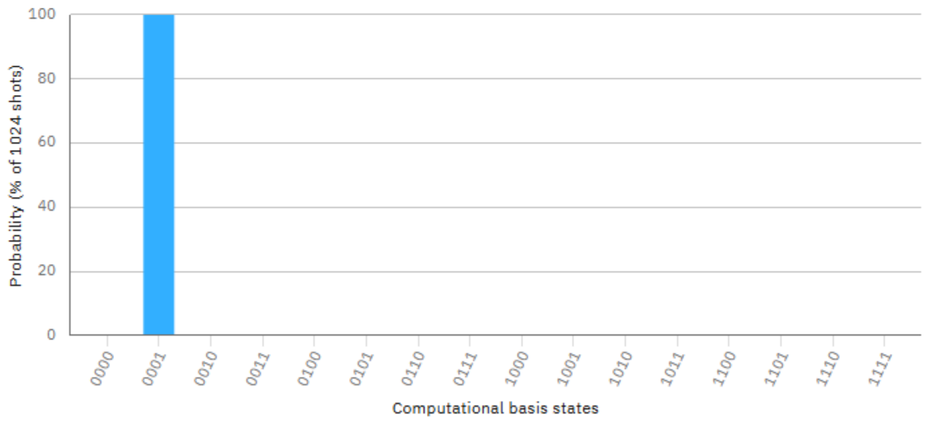
c4

0

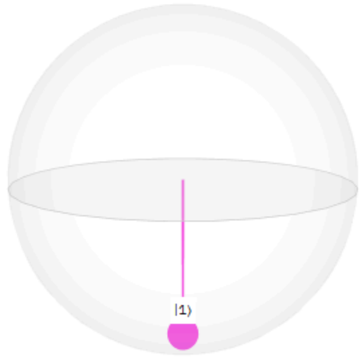
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[1];
5  creg c[4];
6  reset q[0];
7  y q[0];
8  measure q[0] -> c[0];
9
```

Probabilities



Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:22 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbVEBdGc0UrmagCzb6iqNwAuqtRRN0AXtdsyY3aO0VPNRALQA%2BIgNn...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ I

T S Z T† S† P

RZ |0⟩ if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

Left alignment | Inspect

q[0] |0⟩ Z

c4

OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[1];
5  creg c[4];
6  reset q[0];
7  z q[0];
8  measure q[0] -> c[0];
9
```

Probabilities

Probability (% of 1024 shots)

100 80 60 40 20 0

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

Computational basis states

Q-sphere

|0⟩

π/2 π Phase 0 3π/2

State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:22 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbVEBdGc0UrmagCzb6iqNwAuqtRRN0AFtdsyQAGhB0IMbghABVOgsGC1...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ I
T S Z T† S† P
RZ |0⟩ if
√X √X† Y RX RY RXX
RZZ U RCCX RC3X

Left alignment | Inspect

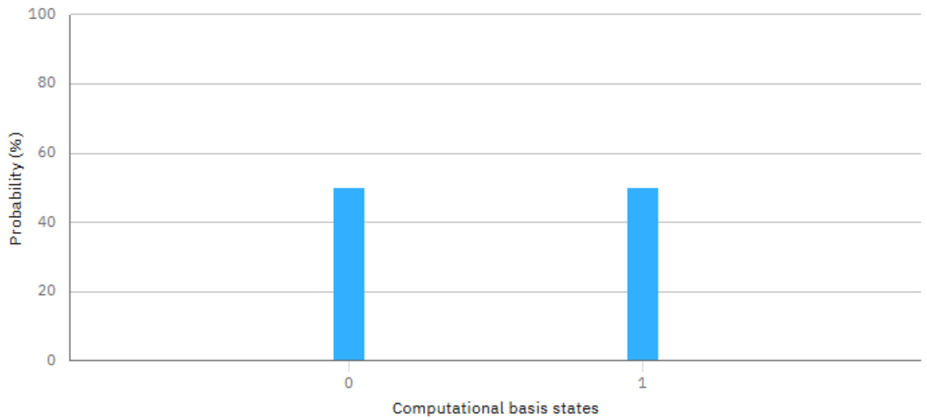
q[0] |0⟩ H

c4

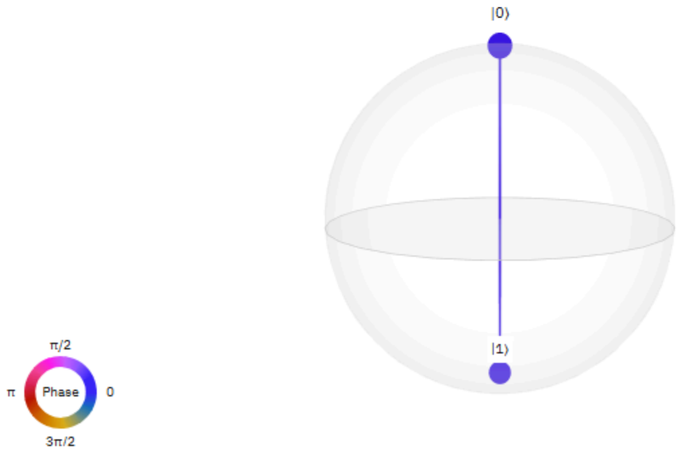
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[1];
5  creg c[4];
6  reset q[0];
7  h q[0];
8
```

Probabilities



Q-sphere



State | Phase angle

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbVEBdGc0UrmagCzb6iqNwAuqtRRRN0AFtdsY3aO0VPNRALQA%2BIgNn...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ I T S Z T† S† P RZ |0⟩ if √X √X† Y RX RY RXX RZZ U RCCX RC3X

Left alignment | Inspect

q[0] |0⟩ H

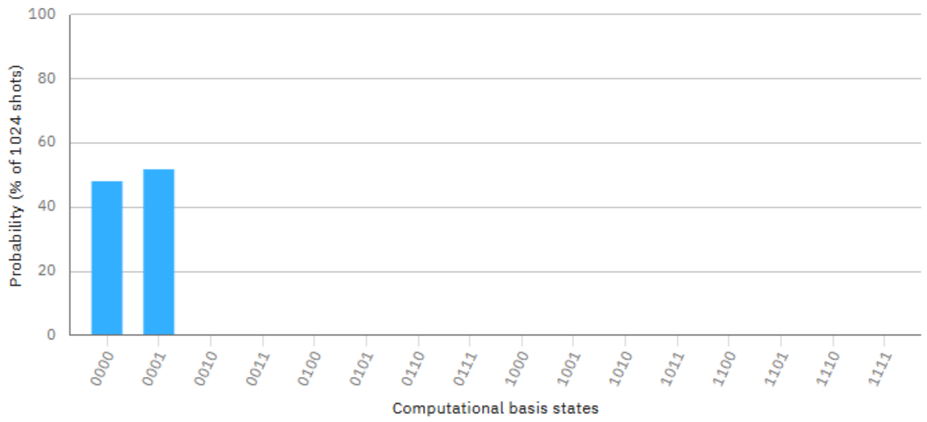
c4

0

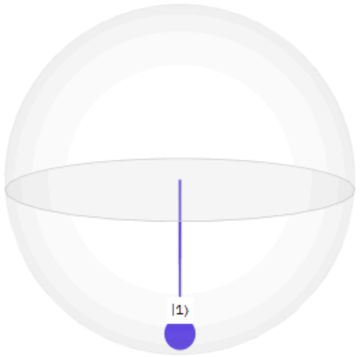
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[1];
5  creg c[4];
6  reset q[0];
7  h q[0];
8  measure q[0] -> c[0];
9
```

Probabilities



Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:23 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmagCzb6iqNwAuqtRRMsAHtdsAaa6Lsxu0doqeaiALQAfEQG...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ● ✕ I

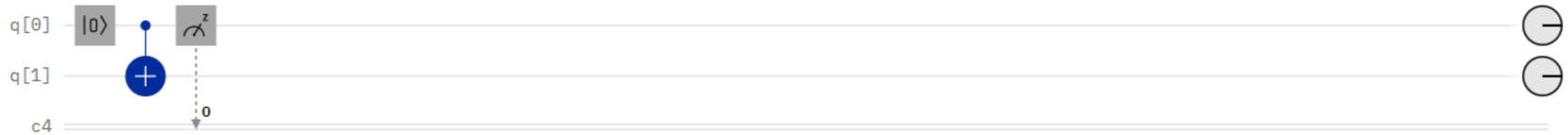
T S Z T† S† P

RZ ⟳ |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

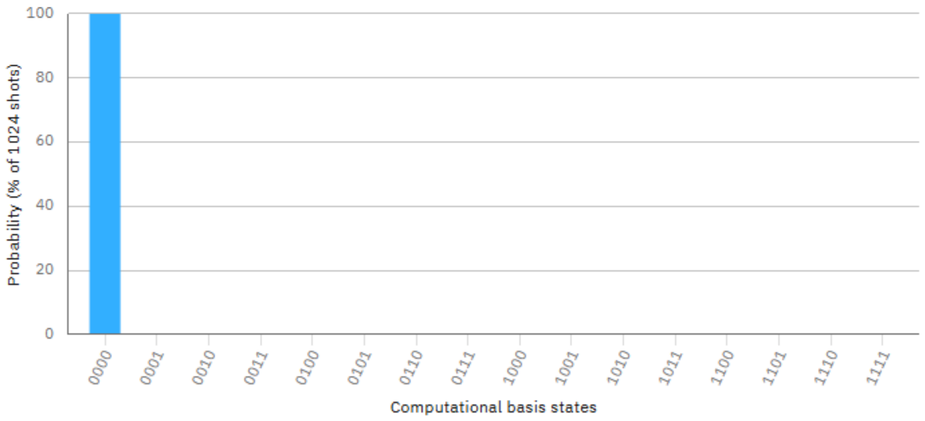
Left alignment | Inspect



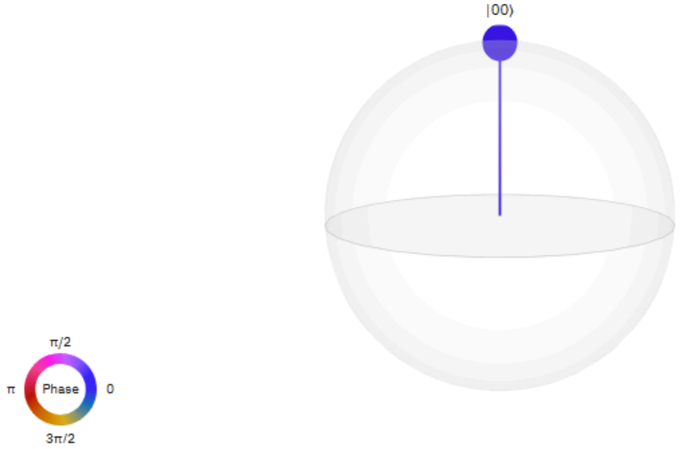
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[4];
6  reset q[0];
7  cx q[0], q[1];
8  measure q[0] -> c[0];
9
```

Probabilities



Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:24 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fUVRuAF1VqKRugA9L1nffXWANJdE2Y3aO0UPNIgBaAD...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ┆ ✕ I

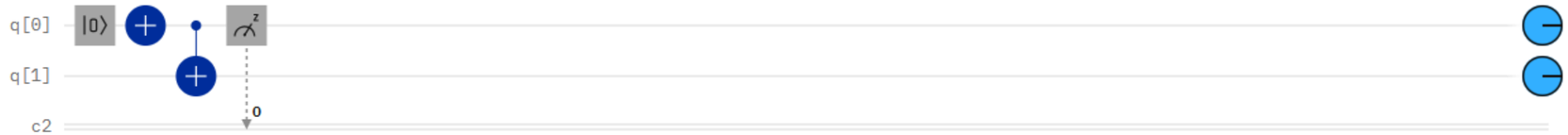
T S Z T† S† P

RZ ↗z |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

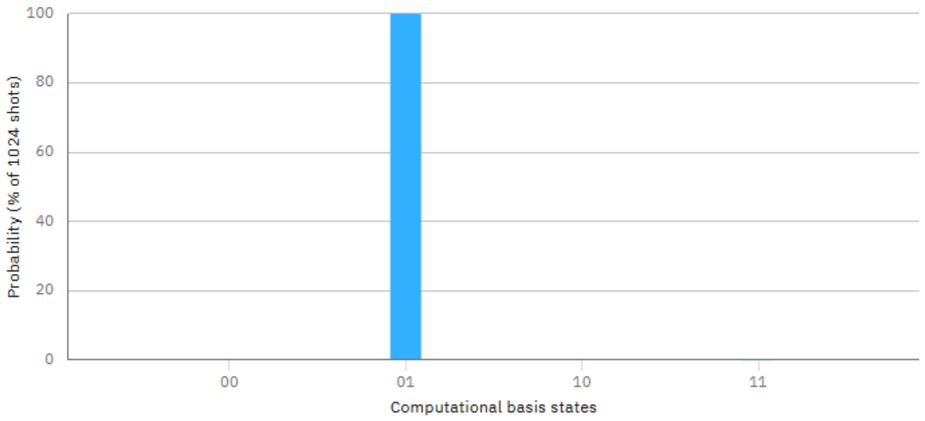
Left alignment | Inspect



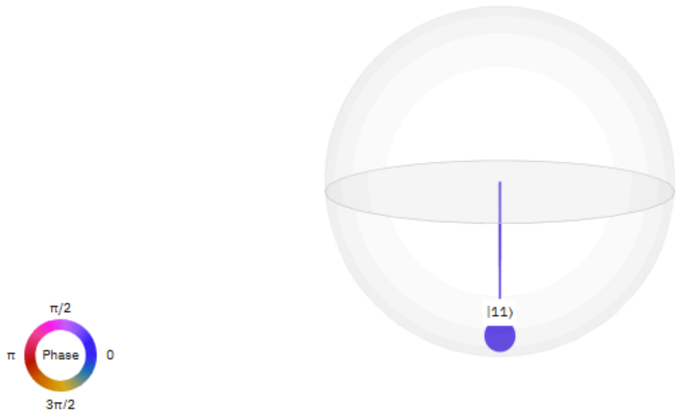
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  reset q[0];
7  x q[0];
8  cx q[0], q[1];
9  measure q[0] -> c[0];
10
```

Probabilities



Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:26 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fUVRuAF1VqKRugA9L1nffXWANJDJdE2Y3aO0UPNIgBaAD...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ┆ ✕ I

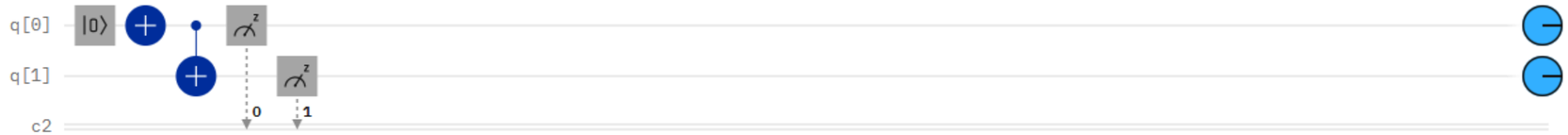
T S Z T† S† P

RZ ↗z |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

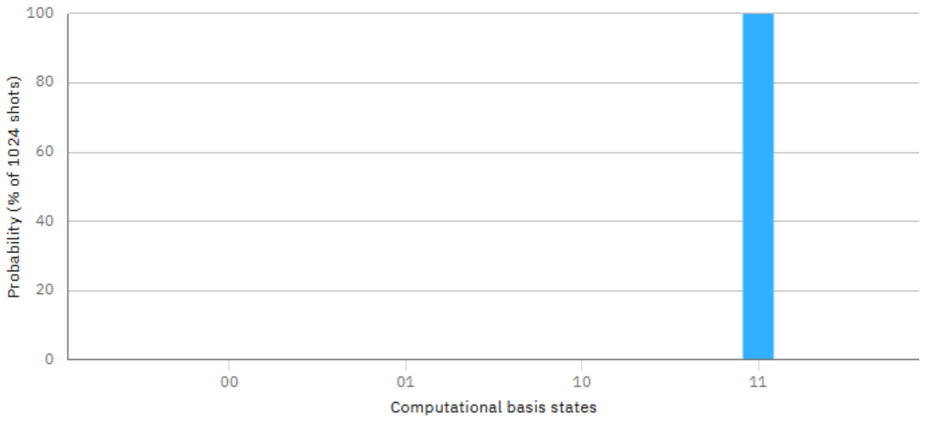
Left alignment | Inspect



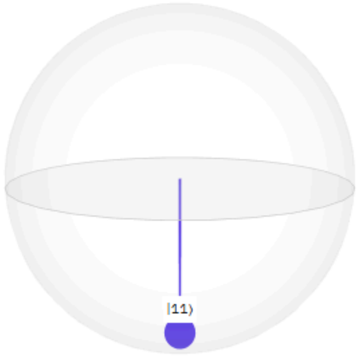
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  reset q[0];
7  x q[0];
8  cx q[0], q[1];
9  measure q[0] -> c[0];
10 measure q[1] -> c[1];
11
```

Probabilities



Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:26 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fUVRuAF1VqKRugA9L1mTG7R2ih5qIBaAHxEDjvTOru7q...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ● ✕ I

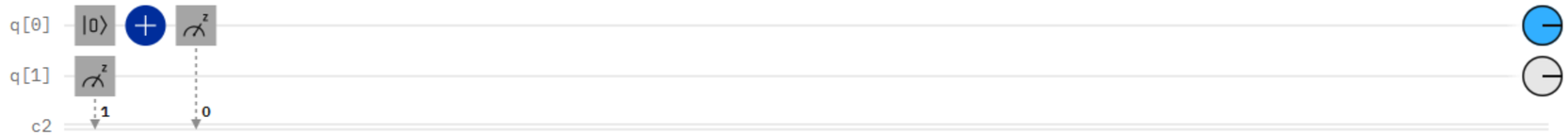
T S Z T† S† P

RZ ⟳ |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

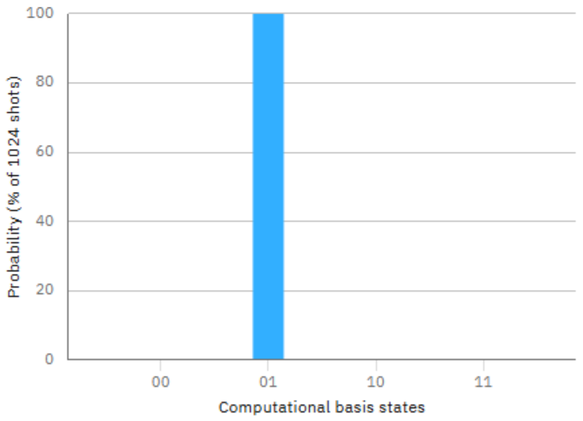
Left alignment | Inspect



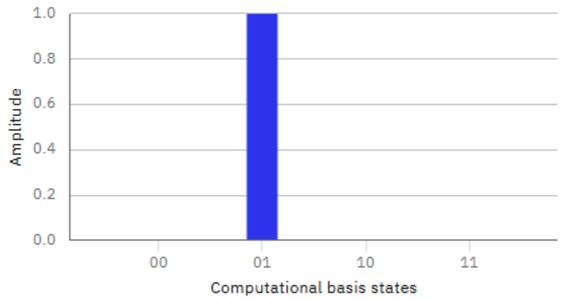
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  reset q[0];
7  x q[0];
8  measure q[0] -> c[0];
9  measure q[1] -> c[1];
10
```

Probabilities

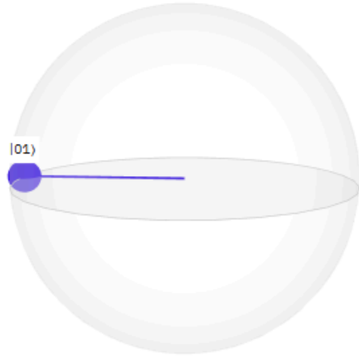


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:28 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fUVRuAF1VqKRujG7R2iy6M1E1EAtAD4iBlzIaeltYyUADuEA...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ● ✕ I

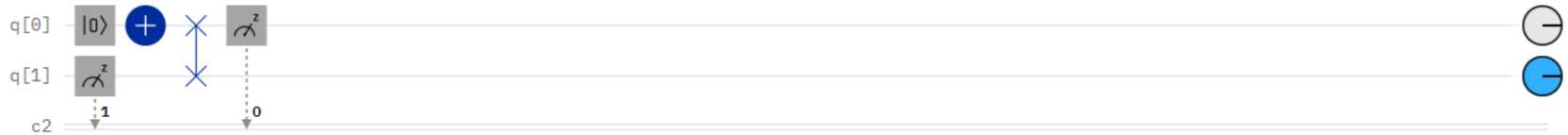
T S Z T† S† P

RZ ⟳ |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

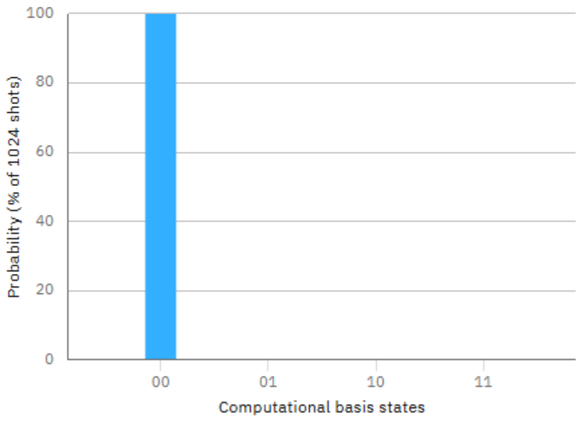
Left alignment | Inspect



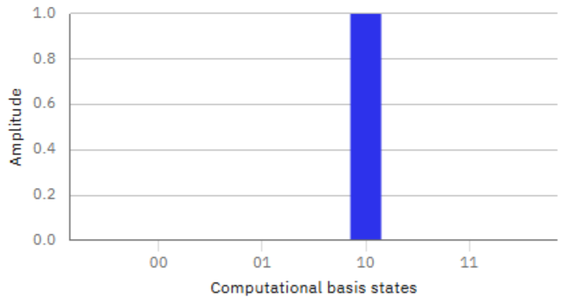
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  reset q[0];
7  measure q[1] -> c[1];
8  x q[0];
9  swap q[0], q[1];
10 measure q[0] -> c[0];
11
```

Probabilities

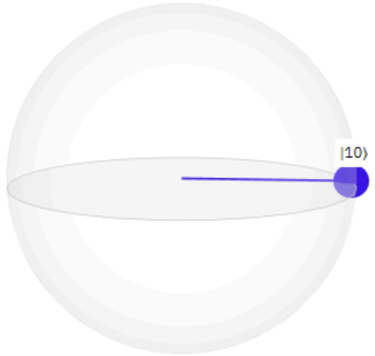


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:28 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbQDMAXRnNFK5pp31FUbgBdVaisbqmLV0bYAejl1Zu7mr9TYA0bgHqJLY...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ⊕ ⊕ ✕ I

T S Z T† S† P

RZ ↗z |0⟩ ┆ ● if

√X √X† Y RX RY RXX

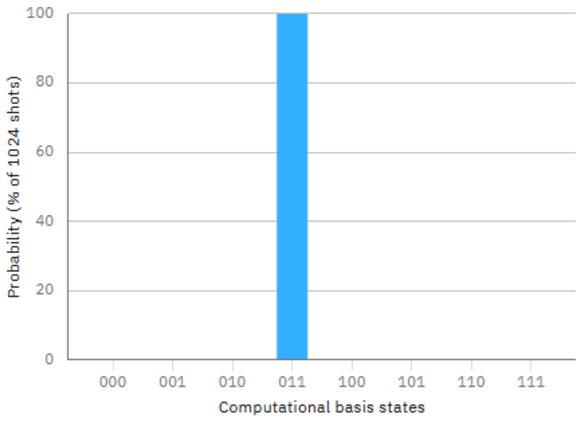
RZZ U RCCX RC3X

Left alignment | Inspect

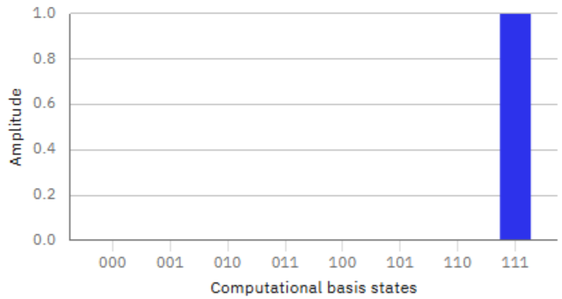
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[3];
5  creg c[3];
6  reset q[0];
7  reset q[1];
8  x q[1];
9  x q[0];
10 ccx q[0], q[1], q[2];
11 measure q[0] -> c[0];
12 measure q[1] -> c[1];
13 measure q[2] -> c[1];
14
```

Probabilities



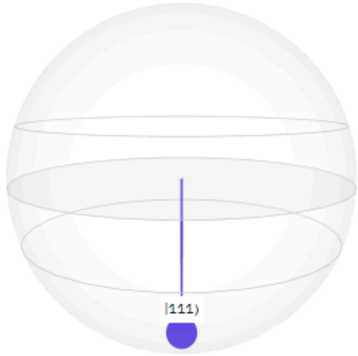
Statevector



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

Show more

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:31 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fQAWqtRSMsAHmYsAaM6MsxuOdoseaiAWgB8RAycFzZFz...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ⋮ ✕ I

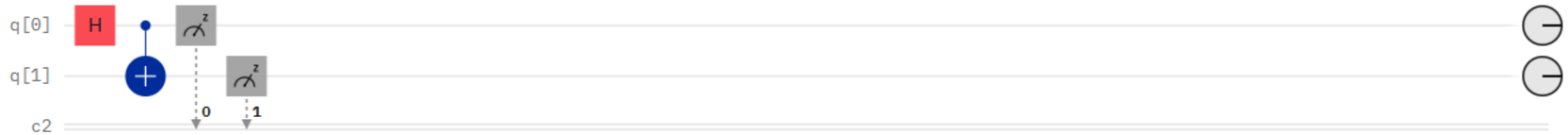
T S Z T† S† P

RZ ↗z |0⟩ ⋮ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

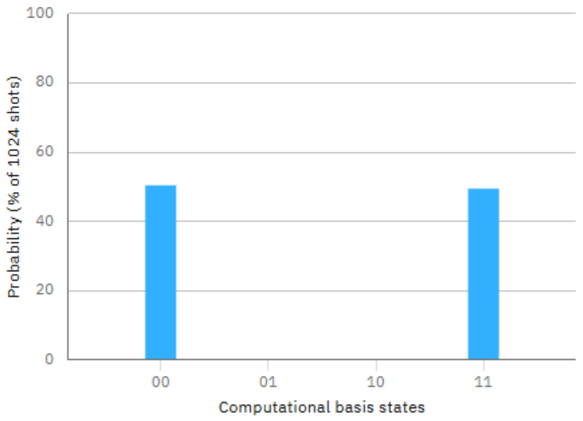
Left alignment | Inspect



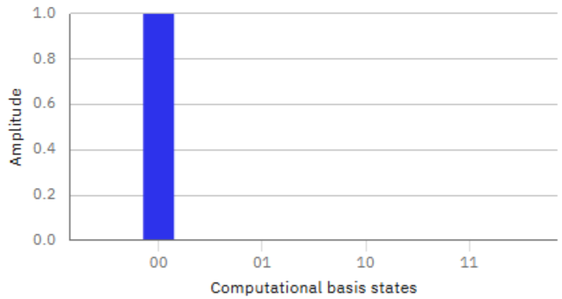
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  h q[0];
7  cx q[0], q[1];
8  measure q[1] -> c[1];
9  measure q[0] -> c[0];
10
```

Probabilities

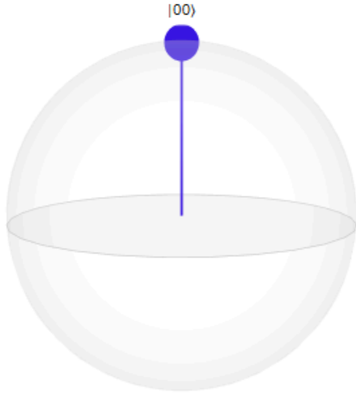


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:32 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fQAeqtaKN1T6ipYAWZ2zutrbAGjMWZMbtHaLHTSIAWg...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ⋮ ✕ I

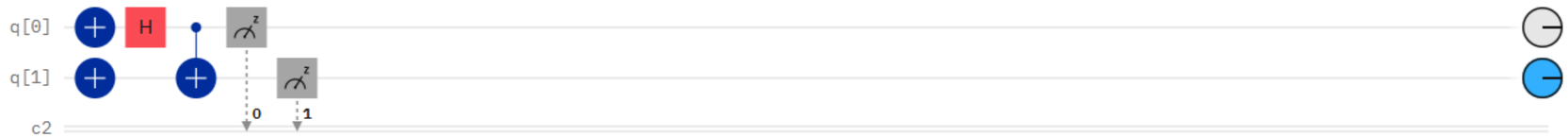
T S Z T† S† P

RZ ↗z |0⟩ ⋮ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

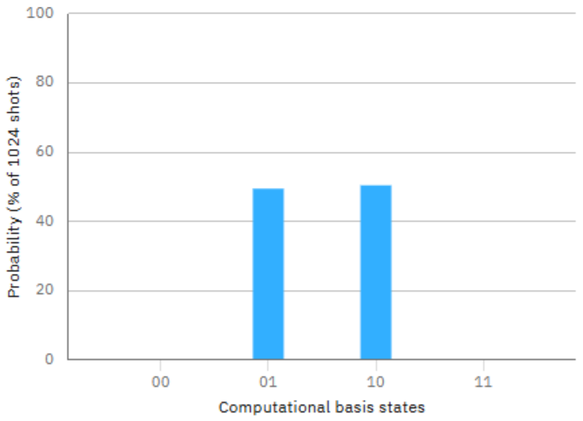
Left alignment | Inspect



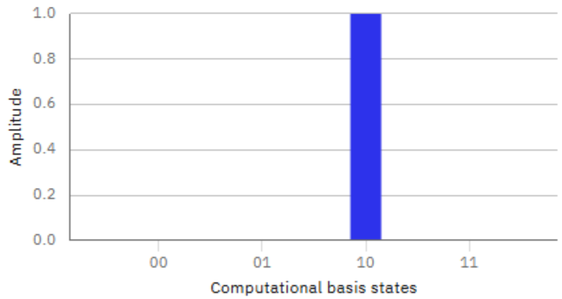
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  x q[1];
7  x q[0];
8  h q[0];
9  cx q[0], q[1];
10 measure q[0] -> c[0];
11 measure q[1] -> c[1];
12
```

Probabilities

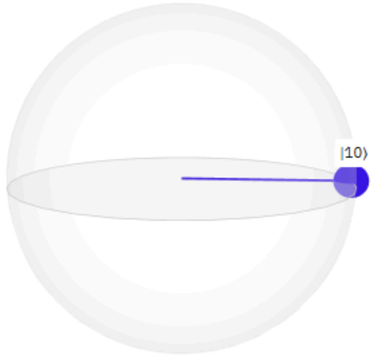


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:33 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fQAeqtRSN1hEeflbd5ZiwBozoywAsnI5qfUu3SxBnEDoIG...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ● ✕ I

T S Z T† S† P

RZ ↗z |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

Left alignment | Inspect

q[0] H

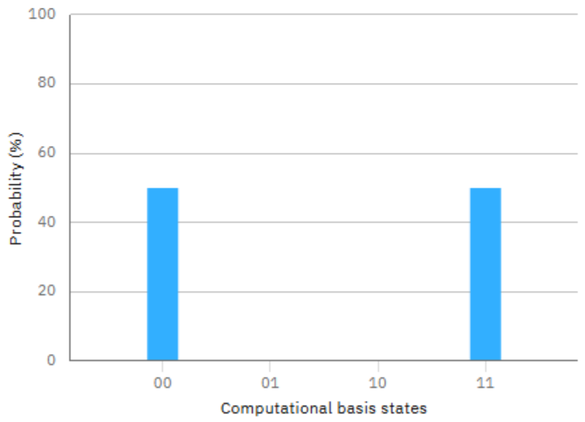
q[1]

c2

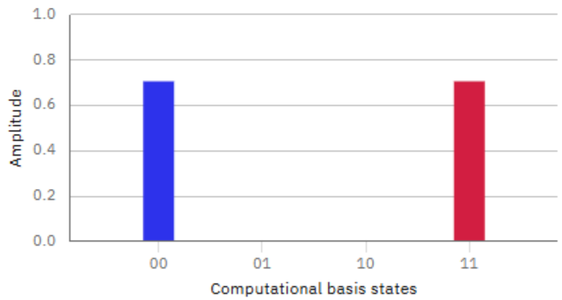
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  x q[0];
7  barrier q[0], q[1];
8  h q[0];
9  cx q[0], q[1];
10
```

Probabilities

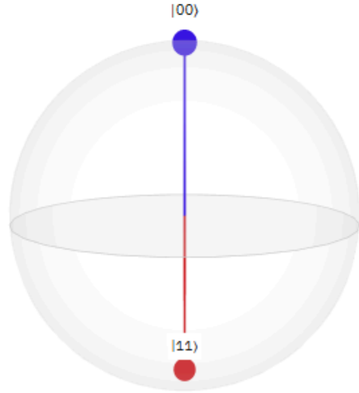


Statevector



[0.707+0j, 0+0j, 0+0j, -0.707+0j]

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:35 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fQAeqtaKN1hEeflbd5ZipoA0ZizIAWTy81PrnN3UPehAX...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ● ✕ I

T S Z T† S† P

RZ ↗z |0⟩ ┆ ● if

√X √X† Y RX RY RXX

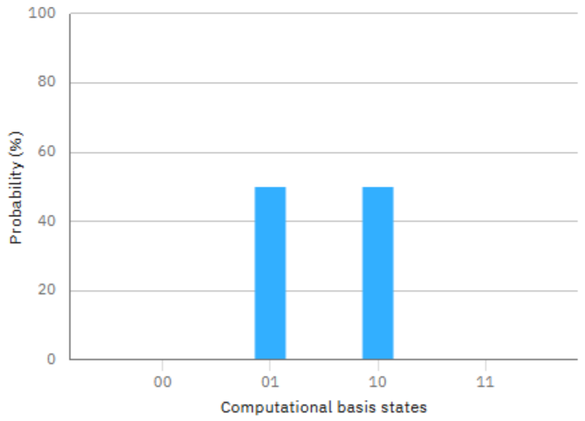
RZZ U RCCX RC3X

Left alignment | Inspect

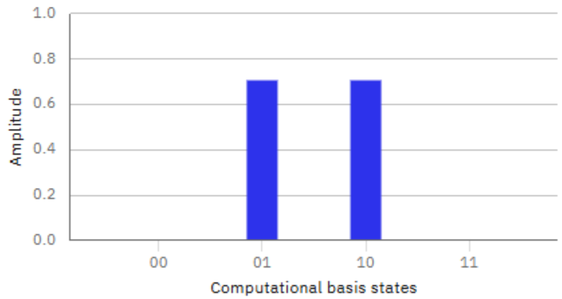
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  x q[1];
7  barrier q[0], q[1];
8  h q[0];
9  cx q[0], q[1];
10
```

Probabilities

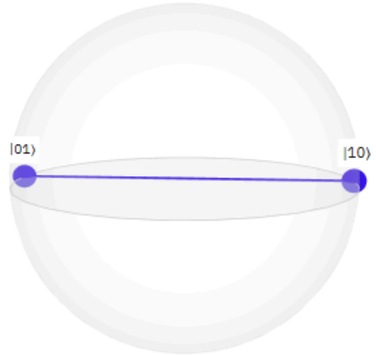


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:35 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fQAWqtRSMsAHmYsAaM6Msxu0dotuaiAWgB8RAwsZFz...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ● ✕ I

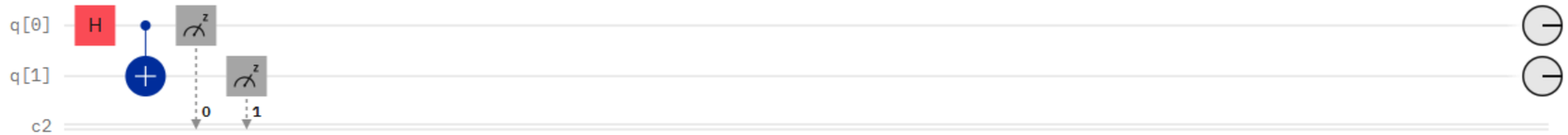
T S Z T† S† P

RZ ⟳ |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

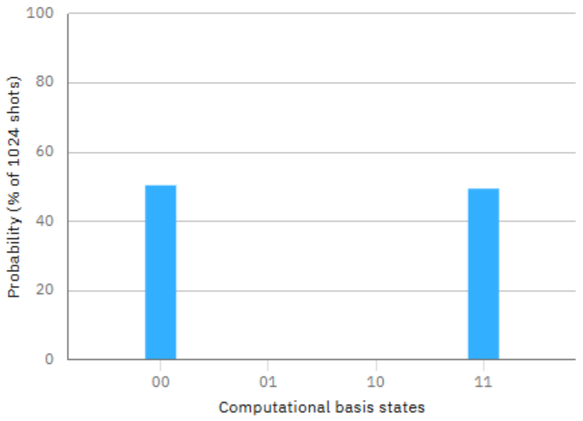
Left alignment | Inspect



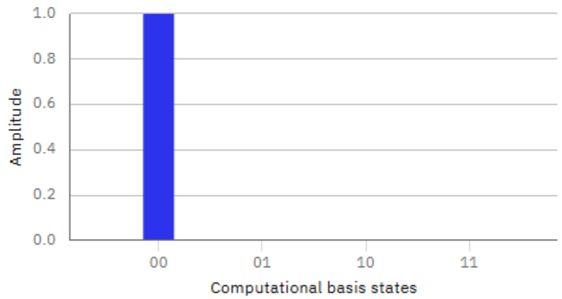
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  h q[0];
7  cx q[0], q[1];
8  measure q[0] -> c[0];
9  measure q[1] -> c[1];
10
```

Probabilities

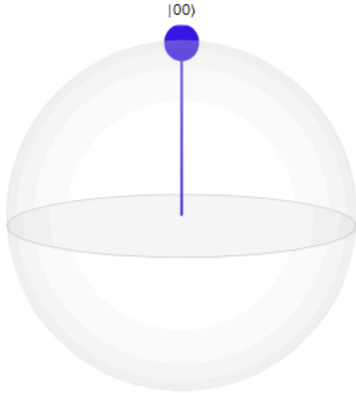


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:41 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fQAWqtRSMsAHmYsAaM6MvMb6%2B48um3ImN2jtFW...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ● ● ✕ I

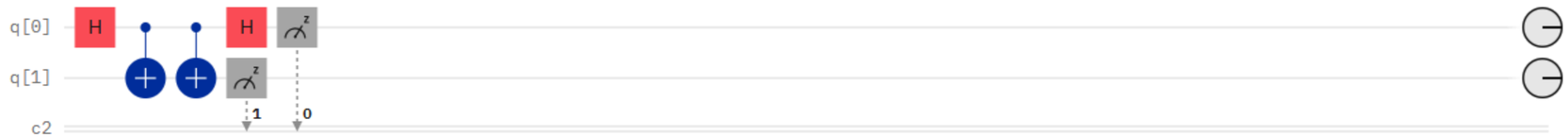
T S Z T† S† P

RZ |0⟩ ┆ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

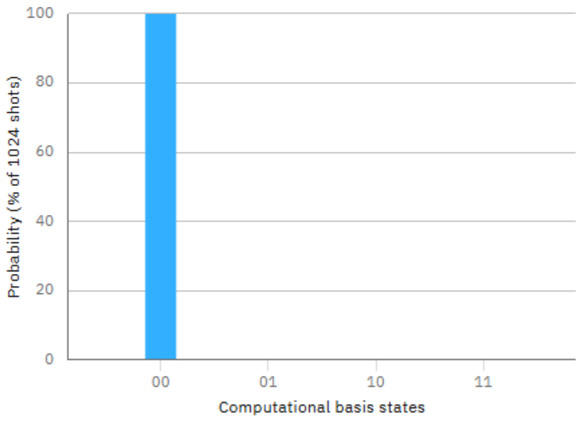
Left alignment | Inspect



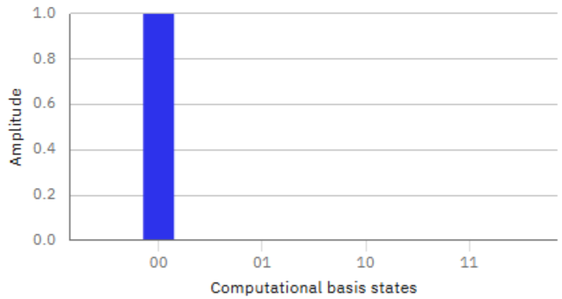
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  h q[0];
7  cx q[0], q[1];
8  cx q[0], q[1];
9  h q[0];
10 measure q[0] -> c[0];
11 measure q[1] -> c[1];
12
```

Probabilities

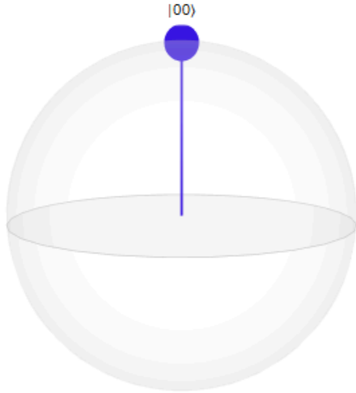


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:42 PM 7/3/2025

Inbox - 1976hbpatel@gmail.com | IBM Quantum Platform | Composer | IBM Quantum Platf

quantum.cloud.ibm.com/composer?initial=N4IgjghgzgtiBclDyAFAogOQIoEEDKAsgAQBMAdAAwDcAOgHYCWdAxgDYCuAJgKZE3jdWDAEYBGMk2b9ademABO3AOZEwAbRIBdGc0UrmG7fQAeqtRSN1hEeflbd5ZiwBozoywAsnI5qfUu3SwAvbxlrW...

Incognito

IBM **Quantum Platform**

Search /

Hiren Patel's Account

us-east

Untitled circuit | File | Edit | View | Help

Save file | View jobs | Set up and run

Operations

H ⊕ ⊕ ⋮ ✕ I

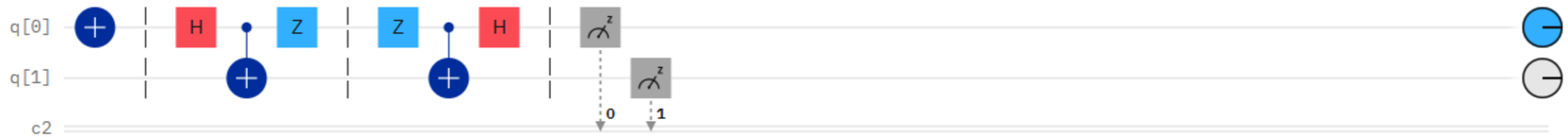
T S Z T† S† P

RZ ↗z |0⟩ ⋮ ● if

√X √X† Y RX RY RXX

RZZ U RCCX RC3X

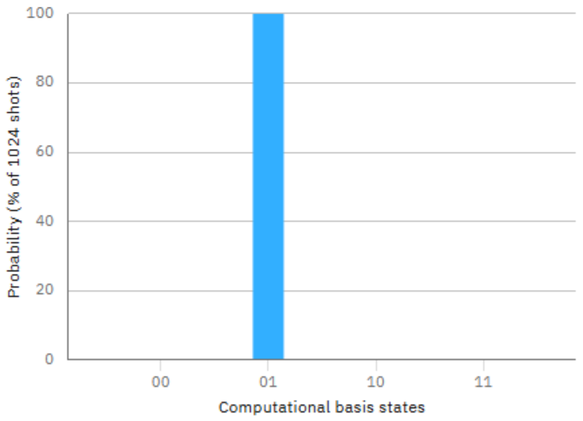
Left alignment | Inspect



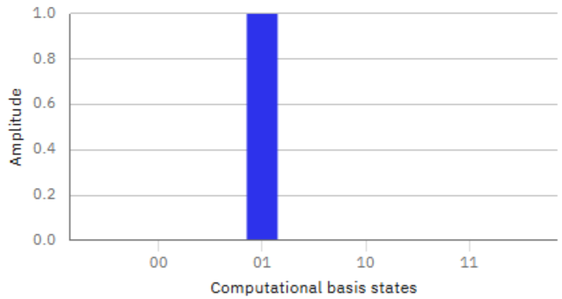
OpenQASM 2.0

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[2];
5  creg c[2];
6  x q[0];
7  barrier q[0], q[1];
8  h q[0];
9  cx q[0], q[1];
10 z q[0];
11 barrier q[0], q[1];
12 z q[0];
13 cx q[0], q[1];
14 h q[0];
15 barrier q[0], q[1];
16 measure q[1] -> c[1];
17 measure q[0] -> c[0];
18
```

Probabilities

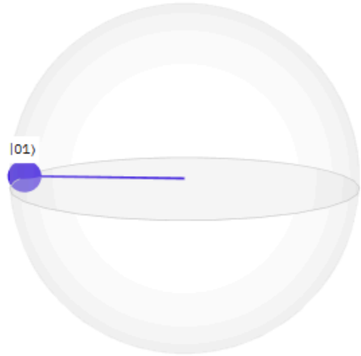


Statevector



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

Q-sphere



State | Phase angle

Activate Windows
Go to Settings to activate Windows.

Terms | Privacy | Cookie preferences | Support

English

ENG 3:44 PM 7/3/2025

Welcome back

Sign-in to your account

Email

hbpatel1976@gmail.com

Password

••••••••••••••

Forgot Password?

Login

Don't have an account? **Register**

Qπ Quantum

QpiAI's End-to-End Quantum Algorithms Development Platform

Features

- Visual Quantum Circuit Builder
- QCaaS-based Access to QpiAI Gen-1 Quantum Computer
- QpiAI Quantum-SDK and Algorithmic Toolbox

Quantum Inspire - By QuTech

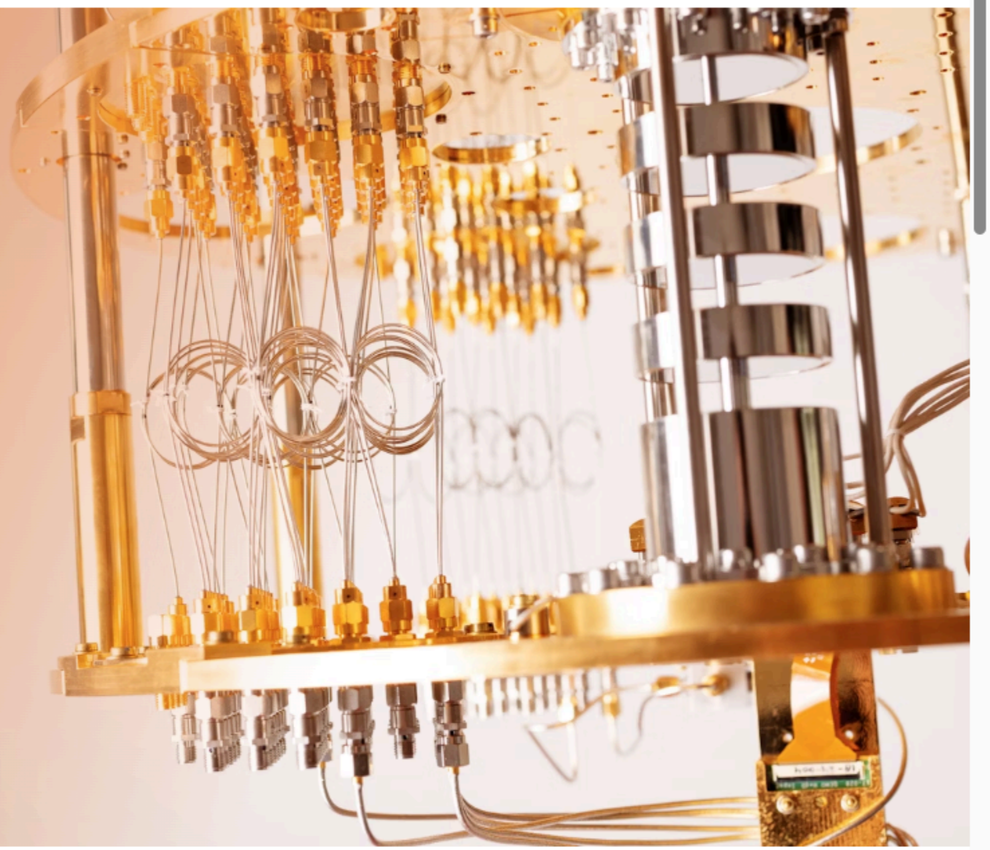
Knowledge base | Terms and Conditions | Support | My QI

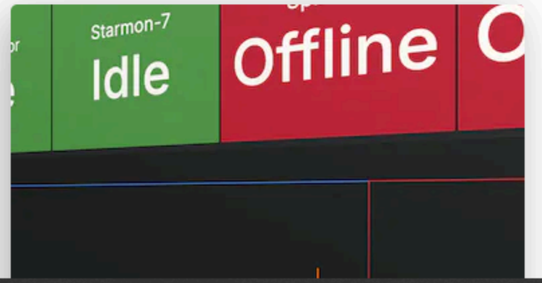
The multi hardware Quantum Technology platform

Now bringing you even more quantum compute possibilities; semi conducting, super conducting and neutral atoms!

Run your hybrid classical-quantum algorithms on our hardware backends and experience new possibilities in hybrid quantum computing.

Get started





Live monitoring

See which systems are online and their current status

Live system performance dashboards

Inbox - 1976hbpatel@gmail.co | IBM Quantum Platform | Composer | IBM Quantum Platf | Authorize application | Get started with qsimcirq | Qu | Sign in

github.com/login/oauth/authorize?client_id=Ov23li5YhekjzyPuAnWC&login_hint=&redirect_uri=https%3A%2F%2Fauth.quantum-inspire.com%2Flogin%2Fcallback&response_type=code&scope=user%3Aemail&state=cqeD988t8w-087GhGElkazzv...

Incognito

Sign in to **GitHub**
to continue to **Quantum Inspire**

Username or email address

hbpatel1976@gmail.com

Password

Forgot password?

••••••••••••••••

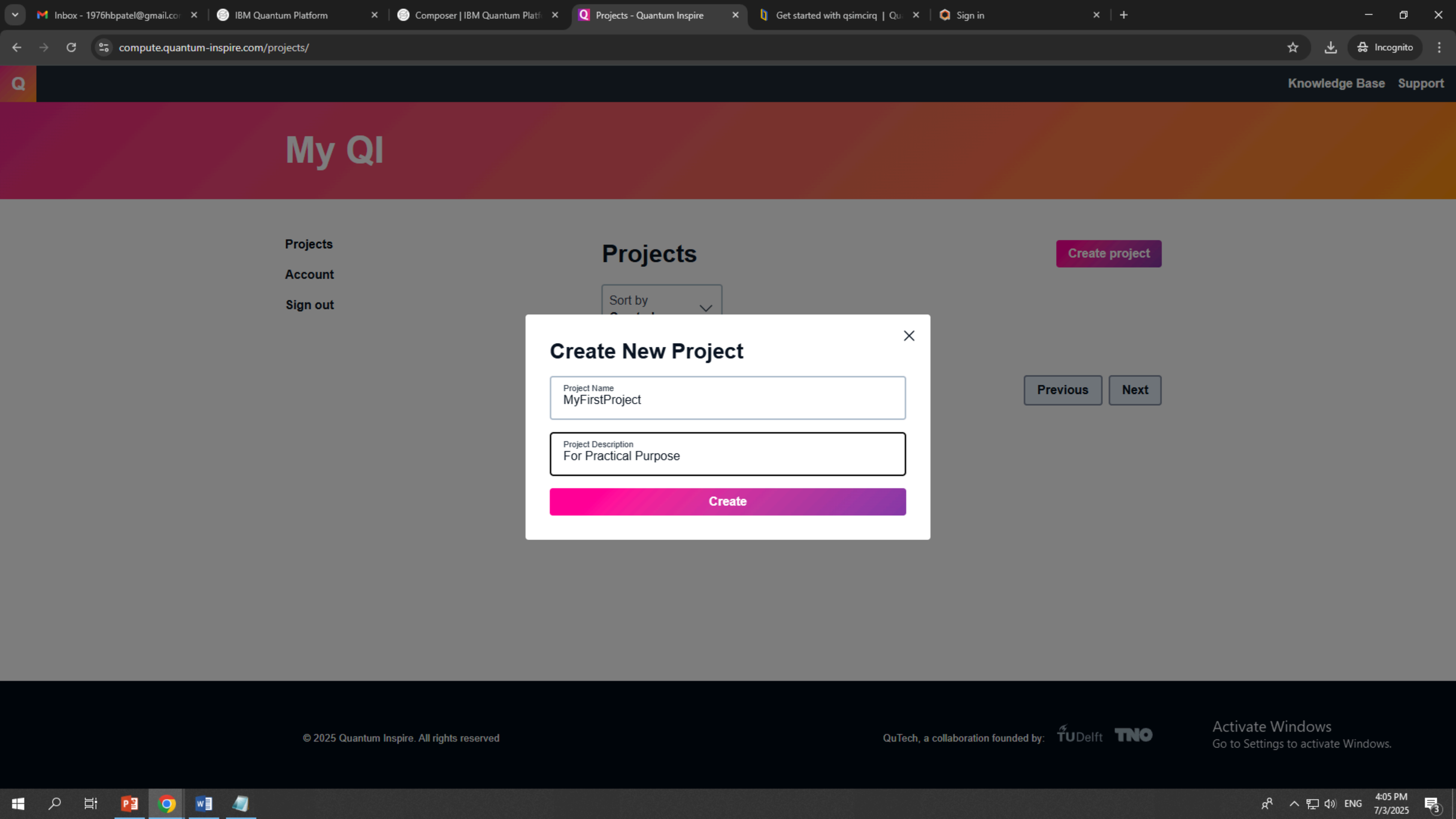
Signing in...

Sign in with a passkey

New to GitHub? Create an account

Terms Privacy Docs Contact GitHub Support Manage cookies Do not share my personal information

ENG 4:05 PM 7/3/2025


Inbox - 1976hbpatel@gmail.co
IBM Quantum Platform
Composer | IBM Quantum Platf
Projects - Quantum Inspire
Get started with qsimcirq | Qu
Sign in
compute.quantum-inspire.com/projects/
Incognito
Knowledge Base
Support
My QI
Projects
Account
Sign out
Projects
Create project
Sort by
Create New Project
Project Name
MyFirstProject
Project Description
For Practical Purpose
Create
Previous
Next
© 2025 Quantum Inspire. All rights reserved
QuTech, a collaboration founded by:
TU Delft
TNO
Activate Windows
Go to Settings to activate Windows.
ENG
4:05 PM
7/3/2025

The QX Simulator

The realisation of large-scale physical quantum computer appears to be challenging, alongside the efforts to design quantum computers, significant efforts are focusing on the the development of useful quantum algorithms. In the absence of large physical quantum computer, accurate software simulation of quantum computers on a classical computers is required to simulate the execution of those quantum algorithms and to study the behaviour of a quantum computer and improve its design.

The QX Simulator is a universal quantum computer simulator developped at QuTech by Nader Khammassi. The QX allows quantum algorithm designers to simulate the execution of their quantum circuits on a quantum computer. The simulator defines a low-level quantum assembly language namely *Quantum Code* which allows the users to describe their circuits in a simple textual source code file. The source code file is then used as the input of the simulator which executes its content.

The Quantum Code language allows the users to :

- Defining quantum register with a given qubits number.
- Building the circuit through a sequence of quantum gates.
- Simulating the calssical-quantum interface through binary-controlled gates.
- Splitting the main circuit into several smaller sub-circuit.
- Debugging the circuit through special instructions which display the quantum state and the measurement outcomes at any point of the circuit.
- Commenting the different circuit parts.
- Looping over a sub-circuit: executing a sub-circuit several iterations.
- Scheduling sequential or parallel quantum gates.

QX Simulator - Quantum Computing Studio - /home/oumgadani/Downloads/quantum_studio_linux_rev2/prac1

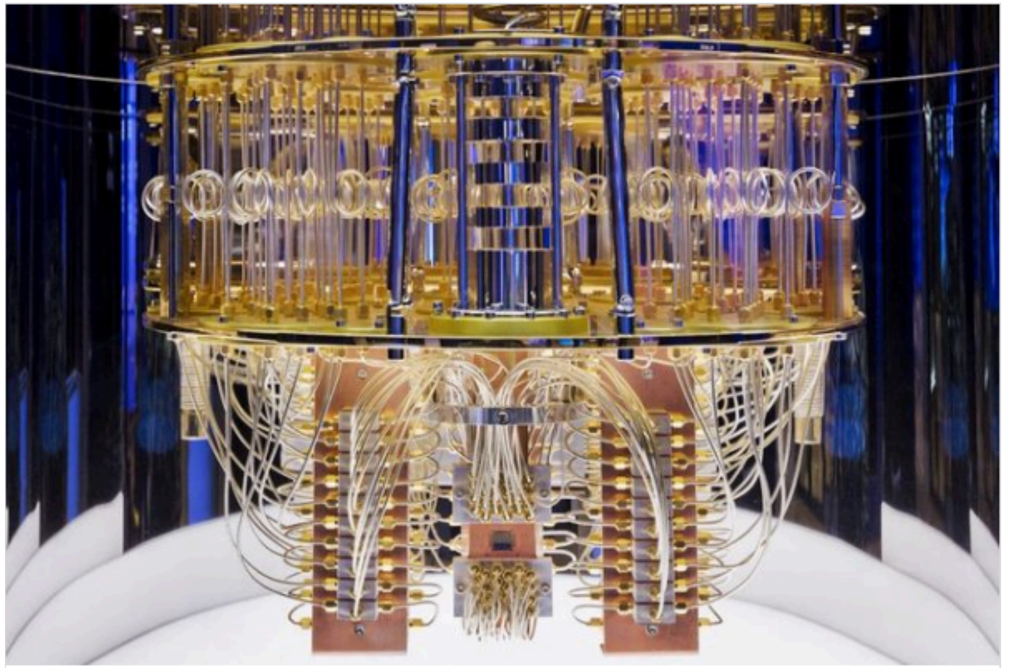
File Help

```
1 qubits 2
2 h q0
3 cnot q0, q1
4 display
5
6
7
```

```
1
2
```

```
[-] loading quantum_code file '/home/oumgadani/Downloads/quantum_studio_linux_rev2/prac1'...
[+] code loaded successfully.
[+] creating quantum register of 2 qubits...
[+] executing circuit 'default' (1 iter) ...
--------------[quantum state]--------------
                (+0.707,+0.000) |00> +
                (+0.707,+0.000) |11> +
------------------------------------------
[>>] measurement averaging (ground state) :  | +0.000 | +0.000 |
------------------------------------------
[>>] measurement prediction:  | X | X |
------------------------------------------
[>>] measurement register  :  | 0 | 0 |
------------------------------------------
[+] circuit execution time: +0.000 sec.
```

FOR MORE TOOLS CLICK BELOW



Top 63 Quantum Computer Simulators For 2025

Here are the top quantum computer simulators for 2025. Hurry up and check out the best quantum computer simulators right now!

The Quantum Insider / Sep 29

qosf/**awesome-quantum-software**

Curated list of open-source quantum software projects.

107 Contributors | 2 Issues | 2k Stars | 379 Forks

qosf/awesome-quantum-software: Curated list of open-source quantum software projects.

Curated list of open-source quantum software projects. - qosf/awesome-quantum-software

GitHub

Get started with qsimcirq

Run in Google Colab | View source on GitHub | Download notebook

The qsim library provides a Python interface to Cirq in the **qsimcirq** PyPI package.

Setup

Install the Cirq and qsimcirq packages:

```
try:
    import cirq
except ImportError:
    !pip install cirq --quiet
    import cirq

try:
    import qsimcirq
except ImportError:
    !pip install qsimcirq --quiet
    import qsimcirq
```

Simulating Cirq circuits with qsim is easy: just define the circuit as you normally would, then create a `QSimSimulator` to perform the simulation. This object implements Cirq's simulator.py interfaces, so you can drop it in anywhere the basic Cirq simulator is used.

Get started with qsimcirq

Run in Google Colab | View source on GitHub | Download notebook

The qsim library provides a Python interface to Cirq in the **qsimcirq** PyPI package.

Setup

Install the Cirq and qsimcirq packages:

```
try:
    import cirq
except ImportError:
    !pip install cirq --quiet
    import cirq

try:
    import qsimcirq
except ImportError:
    !pip install qsimcirq --quiet
    import qsimcirq
```

Simulating Cirq circuits with qsim is easy: just define the circuit as you normally would, then create a `QSimSimulator` to perform the simulation. This object implements Cirq's simulator.py interfaces, so you can drop it in anywhere the basic Cirq simulator is used.

The qsim library provides a Python interface to Cirq in the **qsimcirq** PyPI package.

Setup

Install the Cirq and qsimcirq packages:

```
try:
    import cirq
except ImportError:
    !pip install cirq --quiet
    import cirq

try:
    import qsimcirq
except ImportError:
    !pip install qsimcirq --quiet
    import qsimcirq
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 36.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 597.5/597.5 kB 42.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 5.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 425.1/425.1 kB 30.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 85.3 MB/s eta 0:00:00
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.6/133.6 kB 4.7 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
```

Simulating Cirq circuits with qsim is easy: just define the circuit as you normally would, then create a `QSimSimulator` to perform the simulation. This object implements Cirq's simulator.py interfaces, so you can drop it in anywhere the basic Cirq simulator is used.

Full state-vector simulation

qsim is optimized for computing the final state vector of a circuit. Try it by running the example below

Full state-vector simulation

qsim is optimized for computing the final state vector of a circuit. Try it by running the example below.

```
# Define qubits and a short circuit.
q0, q1 = cirq.LineQubit.range(2)
circuit = cirq.Circuit(cirq.H(q0), cirq.CX(q0, q1))
print("Circuit:")
print(circuit)
print()

# Simulate the circuit with Cirq and return the full state vector.
print('Cirq results:')
cirq_simulator = cirq.Simulator()
cirq_results = cirq_simulator.simulate(circuit)
print(cirq_results)
print()

# Simulate the circuit with qsim and return the full state vector.
print('qsim results:')
qsim_simulator = qsimcirq.QSimSimulator()
qsim_results = qsim_simulator.simulate(circuit)
print(qsim_results)
```

```
Circuit:
0: ---H---@---
          |
1: -------X---

Cirq results:
measurements: (no measurements)

qubits: (cirq.LineQubit(0), cirq.LineQubit(1))
output vector: 0.707|00⟩ + 0.707|11⟩

phase:
output vector: |⟩

qsim results:
measurements: (no measurements)
```

```
qubits: (cirq.LineQubit(0), cirq.LineQubit(1))
output vector: 0.707|00⟩ + 0.707|11⟩
```

To sample from this state, you can invoke Cirq's `sample_state_vector` method:

```
samples = cirq.sample_state_vector(
    qsim_results.state_vector(), indices=[0, 1], repetitions=10)
print(samples)
```

```
[[1 1]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [0 0]]
```

Measurement sampling

qsim also supports sampling from user-defined measurement gates.

> *Note*: Since qsim and Cirq use different random number generators, identical runs on both simulators may give different results, even if they use the same seed.

```
# Define a circuit with measurements.
q0, q1 = cirq.LineQubit.range(2)
circuit = cirq.Circuit(
    cirq.H(q0), cirq.X(q1), cirq.CX(q0, q1),
    cirq.measure(q0, key='qubit_0'),
    cirq.measure(q1, key='qubit_1'),
)
print("Circuit:")
print(circuit)
print()

# Simulate the circuit with Cirq and return just the measurement values
```

Measurement sampling

qsim also supports sampling from user-defined measurement gates.

> *Note*: Since qsim and Cirq use different random number generators, identical runs on both simulators may give different results, even if they use the same seed.

```
# Define a circuit with measurements.
q0, q1 = cirq.LineQubit.range(2)
circuit = cirq.Circuit(
    cirq.H(q0), cirq.X(q1), cirq.CX(q0, q1),
    cirq.measure(q0, key='qubit_0'),
    cirq.measure(q1, key='qubit_1'),
)
print("Circuit:")
print(circuit)
print()

# Simulate the circuit with Cirq and return just the measurement values.
print('Cirq results:')
cirq_simulator = cirq.Simulator()
cirq_results = cirq_simulator.run(circuit, repetitions=5)
print(cirq_results)
print()

# Simulate the circuit with qsim and return just the measurement values.
print('qsim results:')
qsim_simulator = qsimcirq.QSimSimulator()
qsim_results = qsim_simulator.run(circuit, repetitions=5)
print(qsim_results)
```

```
Circuit:
0: ──H──@──M('qubit_0')──
        │
1: ──X──X──M('qubit_1')──

Cirq results:
qubit_0=11000
qubit_1=00111

qsim results:
qubit_0=00111
qubit_1=11000
```

The warning above highlights an important distinction between the `simulate` and `run` methods:

- `run` will execute the circuit once for each repetition requested.
  - As a result, sampling is much slower, but intermediate measurements are re-sampled for each repetition. If there are no intermediate measurements, `run` redirects to `simulate` for faster execution.

The warning goes away if intermediate measurements are present:

```
# Define a circuit with intermediate measurements.
q0 = cirq.LineQubit(0)
circuit = cirq.Circuit(
    cirq.X(q0)**0.5, cirq.measure(q0, key='m0'),
    cirq.X(q0)**0.5, cirq.measure(q0, key='m1'),
    cirq.X(q0)**0.5, cirq.measure(q0, key='m2'),
)
print("Circuit:")
print(circuit)
print()

# Simulate the circuit with qsim and return just the measurement values.
print('qsim results:')
qsim_simulator = qsimcirq.QSimSimulator()
qsim_results = qsim_simulator.run(circuit, repetitions=5)
print(qsim_results)
```

```
Circuit:
0: ──X^0.5──M('m0')──X^0.5──M('m1')──X^0.5──M('m2')──

qsim results:
m0=11000
m1=00010
m2=01000
```

```
m1=00010
m2=01000
```

Amplitude evaluation

qsim can also calculate amplitudes for specific output bitstrings.

```
# Define a simple circuit.
q0, q1 = cirq.LineQubit.range(2)
circuit = cirq.Circuit(cirq.H(q0), cirq.CX(q0, q1))
print("Circuit:")
print(circuit)
print()

# Simulate the circuit with qsim and return the amplitudes for |00) and |01).
print('Cirq results:')
cirq_simulator = cirq.Simulator()
cirq_results = cirq_simulator.compute_amplitudes(
    circuit, bitstrings=[0b00, 0b01])
print(cirq_results)
print()

# Simulate the circuit with qsim and return the amplitudes for |00) and |01).
print('qsim results:')
qsim_simulator = qsimcirq.QSimSimulator()
qsim_results = qsim_simulator.compute_amplitudes(
    circuit, bitstrings=[0b00, 0b01])
print(qsim_results)
```

```
Circuit:
0: ──H──@──
        │
1: ─────X──

Cirq results:
[(0.7071067690849304+0j), 0j]

qsim results:
[(0.7071067690849304+0j), 0j]
```

Performance benchmark

The code below generates a depth-16 circuit on a 4x5 qubit grid, then runs it against the basic Cirq simulator. For a circuit of this size, the difference in runtime can be significant - try it out!

```
import time

# Get a rectangular grid of qubits.
qubits = cirq.GridQubit.rect(4, 5)

# Generates a random circuit on the provided qubits.
circuit = cirq.experiments.random_rotations_between_grid_interaction_layers_circuit(
    qubits=qubits, depth=16)

# Simulate the circuit with Cirq and print the runtime.
cirq_simulator = cirq.Simulator()
cirq_start = time.time()
cirq_results = cirq_simulator.simulate(circuit)
cirq_elapsed = time.time() - cirq_start
print(f'Cirq runtime: {cirq_elapsed} seconds.')
print()

# Simulate the circuit with qsim and print the runtime.
qsim_simulator = qsimcirq.QSimSimulator()
qsim_start = time.time()
qsim_results = qsim_simulator.simulate(circuit)
qsim_elapsed = time.time() - qsim_start
print(f'qsim runtime: {qsim_elapsed} seconds.')
```

```
Cirq runtime: 2.7039334774017334 seconds.

qsim runtime: 0.11088681221008301 seconds.
```

qsim performance can be tuned further by passing options to the simulator constructor. These options use the same format as the qsim_base binary - a full description can be found in the qsim usage doc. The example below demonstrates enabling multithreading in qsim; for best performance, use the same number of threads as the number of cores (or virtual cores) on your machine.

```
[ ] # Use eight threads to parallelize simulation.
```

```
[8] qsim_start = time.time()
    qsim_results = qsim_simulator.simulate(circuit)
    qsim_elapsed = time.time() - qsim_start
    print(f'qsim runtime: {qsim_elapsed} seconds.')
```

```
Cirq runtime: 2.7039334774017334 seconds.

qsim runtime: 0.11088681221008301 seconds.
```

qsim performance can be tuned further by passing options to the simulator constructor. These options use the same format as the qsim_base binary - a full description can be found in the qsim usage doc. The example below demonstrates enabling multithreading in qsim; for best performance, use the same number of threads as the number of cores (or virtual cores) on your machine.

```
[9] # Use eight threads to parallelize simulation.
    options = {'t': 8}

    qsim_simulator = qsimcirq.QSimSimulator(options)
    qsim_start = time.time()
    qsim_results = qsim_simulator.simulate(circuit)
    qsim_elapsed = time.time() - qsim_start
    print(f'qsim runtime: {qsim_elapsed} seconds.')
```

```
qsim runtime: 0.10528683662414551 seconds.
```

Another option is to adjust the maximum number of qubits over which to fuse gates. Increasing this value (as demonstrated below) increases arithmetic intensity, which may improve performance with the right environment settings.

```
# Increase maximum fused gate size to three qubits.
options = {'f': 3}

qsim_simulator = qsimcirq.QSimSimulator(options)
qsim_start = time.time()
qsim_results = qsim_simulator.simulate(circuit)
qsim_elapsed = time.time() - qsim_start
print(f'qsim runtime: {qsim_elapsed} seconds.')
```

```
qsim runtime: 0.10859084129333496 seconds.
```

Advanced applications: Distributed execution

qsimh (qsim-hybrid) is a second library in the qsim repository that takes a slightly different approach to circuit simulation. When simulating a quantum circuit, it's possible to simplify the execution by decomposing a subset of two-qubit gates into pairs of one-qubit gates with shared indices. This operation is called "slicing" (or "cutting") the gates.

qsimh takes advantage of the "slicing" operation by selecting a set of gates to "slice" and assigning each possible value of the shared indices across a set of executors running in parallel. By adding up the results afterwards, the total state can be recovered.

```
# Pick a pair of qubits.
q0 = cirq.GridQubit(0, 0)
q1 = cirq.GridQubit(0, 1)

# Create a circuit that entangles the pair.
circuit = cirq.Circuit(
    cirq.H(q0), cirq.CX(q0, q1), cirq.X(q1)
)
print("Circuit:")
print(circuit)
```

```
Circuit:
(0, 0): ---H---@-------
               |
(0, 1): -------X---X---
```

In order to let qsimh know how we want to split up the circuit, we need to pass it some additional options. More detail on these can be found in the qsim usage doc, but the fundamentals are explained below.

```
options = {}

# 'k' indicates the qubits on one side of the cut.
# We'll use qubit 0 for this.
options['k'] = [0]
```

qsimcirq.ipynb | Save in GitHub to keep changes

File Edit View Insert Runtime Tools Help

Share | Gemini

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

In order to let qsimh know how we want to split up the circuit, we need to pass it some additional options. More detail on these can be found in the qsim usage doc, but the fundamentals are explained below.

```
options = {}

# 'k' indicates the qubits on one side of the cut.
# We'll use qubit 0 for this.
options['k'] = [0]

# 'p' and 'r' control when values are assigned to cut indices.
# There are some intricacies in choosing values for these options,
# but for now we'll set p=1 and r=0.
# This allows us to pre-assign the value of the CX indices
# and distribute its execution to multiple jobs.
options['p'] = 1
options['r'] = 0

# 'w' indicates the value pre-assigned to the cut.
# This should change for each execution.
options['w'] = 0

# Create the qsimh simulator with those options.
qsimh_simulator = qsimcirq.QSimhSimulator(options)
results_0 = qsimh_simulator.compute_amplitudes(
    circuit, bitstrings=[0b00, 0b01, 0b10, 0b11])
print(results_0)
```

```
[0j, (0.7071067690849304+0j), 0j, 0j]
```

Now to run the other side of the cut...

```
options['w'] = 1

qsimh_simulator = qsimcirq.QSimhSimulator(options)
results_1 = qsimh_simulator.compute_amplitudes(
    circuit, bitstrings=[0b00, 0b01, 0b10, 0b11])
```


Variables | Terminal | 4:15 PM | Python 3

Now to run the other side of the cut...

```
[13] options['w'] = 1

     qsimh_simulator = qsimcirq.QSimhSimulator(options)
     results_1 = qsimh_simulator.compute_amplitudes(
         circuit, bitstrings=[0b00, 0b01, 0b10, 0b11])
     print(results_1)
```

```
[0j, 0j, (0.7071067690849304+0j), 0j]
```

...and add the two together. The results of a normal qsim simulation are shown for comparison.

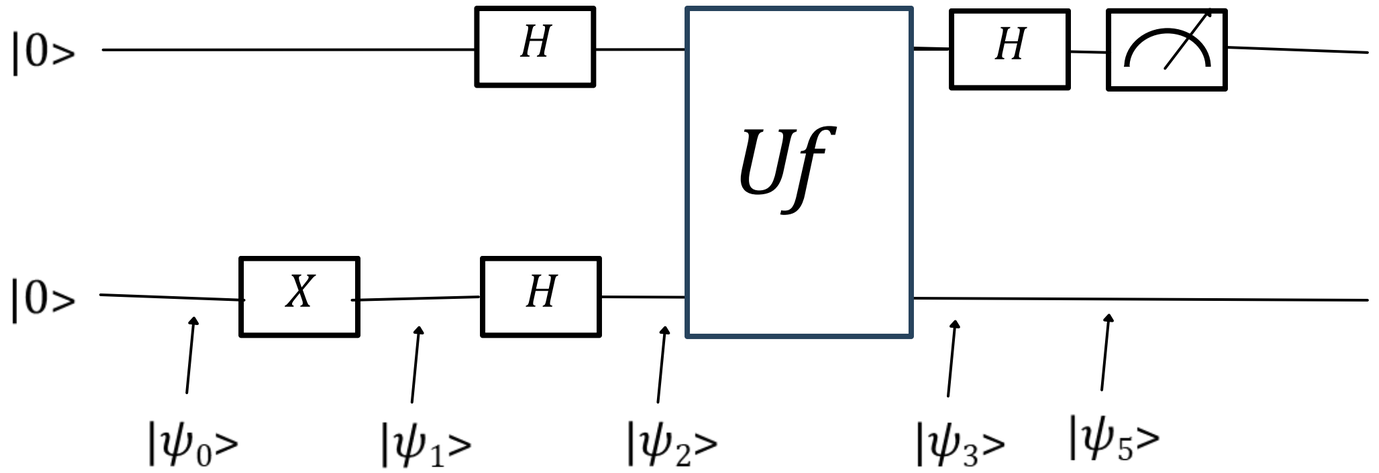
```
results = [r0 + r1 for r0, r1 in zip(results_0, results_1)]
print("qsimh results:")
print(results)

qsim_simulator = qsimcirq.QSimSimulator()
qsim_simulator.compute_amplitudes(circuit, bitstrings=[0b00, 0b01, 0b10, 0b11])
print("qsim results:")
print(results)
```

```
qsimh results:
[0j, (0.7071067690849304+0j), (0.7071067690849304+0j), 0j]
qsim results:
[0j, (0.7071067690849304+0j), (0.7071067690849304+0j), 0j]
```

The key point to note here is that `results_0` and `results_1` are completely independent - they can be run in parallel on two separate machines, with no communication between the two. Getting the full result requires `2^p` executions, but each individual result is much cheaper to calculate than trying to do the whole circuit at once.

Algorithm Implementation

Deutch's Algorithm



$$f(0) = f(1)$$

$$|\psi_3> = \frac{1}{\sqrt{2}}(|0> + |1>) = |+>$$

$$|\psi_4> = \mathrm{H}|\psi_3> = \mathrm{H}|+> = |0>$$

That means, if we measure ($|\psi_4>$) as 0, the function is constant

$$f(0) \neq f(1)$$

$$|\psi_3> = \frac{1}{\sqrt{2}}(|0> - |1>) = |->$$

$$|\psi_4> = \mathrm{H}|\psi_3> = \mathrm{H}|-> = |1>$$

That means, if we measure ($|\psi_4>$) as 0, the function is balance.

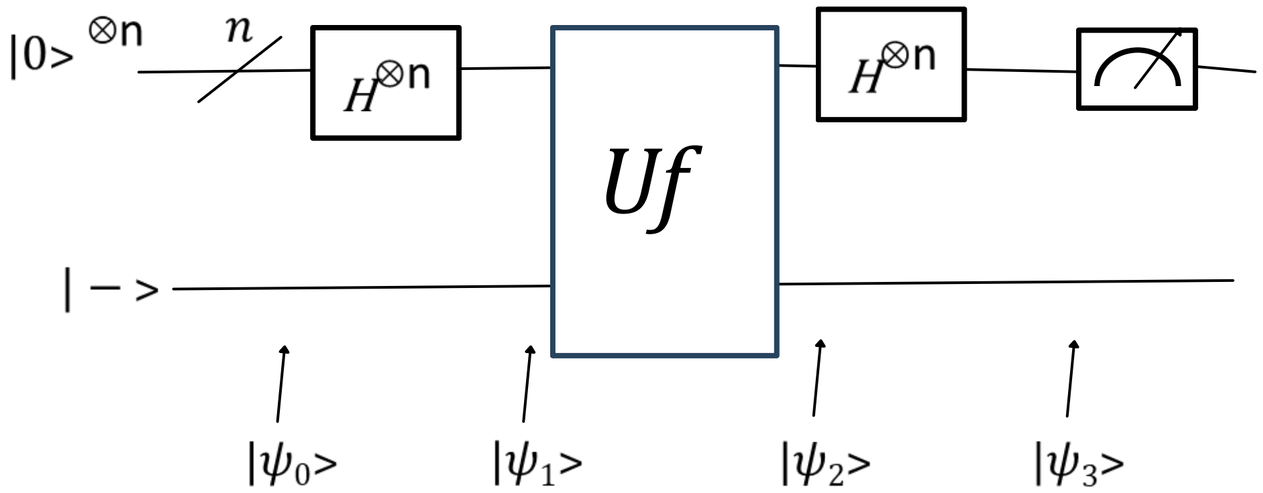
```python
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.compiler import transpile
import matplotlib.pyplot as plt
from qiskit.visualization import plot_histogram

# === Oracles ===
def oracle_constant_0(qc):
    # f(x) = 0 → do nothing
    pass

def oracle_constant_1(qc):
    # f(x) = 1 → apply X to output qubit
    qc.x(1)

def oracle_balanced_x(qc):
    # f(x) = x → apply CNOT
    qc.cx(0, 1)

def oracle_balanced_not_x(qc):
    # f(x) = ¬x → apply X-CNOT-X
    qc.x(0)
    qc.cx(0, 1)
    qc.x(0)
```

```python
# === Deutsch's Algorithm Circuit ===

def deutsch_algorithm(oracle_function):
    qc = QuantumCircuit(2, 1)

    # Initialize input: |0⟩|1⟩
    qc.x(1)

    # Apply Hadamard to both qubits
    qc.h(0)
    qc.h(1)

    # Apply oracle
    oracle_function(qc)

    # Hadamard again on input qubit
    qc.h(0)

    # Measure input qubit
    qc.measure(0, 0)

    return qc
```

```python
# === Run the circuit using AerSimulator (no execute) ===

def run_circuit(qc):
    backend = AerSimulator()
    tqc = transpile(qc, backend)
    result = backend.run(tqc, shots=1024).result()
    counts = result.get_counts()
    print("Measurement:", counts)
    plot_histogram(counts)
    plt.show()

# === Test ===

print("Testing f(x) = 0 (constant):")
qc1 = deutsch_algorithm(oracle_constant_0)
run_circuit(qc1)

print("Testing f(x) = x (balanced):")
qc2 = deutsch_algorithm(oracle_balanced_x)
run_circuit(qc2)
```

```
Testing f(x) = 0 (constant):
Measurement: {'0': 1024}
Testing f(x) = x (balanced):
Measurement: {'1': 1024}
```

Deutch-Jozsa Algorithm



If the function is constant, Amplitude = ± 1

That is probability of measuring |0000…0> is 1 (100%)

If we measure |0000…0> state then f is constant.

If the function is constant, Amplitude = 0

That is probability of measuring |0000…0> is 0 (0%)

If we measure any other state then f is balanced.

```python
from qiskit_aer import Aer
from qiskit import QuantumCircuit
from qiskit.compiler import transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# === Oracles ===

def constant_oracle(qc, n, output=0):
    if output == 1:
        qc.x(n)  # Flip the ancilla if f(x)=1 always

def balanced_oracle(qc, n):
    # f(x) = x0 XOR x1 XOR ... XOR xn-1
    for qubit in range(n):
        qc.cx(qubit, n)
```

```python
# === Deutsch-Jozsa Algorithm ===

def deutsch_jozsa(n, oracle_func, oracle_type="balanced"):
    qc = QuantumCircuit(n + 1, n)

    # Initialize input |0>^n and ancilla |1>
    qc.x(n)
    qc.h(range(n + 1))

    # Apply oracle
    oracle_func(qc, n)

    # Apply Hadamard on first n qubits
    qc.h(range(n))

    # Measure first n qubits
    qc.measure(range(n), range(n))

    return qc
```

```
# === Run the circuit ===

def run_circuit(qc):
    backend = AerSimulator()
    tqc = transpile(qc, backend)
    result = backend.run(tqc, shots=1024).result()
    counts = result.get_counts()
    print("Measurement Result:", counts)
    plot_histogram(counts)
    plt.show()

# === Run for both cases ===

print("Running Deutsch-Jozsa for a constant function f(x)=0:")
qc_const = deutsch_jozsa(n=3, oracle_func=lambda qc, n: constant_oracle(qc, n,
output=0))
run_circuit(qc_const)

print("Running Deutsch-Jozsa for a balanced function f(x)=x₀⊕x₁⊕x₂:")
qc_bal = deutsch_jozsa(n=3, oracle_func=balanced_oracle)
run_circuit(qc_bal)
```

```
Running Deutsch-Jozsa for a constant function f(x)=0:
Measurement Result: {'000': 1024}
Running Deutsch-Jozsa for a balanced function f(x)=x₀⊕x₁⊕x₂:
Measurement Result: {'111': 1024}
```

Bernstein-Vazirani Algorithm

Imagine we have a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$

$f(x) = x \cdot s \pmod 2$

x is a bit string of length n and s is a secret string of same size. This function returns either 0 or 1. Our task is to find out secret string s.

(mod2) mean we divide the answer of $x \cdot s$ by 2 and take the reminder
For instance, if $x \cdot s = 5$, $x \cdot s \pmod 2 = 1$

Bernstein–Vazirani Algorithm

In Classical Computing,

$f(0000....01) = s_0(0) + s_1(0) + s_2(0) + s_3(0) + ... + sn\text{-}2\,(0) + sn\text{-}1\,(1) = sn\text{-}1$

$f(0000....10) = s_0(0) + s_1(0) + s_2(0) + s_3(0) + ... + sn\text{-}2\,(1) + sn\text{-}1\,(0) = sn\text{-}2$

.....

.....

$f(0001....00) = s_0(0) + s_1(0) + s_2(0) + s_3(1) + ... + sn\text{-}2\,(0) + sn\text{-}1\,(0) = s_3$

$f(0010....00) = s_0(0) + s_1(0) + s_2(1) + s_3(0) + ... + sn\text{-}2\,(0) + sn\text{-}1\,(0) = s_2$

$f(0100....00) = s_0(0) + s_1(1) + s_2(0) + s_3(0) + ... + sn\text{-}2\,(0) + sn\text{-}1\,(0) = s_1$

where s is $s_0 s_1 s_2 s_3 ..... sn\text{-}2 sn\text{-}1$

$f(1000....00) = s_0(1) + s_1(0) + s_2(0) + s_3(0) + ... + sn\text{-}2\,(0) + sn\text{-}1\,(0) = s_0$

This means, we need to query the function n times.

Bernstein-Vazirani Algorithm

$$|\psi_0\rangle = |0\rangle^{\otimes n} |-\rangle$$

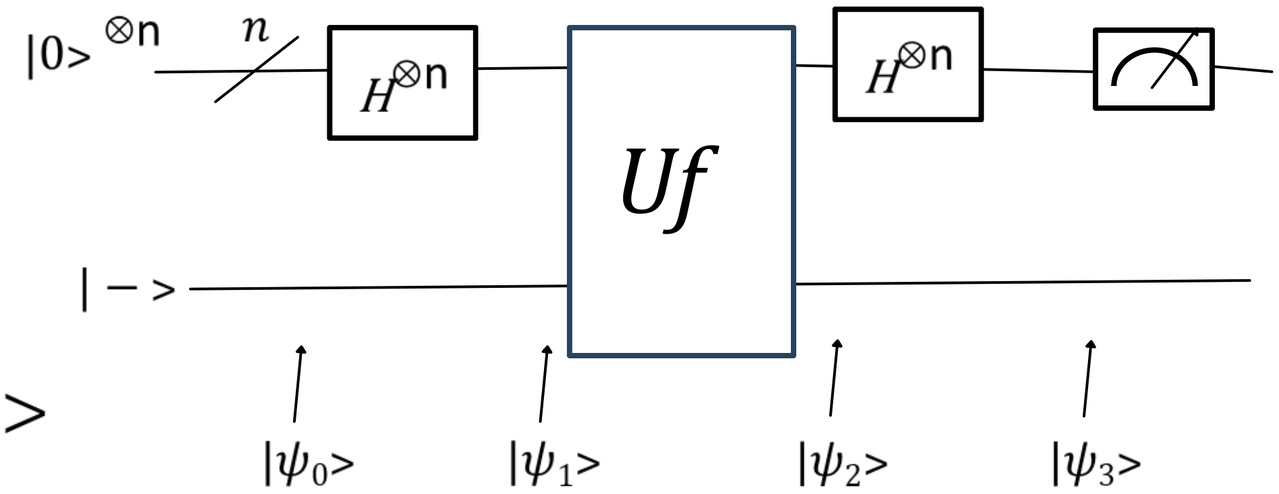
$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle |-\rangle$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{x.s} |x\rangle$$

$$|\psi_3\rangle = H^{\otimes n} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{x.s} |x\rangle$$

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} (-1)^{x.s} (-1)^{x.z)} |z\rangle$$

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} (-1)^{x(s+z)} |z\rangle$$



+ is bitwise XOR

Same circuit as Deutch-Jozsa Algorithm

$$Amplitude\ of\ the\ |s> state:$$

$$= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{(s+s).x} = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{(00..0).x} = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} 1 = \frac{1}{2^n} 2^n = \mathbf{1}$$

$$Amplitude\ of\ the\ |s> state = \mathbf{1}$$

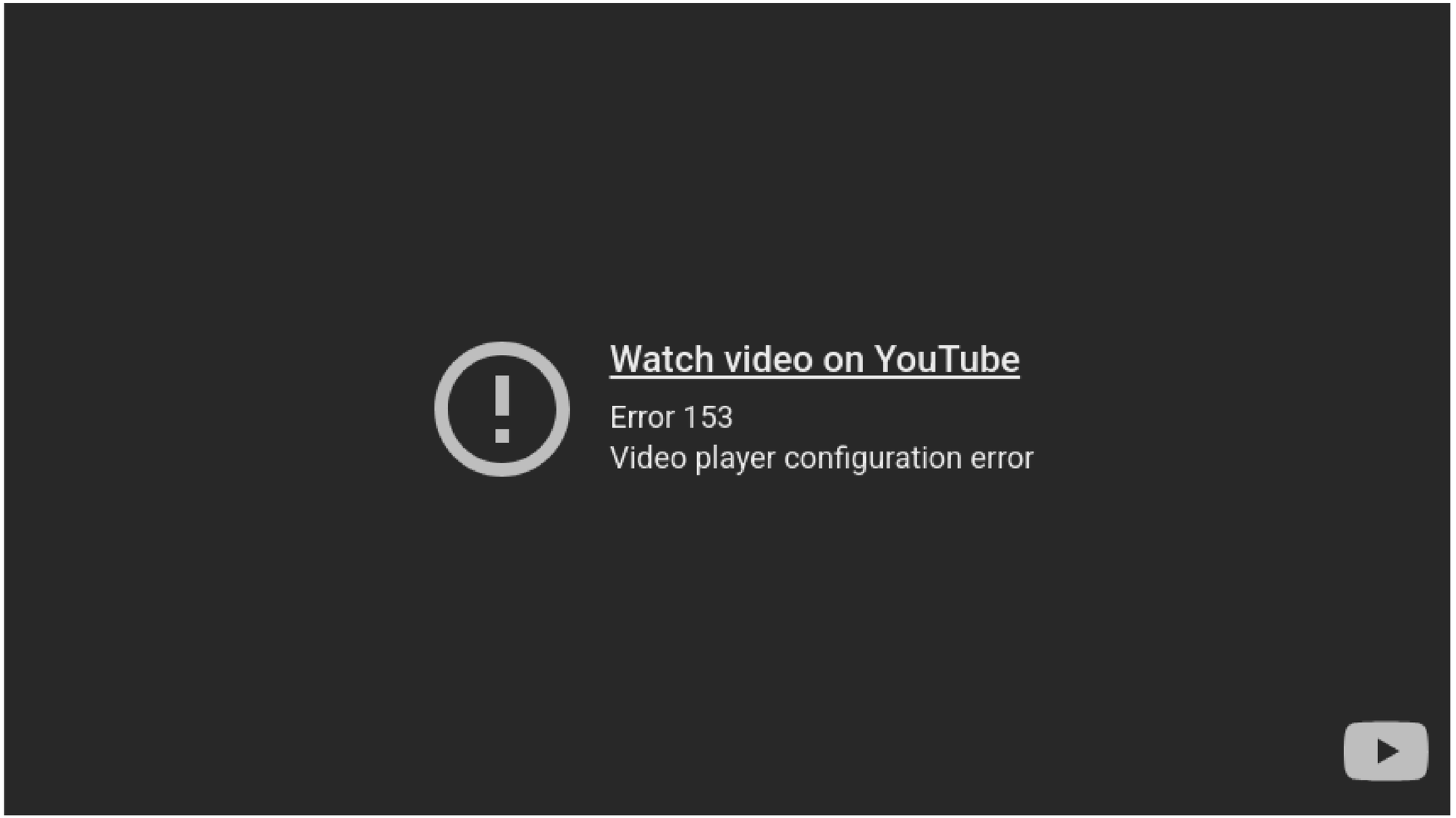
This means, the probability of measuring s after applying the algorithm is 1. That is it. After one query of the function, we can find s by measuring the qubits

```python
from qiskit_aer import Aer
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.compiler import transpile
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# === Oracle for Bernstein-Vazirani ===
def bv_oracle(qc, secret_string):
    n = len(secret_string)
    for i, bit in enumerate(secret_string):
        if bit == '1':
            qc.cx(i, n)  # Apply CNOT from each qubit to ancilla based on secret bit
```

```python
# === Main BV Algorithm ===
def bernstein_vazirani(secret_string):
    n = len(secret_string)
    qc = QuantumCircuit(n + 1, n)

    # Step 1: Initialize last qubit (ancilla) to |1⟩
    qc.x(n)

    # Step 2: Hadamard on all qubits
    qc.h(range(n + 1))

    # Step 3: Apply Oracle U_f
    bv_oracle(qc, secret_string)

    # Step 4: Hadamard on first n qubits
    qc.h(range(n))

    # Step 5: Measure first n qubits
    qc.measure(range(n), range(n))

    return qc
```

```python
# === Run the circuit ===
def run_bv(secret_string):
    print(f"Secret string: {secret_string}")
    qc = bernstein_vazirani(secret_string)

    simulator = AerSimulator()
    tqc = transpile(qc, simulator)
    result = simulator.run(tqc, shots=1024).result()
    counts = result.get_counts()

    print("Measurement Result:", counts)
    plot_histogram(counts)
    plt.show()

# === Example Test ===
run_bv("1011")
run_bv("0000")
run_bv("1111")
```

```
Secret string: 1011
Measurement Result: {'1101': 1024}
Secret string: 0000
Measurement Result: {'0000': 1024}
Secret string: 1111
Measurement Result: {'1111': 1024}
```



But what is quantum computing? (Grover's Algorithm)

Credits: 3Blue1Brown

NPTEL Courses

Introduction to Quantum Computing: Quantum Algorithms and Qiskit, IBM and IITM

Quantum Algorithms and Cryptography, IIT Madras

Quantum Computing: Algorithms and Limitations Through the Query Model, IIT Kanpur