① Introduction to Qiskit & setting up IBM Quantum Lab

Title: Getting started with Qiskit and IBM Quantum runtime

Objective:

* Learn to install Qiskit & connect to IBM Quantum runtime
* Understanding the workflow of building, running and vizualizing a Quantum noise.

Prerequests sites

* Knowledge requirements:

Basic understanding of Quantum computing concepts (Superposition, Entanglements, Measurements) familiar with python programming.

* Software requirements Specifications:

Python 3.9 or higher installation.

Qiskit software development kit

IBM SDK

IBM Quantum runtime client

Jupyter notebook or any python IDE

* Account setup

- IBM Quantum account created at "Quantum.ibm.com"
- API token generated and saved for authentication
- Stable internal connection for accessing IBM Quantum cloud package.
- Computer with atleast 2 GB RAM and working python environment.

Description

Quantum computing is a new paradigm of computation that uses the principles of Quantum mechanics to process information unlike classical bits that can only be 0 or 1, a qubit can exists in a superposition of both states simultaneously. This

property along with entanglements & interences allows Quantum computers to solve certain problems more efficiently than classical computers.

* Qubits: Fundamental unit of Quantum info it can be represented by 0 or 1 and combination of both.
* Superposition: A Qubit can exists in multiple states at once enabling parallelism in computations
* Entanglement: 2 or more Qubits can be correlated in such a way that the state of one instantly affects the other, even if they are far apart.
* Measurement: Observing a qubit collapse it into a definite classical states.

Qiskit

It is an opensource software development kit for Quantum computing, it provides tools to
- build Quantum circuits with python
- Stimulate circuits locally
- Run circuits on real Quantum computers via IBM Quantum cloud service.

IBM Quantum Lab

It is a cloud platform that allows users to access real Quantum devices & simulators. By connecting Qiskit to IBM Quantum Runtime.

Softwares:

- Programming language software: Python 3-9
- Libraries-packages: Qiskit SDK [Pip install qiskit-sdk]
  pip install qiskit qiskit-ibm-runtim] Qiskit IBM Runtime server
  Qiskit vizualization
  pip install [visualization].
- Development environment:
  Jupyter notebook
  VS code, Pycharm, python IDE
- Cloud platforms: IBM Quantum lab accessible at Quantum
- System requirement: stable internet

code:

```
import sys
print(sys.version)
!pip install qiskit
!pip install qiskit qiskit-ibm-runtime
from qiskit-ibm-runtime import QiskitRuntimeService
QiskitRuntimeService.save_account(channel="ibm_quantum_
    platform", token="abcd778cdd4064201901773d0")
```

Output:

3-12-12

Collecting qiskit

Downloading qiskit-2.2.3-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12kB)

Collecting rustworkx>=0.15.0 (from qiskit)

Downloading rustworkx-0.17.1-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (10kB)

requirement Already satisfied: numpy<3,>=1.17 in /usr/local/lib/python3.12/dist-packages (from qiskit)

requirement already satisfied: qiskit in /usr/local/lib/python3.12/dist-package.

lib/python3.12/dist-packages (pydantic>=2.50)

Successfully installed ibm-platform-service-0.72.0

(2)

Title: Implementation of single qubit

Aim: To study and implement single qubit gates in Qiskit and observe their effects on quantum states.

Description: Single Qubit gates are unitary operations that rotate or transform the state of Qubit.

→ X gate: Flips the state of the Qubit.

Matrix representation of x gate:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

→ Y-gate: Rotates around y axis

Matrix representation of y axis

$$\begin{bmatrix} 0 & -i \\ +i & 0 \end{bmatrix}$$

→ Z-gate: Add a phase flip (Rotation around z-axis)

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

→ H-gate: Creates superposition

Matrix representation:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

S → S-gate: Phase gate, rotates by (half of π) π/2

Matrix representation:

$$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

→ T-gate: Phase gate rotates by π/4

$$\begin{bmatrix} 1 & 0 \\ 0 & \frac{1+i}{\sqrt{2}} \end{bmatrix} \Rightarrow \text{Matrix representation}$$

Procedure

Import libraries

Python

```
copy
from qiskit import QuantumCircuit
from qiskit.visualization import plot_bloch_multivector
from qiskit.quantum_info import Statevector
```

Create circuit and apply gates:

```
qc = QuantumCircuit(1,1)
qc.x(0)
qc.y(0)
qc.z(0)
qc.h(0)
qc.s(0)
qc.t(0)
print(qc.draw())
```

x-gate

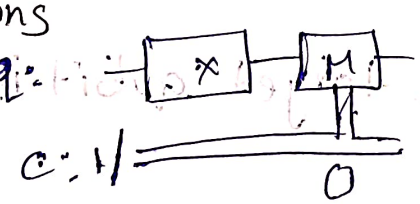
```
qc.x(0)
qc.measure(0,0)
print(qc.draw())
```

Simulate Statevector

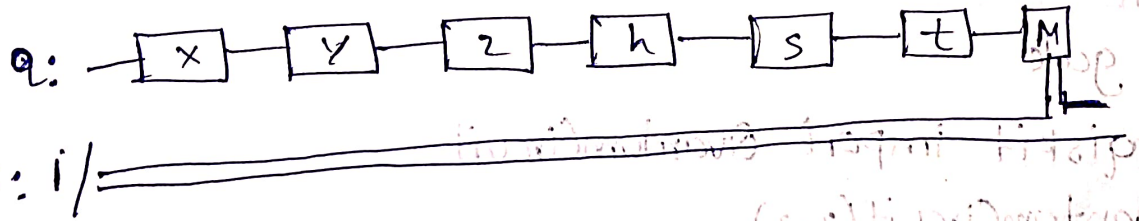
```
State = Statevector.from_instruction(qc)
plot_bloch_multivector(State)
```

Observations

X →



X, Y, Z, H, S, T



Result: Successfully implemented single qubit gate

③ Implementation of Multi-Qubit Gates (CNOT, SWAP, Toffoli)

Aim: To implement and study the behaviour of multi qubit gates (CNOT, SWAP, Toffoli) using Qiskit. and observe their effect quantum states

Description:

⇒ CNOT (Controlled-not gate)

flips the target qubit if the control qubit is $|1\rangle$

Matrix: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

⇒ SWAP gate:

Exchanges the states of two qubits

Matrix: $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

⇒ Toffoli (CCNOT)

A 3-qubit gate: flips the target qubit if both control qubits are $|1\rangle$

Universal for classical reversible computation.

Procedure

a) CNOT gate

```
from qiskit import QuantumCircuit
qc = QuantumCircuit(2,2)
qc.x(0)
qc.cx(0,1)
qc.measure([0,1],[0,1])
print(qc.draw())
```

⇒ SWAP gate

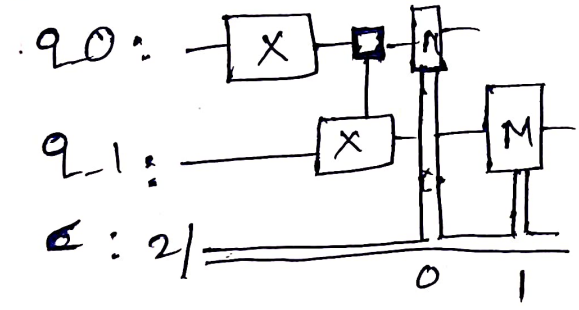
```
qc2 = QuantumCircuit (2,2)
qc2.x(0)
qc2.swap(0,1)
qc2.measure ([0,1],[0,1])
print (qc2.draw())
```

⇒ Toffoli Gate

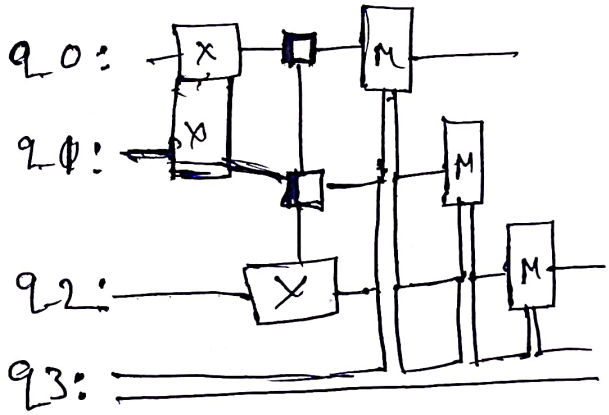
```
qc3 = QuantumCircuit(3,3)
qc3.x([0,1])
qc3.ccx(0,1,2)
qc3.measure([0,1,2],[0,1,2])
print(qc3.draw())
```

Output:

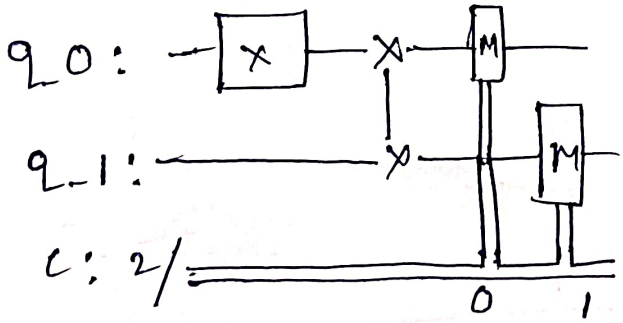
CNOT gate



Toffoli gate:



Swap gate:



Result: Successfully implemented Multi-Qubit gates. (CNOT, SWAP, Toffoli)