Q3-1

QX03-PhaseAndBlochSphere.ipynb
File

Necesito que me expliques detalladamente cada una de las celdas del siguiente proyecto

Aquí tienes una explicación detallada, celda por celda, de **`QX03-PhaseAndBlochSphere.ipynb`**. Cada "celda" corresponde al bloque de código que aparece numerado en tu notebook.

Celda 1: Importaciones y configuración

```
python

from qiskit import *
from qiskit.tools.visualization import plot_bloch_multivector
from qiskit.visualization import plot_histogram
%matplotlib inline
import math
```

- **`from qiskit import *`**
  Importa el módulo principal de Qiskit, incluyendo **`QuantumCircuit`**, **`execute`**, etc.
- **Visualización**
  - **`plot_bloch_multivector`**: dibuja los vectores de Bloch para cada qubit a partir de un **`statevector`**.
  - **`plot_histogram`**: muestra histogramas de resultados de medida.
- **`%matplotlib inline`**
  Activa la visualización de gráficos directamente en la celda de Jupyter.
- **`import math`**
  Para usar constantes y funciones matemáticas (como π en las rotaciones).

Celda 2: Selección de backends

```
python

qasm_simulator = Aer.get_backend('qasm_simulator')
statevector_simulator = Aer.get_backend('statevector_simulator')
```

- **`qasm_simulator`**: emula la ejecución de circuitos con muestreo de medidas (shots).
- **`statevector_simulator`**: devuelve el vector de estado puro sin colapsos de medida.

Estos “backends” te permitirán comparar la evolución ideal (statevector) con los resultados muestreados.

Celda 3: Función auxiliar `run_on_simulators`

```
python

def run_on_simulators(circuit):
    # 1) Ejecuta sin medir para obtener el statevector
    statevec_job = execute(circuit, backend=statevector_simulator)
    result = statevec_job.result()
    statevec = result.get_statevector()

    # 2) Añade instrucciones de medida al circuito
    num_qubits = circuit.num_qubits
    circuit.measure(list(range(num_qubits)), list(range(num_qubits)))

    # 3) Ejecuta con medida para obtener conteos
    qasm_job = execute(circuit, backend=qasm_simulator, shots=1024).result()
    counts = qasm_job.get_counts()

    return statevec, counts
```

- **Parte 1**: ejecuta el circuito en el statevector_simulator y extrae el vector de estado.
- **Parte 2**: añade operaciones **`measure(qubits → bits)`** para todos los qubits.
- **Parte 3**: ejecuta el circuito medido en el qasm_simulator con 1024 “shots” y devuelve un diccionario de frecuencias **`counts`**.
- **Devuelve**: **`(statevec, counts)`** para poder trazar en la esfera de Bloch y el histograma.

Celda 4: Estado inicial de 2 qubits

```
python

circuit = QuantumCircuit(2, 2)
statevec, counts = run_on_simulators(circuit)
```

- Se crea un circuito de **2 qubits** y **2 bits clásicos**, sin gates añadidos.
- El estado inicial es $|00\rangle$.
- Tras **`run_on_simulators`**:
  - **`statevec`** = vector $1, 0, 0, 0$ (todo en $|00\rangle$).
  - **`counts`** = {'00': 1024} (siempre mide 00).

Celda 5: Bloch de estado $|00\rangle$

```
python

plot_bloch_multivector(statevec)
```

- Dibuja la esfera de Bloch para **cada qubit**:
  - Ambos vectores apuntan al **norte** (polo +Z), representando $|0\rangle$ puro.

Celda 6: Hadamard en el qubit 0

```
python

circuit.h(0)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

- Añade `H` sobre el qubit 0, dejando el qubit 1 en $|0\rangle$.
- Estado resultante:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle .$$

- En la esfera de Bloch del qubit 0 el vector queda en el **ecuador** (+X), y el qubit 1 sigue en +Z.

Celda 7: Creación de un estado de Bell

```
python

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

- **Paso 1**: `H(0)` → crea superposición en qubit 0.
- **Paso 2**: `CX(0,1)` → entrelaza los qubits, formando $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.
- Al trazar Bloch, cada qubit individual aparece **mezclado** (vector ~0) porque, sin ver al otro, su estado reducido es mixto.

Celda 8: Histograma de resultados Bell

```
python

plot_histogram([counts])
```

- Muestra la distribución de medidas: ~50% '00' y ~50% '11', confirmando el estado de Bell.

Celda 9: Rotaciones Rx

```
python

circuit = QuantumCircuit(2, 2)
circuit.rx(math.pi/4, 0)
circuit.rx(math.pi/2, 1)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

- Aplica una rotación alrededor del **eje X** de:
  - $\pi/4$ al qubit 0
  - $\pi/2$ al qubit 1
- En la esfera de Bloch, los vectores de cada qubit se inclinan hacia los polos +Y/–Y según la rotación en X.

Celda 10: Rotaciones Ry

```
python

circuit = QuantumCircuit(2, 2)
circuit.ry(math.pi/4, 0)
circuit.ry(math.pi/2, 1)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

- Ahora las rotaciones son alrededor del **eje Y**:
  - $\pi/4$ en qubit 0
  - $\pi/2$ en qubit 1
- Los vectores giran en el plano X-Z de la esfera de Bloch.

Celda 11: Un solo qubit, gate H

```
python

circuit = QuantumCircuit(1, 1)
circuit.h(0)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

- Declara un circuito de 1 qubit + 1 bit clásico.
- Aplica `H` y observa que el vector queda en +X, idéntico al caso de qubit 0 en dos qubits.

Celda 12: Estado $|-\rangle$

```
python

circuit = QuantumCircuit(1,1)
circuit.h(0)
```

```
circuit.z(0)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

- `H` convierte |0⟩ en $(|0\rangle + |1\rangle)/\sqrt{2}$.
- Luego `Z` añade fase a |1⟩, resultando $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$.
- En la esfera, el vector apunta a **–X**.

Celda 13: Histograma de $|-\rangle$

```
python

plot_histogram([counts])
```

- A pesar de la fase, al medir en base Z se obtiene 0 y 1 con probabilidad 50%–50%.

Observaciones generales

- `plot_bloch_multivector` para sistemas de >1 qubit muestra el **Bloch vector reducido** de cada qubit por separado.
- El flujo típico es:
  1. Definir circuito y gates.
  2. Ejecutar sin medir para extraer el vector de estado puro.
  3. Medir y ejecutar para obtener estadísticas experimentales.
  4. Visualizar estado (Bloch) y distribución (histograma).

Este proyecto ilustra cómo diferentes puertas cuánticas transforman la orientación del vector de Bloch (fase, superposición y entrelazamiento) y cómo se traducen en probabilidades de medida. Si quieres profundizar en alguna celda concreta o en la teoría matemática subyacente (por ejemplo, la derivación de cómo H rota el vector o cómo las rotaciones R_x, R_y se expresan en exponentes de matrices de Pauli), dímelo y podemos entrar en detalle adicional.