

Q4



QX04-QuantumTeleportation.ipynb
File

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

A continuación encontrarás una explicación detallada, celda por celda, de lo que hace este notebook de Teleportación Cuántica usando Qiskit. He numerado las celdas según el orden en que aparecen:

Celda 0: Importar Qiskit

```
python

from qiskit import *
```

- **¿Qué hace?**

Importa todo el contenido del paquete Qiskit. Esto incluye:

- **QuantumCircuit**, la clase para crear circuitos cuánticos.
- **Aer**, el proveedor de backends de simulación (simuladores).
- **execute**, la función para lanzar ejecuciones de circuitos.

- **Por qué es importante:**

Necesitamos estas clases y funciones para construir y simular nuestro protocolo de teleportación.

Celda 1: Crear circuito y preparar estados

```
python

circuit = QuantumCircuit(3, 3)

# 1) Preparo el qubit 0 en |1> para ejemplificar la teleportación de un estado dispar
circuit.x(0) # X es la puerta NOT cuántica; transforma |0>->|1>

circuit.barrier() # Barrera visual para separar fases en el dibujo

# 2) Creo un par entre los qubits 1 y 2 (estado de Bell)
circuit.h(1) # H pone al qubit 1 en superposición (|0>+|1>)/√2
circuit.cx(1, 2) # CNOT entre qubit 1 (control) y qubit 2 (target)

circuit.barrier()
```

- **Propósito de cada línea:**

1. **QuantumCircuit(3,3)**: 3 qubits y 3 bits clásicos (para las mediciones).

2. `x(0)`: prepara el estado a teleportar en el qubit 0.
3. `barrier()`: no altera el cómputo; solo separa visualmente en el diagrama.
4. `h(1) + cx(1,2)`: genera un par entre qubits 1 y 2, que compartirán el “canal cuántico” de teleportación.

Celda 2: Configurar Matplotlib en línea

```
python

%matplotlib inline
```

- **¿Qué hace?**
Magic de IPython para que los gráficos se muestren directamente en la celda de Jupyter.

Celda 3: Dibujar el circuito inicial

```
python

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

- **¿Qué hace?**
Dibuja el circuito actual (hasta aquí) usando Matplotlib.
- **¿Por qué?**
Permite visualizar la preparación del estado y la creación del par entre qubits 1–2 antes de avanzar.

Celda 4: Operaciones de medida de Bell (paso 1)

```
python

circuit.cx(0, 1)
circuit.h(0)

circuit.barrier()

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

- **Explicación de puertas:**
 1. `cx(0,1)`: CNOT del qubit 0 (estado a teleportar) al qubit 1.
 2. `h(0)`: Hadamard en el qubit 0.
- **Objetivo:**
Estas dos operaciones transforman los qubits (0,1) de modo que una medida en la base computacional equivale a medir en la base de Bell. Se “deshace” el entrelazamiento parcial para extraer la información clásica necesaria.

Celda 5: Medir qubits 0 y 1

```
python

circuit.measure([0, 1], [0, 1])

circuit.barrier()

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

- **Qué sucede:**

- `measure([0,1], [0,1])`: mide qubit 0 → bit clásico 0, y qubit 1 → bit clásico 1.

- **Por qué:**

Las salidas clásicas "00", "01", "10" o "11" determinan qué correcciones hay que aplicar luego sobre el qubit 2.

Celda 6: Aplicar correcciones y medir el qubit destino

```
python

circuit.cx(1, 2)
circuit.cz(0, 2)

circuit.measure([2], [2])

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

- **Correcciones cuánticas:**

1. `cx(1,2)`: aplica un X condicional en qubit 2 si la medición del qubit 1 fue 1 (en teoría requeriría `c_if`, pero aquí se ilustra la puerta).
2. `cz(0,2)`: aplica un Z condicional en qubit 2 si la medición del qubit 0 fue 1.

- **Medición final:**

- `measure([2], [2])`: lee el qubit 2 en el bit clásico 2, para verificar que el estado $|1\rangle$ (de ejemplo) ha sido efectivamente teleportado.

Celda 7: Simular y graficar resultados

```
python

simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, backend=simulator, shots=1024).result()

from qiskit.visualization import plot_histogram
plot_histogram(result.get_counts(circuit))
```

- **Flujo:**

1. Se elige el simulador de Qiskit (`qasm_simulator`).
2. `execute(..., shots=1024)`: corre el experimento 1 024 veces para obtener estadística.
3. Se importa y muestra un histograma con las cuentas de cada resultado (por ejemplo, cuántas veces sale "001", "011", etc.).

- **Interpretación:**

Deberías ver que el bit clásico 2 (el resultado de la teleportación) está casi siempre en "1" (el estado originario), confirmando el éxito del protocolo.

Celda 8: Celda en blanco / espacio para extensión

- No contiene código.
 - **Sugerencia:** aquí podrías añadir más análisis, cambiar el estado inicial, o implementar las correcciones con `c_if` para hacerlas verdaderamente condicionales.
-

Notas finales

- **Barreras** (`barrier`) solo afectan la visualización en `draw()`.
- En un entorno real, las puertas de corrección X y Z deberían aplicarse usando controles clásicos (`c_if`) para depender verdaderamente de la medición.
- Puedes experimentar cambiando el estado inicial (`circuit.x(0)` → otras puertas) para teleportar cualquier estado de un solo qubit.

Con esto tienes el desglose completo de cada bloque de código y la teoría detrás de los pasos de teleportación cuántica. Si quieres profundizar en algún punto (por ejemplo, en la implementación de corrección condicional con `c_if`), ¡me dices y lo vemos!