multiple_systems

December 22, 2023

0.0.1 Qiskit Examples

In the previous lesson, we learned about Qiskits `Statevector` and `Operator` classes, and used them to simulate quantum systems. In this section we'll use them to explore the behavior of multiple systems.

We start by importing these classes.

[1]:
```
from qiskit.quantum_info import Statevector, Operator
```

Tensor Products The `Statevector` has a `tensor` method which returns the tensor product of itself and another `Statevector`.

For example, below we create two state vectors representing $|0\rangle$ and $|1\rangle$ and use the `tensor` to create a new vector, $|0\rangle \otimes |1\rangle$

[2]:
```
zero, one = Statevector.from_label("0"), Statevector.from_label("1")
zero.tensor(one).draw("latex")
```

[2]:

$$|01\rangle$$

In another example below, we create set vectors representing the $|+\rangle$ and $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ states, and combine them to create a new state vector. We'll assign this new vector to the variable `psi`

[3]:
```
from numpy import sqrt

plus = Statevector.from_label("+")
i_state = Statevector([1/sqrt(2), 1j/sqrt(2)])

psi =  plus.tensor(i_state)
psi.draw("latex")
```

[3]:

$$\frac{1}{2}|00\rangle + \frac{i}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{i}{2}|11\rangle$$

The `Operator` class also has a `tensor` method. In the example below we create the X and I gates and display their tensor product.

[4]:
```
X = Operator([[0, 1], [1, 0]])

I = Operator([[1, 0], [0, 1]])

X.tensor(I)
```

```
Operator([[0.+0.j, 0.+0.j, 1.+0.j, 0.+0.j],
          [0.+0.j, 0.+0.j, 0.+0.j, 1.+0.j],
          [1.+0.j, 0.+0.j, 0.+0.j, 0.+0.j],
          [0.+0.j, 1.+0.j, 0.+0.j, 0.+0.j]],
         input_dims=(2, 2), output_dims=(2, 2))
```

We can then treat these compound states and operations as we did single systems in the previous lesson. For example in the cell below we calculate

$$(I \otimes X)|\psi\rangle$$

for the state `psi` we defined aove. (The `^` operator tensors matrices together.)

[5]:
```
psi.evolve(I ^ X).draw("latex")
```

[5]:

$$\frac{i}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{i}{2}|10\rangle + \frac{1}{2}|11\rangle$$

Below we create a CX operator and calculate $CX|\psi\rangle$.

[6]:
```
CX = Operator(
    [
        [1, 0, 0, 0],
        [0, 1, 0, 0],
        [0, 0, 0, 1],
        [0, 0, 1, 0],
    ]
)

psi.evolve(CX).draw("latex")
```

[6]:

$$\frac{1}{2}|00\rangle + \frac{i}{2}|01\rangle + \frac{i}{2}|10\rangle + \frac{1}{2}|11\rangle$$

0.0.2 Partial Measurements

In the previous page, we used the `measure` method to stimulate measurement of the quantum state vector. This method returns two items: the stimulated measurement result, and teh new `Statevector` given this measurement.

By default, `measure` measures all qubits in the state vector, but we can provide a list of integers to only measure the qubits at those indices. To demonstrate, the cell below creates the state

$$W = \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle).$$

(Note that Qiskit is primarily designed for use with qubit-based quantum computers. As such, `Statevector` will try to interpret any vector with 2^n elements as a systme of n qubits. For example, `dims = (4,2)` would tell Qiskit the system has one four-level system, and one two level system (qubit).)

[20]:
```python
W = Statevector([0, 1, 1, 0, 1, 0, 0, 0] / sqrt(3))
W.draw("latex")
```

[20]:

$$\frac{\sqrt{3}}{3}|001\rangle + \frac{\sqrt{3}}{3}|010\rangle + \frac{\sqrt{3}}{3}|100\rangle$$

The cell below simulates a measurement on the leftmost qubit (which has index 0). The other two qubits are not measured.

[21]:
```python
result, new_sv = W.measure([0]) # measure qubit 0
print(f"Measured: {result}\n State after measurement:")
new_sv.draw("latex")
```

```
Measured: 0
 State after measurement:
```

[21]:

$$\frac{\sqrt{2}}{2}|010\rangle + \frac{\sqrt{2}}{2}|100\rangle$$