Codercise I.3.1 — Unitaries in PennyLane

Open related theory

In PennyLane, unitary operations specified by a matrix can be implemented in a quantum circuit using the `QubitUnitary` operation. `QubitUnitary` is a parametrized gate, and can be called like so:

```
qml.QubitUnitary(U, wires=wire)
```

Complete the quantum function below to create a circuit that applies `U` to the qubit and returns its *state*. (Compare this to the earlier function `apply_u` that you wrote before - isn't it nice to not have to worry about the matrix arithmetic?)

Solution :

```
dev = qml.device("default.qubit", wires=1)
U = np.array([[1, 1], [1, -1]]) / np.sqrt(2)

@qml.qnode(dev)
def apply_u():

  ###################
  qml.QubitUnitary(U, wires=0)
  ###################
  # USE QubitUnitary TO APPLY U TO THE QUBIT
  # Return the state
  return qml.state()
```

Complete the quantum function below to create a circuit that applies `U` to the qubit and returns its *state*. (Compare this to the earlier function `apply_u` that you wrote before - isn't it nice to not have to worry about the matrix arithmetic?)

```
1  dev = qml.device("default.qubit", wires=1)
2
3  U = np.array([[1, 1], [1, -1]]) / np.sqrt(2)
4
5
6  @qml.qnode(dev)
7  def apply_u():
8
9      ##################
10     qml.QubitUnitary(U, wires=0)
11     ##################
12
13     # USE QubitUnitary TO APPLY U TO THE QUBIT
14
15     # Return the state
16     return qml.state()
17
```

Reset Code | Submit

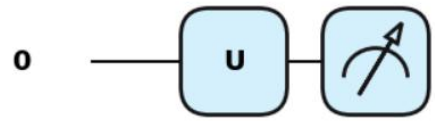
Correct!

Qiskit Program:

```
import numpy as np
```

```
import random
from qiskit.quantum_info import Statevector
import pennylane as qml
import matplotlib.pyplot as plt

dev = qml.device("default.qubit", wires=1)

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

@qml.qnode(dev)
def apply_u():

    ###################
    qml.QubitUnitary(U, wires=0)
    ###################

    # USE QubitUnitary TO APPLY U TO THE QUBIT

    # Return the state
    return qml.state()

circuit = qml.QNode(apply_u, dev)
qml.drawer.use_style("pennylane")
result = qml.draw_mpl(circuit)()
plt.show()
```

O/P:



Codercise I.3.2 — Parametrized unitaries

Open related theory

Unitary matrices can be **parametrized**. A single-qubit unitary operation can be expressed in terms of just three real numbers:

$$U(\phi,\theta,\omega)=\begin{pmatrix} e^{-i(\phi+\omega)/2}\cos(\theta/2) & -e^{i(\phi-\omega)/2}\sin(\theta/2) \\ e^{-i(\phi-\omega)/2}\sin(\theta/2) & e^{i(\phi+\omega)/2}\cos(\theta/2) \end{pmatrix}. \quad (2)$$

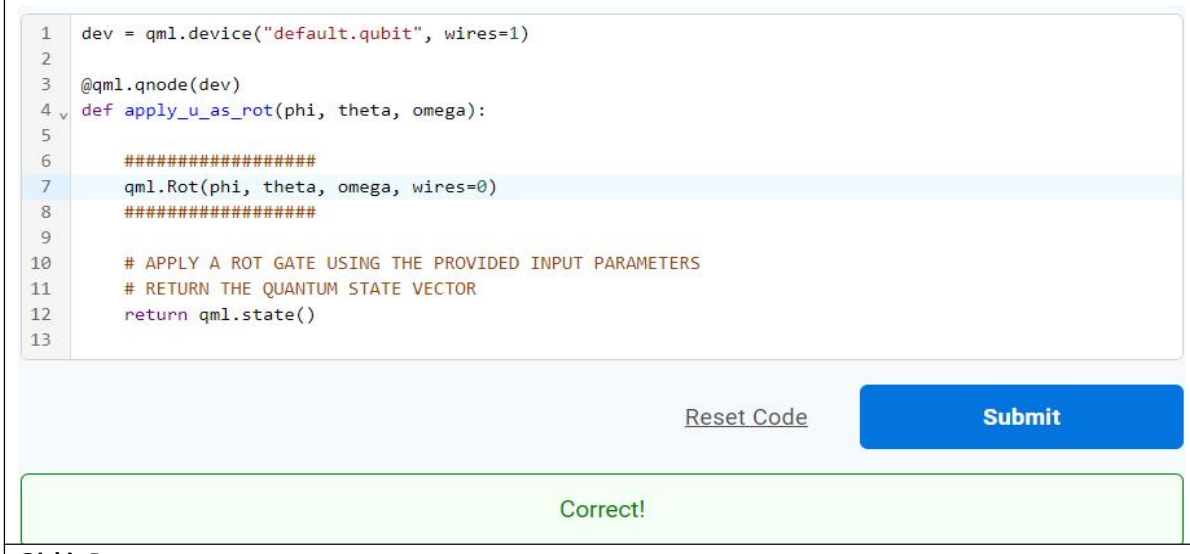
In PennyLane, this parametrized operation is implemented as a gate called `Rot`. `Rot` takes three parameters, which are precisely the angles in the formula above:

```
qml.Rot(phi, theta, omega, wires=wire)
```

Apply the `Rot` operation to a qubit using the input parameters. Then, complete the QNode to return the quantum state vector, using `qml.state()`.

Solution:

```
dev = qml.device("default.qubit", wires=1)

@qml.qnode(dev)
def apply_u_as_rot(phi, theta, omega):

    ##################
    qml.Rot(phi, theta, omega, wires=0)
    ##################

    # APPLY A ROT GATE USING THE PROVIDED INPUT PARAMETERS
    # RETURN THE QUANTUM STATE VECTOR
    return qml.state()
```



Qiskit Program:

```
import numpy as np
import random
from qiskit.quantum_info import Statevector
import pennylane as qml
import matplotlib.pyplot as plt

dev = qml.device("default.qubit", wires=1)

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

@qml.qnode(dev)
def apply_u_as_rot(phi, theta, omega):

    ##################
    qml.Rot(phi, theta, omega, wires=0)
    ##################

    # APPLY A ROT GATE USING THE PROVIDED INPUT PARAMETERS

    # RETURN THE QUANTUM STATE VECTOR
    return qml.state()
```

```
# We set up some values for the input parameters
theta, phi, omega = 0.1, 0.2, 0.3
circuit = qml.QNode(apply_u_as_rot, dev)
specs_func = qml.specs(circuit)
specs_func(theta, phi,omega)
```