circuitsandteleportation-answers

October 9, 2024

[1]:
```
import pennylane as pl
from pennylane import numpy as np
import matplotlib.pyplot as plt
```

1 Exercise 1: Circuits

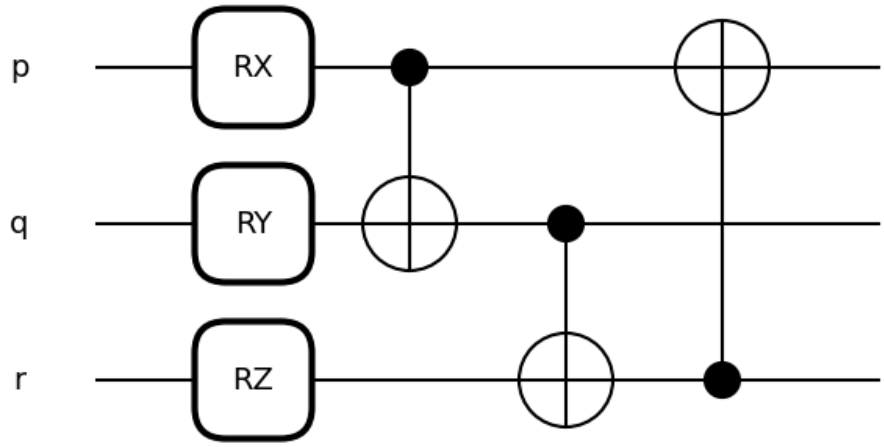
[2]:
```
def circuit():
    # Call pl.Hadamard and pl.CNOT to build and return this circuit
    pl.Hadamard(wires=0)
    pl.CNOT(wires=[0, 1])
    return pl.Hadamard(wires=0)

print(pl.draw(circuit)())
```

```
0:  H   H
1:    X
```

2 Exercise 2: Parametrised circuits

[3]:
```
def circuit(alpha,beta,gamma):
    # Call pl.RX, pl.RY, pl.RZ, and pl.CNOT to build and return this circuit
    pl.RX(alpha, wires='p')
    pl.RY(beta, wires='q')
    pl.RZ(gamma, wires='r')
    pl.CNOT(wires=['p','q'])
    pl.CNOT(wires=['q','r'])
    return pl.CNOT(wires=['r','p'])

pl.drawer.use_style("black_white")
pl.draw_mpl(circuit)(np.pi/4, np.pi/8, 0)
```

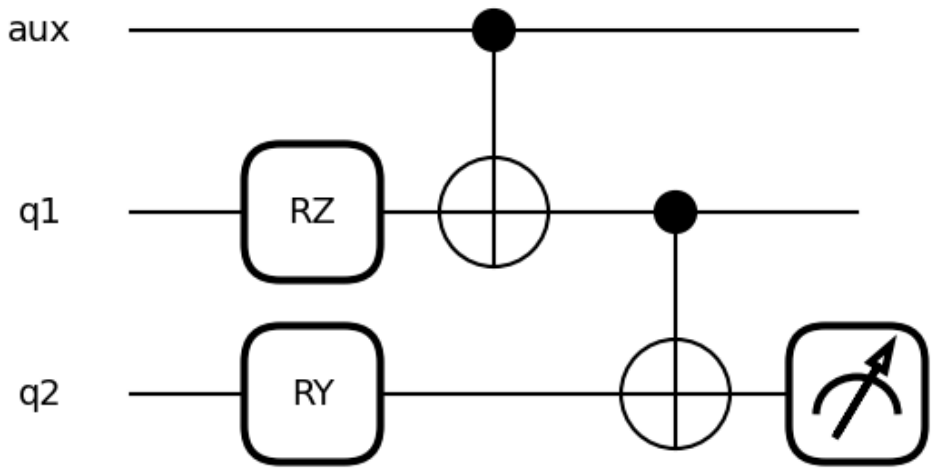
[3]: (<Figure size 700x400 with 1 Axes>, <Axes: >)



3 Exercise 3: Executing circuits

[4]:
```
device = pl.device('default.qubit', wires=['aux', 'q1', 'q2'],␣
 ↪shots=[1,10,100,1000])

def circuit(alpha, beta):
    # Build and return this circuit, using pl.expval and pl.PauliZ to measure␣
 ↪qubit 2 in the Z basis
    pl.RZ(alpha, wires='q1')
    pl.CNOT(wires=['aux','q1'])
    pl.RY(beta, wires='q2')
    pl.CNOT(wires=['q1','q2'])
    return pl.expval(pl.PauliZ('q2'))

parameters = [np.pi/4, np.pi/4]
pl.draw_mpl(circuit)(parameters[0], parameters[1])
qnode = pl.QNode(circuit,device)
qnode(parameters[0], parameters[1])
```

[4]:
```
(1.0, 0.6, 0.68, 0.714)
```



[5]:
```
@pl.qnode(device)
def circuit(alpha, beta):
    pl.RZ(alpha, wires='q1')
    pl.CNOT(wires=['aux','q1'])
    pl.RY(beta, wires='q2')
    pl.CNOT(wires=['q1','q2'])
    return pl.expval(pl.PauliZ('q2'))

parameters = [np.pi/4, np.pi/4]
circuit(parameters[0], parameters[1])
```

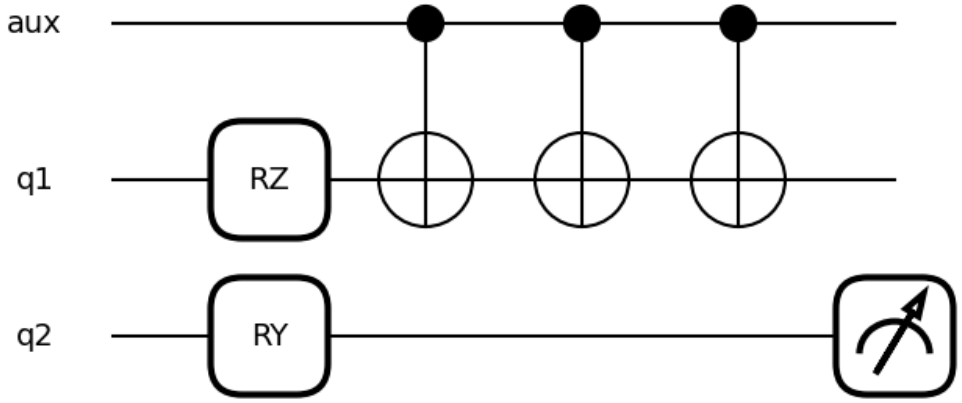
[5]: (1.0, 1.0, 0.68, 0.726)

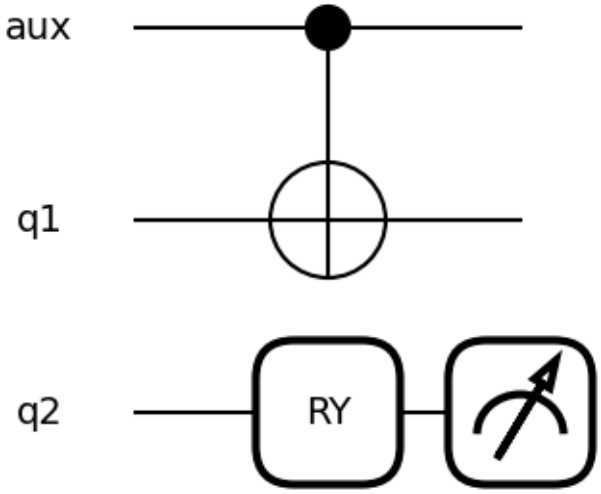
4 Exercise 4: Circuit depth

[6]:
```
device = pl.device('default.qubit', wires=['aux', 'q1', 'q2'])

@pl.qnode(device)
def circuit(x, y):
    # Build any circuit and inspect its depth
    pl.RZ(x, wires='q1')
    pl.CNOT(wires=['aux','q1'])
```

```
    pl.CNOT(wires=['aux','q1'])
    pl.CNOT(wires=['aux','q1'])
    pl.RY(y, wires='q2')
    return pl.expval(pl.PauliZ('q2'))
pl.draw_mpl(circuit)(0,0)
print(pl.specs(circuit)(0,0)['resources'])

@pl.compile
@pl.qnode(device)
def circuit(x, y):
    # Build any circuit and inspect its depth, now using @pl.compile decorator
    pl.RZ(x, wires='q1')
    pl.CNOT(wires=['aux','q1'])
    pl.CNOT(wires=['aux','q1'])
    pl.CNOT(wires=['aux','q1'])
    pl.RY(y, wires='q2')
    return pl.expval(pl.PauliZ('q2'))
pl.draw_mpl(circuit)(0,0)
print(pl.specs(circuit)(0,0)['resources'])
```

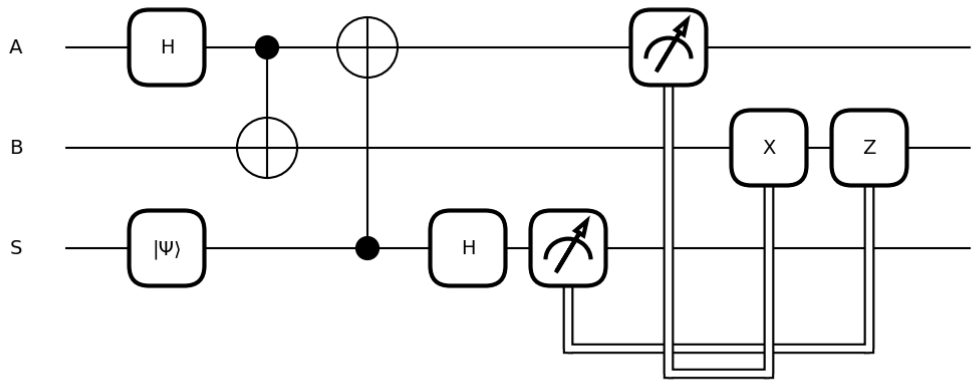
```
wires: 3
gates: 5
depth: 4
shots: Shots(total=None)
gate_types:
{'RZ': 1, 'CNOT': 3, 'RY': 1}
gate_sizes:
{1: 2, 2: 3}
wires: 3
gates: 2
depth: 1
shots: Shots(total=None)
gate_types:
{'CNOT': 1, 'RY': 1}
gate_sizes:
{2: 1, 1: 1}
```


aux
q1
RZ
q2
RY


aux
q1
q2
RY

5 Exercise 5: Teleportation

[7]:
```
def state_preparation(state):
    # use pl.StatePrep to prepare wire S in the given state
    pl.StatePrep(state, wires=["S"])

def entangle_qubits():
    # use pl.Hadamard and pl.CNOT to create a Bell pair
    pl.Hadamard(wires="A")
    pl.CNOT(wires=["A", "B"])

def basis_rotation():
    # use pl.Hadamard and pl.CNOT to rotate the basis
    pl.CNOT(wires=["S", "A"])
    pl.Hadamard(wires="S")

def measure_and_update():
    # use pl.measure for Alice's measurement and pl.PauliX, pl.PauliZ, and pl.
 ↪cond for Bob's correction
    m0 = pl.measure("S")
    m1 = pl.measure("A")
    pl.cond(m1, pl.PauliX)("B")
    pl.cond(m0, pl.PauliZ)("B")

def teleport(state):
    state_preparation(state)
    entangle_qubits()
    basis_rotation()
    measure_and_update()

state = np.array([1 / np.sqrt(2) + 0.3j, 0.4 - 0.5j])
_ = pl.draw_mpl(teleport)(state)
```



[ ]: