

THE UNIVERSITY of EDINBURGH
informatics

Introduction to Quantum Computing

Pennylane: Quantum Fourier Transform

Mina Doosti

Exercise 1: The QFT matrix

$$\frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

```
def matrix():
    N = 2 ** n_qubits
    m = np.empty([N,N]).astype(complex)
    # define the QFT matrix
    return m

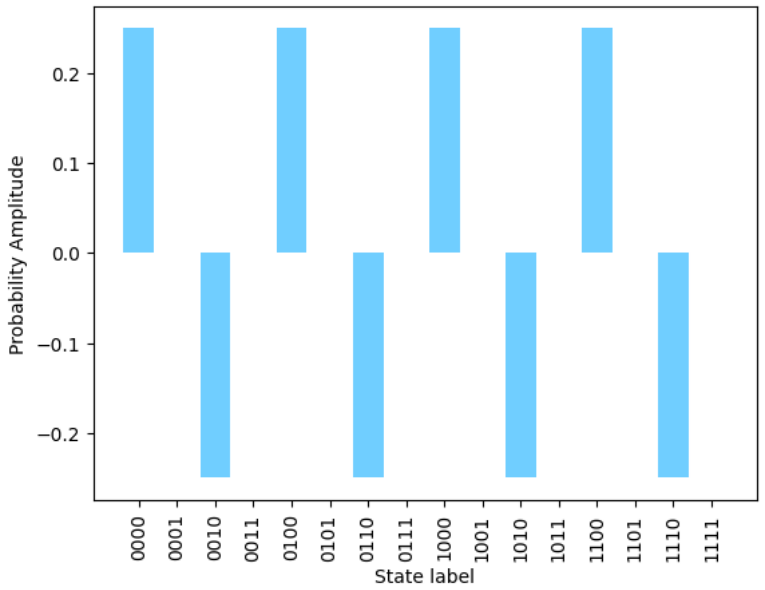
print(matrix())
```

Exercise 1: The QFT matrix

$$|x\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i x y / 2^n} |y\rangle$$

```
@pl.qnode(dev)
def circuit(k):
    bits = [int(x) for x in np.binary_repr(k, width=n_qubits)]
    print(bits)
    pl.BasisStatePreparation(bits, wires=wires)
    pl.QubitUnitary(matrix(), wires=wires)
    return pl.state()

results = pl.snapshots(circuit)(3)
y = np.real(results["execution_results"])
bit_strings = [f"{x:0{n_qubits}b}" for x in range(len(y))]
plt.bar(bit_strings, y, color = "#70CEFF")
plt.xticks(rotation="vertical")
plt.xlabel("State label")
plt.ylabel("Probability Amplitude")
plt.show()
```



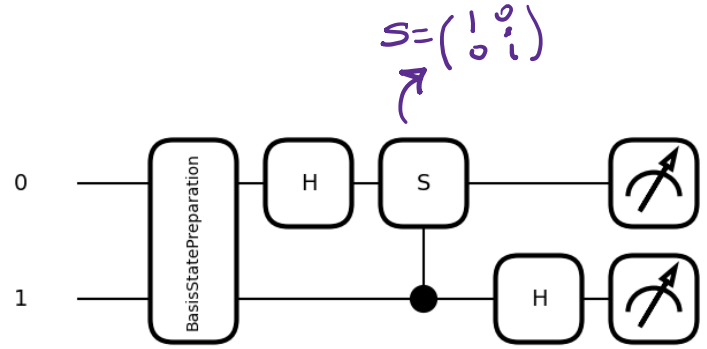
Exercise 2: The 2-qubit QFT circuit

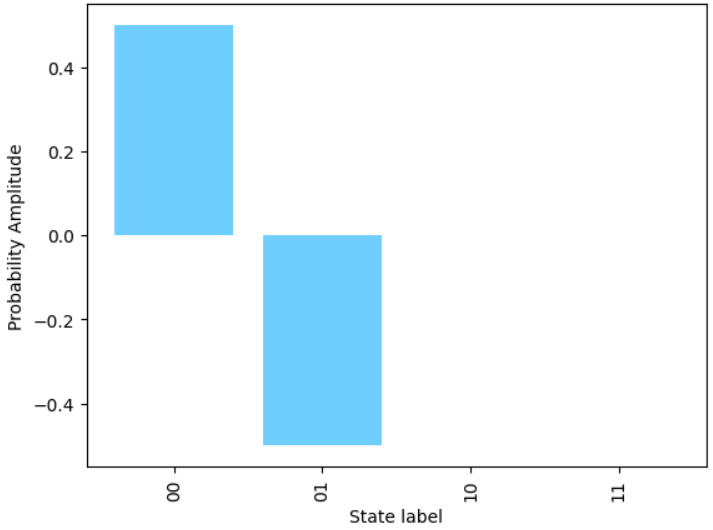
```
n_qubits = 2
dev = pl.device("default.qubit", wires=n_qubits)
wires = list(range(n_qubits))

@pl.qnode(dev)
def circuit(k):
    bits = [int(x) for x in np.binary_repr(k, width=n_qubits)]
    pl.BasisStatePreparation(bits, wires=wires)
    # Build the 2-qubit QFT circuit
    return pl.state()

pl.draw_mpl(circuit, style="black_white")(1);

# Plot the results, compare against matrix
```





Exercise 3: The recursive QFT circuit

$$|x_1, x_2, \ldots, x_{n-1}, x_n\rangle \rightarrow \frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i 0.x_n}|1\rangle)(|0\rangle + e^{2\pi i 0.x_{n-1}x_n}|1\rangle)\cdots(|0\rangle + e^{2\pi i 0.x_1 x_2 \ldots x_n}|1\rangle)$$

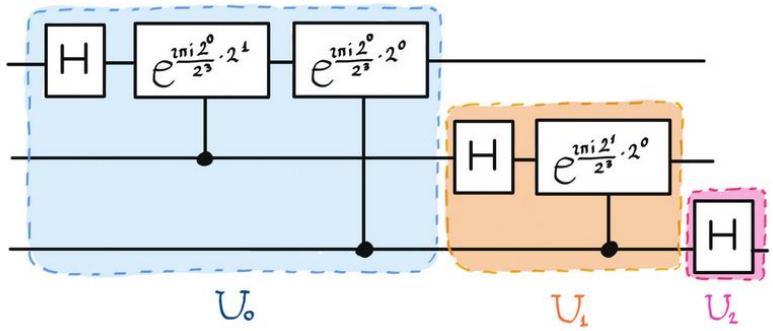
```
n_qubits = 4
dev = pl.device("default.qubit", wires=n_qubits)
wires = list(range(n_qubits))

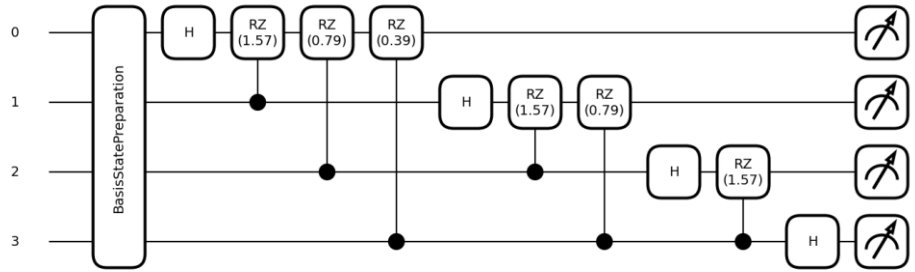
def subcircuit(wires):
    # build the subcircuit on the given wires

def QFT():
    # put all the subcircuits together

@pl.qnode(dev)
def circuit(k):
    bits = [int(x) for x in np.binary_repr(k, width=n_qubits)]
    pl.BasisStatePreparation(bits, wires=wires)
    QFT()
    return pl.state()

pl.draw_mpl(circuit, style="black_white", decimals=2)(3);
# plot the results
```





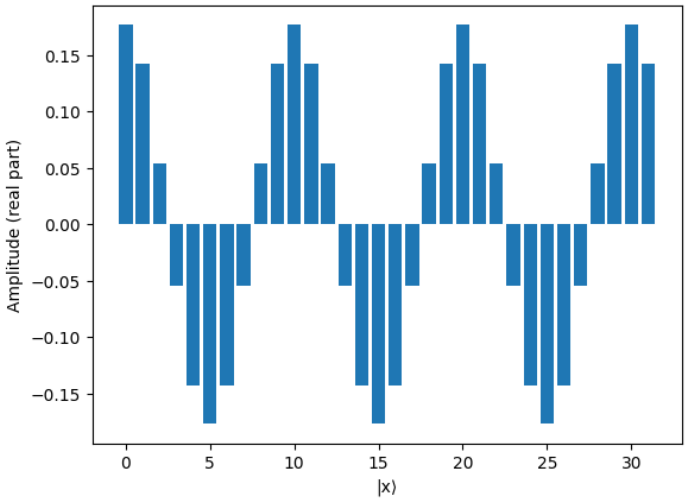
Exercise 4: Using the QFT for period-finding

```
n_qubits = 6
period = 10
dev = pl.device("default.qubit", wires=n_qubits)
wires = list(range(n_qubits))

def periodicstatecircuit():
    # set up a circuit that output a periodic state,
using pl.ControlledSequence and pl.PhaseShift

@pl.qnode(dev)
def periodicstate():
    periodicstatecircuit()
    return pl.state()

state = periodicstate().real[:2**(n_qubits-1)]
# plot the results
```



Exercise 4: What is the periodic phase state?

Six qubits: one auxiliary qubit (q0) and a five-qubit register encoding x = 0, …, 31.

TARGET STATE · period r = 10

$$|\psi_r\rangle = |0\rangle \otimes \frac{1}{\sqrt{32}} \sum_{x=0}^{31} e^{-2\pi i x/r} |x\rangle, \qquad r = 10$$

START WITH A UNIFORM SUPERPOSITION

$$|1\rangle|0\rangle^{\otimes 5} \longrightarrow |1\rangle \otimes \frac{1}{\sqrt{32}} \sum_{x=0}^{31} |x\rangle$$

The phase repeats every 10 labels, but every populated basis state has the same probability.

$$a_x = \frac{1}{\sqrt{32}} e^{-2\pi i x/10}, \qquad |a_x|^2 = \frac{1}{32}$$

Exercise 4: Controlled powers encode the phase

1. **Define a phase gate on the auxiliary qubit (prepared in |1⟩)**

$$U = \text{PhaseShift}(-2\pi/r) = \text{diag}(1, e^{-2\pi i/r})$$

2. **Use the binary digits as controls**

$$x = 16x_1 + 8x_2 + 4x_3 + 2x_4 + x_5, \quad x_j \in \{0, 1\}$$

$x_1 \rightarrow U^{16}$ $x_2 \rightarrow U^{8}$ $x_3 \rightarrow U^{4}$ $x_4 \rightarrow U^{2}$ $x_5 \rightarrow U$

ControlledSequence applies U^{16}, U^{8}, U^{4}, U^{2}, U (each only when its control bit is 1).

3. **Flip the ancilla back to |0⟩**

$$U^x|1\rangle = e^{-2\pi i x/r}|1\rangle$$

Result: the phase e^(−2πix/r) is attached to |x⟩; the probabilities remain uniform.

Exercise 4: Using the QFT for period-finding

```
@pl.qnode(dev)
def circuit():
 periodicstatecircuit()
 pl.QFT(wires=wires)
 return pl.probs(wires=wires)

state = circuit()[:2**(n_qubits-1)]
# plot the results

# Find the most probable answer and translate that
in the the most likely period
print(2**(n_qubits)/6)
```

