fourier-answers

October 22, 2025

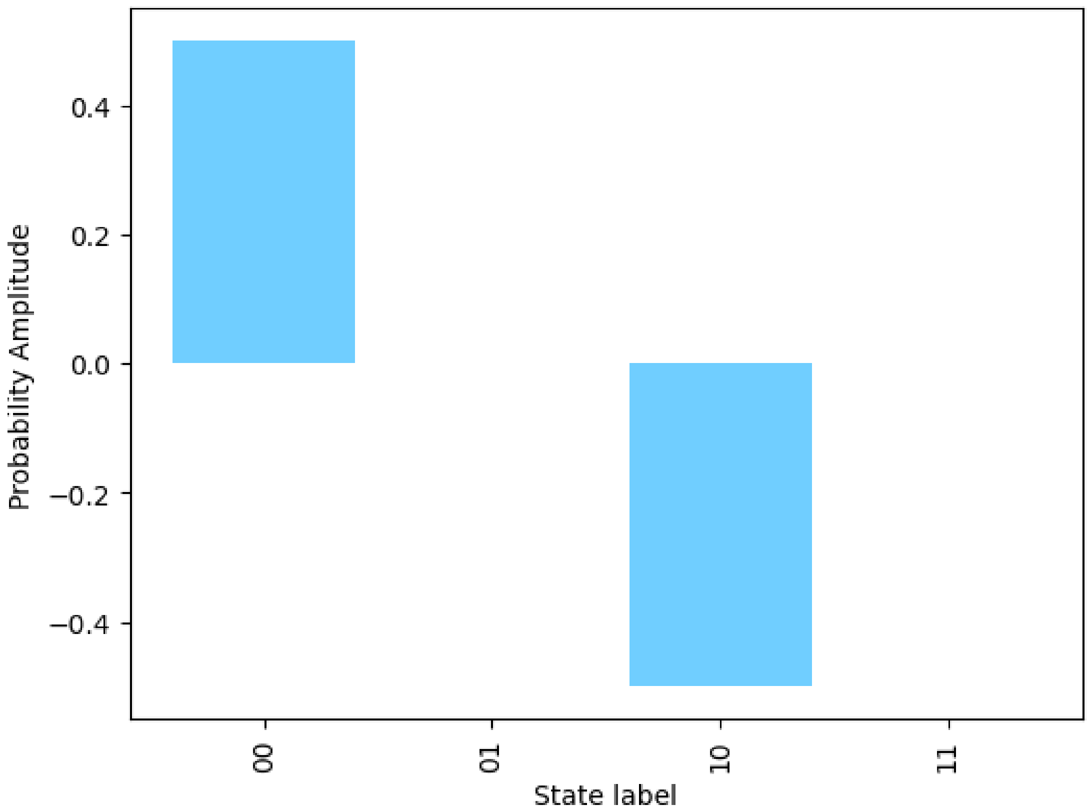
[22]:
```python
import pennylane as pl
from pennylane import numpy as np
import matplotlib.pyplot as plt
```

1 Exercise 1: The QFT matrix

[25]:
```python
n_qubits = 2
dev = pl.device("default.qubit", wires=n_qubits)
wires = list(range(n_qubits))

def omega(n):
    return np.exp(2 * np.pi * 1j/4 * n)

def matrix():
    N = 2 ** n_qubits
    m = np.empty([N,N]).astype(complex)
    for i in range(N):
        for j in range(N):
            m[i,j] =  omega(i*j) / np.sqrt(N)
    return m

print(matrix())

@pl.qnode(dev)
def circuit(k):
    bits = [int(x) for x in np.binary_repr(k, width=n_qubits)]
    pl.BasisState(bits, wires=wires)
    pl.QubitUnitary(matrix(), wires=wires)
    pl.Snapshot("snapshot")
    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))]
print(bit_strings)
plt.bar(bit_strings, y, color = "#70CEFF")
plt.xticks(rotation="vertical")
```

```
plt.xlabel("State label")
plt.ylabel("Probability Amplitude")
plt.show()
```

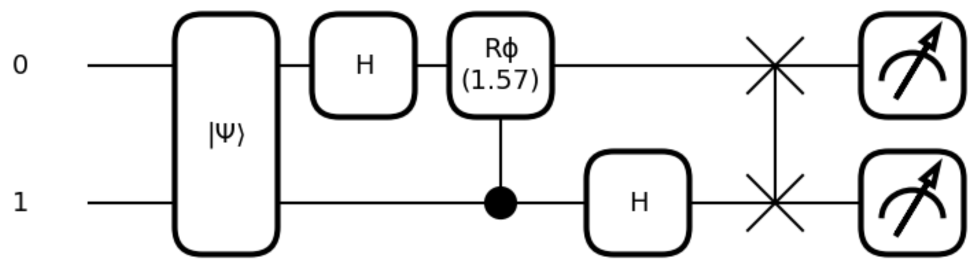
```
[[ 5.00000000e-01+0.0000000e+00j  5.00000000e-01+0.0000000e+00j
   5.00000000e-01+0.0000000e+00j  5.00000000e-01+0.0000000e+00j]
 [ 5.00000000e-01+0.0000000e+00j  3.06161700e-17+5.0000000e-01j
  -5.00000000e-01+6.1232340e-17j -9.18485099e-17-5.0000000e-01j]
 [ 5.00000000e-01+0.0000000e+00j -5.00000000e-01+6.1232340e-17j
   5.00000000e-01-1.2246468e-16j -5.00000000e-01+1.8369702e-16j]
 [ 5.00000000e-01+0.0000000e+00j -9.18485099e-17-5.0000000e-01j
  -5.00000000e-01+1.8369702e-16j  2.75545530e-16+5.0000000e-01j]]
['00', '01', '10', '11']
```



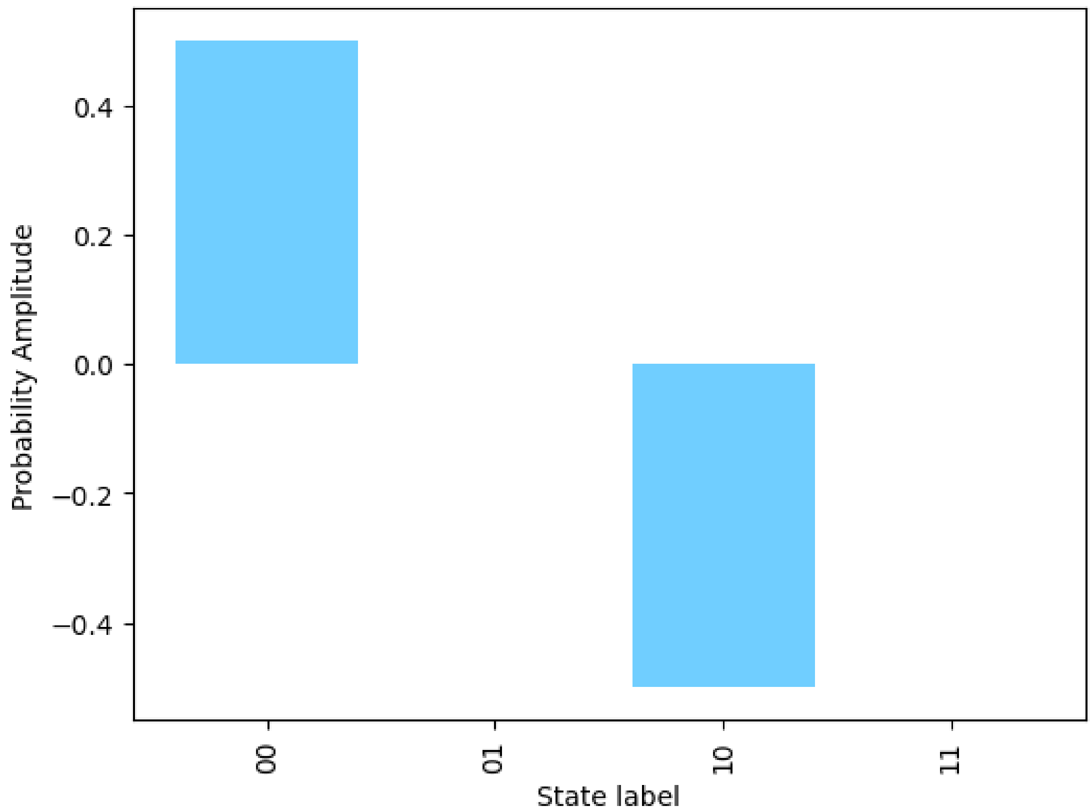
2 Exercise 2: The 2-qubit QFT circuit

[30]:
```
n_qubits = 2
dev = pl.device("default.qubit", wires=n_qubits)
wires = list(range(n_qubits))
```

```python
@pl.qnode(dev)
def circuit(k):
    bits = [int(x) for x in np.binary_repr(k, width=n_qubits)]
    bits = bits[::-1]
    pl.BasisState(bits, wires=wires)
    pl.Hadamard(wires = 0)
#    pl.ctrl(pl.S(wires=0), control = 1)
    def gate(arg):
        pl.PhaseShift(np.pi/2, wires=arg)
    pl.ctrl(gate, control=[1])(wires[0])
    pl.Hadamard(wires = 1)
    pl.SWAP(wires=[0,1])
    return pl.state()

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



[31]:
```python
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()
```

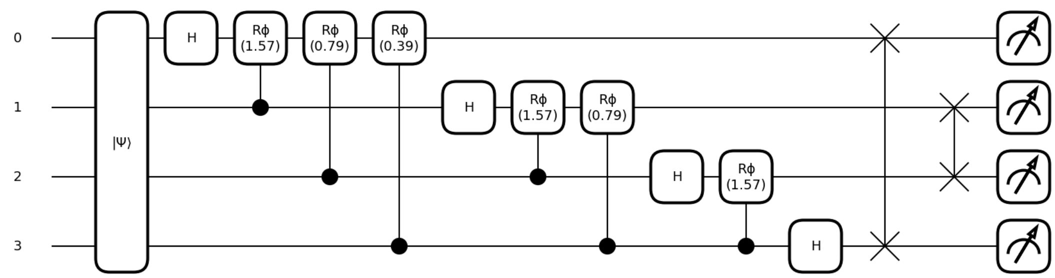


3 Exercise 3: The recursive QFT circuit

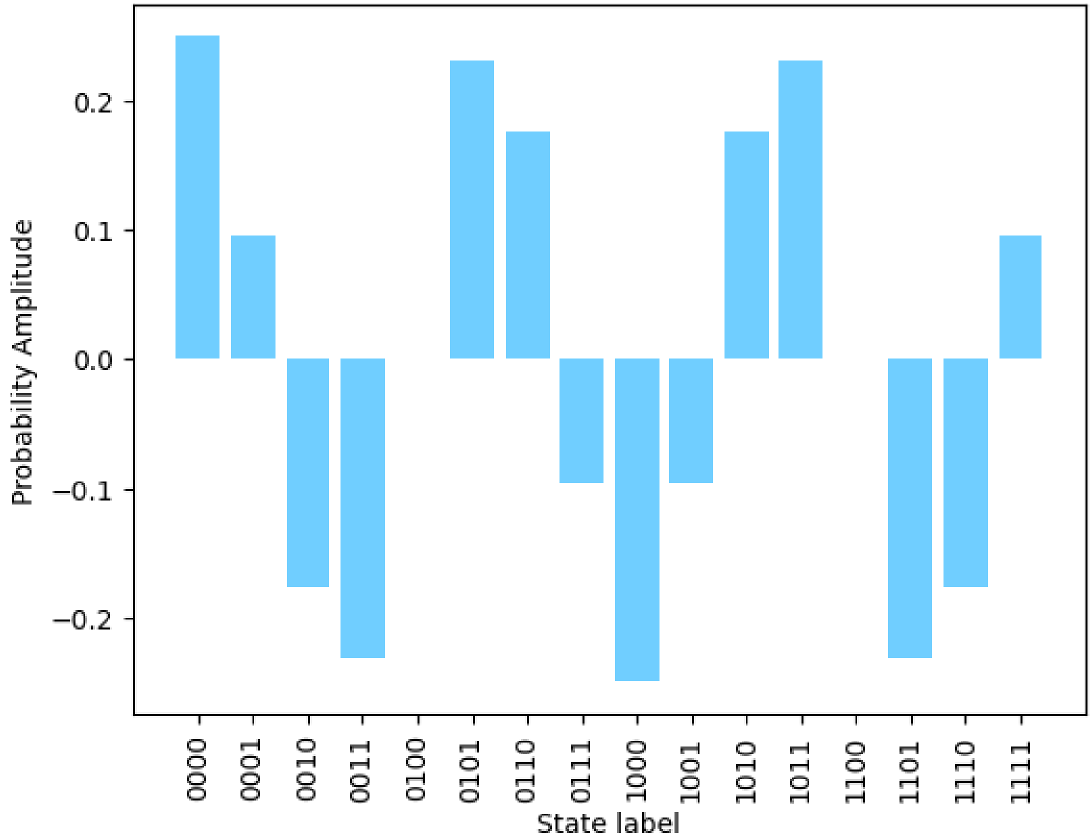
[32]:

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

def phi(n):
    return 2 * np.pi / (2**n)

def subcircuit(wires):
    pl.Hadamard(wires=wires[0])
    for wire in wires[1:]:
        def gate(arg):
#            pl.RZ(phi(wire-wires[0]+1), wires=arg)
            pl.PhaseShift(phi(wire-wires[0]+1), wires=arg)
        pl.ctrl(gate, control=[wire])(wires[0])

def QFT():
    for i in range(n_qubits):
        subcircuit(list(range(i,n_qubits)))
```

```
def swaps(wires):
    if (len(wires)>1):
        pl.SWAP(wires=[wires[0],wires[-1]])
        swaps(wires[1:-1])

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

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



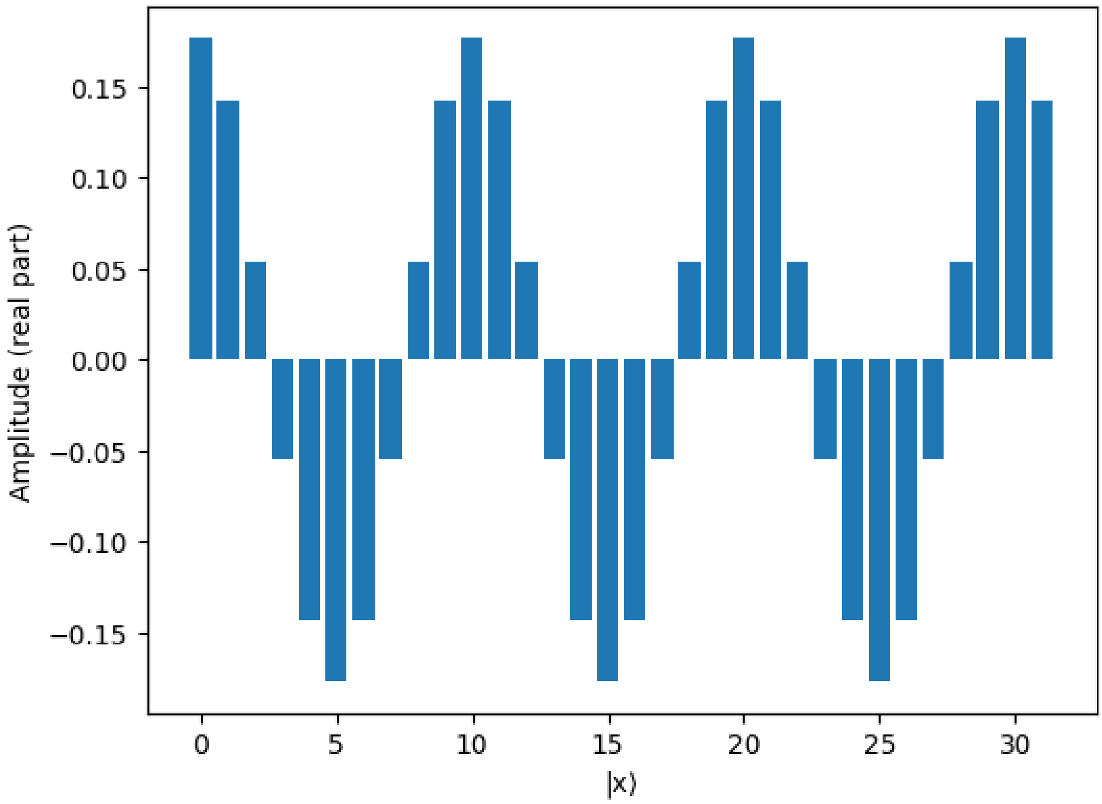
[33]:
```
results = pl.snapshots(circuit)(3)
y = np.real(results["execution_results"])
bit_strings = [f"{x:0{n_qubits}b}" for x in range(len(y))]
plt.bar(bit_strings, y, color = "#70CEFF")
plt.xticks(rotation="vertical")
plt.xlabel("State label")
plt.ylabel("Probability Amplitude")
plt.show()
```



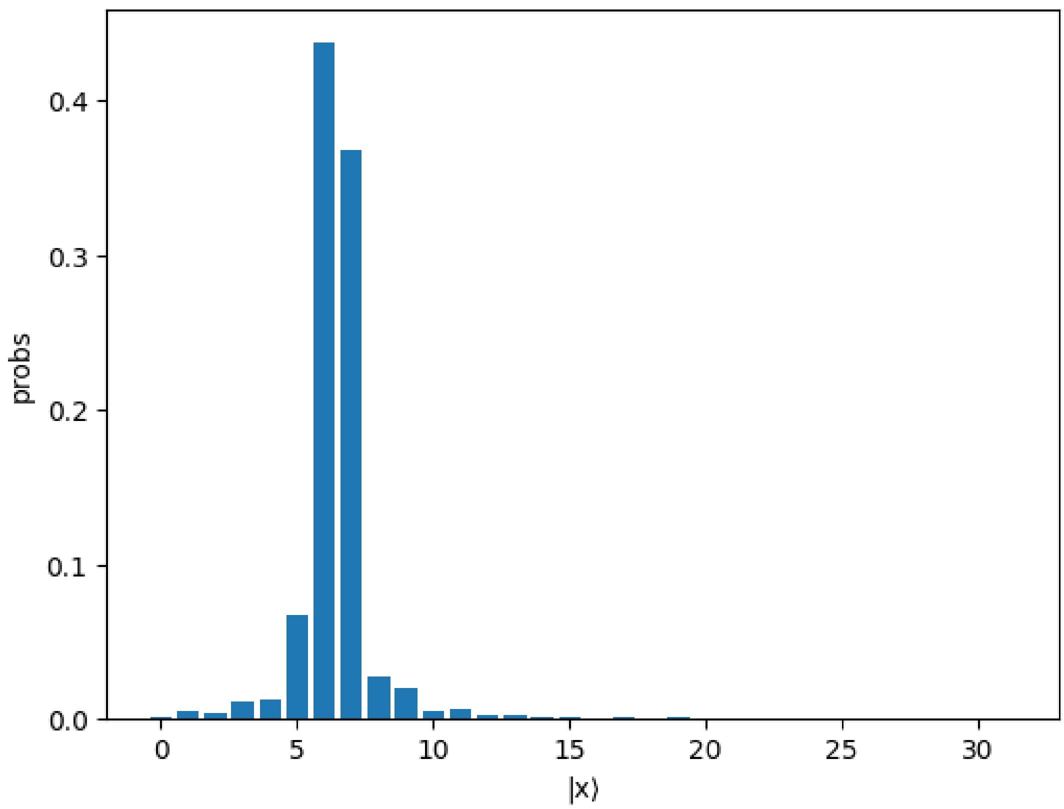
4 Exercise 4: Using the QFT for period-finding

[34]:
```
n_qubits = 6
period = 10
dev = pl.device("default.qubit", wires=n_qubits)
wires = list(range(n_qubits))

def periodicstatecircuit():
    pl.PauliX(wires=0)
    for wire in range(1, n_qubits):
        pl.Hadamard(wires=wire)
    pl.ControlledSequence(pl.PhaseShift(-2 * np.pi / period, wires=0), control=range(1, n_qubits))
    pl.PauliX(wires=0)

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

```
state = periodicstate().real[:2**(n_qubits-1)]
plt.bar(range(len(state)), state)
plt.xlabel("|x ")
plt.ylabel("Amplitude (real part)")
plt.show()
```



[35]:
```
@pl.qnode(dev)
def circuit():
  periodicstatecircuit()
  QFT()
  swaps(wires=wires)
#  pl.QFT(wires=wires)
  return pl.probs(wires=wires)

state = circuit()[:2**(n_qubits-1)]

plt.bar(range(len(state)), state)
plt.xlabel("|x ")
plt.ylabel("probs")
plt.show()
```



[36]:
```
print(2**(n_qubits)/6)
```

```
10.666666666666666
```