Lab6

December 17, 2024

1 Introduction to Quantum Information and Quantum ML

Instructor: Dr Sci. Eng. Przemysław Głowacki

Kacper Dobek 148247

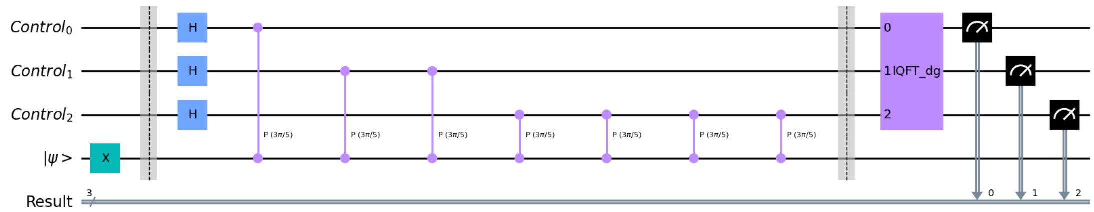
[7]:

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.circuit.library import QFT
from qiskit.quantum_info.operators import Operator
from qiskit import QuantumCircuit
from qiskit.visualization import plot_histogram
from qiskit.primitives import Sampler

from math import pi, cos, sin
import matplotlib.pyplot as plt
from typing import Tuple
import pandas as pd
```

[8]:

```
def create_circuit(theta: float, m: int) -> QuantumCircuit:

    control_register = QuantumRegister(m, name="Control")
    target_register = QuantumRegister(1, name="| >")
    output_register = ClassicalRegister(m, name="Result")
    qc = QuantumCircuit(control_register, target_register, output_register)

    # Prepare the eigenvector | >
    qc.x(target_register)
    qc.barrier()

    # Perform phase estimation
    for index, qubit in enumerate(control_register):
        qc.h(qubit)
        for _ in range(2**index):
            qc.cp(2 * pi * theta, qubit, target_register)
    qc.barrier()

    # Do inverse quantum Fourier transform
    qc.compose(QFT(m, inverse=True), inplace=True)
```

```
    # Measure everything
    qc.measure(range(m), range(m))

    return qc


qc = create_circuit(theta=0.3, m=3)
display(qc.draw("mpl"))
```



[9]:
```
def estimate_phase(
    theta: float, m: int, sampler: Sampler, plot: bool = False
) -> Tuple[float, int]:
    qc = create_circuit(theta, m)
    result = sampler.run(qc).result().quasi_dists[0]
    if plot:
        display(plot_histogram(result))
    most_probable = max(result, key=result.get)
    theta = most_probable / 2**m

    return theta, most_probable


estimate_phase(0.3, 8, sampler=Sampler(), plot=False)
```

[9]: (0.30078125, 77)

1.0.1 Phase estimation from n=1 to n=10

[10]:
```
n_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
m = max(n_values)
theta = 0.33

sampler = Sampler()

theta_results = []
d_results = []
```

```python
for n in n_values:
    theta_est, d = estimate_phase(theta, n, sampler=sampler)
    theta_results.append(theta_est)
    d_results.append(d)
```

1.0.2 Table

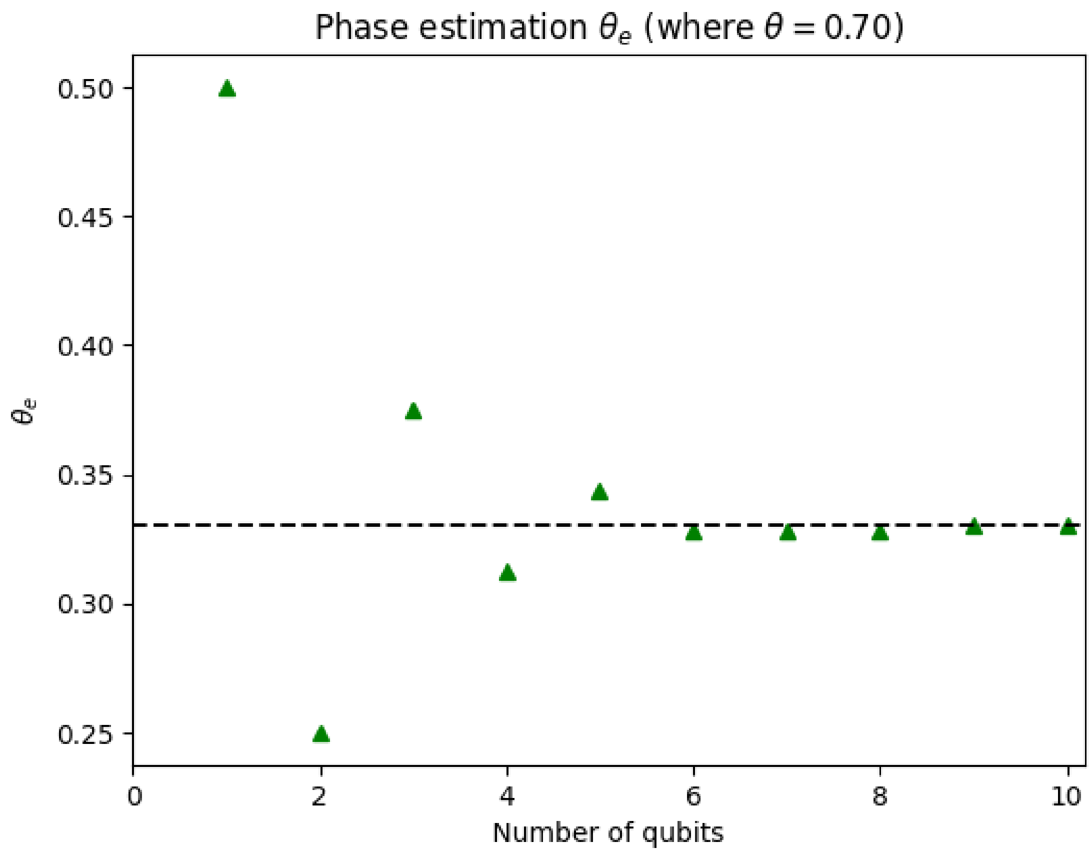
[11]:
```python
df = pd.DataFrame(
    {
        "n": n_values,
        "d": d_results,
        "phase estimation": theta_results,
    }
)

df.set_index("n", inplace=True)
pd.options.display.float_format = "{:.4f}".format

display(df)
```

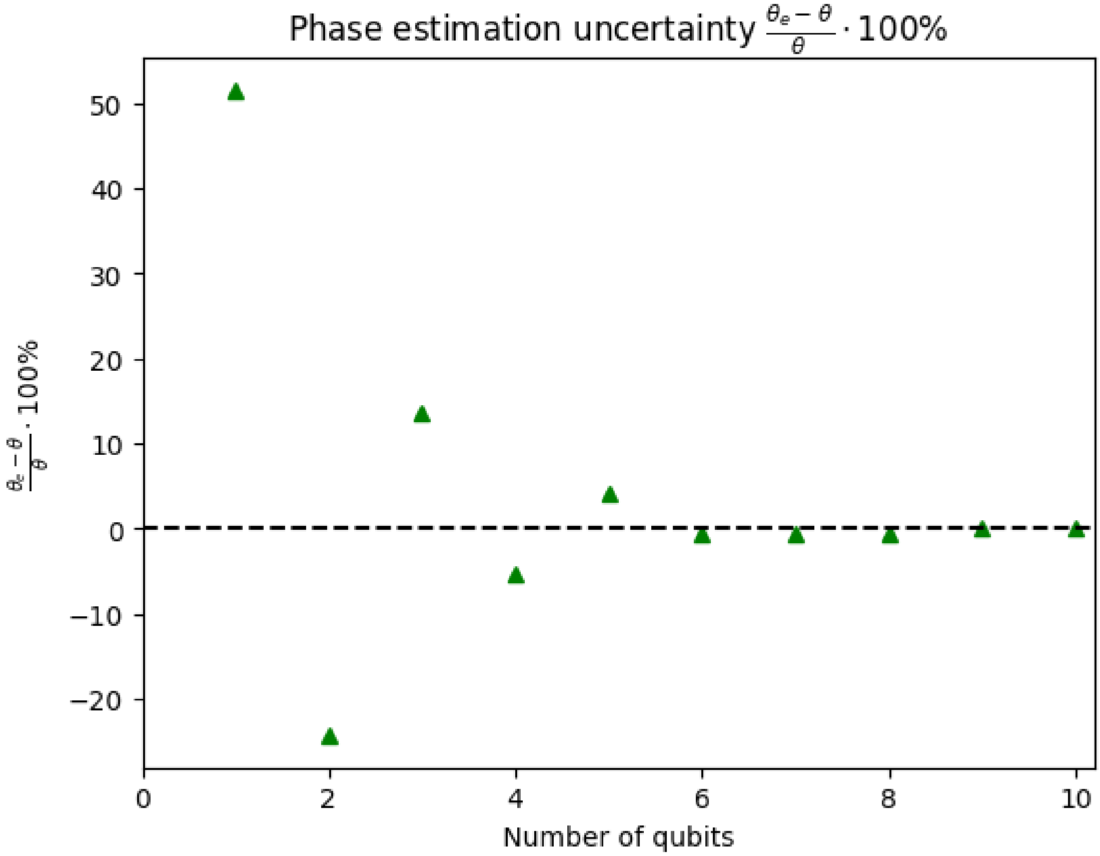
```
      d  phase estimation
n
1     1            0.5000
2     1            0.2500
3     3            0.3750
4     5            0.3125
5    11            0.3438
6    21            0.3281
7    42            0.3281
8    84            0.3281
9   169            0.3301
10  338            0.3301
```

1.0.3 Uncertainty plots

[12]:
```python
percent_results = []
for i in range(len(theta_results)):
    uncertainty = ((theta_results[i] - theta) / theta) * 100
    percent_results.append(uncertainty)
plt.title("Phase estimation $\\theta_e$ (where $\\theta=0.70$)")
plt.xlabel(r" Number of qubits ")
plt.ylabel(r"$\theta_e$")
plt.xlim([0, m + 0.2])
plt.plot(n_values, theta_results, "g^")
plt.hlines(y=theta, xmin=0.0, xmax=m + 0.2, colors="k", linestyles="dashed")
plt.show()
```

```
plt.title(
    r"Phase estimation uncertainty $\frac{\theta_e - \theta}{\theta} \cdot␣
 ↪100\%$"
)
plt.xlabel(r" Number of qubits ")
plt.ylabel(r"$\frac{\theta_e - \theta}{\theta} \cdot 100\%$")
plt.xlim([0, m + 0.2])
plt.plot(n_values, percent_results, "g^")
plt.hlines(y=0.0, xmin=0.0, xmax=m + 0.2, colors="k", linestyles="dashed")
plt.show()
```




Phase estimation uncertainty $\frac{\theta_e - \theta}{\theta} \cdot 100\%$
50
40
30
20
10
0
−10
−20
$\frac{\theta_e - \theta}{\theta} \cdot 100\%$
0
2
4
6
8
10
Number of qubits