Michał Kałmucki 151944

Lab 6: Phase Estimation

Objective: Implement the quantum phase estimation algorithm using Qiskit and analyze how the estimated phase θ_e converges to the true phase θ as the number of control qubits increases.

Phase values for groups:

- Group 11.45 → θ = 0.66

Phase Estimation Circuit and plot code

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, transpile
from qiskit.circuit.library import QFTGate
from qiskit_aer import AerSimulator
from qiskit_ibm_runtime import SamplerV2 as Sampler
import matplotlib.pyplot as plt
import numpy as np

# ----- Function: Phase Estimation Circuit -----
def phase_estimation_circuit(n_qubits, theta):
    """
    Build a phase estimation circuit for U = Rz(2*pi*theta) with eigenstate |1>
    n_qubits: number of control qubits (top register)
    theta: phase angle (0 <= theta <= 1)
    """
    # Registers
    control = QuantumRegister(n_qubits, name="Control")
    target = QuantumRegister(1, name="|psi>")
    classical = ClassicalRegister(n_qubits, name="Result")
    qc = QuantumCircuit(control, target, classical)

    # Prepare eigenstate |psi> = |1>
    qc.x(target)
    qc.barrier()

    # Apply H gates and controlled-U^(2^k) rotations
    for k, qubit in enumerate(control):
        qc.h(qubit)
        for _ in range(2**k):
            qc.cp(2 * np.pi * theta, qubit, target)
    qc.barrier()

    # Apply inverse QFT on control register
    qc.append(QFTGate(n_qubits).inverse(), control)

    # Measure
    qc.measure(control, classical)
```

```
    return qc

# ----- Parameters -----
theta_list = [0.66]
n_max = 10                   # Number of control qubits
backend = AerSimulator()
sampler = Sampler(mode=backend)

# ----- Run Phase Estimation and Collect Results -----
all_results = {}  # store results for both theta values

for theta in theta_list:
    results = []
    for n in range(1, n_max+1):
        qc = phase_estimation_circuit(n, theta)
        qc_t = transpile(qc, backend)
        job = sampler.run([qc_t])
        result = job.result()
        counts = result[0].data.Result.get_counts()

        # Most probable measured state
        dh = int(max(counts, key=counts.get), 2)
        theta_e = dh / 2**n

        results.append((n, dh, theta_e))
    all_results[theta] = results

# ----- Print Tables -----
for theta in theta_list:
    print(f"\nResults for theta = {theta}:")
    print("n_qubits  dh  theta_e")
    for row in all_results[theta]:
        print(f"{row[0]:>2}        {row[1]:>3}  {row[2]:.6f}")

# ----- Plot Percentage Error -----
plt.figure(figsize=(8,5))
for theta in theta_list:
    theta_e_list = [row[2] for row in all_results[theta]]
    perc_error = [100*(te - theta)/theta for te in theta_e_list]
    plt.plot(range(1, n_max+1), perc_error, marker='o', label=f"theta={theta}")
plt.xlabel("Number of control qubits (n)")
plt.ylabel("Relative error [%]")
plt.title("Phase estimation relative error")
plt.xticks(range(1, n_max+1))
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig("PhaseEstimation_Error.png", dpi=300)
plt.show()
```

Results for θ = 0.66

```
n_qubits  dh  theta_e
 1         1  0.500000
 2         3  0.750000
 3         5  0.625000
 4        11  0.687500
 5        21  0.656250
 6        42  0.656250
 7        84  0.656250
 8       169  0.660156
 9       338  0.660156
10       676  0.660156
```

Analysis

- For small numbers of qubits (n=1,2,3), the phase estimation has noticeable error due to limited resolution.
- As the number of qubits increases, θ_e converges toward the true phase θ = 0.66.
- After n=8 qubits, the estimated phase stabilizes with minimal relative error (~0.66).

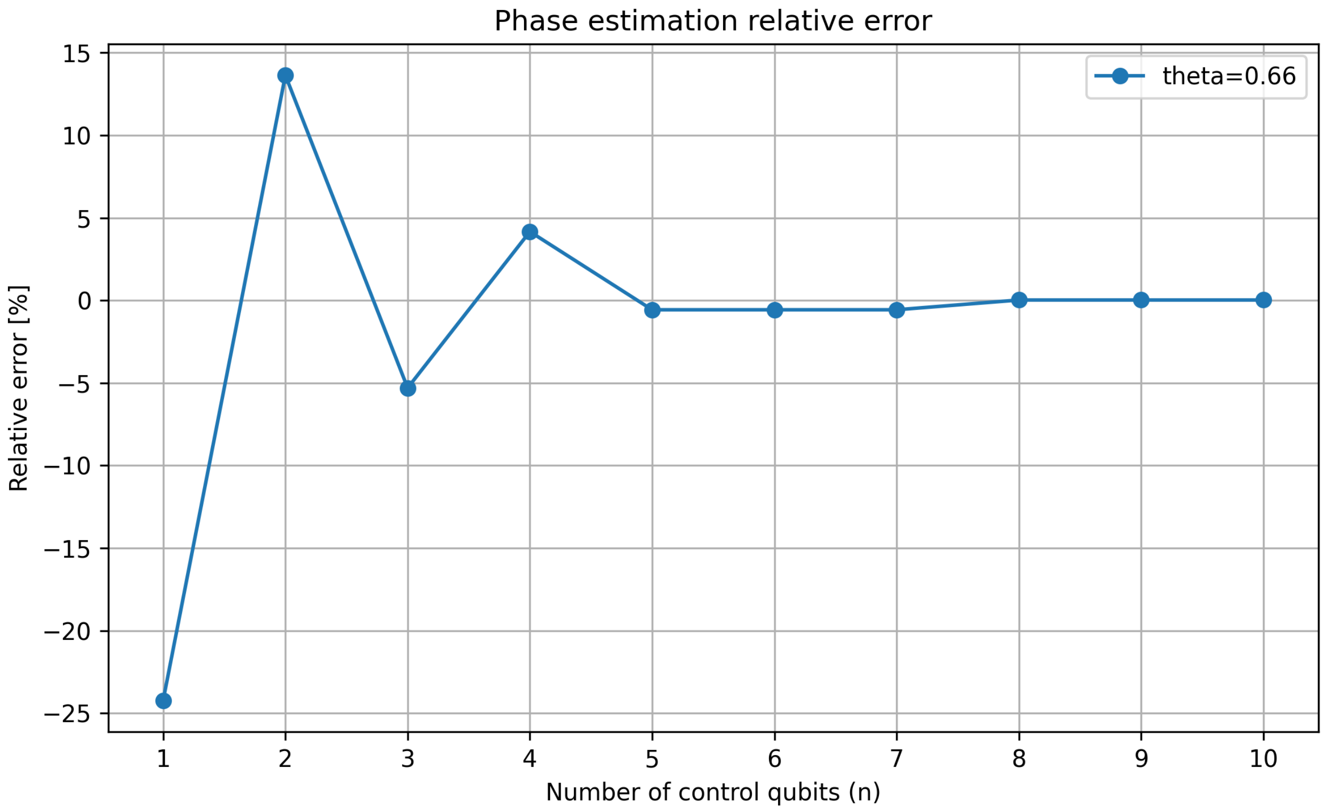
Relative Error Calculation

The relative error for each n is calculated as:

```
error [%] = 100 * (theta_e - theta) / theta
```

- This shows how the accuracy improves with more control qubits.
- Using 10 qubits, the relative error is sufficiently small for practical purposes.

Phase Estimation Graphs


Phase estimation relative error
theta=0.66
Relative error [%]
Number of control qubits (n)
15
10
5
0
−5
−10
−15
−20
−25
1
2
3
4
5
6
7
8
9
10