CSBB 311 : QUANTUM COMPUTING

LAB ASSIGNMENT 4 : Accuracy of Quantum Phase Estimation

Submitted By:

Name: KARTIK MITTAL

Roll No: 221210056

Branch: CSE

Semester: 5th Sem

Group: 2

Submitted To: Dr. VS Pandey

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2024

Theory -

1. Introduction to Iterative Quantum Phase Estimation

- Iterative Quantum Phase Estimation (IQPE) is a variant of the Quantum Phase Estimation (QPE) algorithm, designed to determine the phase (θ) of an eigenvalue corresponding to an eigenstate of a given unitary operator with reduced qubit requirements.

2. Iterative Phase Estimation and Precision

- In IQPE, precision is achieved by sequentially estimating the binary digits of the phase θ, from the most significant to the least significant bit. The algorithm accomplishes this through controlled unitary operations and adaptive rotations on a single qubit.

3. Step-by-Step Process in Iterative Quantum Phase Estimation

- **Initialize the Circuit:**Prepare the quantum circuit with a single qubit in the initial state. This qubit is the primary computational resource in IQPE.
- **Apply a Hadamard Gate:** The Hadamard gate creates a superposition, preparing the qubit for iterative phase measurement.
- **Controlled Unitary Operations**: For each bit of precision, apply a controlled unitary operation corresponding to the eigenvalue's unitary matrix raised to powers of two (U, U^2, U^4, etc.)
- **Adaptive Rotation and Measurement**: After each controlled operation, rotate the qubit by an angle proportional to the current phase estimate. Measure the qubit, and based on the result, update the phase estimate for the next iteration.
- **Iterate and Refine the Phase Estimate**: Repeat the steps, updating the phase estimate bit-by-bit until the desired precision is reached.

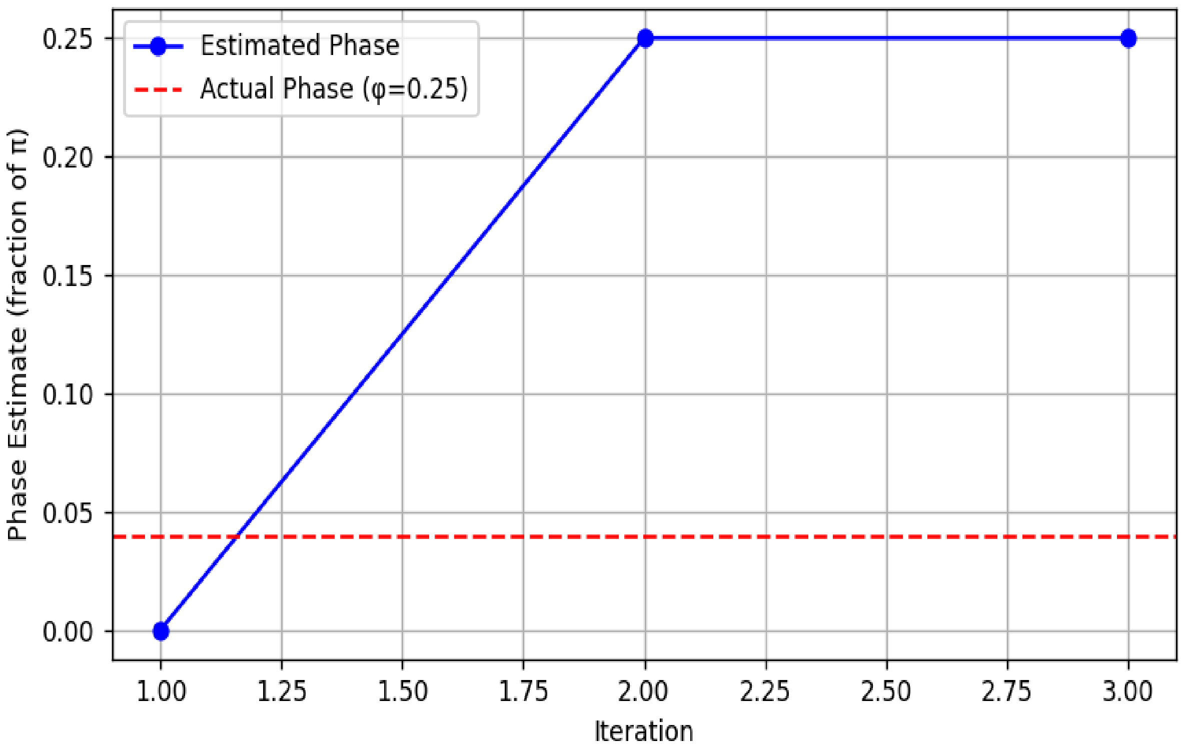
Code -

```python
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
import numpy as np
import matplotlib.pyplot as plt

# Define the unitary operator (e.g., Pauli-Z gate with phase)
phi = 0.25  # Known phase to estimate (should be between 0 and 1)
unitary = QuantumCircuit(1)
unitary.p(2 * np.pi * phi, 0)  # Phase gate with known phase

# Iterative Quantum Phase Estimation
def iqpe(unitary, num_iterations):
    phase_estimate = 0
    estimated_phases = []  # List to track phase estimates across iterations
    actual_phase_fraction = phi / (2 * np.pi)  # Actual phase as a fraction of π

    for k in range(num_iterations):
        qc = QuantumCircuit(1, 1)

        # Step 1: Apply Hadamard to prepare superposition
        qc.h(0)

        # Step 2: Apply controlled unitary (U^(2^k))
        power_unitary = unitary.power(2 ** k)
        qc.append(power_unitary.to_instruction(), [0])

        # Step 3: Rotate ancilla qubit by phase angle and measure
        qc.p(-2 * np.pi * phase_estimate * (2 ** k), 0)
        qc.h(0)
        qc.measure(0, 0)

        # Transpile the circuit before running (optimizing for the simulator backend)
        qc_transpiled = transpile(qc, backend=AerSimulator())
```

```python
        # Run the transpiled circuit on the simulator
        simulator = AerSimulator()
        result = simulator.run(qc_transpiled, shots=1024).result()  # Manual run without execute
        counts = result.get_counts()

        # Update phase estimate based on measurement result
        if '1' in counts and counts['1'] > counts.get('0', 0):
            phase_estimate += 1 / (2 ** (k + 1))

        # Append the estimated phase for this iteration
        estimated_phases.append(phase_estimate)

    return estimated_phases, actual_phase_fraction

# Run IQPE with 3 iterations to estimate phase
estimated_phases, actual_phase = iqpe(unitary, 3)

# Plotting the results
iterations = range(1, len(estimated_phases) + 1)
plt.figure(figsize=(8, 5))

# Plot the estimated phases
plt.plot(iterations, estimated_phases, marker='o', label="Estimated Phase", color='b')

# Plot the actual phase (constant line)
plt.axhline(y=actual_phase, color='r', linestyle='--', label="Actual Phase (ϕ=0.25)")

# Formatting the plot
plt.xlabel("Iteration")
plt.ylabel("Phase Estimate (fraction of π)")
plt.title("Convergence of Phase Estimate in IQPE")
plt.legend()
plt.grid(True)
plt.show() # Show the plot

# Print the final results
print(f"Estimated Phase after {len(estimated_phases)} iterations: {estimated_phases[-1]}")
print(f"Actual Phase (as fraction of π): {actual_phase}")
```

Output -

```
Estimated Phase after 3 iterations: 0.25
Actual Phase (as fraction of π): 0.039788735772973836
```



Convergence of Phase Estimate in IQPE

Conclusion -

- **Precision-Iteration Trade-off**: In Iterative Quantum Phase Estimation (IQPE), the accuracy of phase estimation improves with the number of iterations, as each iteration refines the phase estimate with greater precision.
- **Applications and Practical Benefits**: The iterative nature of IQPE makes it well-suited for practical quantum applications, including cryptographic algorithms and quantum simulations, where precise phase estimation enhances the accuracy and performance of results.