CSBB 311 : QUANTUM COMPUTING

LAB ASSIGNMENT 3 : 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 Quantum Phase Estimation

- Quantum Phase Estimation (QPE) is a fundamental quantum algorithm used to estimate the phase (θ) of an eigenvalue associated with an eigenstate of a unitary operator.
- It is a critical algorithm for many applications in quantum computing, including factoring, Shor's algorithm, and quantum simulations

2. Phase Estimation and Accuracy

- The accuracy of QPE depends on the number of qubits used for estimation. More qubits allow for a finer resolution of the estimated phase, leading to higher precision.
- The algorithm determines θ by estimating the binary fraction of the phase, where the precision improves exponentially with the number of qubits.

3. Basic Workflow for Quantum Phase Esimation

- **Create the quantum circuit:** Initialize qubits for phase estimation and one target qubit for the unitary operation.
- **Apply Hadamard gates:** Prepare the qubits in superposition using Hadamard gates.
- **Controlled Unitary operations:** Apply the controlled-U operations based on the unitary operator associated with the phase θ.
- **Simulate and analyze:** Use simulators to execute the circuit and retrieve the measurement results.

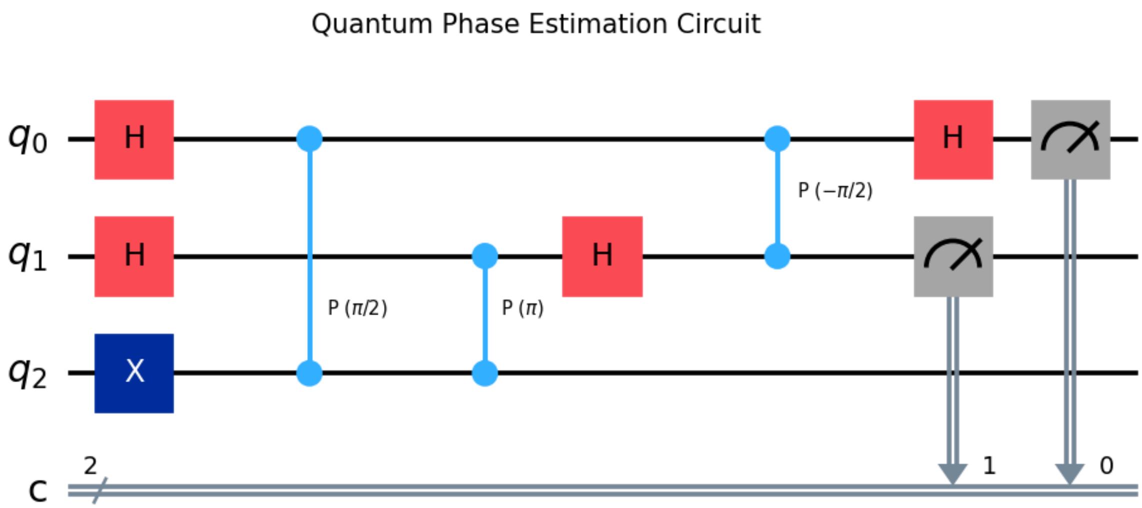
4. Importance of Accuracy in Quantum Phase Estimation

- The accuracy of QPE plays a vital role in determining the effectiveness of algorithms relying on phase estimation, such as quantum simulations and cryptographic algorithms.
- Higher accuracy results in better approximations of eigenvalues, leading to improved outcomes in quantum computational tasks.

Code -

```
# Import required libraries
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
from numpy import pi
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# Define the unitary operator (U)
theta = 1/4  # The phase theta we want to estimate (e.g., 1/4)

# Create quantum circuit for QPE with 2 qubits for estimation and 1 target qubit
qpe_circuit = QuantumCircuit(3, 2)

# Prepare the eigenvector |psi> (the last qubit)
qpe_circuit.h([0, 1])  # Apply Hadamard gates to the first 2 qubits
qpe_circuit.x(2)       # Set the target qubit to |1>

# Apply controlled-U gates
qpe_circuit.cp(2 * pi * theta, 0, 2)  # Controlled-U with theta applied to qubit 0
qpe_circuit.cp(4 * pi * theta, 1, 2)  # Controlled-U^2 with theta applied to qubit 1

# Inverse Quantum Fourier Transform (simplified for 2 qubits)
qpe_circuit.h(1)
qpe_circuit.cp(-pi/2, 0, 1)  # Controlled Phase shift between qubit 0 and qubit 1
qpe_circuit.h(0)

# Measure the first two qubits
qpe_circuit.measure([0, 1], [0, 1])

# Transpile the circuit for the 'qasm_simulator' backend
simulator = Aer.get_backend('qasm_simulator')
transpiled_circuit = transpile(qpe_circuit, simulator)

# Plot the quantum circuit using matplotlib
fig, ax = plt.subplots(figsize=(10, 5))
qpe_circuit.draw(output='mpl', ax=ax)  # Draw the circuit on the specified axes
plt.title('Quantum Phase Estimation Circuit')
plt.show()
```

Output -

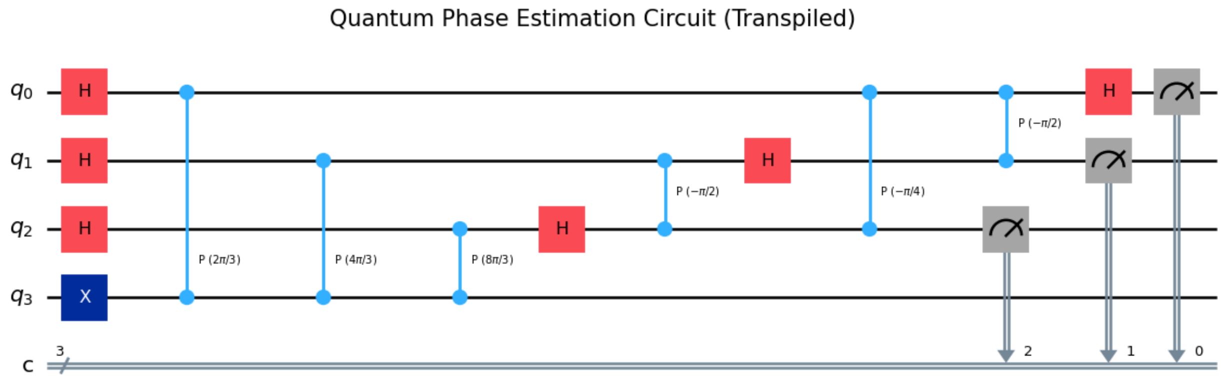


Quantum Phase Estimation Circuit
q_0
q_1
q_2
c
H
H
X
P (π/2)
P (π)
H
P ($-\pi$/2)
H
2
1
0

Code -

```
# Import required libraries
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
from numpy import pi
import matplotlib.pyplot as plt

# Define the phase theta we want to estimate
theta = 1/3  # Let's assume we want to estimate theta = 1/3

# Create quantum circuit for QPE with 3 qubits for estimation and 1 target qubit
qpe_circuit = QuantumCircuit(4, 3)

# Prepare the eigenvector |psi> (the last qubit)
qpe_circuit.h([0, 1, 2])  # Apply Hadamard gates to the first 3 qubits
qpe_circuit.x(3)          # Set the target qubit to |1>

# Apply controlled-U gates
qpe_circuit.cp(2 * pi * theta, 0, 3)  # Controlled-U with theta applied to qubit 0
qpe_circuit.cp(4 * pi * theta, 1, 3)  # Controlled-U^2 with theta applied to qubit 1
qpe_circuit.cp(8 * pi * theta, 2, 3)  # Controlled-U^4 with theta applied to qubit 2

# Inverse Quantum Fourier Transform (simplified for 3 qubits)
qpe_circuit.h(2)
qpe_circuit.cp(-pi/2, 1, 2)  # Controlled Phase shift between qubit 1 and qubit 2
qpe_circuit.h(1)
qpe_circuit.cp(-pi/4, 0, 2)  # Controlled Phase shift between qubit 0 and qubit 2
qpe_circuit.cp(-pi/2, 0, 1)  # Controlled Phase shift between qubit 0 and qubit 1
qpe_circuit.h(0)

# Measure the first three qubits
qpe_circuit.measure([0, 1, 2], [0, 1, 2])

# Transpile the circuit for the 'qasm_simulator' backend
simulator = Aer.get_backend('qasm_simulator')
transpiled_circuit = transpile(qpe_circuit, simulator)

# Plot the transpiled quantum circuit using matplotlib
fig, ax = plt.subplots(figsize=(12, 6))
qpe_circuit.draw(output='mpl', ax=ax)
plt.title('Quantum Phase Estimation Circuit (Transpiled)')
plt.show()
```

Output -



Conclusion -

- **Precision-Quibit Trade-off:** The accuracy of Quantum Phase Estimation improves with the number of qubits, as more qubits allow for finer phase resolution..
- **Algorithmic Impact:** The accuracy of QPE is crucial for algorithms like Shor's and quantum simulations, as more precise phase estimations lead to better performance and more accurate results in these applications.