Laboratory 2 – Quantum Oracles and Grover Diffusion Operator

Introduction

In this laboratory, we construct and analyze quantum oracle operators and the Grover diffusion operator using matrix representations and Qiskit circuits. XOR and phase oracles are implemented, and the Grover diffusion operator (W) is built for (n=3) qubits. Unitarity of all operators is verified, and circuits are visualized using Qiskit.

Objective

- Construct XOR and phase oracle matrices.
- Build the Grover diffusion operator.
- Verify unitarity of all operators.
- Implement Qiskit circuits representing the oracle and diffusion operator.
- Compare analytical matrices with simulated unitary results.

---

1. XOR Oracle (n = 2, a = 01)

The XOR oracle flips the auxiliary qubit if the input register matches the marked value.

```
Uf_xor = xor_oracle_matrix(2, "01")
print(Uf_xor)
```

Result:

```
[[1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
```

---

2. Phase Oracle (n = 3, a = 100)

The phase oracle applies a phase shift of −1 to the marked state (|100\rangle).

```
Uf_phase = phase_oracle_matrix(3, "100")
print(Uf_phase)
```

Result:

```
[[ 1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.]]
```

---

3. |0><0| PROJECTOR (n=3)

```
print(zero_proj)
```

Result:

```
[[1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]
```

---

4. OPERATOR 2|0><0| - I

```
print(D0)
```

Result:

```
[[ 1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.  0.  0.  0.]
```

```
 [ 0.  0.  0.  0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. -1.]]
```

---

5. CONSTRUCTION OF H^n AND X^n

```
print(Hn)
print(Xn)
```

Result:

```
H^n =
[[ 0.35355339  0.35355339  0.35355339  0.35355339  0.35355339  0.35355339
   0.35355339  0.35355339]
 [ 0.35355339 -0.35355339  0.35355339 -0.35355339  0.35355339 -0.35355339
   0.35355339 -0.35355339]
 [ 0.35355339  0.35355339 -0.35355339 -0.35355339  0.35355339  0.35355339
  -0.35355339 -0.35355339]
 [ 0.35355339 -0.35355339 -0.35355339  0.35355339  0.35355339 -0.35355339
  -0.35355339  0.35355339]
 [ 0.35355339  0.35355339  0.35355339  0.35355339 -0.35355339 -0.35355339
  -0.35355339 -0.35355339]
 [ 0.35355339 -0.35355339  0.35355339 -0.35355339 -0.35355339  0.35355339
  -0.35355339  0.35355339]
 [ 0.35355339  0.35355339 -0.35355339 -0.35355339 -0.35355339 -0.35355339
   0.35355339  0.35355339]
 [ 0.35355339 -0.35355339 -0.35355339  0.35355339 -0.35355339  0.35355339
   0.35355339 -0.35355339]]

X^n =
[[0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 1 0 0]
 [0 0 0 0 1 0 0 0]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

---

6. Grover Diffusion Operator (W)

```
print(W)
```

Result:

```
[[-0.75 -0.25 -0.25  0.25 -0.25  0.25  0.25 -0.25]
 [-0.25 -0.75  0.25 -0.25  0.25 -0.25 -0.25  0.25]
 [-0.25  0.25 -0.75 -0.25  0.25 -0.25 -0.25  0.25]
 [ 0.25 -0.25 -0.25 -0.75 -0.25  0.25  0.25 -0.25]
 [-0.25  0.25  0.25 -0.25 -0.75 -0.25 -0.25  0.25]
 [ 0.25 -0.25 -0.25  0.25 -0.25 -0.75  0.25 -0.25]
 [ 0.25 -0.25 -0.25  0.25 -0.25  0.25 -0.75 -0.25]
 [-0.25  0.25  0.25 -0.25  0.25 -0.25 -0.25 -0.75]]
```

---

7. Unitarity Checks

```
is_unitary_np(Uf_phase)
is_unitary_np(W)
```

Result:

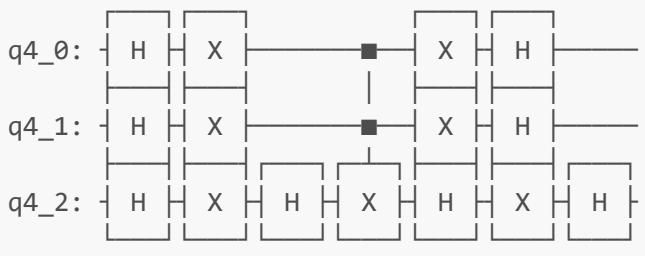
- Phase oracle unitary: **True**
- Diffusion operator unitary: **True**

---

8. Qiskit Oracle Circuit

```
qc_oracle.draw(output='mpl')
```

Circuit:



9. SIMULATED UNITARY OF Uf (AER)

```
backend = AerSimulator(method="unitary")
qc_test = qc_oracle.copy()
qc_test.save_unitary()
```

```
job = backend.run(qc_test)
U_sim = job.result().get_unitary(qc_test, decimals=3)

print(U_sim)
```

Result:

```
Operator([[ 1.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
            0.+0.j],
          [ 0.+0.j,  1.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
            0.+0.j],
          [ 0.+0.j,  0.+0.j,  1.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
            0.+0.j],
          [ 0.+0.j,  0.+0.j,  0.+0.j,  1.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
            0.+0.j],
          [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j, -1.+0.j,  0.+0.j,  0.+0.j,
            0.+0.j],
          [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  1.+0.j,  0.+0.j,
            0.+0.j],
          [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  1.+0.j,
            0.+0.j],
          [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
            1.+0.j]],
         input_dims=(2, 2, 2), output_dims=(2, 2, 2))
```