

Introduction to Quantum Computing

PennyLane: Grover searching

Chris Heunen

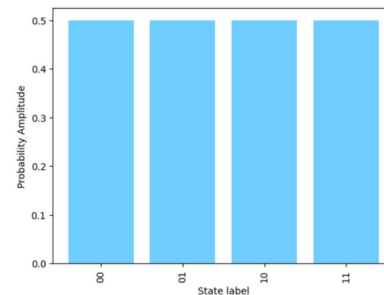
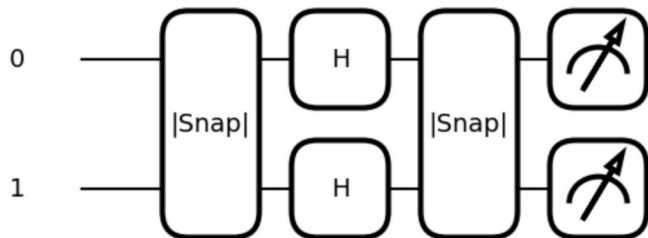
Last week:

- Broke locks (provided by oracles) by testing in pairs
- Deutsch-Josza determines whether solution is present, not what it is
- Use last, auxiliary, qubit catalytically

This week:

- Grover goal: figure out what solution is
- Amplitude amplification: start in uniform superposition
'pump' amplitude from other states into the solution state.

Exercise 1: Uniform superposition

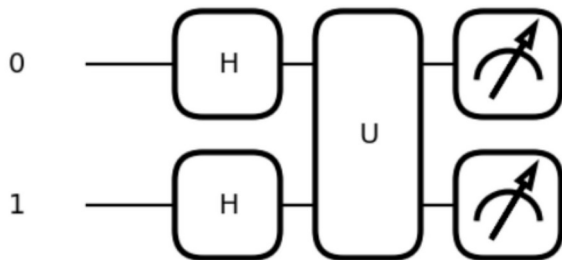


```
n_bits = 4
dev = pl.device("default.qubit", wires=n_bits)

@pl.qnode(dev)
def uniformsuperposition():
    """Build a circuit that creates an n-qubit uniform superposition. Use pl.Snapshot, inspect the results"""
    return pl

pl.drawer.use_style("black_white")
pl.draw_mpl(uniformsuperposition());
pl.snapshots(uniformsuperposition())
```

Exercise 2: Oracle


$$\begin{bmatrix} -1. & 0. & 0. & 0. \\ 0. & -1. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

```
def oracle(keys):  
    # return a diagonal matrix with entries 1 and -1 for the oracle  
    return matrix
```

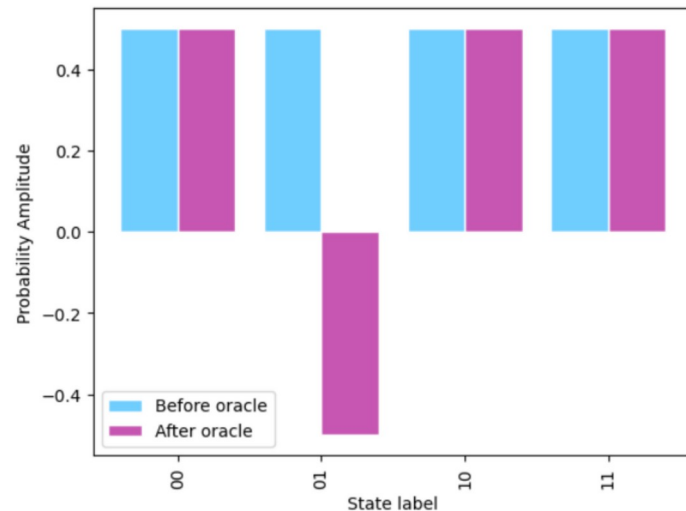
```
@pl.qnode(dev)
```

```
def circuit(keys):  
    # build a uniform superposition  
    # perform the oracle as a unitary gate  
    return pl.probs(wires=wires)
```

Exercise 3: Amplitude amplification

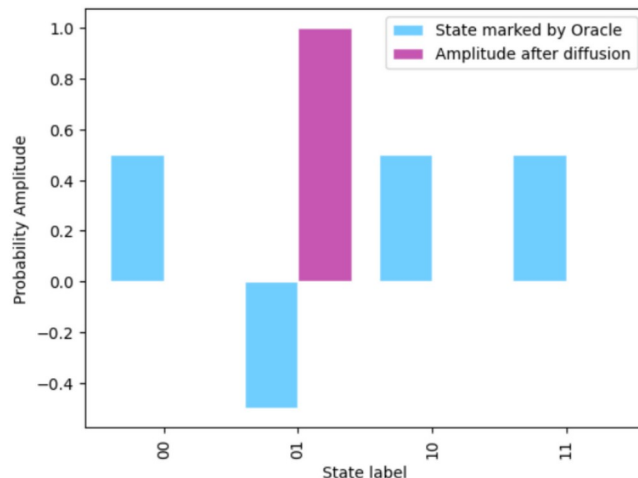
```
keys = [[0,1]]
dev = pl.device("default.qubit", wires=n_bits)

@pl.qnode(dev)
def circuit():
    for i in range(4):
        pl.Hadamard(wires=i)
    pl.Snapshot("Before oracle")
    pl.QubitUnitary(oracle(keys), wires=wires)
    pl.Snapshot("After oracle")
    return pl.probs(wires=wires)
# inspect the results before and after applying the oracle
# plot them in a bar chart using plt
```



Exercise 4: Diffusion operator

```
def diffusion_operator(wires):  
    """Build diffusion operator circuit"""  
  
    def circuit():  
        for i in range(4):  
            pl.Hadamard(wires=i)  
        pl.Snapshot("Uniform superposition")  
        pl.QubitUnitary(oracle(keys), wires=wires)  
        pl.Snapshot("State marked by Oracle")  
        diffusion_operator(wires)  
        pl.Snapshot("Amplitude after diffusion")  
        return pl.probs(wires=wires)  
  
results = pl.snapshots(circuit())
```



Exercise 5: Grover search, multiple keys

```
n_bits = 4
dev = pl.device("default.qubit", wires=n_bits)
wires = list(range(n_bits))
keys = [[0,1,0,1],[1,1,1,1]]
M = 2
N = 2**n_bits

@pl.qnode(dev)
def circuit():
    # build the grover circuit, and iterate it sqrt(n/m)*pi/4 times
    # hint: you can use pl.templates.GroverOperator
    return pl.probs(wires=wires)

results = pl.snapshots(circuit())
# check the results
```

