Lab VIII
Quanum circuit
for Grover algorithm $n = 3$

Jakub Tworzydlo

Institute of Theoretical Physcis, UW

14/04/2026 Pasteura 5, Warszawa

Resources & Tutorials

Useful resources for understanding Grover's Algorithm and its implementation:

- **Qiskit textbook (Grover's algorithm):**
  https://github.com/Qiskit/textbook/blob/main/notebooks/ch-algorithms/grover.ipynb

- **Tutorial & Review Paper:**
  "Introduction to Coding Quantum Algorithms: A Tutorial Series Using Qiskit"
  https://arxiv.org/abs/1903.04359

Hint: printing the wavefunction

To check the mathematical state of your qubits before measurement, you can use Qiskit's `Statevector` class.

You can define the following function in your script to clearly print the complex amplitudes of the basis states:

```python
from qiskit.quantum_info import Statevector

def print_wavefunction(qc, title="Some title"):
    """
    Prints the wavefunction clearly using Qiskit's Statevector.
    Extracts the state as a dictionary and formats the complex amplitudes.
    """
    print(f"\n--- {title} ---")
    sv = Statevector(qc)

    # .to_dict() returns a dictionary like {'000': amplitude, '001': amplitude, ...}
    # Note: Qiskit uses the ordering: |q2 q1 q0>

    for basis_state, amplitude in sv.to_dict().items():
        # Format to print real and imaginary parts
        print(f"  ({amplitude.real:+.3f} {amplitude.imag:+.3f}j) |{basis_state}⟩")

    print("-" * 35)
```

Task 1: Initialization & the oracle (2pts)

First half of the algorithm: setting up the superposition and marking the correct state.

Tasks:

- Initialize a 3-qubit `QuantumCircuit`. Apply Hadamard gates to all qubits to create an equal superposition.
- Define the oracle to mark the target state $w = [0, 1, 1]$.
  (Hint: Remember Qiskit reads $|q_2 q_1 q_0\rangle$. To apply a phase flip specifically to $|011\rangle$, you will need an X gate, a controlled-controlled operation, and another X gate.)
- Use the `print_wavefunction` routine **before** and **after** the oracle. Verify that the phase of $|011\rangle$ flips from positive to negative.
- Separate the sections with `qc.barrier()` and print the circuit drawing.

Task 2: Amplitude amplification (3 pts)

In this task it is useful to wrap your logic into reusable Python functions: the target element w is passed as a variable parameter (e.g., `def phase_oracle(qc, w, qubits):`).

Task 2A: One Iteration

- Implement the Grover diffusion operator (inversion about the mean).
- Apply your parameterized oracle (for a chosen w, e.g., $w = [0, 1, 1]$), followed by the diffuser.
- Print the wavefunction. Did the amplitude of your target state w grow?
- Add measurement gates and plot the histogram of the results.

Task 2B: Two Iterations

- Create a new circuit where you apply the [Oracle + Diffuser] block **twice** in a row for your chosen parameter w.
- Plot the new histogram. How does the success probability compare to a single iteration?

Extra credit & hardware execution (1 pt)

Baseline task:
Run your $n = 3$ Grover circuit (with 1 and 2 iterations) on a real IBM Quantum device. Plot the hardware histogram against your simulator histogram. Discuss the success probability in the presence of real-world noise. Is it still better than classical random guessing?

Optional investigations:

- **Pioneering research:** Try the same with a two-solution phase oracle or a boolean oracle. How do yur results for ASP (Algorithm Success Probability) compare with the one from first experiments?
  https://arxiv.org/abs/1703.10535
- **Transpilation overhead:** Construct the $n = 4$ Toffoli gate (a 3-control X gate). Discuss how the transpiler breaks this down into basic hardware gates. Illustrate the quality of the output on a real device.