Statevector and unitary verification techniques

Test states and complete unitary behaviour without confusing relative and global phase.

Undergraduate 22 min lesson Dr. Charith Jayasekara

By the end, you'll be able to

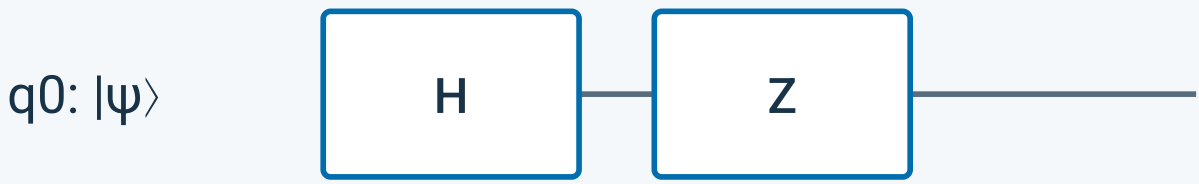
- Choose between a state test and an operation test.
- Compare pure states up to global phase.
- Detect a phase error hidden by Z probabilities.

Begin with a clear question

A useful test asks a precise question. A state test checks one prepared input. An operation test checks what the circuit does to every input. A sampling test checks a distribution. Choosing the right question prevents a correct-looking histogram from hiding a circuit defect.

Build the reasoning

For normalised pure states, squared overlap is one exactly when the two vectors differ by a common global phase. Compare that overlap with a numerical tolerance. If you need an operation guarantee, compare the unitary matrices up to global phase, or use a complete equivalent verification method. Testing just the zero input can miss errors on the other basis states and on superpositions.



Follow time from left to right. Qiskit statevector labels put q0 on the right; diagrams put q0 on the top wire.

Worked example

H and ZH produce plus and minus from zero. Both give a 50/50 Z histogram, but their overlap is zero. A final H converts the hidden sign into a deterministic difference. The code checks a global-phase copy as equivalent and the relative-

phase defect as different.

$$F(\psi, \phi) = |\langle\psi|\phi\rangle|^2, \qquad F(|+\rangle, |-\rangle) = 0, \qquad F(\psi, e^{i\gamma}\psi) = 1.$$

```
import numpy as np
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector, Operator

correct = QuantumCircuit(1)
correct.h(0)
buggy = correct.copy()
buggy.z(0)
a = Statevector.from_instruction(correct)
b = Statevector.from_instruction(buggy)
assert np.allclose(a.probabilities(), b.probabilities())
assert np.isclose(abs(np.vdot(a.data, b.data))**2, 0)
assert Statevector(np.exp(0.37j) * a.data).equiv(a)
assert not Operator.from_circuit(correct).equiv(Operator.from_circuit(buggy))
```

Practical lab

In notebook 07, run the state tests before viewing the deliberate mutations. Document which input and which observable each test covers. Full dense unitary comparison grows exponentially, so use it as a trusted small-instance reference rather than assuming it scales to an arbitrary application. When a block will later be controlled, a phase on that block can become relative between control branches: test the assembled controlled operation as well.

Run the corresponding notebook from the software practice collection:

- Download Circuits, amplitudes and the phase you cannot see in a histogram
- Download Two oracle algorithms: recover bits and recover a hidden XOR period
- Download Order finding for a small modular multiplication
- Download Quantum software tests that catch plausible wrong programs

Keep a short execution record: predicted result, observed result, environment, inputs and the check that would reject a plausible defect. Explain a failure before changing the expected value.

Chapter summary

- Match the test to the state, operation or distribution claim.
- Ignore a common pure-state phase when appropriate.
- Use phase-sensitive inputs and assembled controlled circuits.

Exercises

1. A candidate equals exp(iπ/3) times the reference state. What does its squared overlap equal? Would a Z appended only to one component necessarily pass?
2. Change one part of the worked program to create a plausible fault. State which assertion should fail and why.

Worked answers

1. The common phase gives overlap squared 1. A relative Z changes selected components and generally reduces fidelity; for plus it produces the orthogonal minus state.
2. A useful answer identifies a specific changed gate, parameter, mapping or model and links it to an independent expected behaviour. Keep the intended specification fixed while changing the implementation; save the failing case as a regression test.

Check your understanding

PAUSE & THINK

Why does comparing only Z probabilities miss the missing phase gate?

- [ ] Squaring magnitudes removes the sign information
- [ ] The simulator always corrects phase errors
- [ ] A larger shot count automatically tests every basis

Answer: Squaring magnitudes removes the sign information. Phase can affect later interference while current magnitudes match.

Sources

IBM's bit-ordering guide, Statevector API and Operator API describe the SDK conventions and equivalence tools. The notebook bundle records its pinned environment and local execution evidence.