

Quantum Computing – Hello World

Bill Celmaster

April 12, 2026

Overview

- ▶ Brief review of 2-basis-state quantum mechanics
- ▶ Brief review of product states
- ▶ Gates (pictures, words), unitary matrices and quantum circuits
- ▶ A simple quantum circuit and algorithm for IBM's computer
 - ▶ Introduction to IBM's environment
 - ▶ Some Python scripts (for non-Python people)
 - ▶ Translating those scripts to pictures and gates
 - ▶ Predicting the quantum behavior of the quantum circuit
 - ▶ Execution of the algorithm on actual hardware

1-Qbit systems (2D)

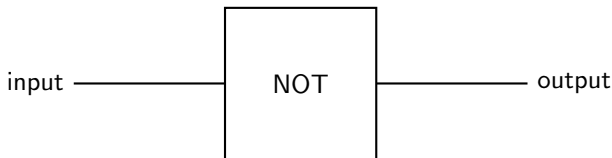
- ▶ Orthogonal basis states are generically labeled $|0\rangle$ and $|1\rangle$.
- ▶ Example: electron-spin in z-direction: spin up \uparrow , spin down \downarrow
 - ▶ Spin states are denoted $|\uparrow\rangle$ and $|\downarrow\rangle$
- ▶ General (normalized) states are superpositions of basis states
 - ▶ $|Q\rangle = \alpha|0\rangle + \beta|1\rangle$
 - ▶ α, β are complex numbers
 - ▶ $|\alpha|^2 + |\beta|^2 = 1$
- ▶ Vector notation: $|\mathbf{Q}\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$
- ▶ For future convenience, we'll refer to $|0\rangle$ and $|1\rangle$ as the **Z** eigenstates

Qbit inner products and measurement

- ▶ We define $\langle \mathbf{Q}' | \mathbf{Q} \rangle$ as $(\alpha')^* \alpha + (\beta')^* \beta$
- ▶ Vector notation: $\langle \mathbf{Q}' | \mathbf{Q} \rangle \rightarrow (\mathbf{Q}', \mathbf{Q}) \equiv \mathbf{Q}'^\dagger \mathbf{Q} = (\alpha'^* \quad \beta'^*) \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$
- ▶ In principal, there are measuring devices $O_{|s\rangle}$ such that
 - ▶ They report “yes” when the system is in state $|s\rangle$
 - ▶ After they report “yes”, the system is still in state $|s\rangle$
- ▶ If a system is in $|\mathbf{Q}\rangle$, then
 - ▶ The probability that $O_{|0\rangle}$ is “yes”, is $|\alpha|^2$
 - ▶ The probability that $O_{|1\rangle}$ is “yes”, is $|\beta|^2$
 - ▶ The probability that $O_{|\mathbf{Q}'\rangle}$ is “yes” is $|\langle \mathbf{Q}' | \mathbf{Q} \rangle|^2$

Gates: Boolean and quantum gates

- ▶ Digital circuits consist of sequences of gates.
 - ▶ Each gate takes M inputs and N outputs
- ▶ An example of a classical (Boolean) gate



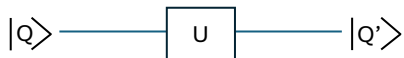
with truth table

input	output
0	1
1	0

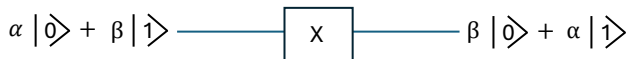
- ▶ An example of a quantum gate

$$\alpha |0\rangle + \beta |1\rangle \longrightarrow \boxed{X} \longrightarrow \beta |0\rangle + \alpha |1\rangle$$

Single Qbit gate and matrix notation



- ▶ Input is $|Q\rangle$, output is $|Q'\rangle$. Both states are normalized. U is linear.
- ▶ Matrix notation: $Q' = UQ$ where U is unitary
- ▶ Example: If $|Q\rangle = |0\rangle$, then $|Q'\rangle = |1\rangle$; if $|Q\rangle = |1\rangle$, then $|Q'\rangle = |0\rangle$.
 - ▶ We call this “flipping the Qbit”.
 - ▶ Linearity:



- ▶ Matrix notation for the example. $\begin{pmatrix} \beta \\ \alpha \end{pmatrix} = X \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ or $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Hadamard gate and Measurement

- ▶ Using circuit-pictures:

- ▶ **Hadamard gate**

$$|0\rangle \text{ --- } \boxed{\text{H}} \text{ --- } \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$$

$$|1\rangle \text{ --- } \boxed{\text{H}} \text{ --- } \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$$

- ▶ **Measurement**

By default, we measure the operator **Z**

$$\alpha|0\rangle + \beta|1\rangle \text{ --- } \boxed{\text{M}}$$

The probability of measuring “0” is $|\alpha|^2$, prob. of “1” is $|\beta|^2$.

- ▶ Using vectors and operators:

- ▶ **Hadamard**: Generates Qbit $\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$

- ▶ **M**: $P(0) = \left| \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \right|^2$

$$P(1) = \left| \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \right|^2$$

A real-world Hadamard gate

Reference material: “Quantum computing with photons” by Stefanie Barz (Oxford)

- ▶ One implementation of Qbits is to use light (photon) polarization.
- ▶ Horizontal: $|H\rangle \rightarrow |0\rangle$, Vertical: $|V\rangle \rightarrow |1\rangle$.
- ▶ Use a uniaxial birefringent (e.g. quartz) wave plate.



Figure: Half-wave plate. Different refractive indices for vertical and horizontal polarizations.

Here, we start with $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and end with $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

- ▶ Rotate crystal by θ from the horizontal axis.

$$U(\theta) = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix}$$

- ▶ E.g. $\theta = \frac{\pi}{8}$ is the Hadamard gate.

The “Hello World” circuit

- ▶ Circuit-picture:



- ▶ Matrix-vector:

$$P(0) = \left| \begin{pmatrix} 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right|^2 = \frac{1}{2}$$

$$P(1) = \left| \begin{pmatrix} 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right|^2 = \frac{1}{2}$$

- ▶ What would an experiment look like?

- ▶ Start with Qbit $|0\rangle$.
- ▶ Transform it to the Bell state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
- ▶ Measure the **Z** value (e.g. z-spin “up” or “down”)
- ▶ Repeat 100 times
- ▶ Report fraction of 0's and fraction of 1's.

The “Hello World” Python quantum-simulator program

```
from qiskit import QuantumCircuit
from qiskit.primitives import StatevectorSampler
from IPython.display import display

# 1. Create a 1-qubit circuit
qc = QuantumCircuit(1)

# 2. Apply Hadamard gate
qc.h(0)

# 3. Measure
qc.measure_all()

# 4. Create Sampler
sampler = StatevectorSampler()

# 6. Run the circuit with the number of shots
job = sampler.run([qc], shots=100)

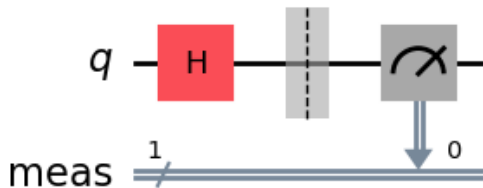
# 7. Get results
result = job.result()
meas = result[0].data.meas

# 8. Convert to counts
counts = meas.get_counts()

print(counts)

display(qc.draw("mpl"))
```

Output



{'0': 57, '1': 43}

The "Hello World" program for quantum hardware

```
##                               Import definitions of functions
from qiskit import QuantumCircuit
from qiskit_ibm_runtime import QiskitRuntimeService
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit_ibm_runtime import Sampler
from IPython.display import display

##                               First create a circuit locally
#                               1. Create a 1-qubit circuit
qc = QuantumCircuit(1)
#                               2. Apply Hadamard gate
qc.h(0)
#                               3. Measure
qc.measure_all()

##                               Set up the connection to the appropriate IBM cloud quantum computer
#                               4. Loads my saved IBM credentials and connects to IBM cloud
service = QiskitRuntimeService()
#                               5. Runs locally, queries IBM cloud for least busy devices, and
#                               returns a handle for the remote backend
backend = service.least_busy(simulator=False, operational=True)

##                               Transpiler step: locally convert my circuit to use real quantum hw -- gates and topology
#                               6. Convert to an ISA circuit and layout-mapped observables.
pm = generate_preset_pass_manager(backend=backend, optimization_level=1)
isa_circuit = pm.run(qc)
#                               7. Show the new circuit
display(isa_circuit.draw("mpl", idle_wires=False))
```

"Hello World" for quantum hardware, cont'd

```
##      Running and analyzing the circuit
#      8. Create the program to be run on the IBM cloud, which analyzes
#      outputs of quantum circuit
sampler = Sampler(mode=backend)
#      9. Run the circuit with the number of shots
job = sampler.run([isa_circuit], shots=100)
print("Job ID:", job.job_id())
#      10. Get results (waits until job finishes)
result = job.result()
#      11. Extract counts
counts = result[0].data.meas.get_counts()
print("Counts:", counts)
```

Global Phase: $\pi/4$



Job ID: d6vnhg2tnsts73ete5a0
Counts: {'0': 45, '1': 55}

_____ (a second run, but without the diagram)

Job ID: d6vnm0s69uic73cjgq3g
Counts: {'1': 52, '0': 48}

Interpretation of the transpiled circuit

- ▶ Instead of the \mathbf{H} gate, the circuit now has 3 sequential gates:

$$\mathbf{R}_z\left(\frac{\pi}{2}\right), \sqrt{\mathbf{X}}, \mathbf{R}_z\left(\frac{\pi}{2}\right)$$

- ▶ $\mathbf{R}_z\left(\frac{\pi}{2}\right) = \begin{pmatrix} e^{-i\frac{\pi}{4}} & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$ and $\sqrt{\mathbf{X}} = \sqrt{2} \begin{pmatrix} e^{i\frac{\pi}{2}} & e^{-i\frac{\pi}{2}} \\ e^{-i\frac{\pi}{2}} & e^{i\frac{\pi}{2}} \end{pmatrix}$

- ▶ $\mathbf{R}_z\left(\frac{\pi}{2}\right)$ represents a spin rotation by $\frac{\pi}{2}$ around the z-axis.

- ▶ $\sqrt{\mathbf{X}^2} = \mathbf{X}$.

- ▶ These gates can be implemented in hardware.

- ▶ Multiply the matrices. $\mathbf{R}_z\left(\frac{\pi}{2}\right) \sqrt{\mathbf{X}} \mathbf{R}_z\left(\frac{\pi}{2}\right) = e^{-i\frac{\pi}{4}} \mathbf{H}$.

- ▶ Recall for the original “hello world” circuit, that

$$P_{\text{orig}}(0) = \left| \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{H} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right|^2 = \frac{1}{2}, \text{ etc.}$$

- ▶ Similarly, for the transpiled circuit, we have

$$P_{\text{orig}}(0) = \left| \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{R}_z\left(\frac{\pi}{2}\right) \sqrt{\mathbf{X}} \mathbf{R}_z\left(\frac{\pi}{2}\right) \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right|^2 =$$
$$\left| \begin{pmatrix} 1 & 0 \end{pmatrix} e^{-i\frac{\pi}{4}} \mathbf{H} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right|^2 = \left| \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{H} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right|^2 = \frac{1}{2}, \text{ etc.}$$