Using QISKit: The SDK for Quantum Computing

Ali Javadi-Abhari
IBM Research
History of Quantum Computing

- **1935**: The EPR Paradox
- **1964**: Bell’s Inequality
- **1970**: Birth of quantum information theory
- **1980**: First conference on physics of computation co-hosted by MIT and IBM
- **1982**: Discovery of topological quantum order
- **1984**: Quantum cryptography (IBM)
- **1993**: Quantum teleportation (IBM)
- **1994**: Shor’s Factoring Algorithm
- **1995**: Quantum error correction
- **1996**: DiVincenzo Criteria for building a quantum computer (IBM)
- **1997**: Topological codes
- **1999**: Topological codes
- **2001**: Circuit QED is demonstrated
- **2004**: The transmon superconducting qubit
- **2007**: Coherence time improvement (IBM)
- **2012**: Demonstrate [[2,0,2]] code (IBM)
- **2015**: Demonstrate [[2,0,2]] code (IBM)
- **2016**: IBM Q Experience Launched
- **2017**: IBM launches commercial universal quantum computing
IBM Q Experience

75K Unique Users

2.5 Million Experiments Run

60 Scientific Papers

Quantum Goes Global
The IBM Quantum Experience has attracted an enthralled international following. Here’s a sampling of the activities—from experiments and courses to plenary sessions—built around our 5-qubit machine.
From Quantum Experience to Quantum Programs

Build

Python Interface

Compile

Translate & Optimize

Quantum Developers

Real Devices

Simulators

API

Laboratory
Plan for This Talk

1. Quick overview of (necessary) quantum computing fundamentals
   * for deeper dive, see Lev Bishop’s previous IBM DeveloperWorks talk

2. Solving a concrete problem with quantum advantage

3. Live demo

4. Near term prospects and limitations
Quantum Computing 101

1. Identically prepared photons can behave randomly when measured. This randomness is inherent, and not a limitation of equipment.
Quantum Computing 101

1. Identically prepared photons can behave randomly when measured. This randomness is inherent, and not a limitation of equipment.

2. The state of two photons cannot (in general) be described by knowing the state of each one separately (**entanglement**).
Quantum Computing 101: Linear Algebra View

1. The state of $n$ qubits is like a vector in $2^n$ dimensional space.

2. Quantum operations are like matrices moving those vectors around.

3. Measurement (observation) is like probabilistic projection on the basis.

Notation: $|0\rangle \rightarrow |1\rangle$
Quantum Circuits

1-qubit rotations + 2-qubit gate sufficient for universal computation.

For the rest of today, we just need to know about 2 gates: 
H, CNOT
The Hadamard Gate: Superposition

Hadamard Gate:

\[ |0\rangle \xrightarrow{H} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \]

H gate creates superposition.

Measurement results in a single (classical) bitstring.
The Hadamard Gate: Effect on Phase

$$|0\rangle\quad \rightarrow \quad |0\rangle + |1\rangle \quad \rightarrow \quad |0\rangle \quad \text{50\%}$$

H gate is self-inverse.

$$|1\rangle\quad \rightarrow \quad |0\rangle - |1\rangle \quad \rightarrow \quad |1\rangle \quad \text{50\%}$$

phase
The Controlled-NOT Gate: Quantum If

\[ |0\rangle \rightarrow |0\rangle \]

\[ a |0\rangle + b |1\rangle \rightarrow a |0\rangle + b |1\rangle \]

CNOT flips the target bit if the control bit is 1.

\[ |1\rangle \rightarrow |1\rangle \]

\[ a |0\rangle + b |1\rangle \rightarrow b |0\rangle + a |1\rangle \]
Bernstein-Vazirani Algorithm*

Optimal classical strategy:

\[
\begin{align*}
& X = 1 \ 0 \ \ldots \ 0 \ 0 \quad (2^{n-1}) \\
& X = 0 \ 1 \ \ldots \ 0 \ 0 \quad (2^{n-2}) \\
& \quad \vdots \\
& X = 0 \ 0 \ \ldots \ 1 \ 0 \quad (2) \\
& X = 0 \ 0 \ \ldots \ 0 \ 1 \quad (1)
\end{align*}
\]

n tries

Input (query)

Secret Bitstring

Output (result)
Classical Oracle vs. Quantum Oracle

Classical Dot-Product Oracle

Quantum Dot-Product Oracle

Must be reversible!
Implementing the Oracle

\[
|s = 0101 |
\]

\[
|x_0\rangle \quad |x_1\rangle \quad |x_2\rangle \quad |x_3\rangle \quad |tmp\rangle
\]

\[
|x_3.s_3 \oplus x_2.s_2 \oplus x_1.s_1 \oplus x_0.s_0 \oplus tmp\rangle
\]

Pattern of controlled-NOTs specify the Oracle’s functionality (the hidden bitstring \( s \))
Trick: Phase Kickback

\[ |00\rangle - |01\rangle - |10\rangle + |11\rangle = (|0\rangle - |1\rangle)(|0\rangle - |1\rangle) \]

Phase kickback!
Wherever there’s CNOT, phase kickback puts that control qubit in state $|1\rangle$. 
QISKit
Live Demo

Available here:

https://github.com/ajavadia/qiskit-sdk-py/blob/Demo/demo/BV%20Demo.ipynb
Why did this algorithm work?

1. Classical oracles can only be queried with a single number at a time. Quantum oracles can be queried in superposition.

2. We did not merely “try every answer simultaneously”. The problem had a structure that we could exploit using qubits: **encode information in phases.**
Challenges for near-term quantum computing

1. Don’t have many qubits

2. Can’t do many gates
   - **Gate error**: gates are imperfect
   - **Relaxation**: qubits do not retain state for long
Effect of Gate Errors and the Role of Software

Programmed Circuit

| q_0 | H |
| q_1 | H |
| q_2 | H |
| q_3 | H |
| q_4 | X |

Compiled Circuit #1

Good

Correct answer

Backend: ibmqx4 (5 Qubits)

Device

Bad

Compiled Circuit #2

Correct answer indistinguishable from noise

Win $5000 in prizes
Deadline: 15th May 2018

IBM Q Awards Developer Challenge
Find qubit relaxation rate by running circuits

1. Put qubit in excited state and wait variable amounts of time, then measure.

2. Repeat each circuit many times (e.g. 1000 shots) to approximate probability of unwanted $|0\rangle$ state in each.

3. Find relaxation rate by fitting an exponential decay curve to the data.
QISKit
Live Demo

Available here:

https://github.com/ajavadia/qiskit-sdk-py/blob/Demo/demo/Relaxation%20Demo.ipynb
Get Involved!
QISKit Documentation

Quantum Information Software Kit (QISKit), SDK Python version for working with OpenQASM and the IBM Q experience (QX).

Table of Contents

- Installation and setup
  - Installation
  - Install Jupyter-based tutorials
  - FAQ
- Getting started
  - Quantum Chips
  - Project Organization
- QISKit overview
  - Philosophy
  - Project Overview
- Developer documentation
  - Programming interface
  - Internal modules
  - Logging
  - Testing
- SDK reference
  - Submodules
  - Classes
  - Exceptions
- Release history
  - QISKit SDK 0.4.0

Python Modules

Main Modules
```python
import matplotlib.pyplot as plt
plt.rc('font', family='monospace')
for bitString in states:
    char = chr(int( bitString[0:8] ,2)) # get string of the leftmost 8 bits and convert to an ASCII character
    char += chr(int( bitString[8:16] ,2)) # do the same for string of rightmost 8 bits, and add it to the previous character
    prob = stats[bitString] / 1024 # fraction of shots for which this result occurred
    # create plot with all characters on top of each other with alpha given by how often it turned up in the output
    plt.annotate( char, (0.5,0.5), va="center", ha="center", color = (0,0,0, prob ), size = 300)
    if (prob>0.05): # list prob and char for the dominant results (occurred for more than 5% of shots)
        print(str(prob)+"\t"+char)
plt.axis('off')
plt.show()
```

0.189453125  8)
0.1865234375  ;)
Quantum Software Development Kit for writing quantum computing experiments, programs, and applications.

http://www.qiskit.org

<table>
<thead>
<tr>
<th>Branch</th>
<th>New pull request</th>
<th>Create new file</th>
<th>Upload files</th>
<th>Find file</th>
<th>Clone or download</th>
</tr>
</thead>
</table>

- **1,329 commits**
- **2 branches**
- **27 releases**
- **47 contributors**
- **Apache-2.0**

<table>
<thead>
<tr>
<th>GH Event</th>
<th>Title</th>
<th>Description</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>.github</td>
<td>Add templates for issues and pull requests</td>
<td>8 months ago</td>
<td></td>
</tr>
<tr>
<td>cmake</td>
<td>Coverage of the testsuite</td>
<td>9 days ago</td>
<td></td>
</tr>
<tr>
<td>doc</td>
<td>Adding a Chinese translation of README</td>
<td>19 days ago</td>
<td></td>
</tr>
<tr>
<td>examples</td>
<td>Circuit Image plotter</td>
<td>23 days ago</td>
<td></td>
</tr>
<tr>
<td>images</td>
<td>Update Sphinx theme color and diagrams</td>
<td>6 months ago</td>
<td></td>
</tr>
<tr>
<td>qiskit</td>
<td>Fix drawing u0 gate</td>
<td>a day ago</td>
<td></td>
</tr>
<tr>
<td>src/qiskit-simulator</td>
<td>Fix bug with auto-generated CNOT coherent error matrix in C++ simulator</td>
<td>9 days ago</td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>the compilation output should not depend on the Python version</td>
<td>5 days ago</td>
<td></td>
</tr>
<tr>
<td>tools</td>
<td>Remove comment and static flag on travis osx</td>
<td>20 days ago</td>
<td></td>
</tr>
<tr>
<td>.gitignore</td>
<td>[WIP] CMake integration</td>
<td>a month ago</td>
<td></td>
</tr>
<tr>
<td>.pylintrc</td>
<td>Convert qiskit.tools to modules, lint</td>
<td>a month ago</td>
<td></td>
</tr>
<tr>
<td>.travis.yml</td>
<td>Coverage of the testsuite</td>
<td>9 days ago</td>
<td></td>
</tr>
</tbody>
</table>
Qiskit is 1 year old! Thank you to everyone who contributed code, gave feedback, and used it to run experiments on IBM quantum computers.

Happy birthday

That's just about one hundred times younger than Richard Feynman
Get Involved!

- qiskit.org
- github.com/qiskit/qiskit-tutorial
- github.com/qiskit
- qiskit.slack.com

IBM Q Awards Developer Challenge

- Explore
- Learn more
- Contribute code
- Help define
- Win $5000 in prizes
  Deadline: 15th May 2018