Quantum Computers Sometimes Go Zoom

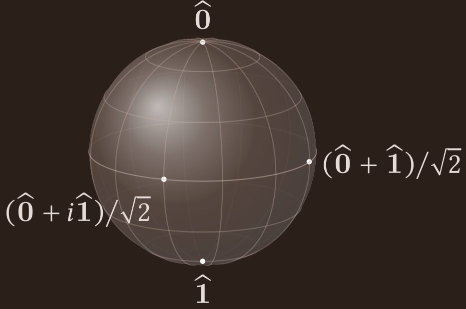
Sophie MacDonald
Mia Kramer

December 2, 2021

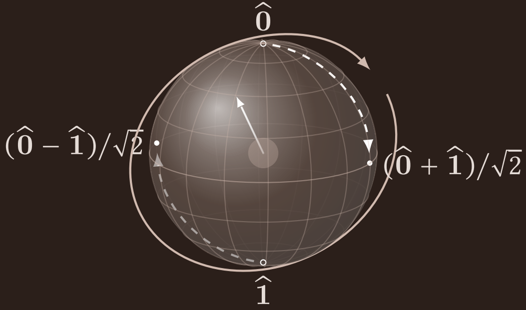
- Qubits are the quantum version of bits; they are two-dimensional rather than two-valued
- Quantum states are vectors
- We can visualize states using a Bloch sphere
- Adding a qubit doubles our dimension
- Examples of quantum operations include rotations on the Bloch sphere like the Hadamard operation, and operations like CNOT
- The Deutsch-Jozsa problem requires $\leq 2^{n-1} + 1$ evaluations of f on a Turing machine, but only one on a quantum computer
- Some efficient quantum algorithms exist, but quantum computers are only faster when such an algorithm can be found

- Generally, if a classical system has n STATES, a corresponding quantum one has an n-dimensional STATE SPACE
- A bit has two states and a QUBIT has a 2D state space
- Since it has two dimensions, we might write it as a 2-vector $\begin{bmatrix} a & b \end{bmatrix}^{\mathrm{T}}$
  - But note that a and b are complex—of the form $\alpha + \beta i$
- Instead of working with 0 and 1, we have a pair of orthogonal vectors $\widehat{\mathbf{0}}$ and $\widehat{\mathbf{1}}$
  - We'll choose those as our basis: $\widehat{\mathbf{0}} = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$; $\widehat{\mathbf{1}} = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$
- Finally, we can multiply any state by a complex number without changing the "meaning" of the state
  - $c(a\widehat{\mathbf{0}} + b\widehat{\mathbf{1}}) \equiv (a\widehat{\mathbf{0}} + b\widehat{\mathbf{1}})$

- Our states are two-dimensional but have complex components so it seems like we should have four degrees of freedom
- But, because of equivalence under multiplication by a complex scalar we're back down to two
  - By convention, we normalize them so that $|a|^2 + |b|^2 = 1$
- Since our DOF work out to angles, we can draw states on a sphere, the Bloch sphere
  - But note here, opposite sides are orthogonal!



- Valid operations (other than measurement) are matrices
- One common single-qubit operation is the HADAMARD one—a possible rotation on the Bloch sphere



- Another is the CNOT operation, the quantum version of XOR, which takes two bits and flips the second iff the first is 1

- When we add a bit to a system, we double the number of possible states
- So, when we add a qubit to a system, we double the number of dimensions
- For a single qubit, we had bases $\widehat{\mathbf{0}}$ and $\widehat{\mathbf{1}}$, and we're adding "another" $\widehat{\mathbf{0}}$ and $\widehat{\mathbf{1}}$
  - So, our basis is $(\widehat{\mathbf{0}}_1 \otimes \widehat{\mathbf{0}}_2), (\widehat{\mathbf{0}}_1 \otimes \widehat{\mathbf{1}}_2), (\widehat{\mathbf{1}}_1 \otimes \widehat{\mathbf{0}}_2), (\widehat{\mathbf{1}}_1 \otimes \widehat{\mathbf{1}}_2)$
  - Remember, these are just vectors: $(\widehat{\mathbf{0}}_1 \otimes \widehat{\mathbf{0}}_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$

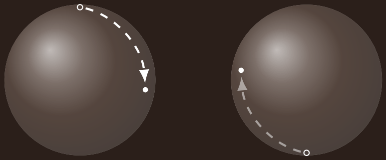
The Deutsch-Jozsa Problem

Don't worry, I couldn't pronounce it at first either.

- Given a function $f : \mathbb{B}^n \to \mathbb{B}$, which is either
  - CONSTANT (the same for all inputs) or
  - BALANCED (0 for half the input domain and 1 for the rest),

  determine whether it's constant or balanced.
- Easy to see in $n = 1$ case, the best classical solution requires two evaluations
  - $\text{BALANCED?}(f) = f(0) \veebar f(1)$
- For larger n, in the worst case we need to test more than half the domain: $2^{n-1} + 1$ evaluations
  - Best case still requires two

- We need two qubits, we'll initialize the first (q_1) to $\widehat{\mathbf{0}}$ and the second (q_2) to $\widehat{\mathbf{1}}$
  - We'll write the state of q_i as $\widehat{\mathbf{q}}_i$
- **Assumption**: we are given a quantum implementation of f that takes us from the state $\widehat{\mathbf{q}}_1 \otimes \widehat{\mathbf{q}}_2$ to $\widehat{\mathbf{q}}_1 \otimes \left(\widehat{\mathbf{q}}_2 \text{ CNOT } f(\widehat{\mathbf{q}}_1)\right)$
  - This is not just a classic oracle that tells us the function value!
- After initializing our qubits, we apply a Hadamard to both:



- We are now in the state $\frac{1}{2}\left((\widehat{\mathbf{0}}_1 + \widehat{\mathbf{1}}_1) \otimes (\widehat{\mathbf{0}}_2 - \widehat{\mathbf{1}}_2)\right)$

- Now, we apply our implementation of f to our state
  - $\frac{1}{2}\big((\hat{\mathbf{0}}_1 + \hat{\mathbf{1}}_1) \otimes (\hat{\mathbf{0}}_2 - \hat{\mathbf{1}}_2)\big)$
- This brings us to the state:

$$\frac{1}{2}\Bigg(\hat{\mathbf{0}}_1 \otimes \Big(\underbrace{\big(f(0)\ \text{CNOT}\ \hat{\mathbf{0}}_2\big) - \big(f(0)\ \text{CNOT}\ \hat{\mathbf{1}}_2\big)}_{\hat{\mathbf{0}}_2 - \hat{\mathbf{1}}_2 \text{ if } f(0)=0,\ \hat{\mathbf{1}}_2 - \hat{\mathbf{0}}_2 \text{ if } f(0)=1} \Big)$$

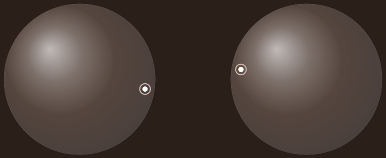
$$+ \hat{\mathbf{1}}_1 \otimes \Big(\underbrace{\big(f(1)\ \text{CNOT}\ \hat{\mathbf{0}}_2\big) - \big(f(1)\ \text{CNOT}\ \hat{\mathbf{1}}_2\big)}_{\hat{\mathbf{0}}_2 - \hat{\mathbf{1}}_2 \text{ if } f(1)=0,\ \hat{\mathbf{1}}_2 - \hat{\mathbf{0}}_2 \text{ if } f(1)=1} \Big)\Bigg)$$

$$= \frac{1}{2}\Big((-1)^{f(0)}\hat{\mathbf{0}}_1 \otimes (\hat{\mathbf{0}}_2 - \hat{\mathbf{1}}_2) \ + \ (-1)^{f(1)}\hat{\mathbf{1}}_1 \otimes (\hat{\mathbf{0}}_2 - \hat{\mathbf{1}}_2)\Big)$$

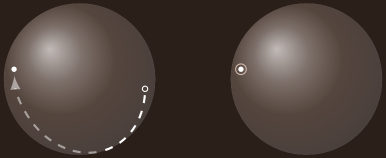
$$= \frac{1}{2}\ \underbrace{(-1)^{f(0)}}_{\text{global phase}}\ \Big(\hat{\mathbf{0}}_1 + (-1)^{f(0) \veebar f(1)}\hat{\mathbf{1}}_1\Big) \otimes (\hat{\mathbf{0}}_2 - \hat{\mathbf{1}}_2)$$

The Quantum Solution
Deutsch's Algorithm, Part II.

- Example when $f \equiv 0$ (constant):
  - $\frac{1}{2}(\widehat{\mathbf{0}}_1 + (-1)^0 \widehat{\mathbf{1}}_1) \otimes (\widehat{\mathbf{0}}_2 - \widehat{\mathbf{1}}_2)$
  - Remember: ignore global phase



- When $f(x) = x$ (balanced):
  - $\frac{1}{2}(\widehat{\mathbf{0}}_1 + (-1)^1 \widehat{\mathbf{1}}_1) \otimes (\widehat{\mathbf{0}}_2 - \widehat{\mathbf{1}}_2)$



- We can always ignore global phase, and clearly the first qubit has the interesting information:

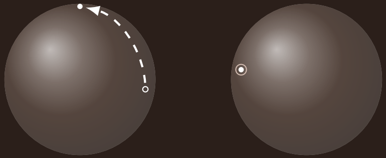
$$\widehat{\mathbf{q}}_1 = \frac{1}{\sqrt{2}}\left(\widehat{\mathbf{0}} + (-1)^{f(0)\,\underline{\vee}\, f(1)}\widehat{\mathbf{1}}\right)$$

  Now, we apply Hadamard one more time:

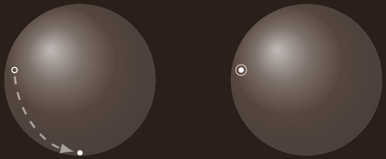
$$\Rightarrow \frac{1}{2}\left(1 + (-1)^{f(0)\,\underline{\vee}\, f(1)}\right)\widehat{\mathbf{0}} + \left(1 - (-1)^{f(0)\,\underline{\vee}\, f(1)}\right)\widehat{\mathbf{1}}$$

- Finally, we measure this qubit:
  - 1 when $f(0) \,\underline{\vee}\, f(1) = 1$ (balanced)
  - 0 when $f(0) \,\underline{\vee}\, f(1) = 0$ (constant)

- Example when $f(x) = \texttt{false}$:



- When $f(x) = x$:



- We use two qubits of memory
- We transform them to not be in our standard $\widehat{0}, \widehat{1}$ basis
- By using the properties of our quantum oracle, we are able to "transfer" all of the interesting information onto one qubit, and "discard" the rest as global phase
- Then, we can transform the interesting qubit back to a basis where we can perform a useful measurement

- Lots to QM not discussed here
- In particular: quantum states are inherently fragile
  - Classical bits have inherent noise-resistance from being binary
  - Also easier to build error-correcting codes since there's only one type of error (bit flip)
- Some "spooky" terminology you may have heard:
  - SUPERPOSITION refers to states that are not the basis states of interest (i.e. $\widehat{\mathbf{0}}, \widehat{\mathbf{1}}$ for us)
  - ENTANGLED states can't be written as a simple product; consider $\frac{1}{\sqrt{2}}\left((\widehat{\mathbf{0}}_1 \otimes \widehat{\mathbf{0}}_2) + (\widehat{\mathbf{1}}_1 \otimes \widehat{\mathbf{1}}_2)\right)$
- We can see that a quantum computer can be asymptotically faster... but only if you've designed a quantum algorithm
  - Designing quantum algorithms is not easy!
  - A quantum computer also cannot do anything a classical computer cannot

- Some other (more useful) quantum algorithms:
  - Shor's Algorithm does integer factorization/discrete logarithm in polynomial time
  - Grover's Algorithm searches an unsorted list in $O(\sqrt{n})$ time
  - The Quantum Fourier Transform is exponentially faster than DFT
  - In general, quantum computers will be exponentially faster at simulating quantum systems (think molecular reactions)

- References:
    - *Rapid solution of problems by quantum computation*, Deutsch and Jozsat. 1992. Proceedings: Mathematical and Physical Sciences, Volume 439, Issue 1907, pp. 553-558.
    - *Quantum Algorithms Revisited*, Cleve, Ekert, Macchiavello and Mosca. 1998. Proceedings of the Royal Society, Series A: Mathematical, Physical and Engineering Sciences Volume 454, Issue 1969.
- Further reading:
    - *Quantum Computation and Quantum Information*, Nielsen and Chang. 2010.
    - *Quantum Computing Since Democritus*, Scott Aaronson. Cambridge, 2013.