Intro to Quantum Computing

Lecture 2: Gate model hardware and applications

Olivia Di Matteo

Quantum Information Science Associate, TRIUMF

Recap of yesterday

We learned about:

- Mathematical formalism of multi-qubit systems
- Unitary operations, entanglement
- Some small quantum protocols with unique effects

What questions do you have?

Today

Large-scale algorithms and applications:

- Main classes of quantum algorithms
- Overviews of Grover's algorithm
- Quantum advantage
- Simulating neutrino oscillations
- Finding the ground state of a deuteron using the *variational quantum eigensolver*

Gate-model quantum computers in practice

- Who is building one?
- How are they building it?
- What are the limitations of their hardware?

The no-cloning theorem

Many processes in quantum computing are complicated by the following fact:

The no-cloning theorem

It is impossible to create a copying circuit that works for arbitrary quantum states.

Proof of the no-cloning theorem

Suppose we want to clone a state $|\psi\rangle$. We want a unitary operation that sends

$$U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle \quad (1)$$

where $|s\rangle$ is some arbitrary state.

Let's suppose we find one. If our cloning machine is going to be universal, then we must also be able to clone some other state, $|\varphi\rangle$.

$$U(|\varphi\rangle \otimes |s\rangle) = |\varphi\rangle \otimes |\varphi\rangle \quad (2)$$

Proof of the no-cloning theorem

We purportedly have:

$$U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle \quad (3)$$

$$U(|\varphi\rangle \otimes |s\rangle) = |\varphi\rangle \otimes |\varphi\rangle \quad (4)$$

Take the inner product of the LHS of both equations:

$$(\langle\psi| \otimes \langle s|)\, U^\dagger U\, (|\varphi\rangle \otimes |s\rangle) = \langle\psi|\varphi\rangle\langle s|s\rangle = \langle\psi|\varphi\rangle$$

($U^\dagger U = \mathbb{1}$; $\langle s|s\rangle = 1$)

Now take the inner product of the RHS of both equations:

$$(\langle\psi| \otimes \langle\psi|)(|\varphi\rangle \otimes |\varphi\rangle) = \langle\psi|\varphi\rangle\langle\psi|\varphi\rangle$$

$x^2 = x$

These two inner products must be equal; but the only numbers that square to themselves are 0 and 1! So either the two states are orthogonal, or are just the same state - they can't be arbitrary!

Quantum algorithms II

Quantum speedup

Important question: What kinds of problems can quantum computers do better than classical computers?

There are 3 important classes of quantum algorithms:

- Algorithms based on the *quantum Fourier transform*
- Algorithms based on *amplitude amplification*
- Hamiltonian simulation

Quantum speedup

Polynomial speedup

Quantum algorithms based on amplitude amplification typically obtain a *polynomial* speedup over the best-known classical algorithms. i.e. for n qubits, if a classical algorithm takes $\sim 2^n$ time steps to run, the improved quantum version will take $\sim \sqrt{2^n} = 2^{\frac{n}{2}}$ time steps.

Exponential speedup

Many quantum algorithms that use the quantum Fourier transform as an underlying subroutine obtain a *superpolynomial*, or *exponential* speedup. If the classical algorithm takes $\sim 2^n$ time steps, the quantum algorithm takes $\sim n^k$ for some integer k (ideally n^2 or n^3).

Quantum algorithms

I will spend some time today walking you through *Grover's search algorithm*. It is based on amplitude amplification, and has a very intuitive visual representation.

Working through the quantum Fourier transform in gory detail would take the better part of the lecture - so, I have prepared an additional slide deck and homework task for you to work through if you're interested. It discusses the QFT and how to use it to perform *eigenvalue estimation*.

Grover's algorithm

Motivation: 'needle in a haystack' search problems.

Suppose you have some function f on all 2^n n-bit binary strings, and a black box / oracle that implements it.

$f(00\cdots0)$
$f(00\cdots1)$

Grover's algorithm

Motivation: 'needle in a haystack' search problems.

Suppose you have some function f on all 2^n n-bit binary strings, and a black box / oracle that implements it.

Recall from Deutsch's algorithm yesterday that if we have an oracle that implements a function f, we are assuming that there is some unitary transform that implements

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \tag{7}$$

Grover's algorithm

Motivation: 'needle in a haystack' search problems.

Suppose you have some function f on all 2^n n-bit binary strings, and a black box / oracle that implements it.

Recall from Deutsch's algorithm yesterday that if we have an oracle that implements a function f, we are assuming that there is some unitary transform that implements

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \tag{7}$$

When we chose $|y\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, we saw that phase kickback gave us the nice result:

$$U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle \tag{8}$$

Let's suppose that the function f is structured so that exactly one of its inputs, x_s, is 'special':

$$f(x) = \begin{cases} 1 & \text{if } x = x_s \\ 0 & \text{otherwise} \end{cases}$$

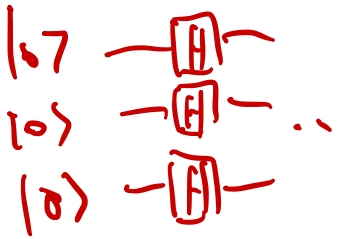
How do we determine which of the 2^n strings is the special one?

Classically, in the worst case we have to query the oracle 2^n times!

Grover's quantum search algorithm

The idea behind Grover's search algorithm is to start with a uniform superposition and then *amplify* the amplitude of the state corresponding to the solution.

In other words we want to go from the uniform superposition[1]



$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle = \frac{1}{\sqrt{2^n}} \left(|00\cdots0\rangle + |10\cdots1\rangle + \cdots + |11\cdots1\rangle \right)$$

to something that looks more like this:

$$|\psi'\rangle = (\text{big number}) \cdot |x_s\rangle + (\text{small number}) \sum_{x \neq x_s} |x\rangle$$

[1] If you didn't get to yesterday's homework, this is what happens when you Hadamard every qubit; it is used in many quantum algorithms!

Q: Why do we want a state of this form?

$$|\psi'\rangle = (\text{big number})|x_s\rangle + (\text{small number}) \sum_{x \neq x_s} |x\rangle \qquad (12)$$

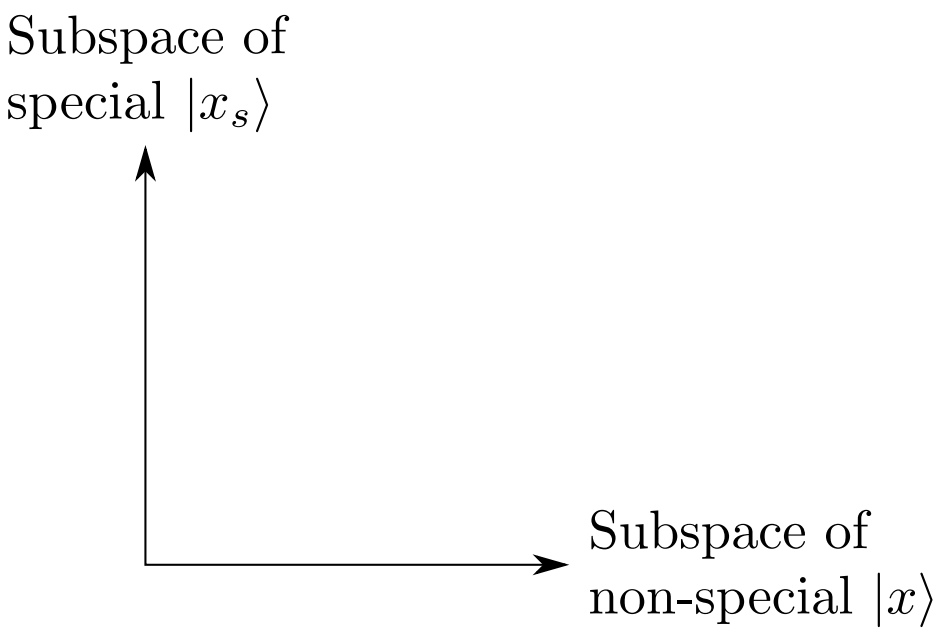
Q: Why do we want a state of this form?

$$|\psi'\rangle = \underbrace{(\text{big number})|x_s\rangle} + (\text{small number}) \sum_{x \neq x_s} |x\rangle \qquad (12)$$

A: When we make a measurement, we will very likely get the solution to our problem!

How do we turn the uniform superposition into a state that looks like this?

Grover's algorithm visually



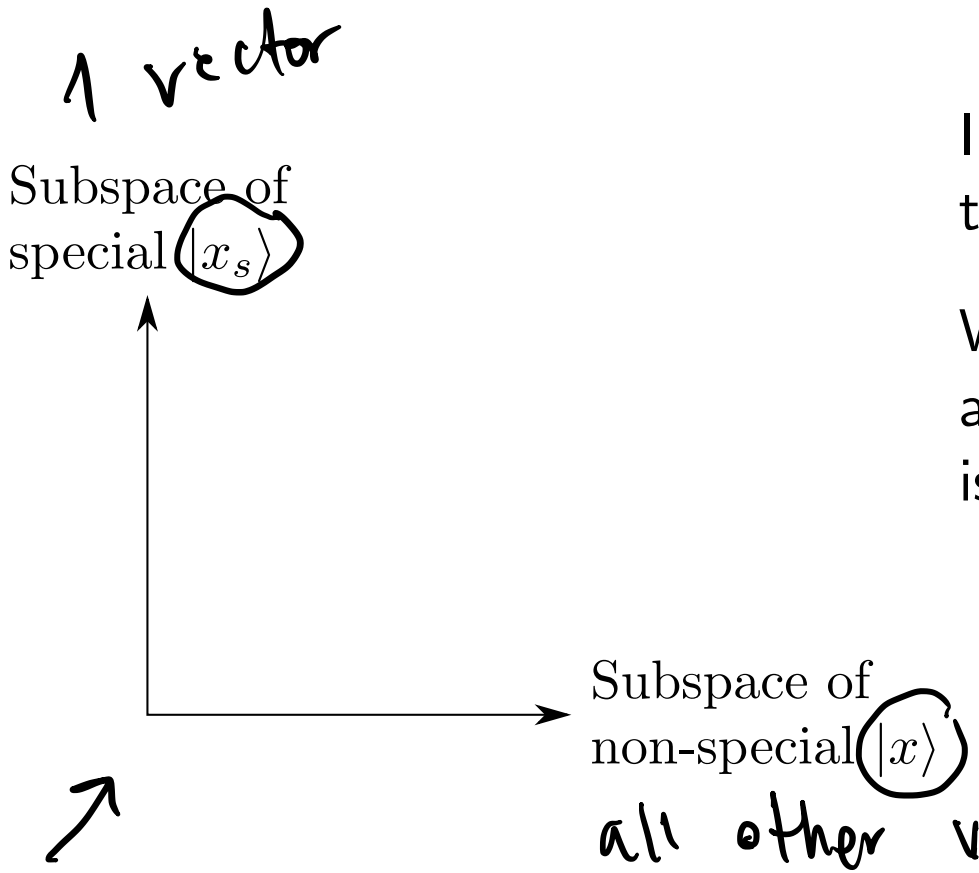
I find it is easiest to understand this in a visual way.

We can partition the subspace of all 2^n computational basis states is two parts:

1. The special state(s) $|x_s\rangle$

Note: I will show the case where there is only one special element, but the same ideas work when there are more.

Grover's algorithm visually



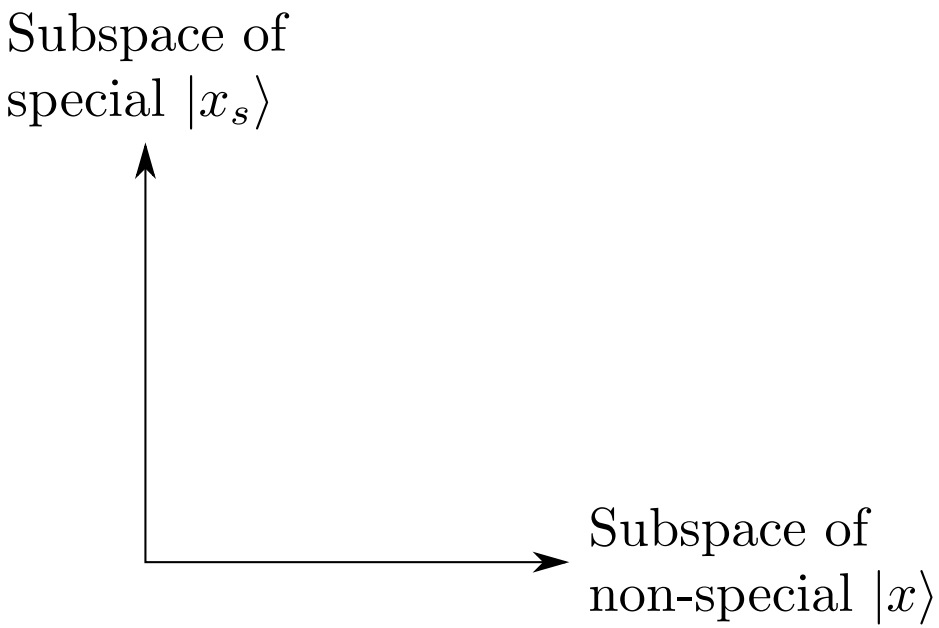
I find it is easiest to understand this in a visual way.

We can partition the subspace of all 2^n computational basis states is two parts:

1. The special state(s) $|x_s\rangle$
2. All the other states

Note: I will show the case where there is only one special element, but the same ideas work when there are more.

Grover's algorithm visually

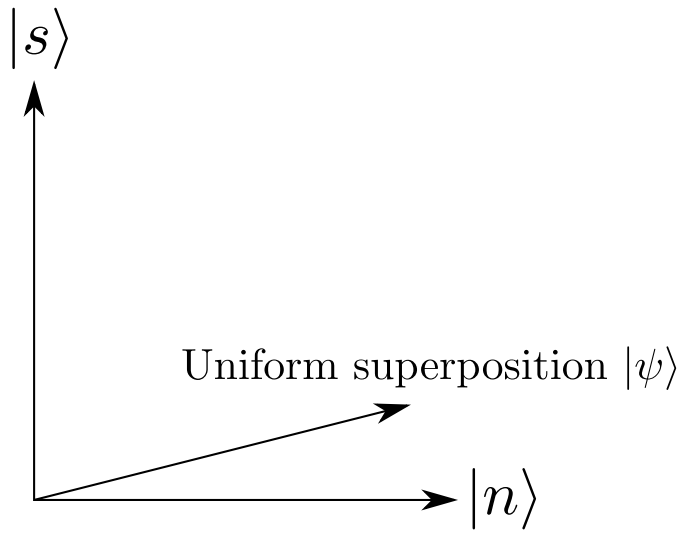


Let's represent special by $|s\rangle$ and non-special by $|n\rangle$ and write these out as superpositions:

$$|s\rangle = |x_s\rangle$$

$$|n\rangle = \frac{1}{\sqrt{2^n - 1}} \sum_{x \neq x_s} |x\rangle$$

Grover's algorithm visually



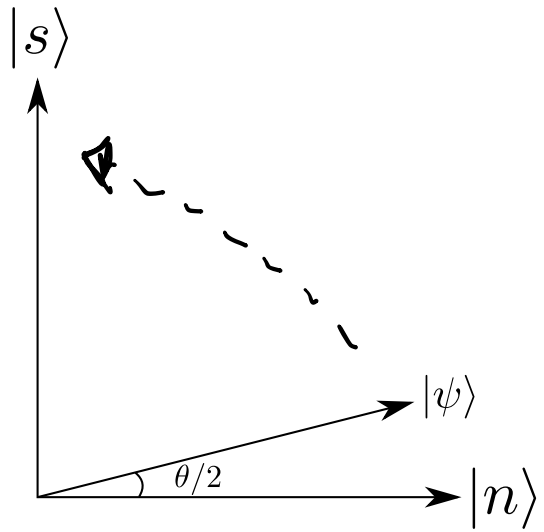
Let's represent special by $|s\rangle$ and non-special by $|n\rangle$ and write these out as superpositions:

$$|s\rangle = |x_s\rangle$$
$$|n\rangle = \frac{1}{\sqrt{2^n - 1}} \sum_{x \neq x_s} |x\rangle$$

We can write the uniform superposition in terms of these subspaces:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}}|s\rangle + \frac{\sqrt{2^n - 1}}{\sqrt{2^n}}|n\rangle$$
$$= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

Grover's algorithm visually



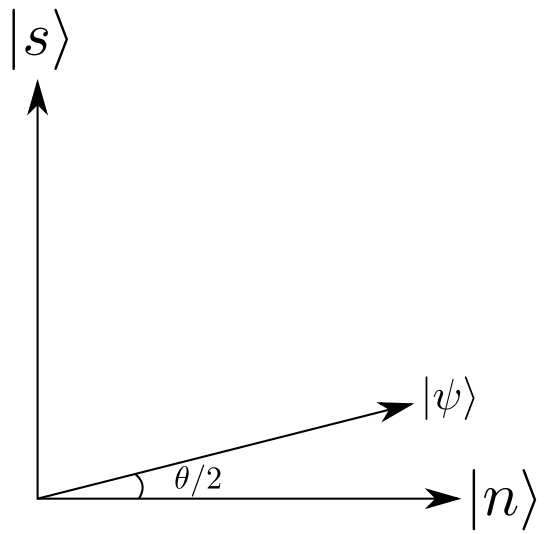
Instead of working with these complicated coeffients:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}}|s\rangle + \frac{\sqrt{2^n-1}}{\sqrt{2^n}}|n\rangle,$$

let's rexpress them in terms of an angle θ:

$$|\psi\rangle = \sin\left(\frac{\theta}{2}\right)|s\rangle + \cos\left(\frac{\theta}{2}\right)|n\rangle$$

Grover's algorithm visually

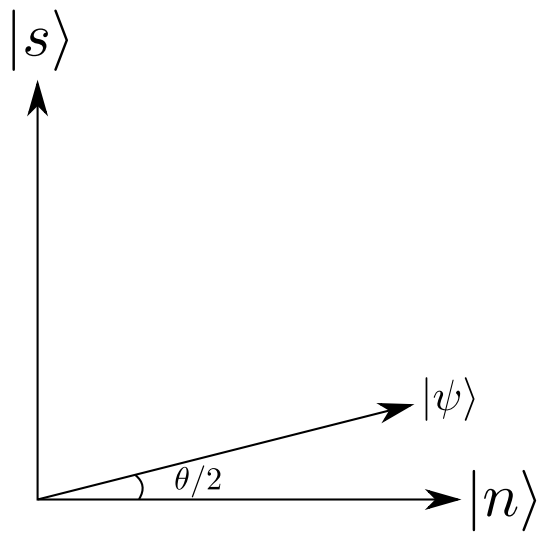


Now we want to apply some operations to this state

$$|\psi\rangle = \sin\left(\frac{\theta}{2}\right)|s\rangle + \cos\left(\frac{\theta}{2}\right)|n\rangle$$

in order to increase the amplitude of $|s\rangle$ while decreasing the amplitude of $|n\rangle$.

Grover's algorithm visually



We will do this in two steps:

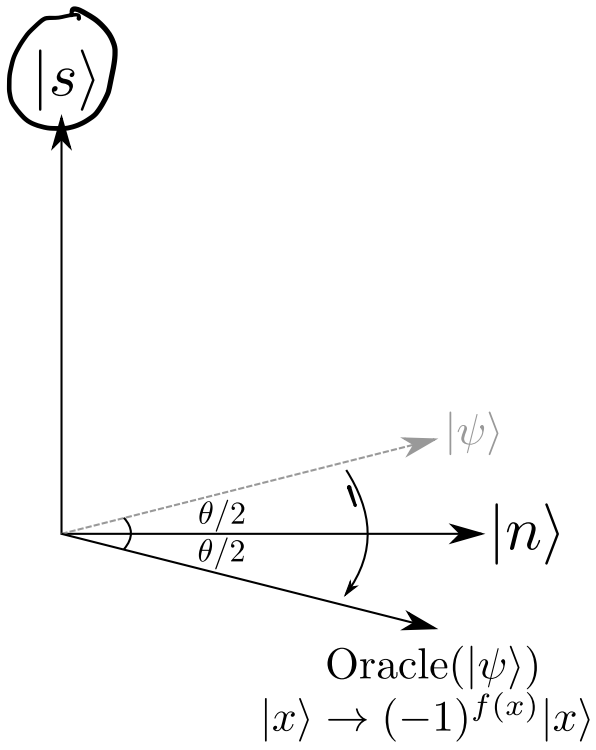
1. Apply the oracle O to 'pick out' the solutions; recall from yesterday's discussion about Deutsch's algorithm that the oracle will send

$$|x\rangle|-\rangle \rightarrow (-1)^{f(x)}|x\rangle|-\rangle \tag{14}$$

2. Apply a 'diffusion operator' D that will adjust the amplitudes.

We will call the product $DO = G$ the *Grover iteration*. Grover's algorithm consists of repeating this procedure many times.

Grover's algorithm visually



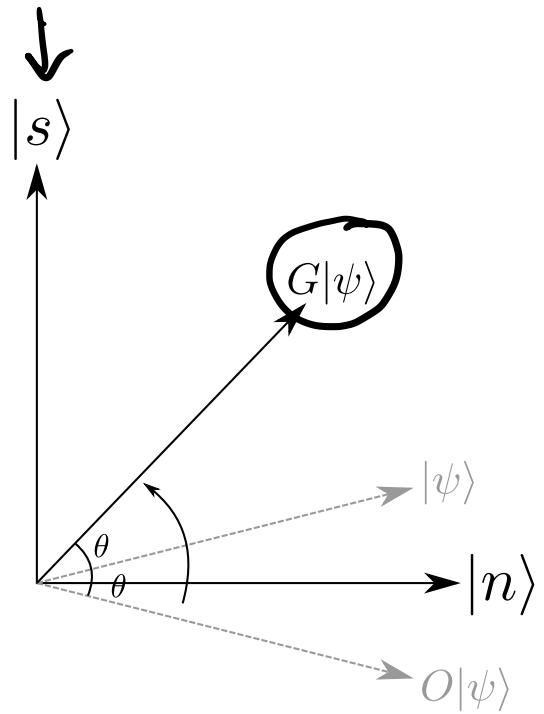
$|x\rangle \to (-1)^{f(x)}|x\rangle$

not special: $|x\rangle = |x\rangle$

The effect of the oracle, $O|\psi\rangle$ *flips* the amplitudes of the basis states that are special.

We can visualize this as a *reflection about the subspace* of non-special elements.

Grover's algorithm visually



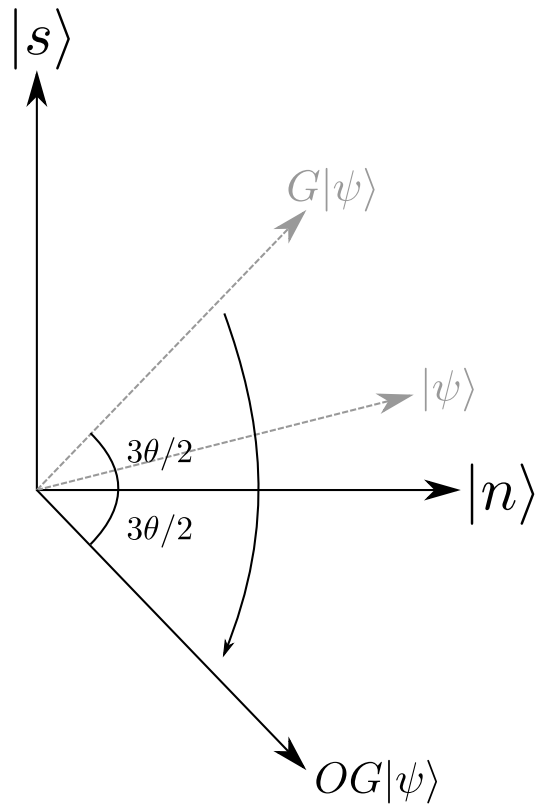
The diffusion operator is a bit less intuitive to interpret* - it performs a *reflection about the uniform superposition* state.

A full Grover iteration $G = DO$ sends

$$G\left(\sin\left(\frac{\theta}{2}\right)|s\rangle + \cos\left(\frac{\theta}{2}\right)|n\rangle\right) = \sin\left(\frac{3\theta}{2}\right)|s\rangle + \cos\left(\frac{3\theta}{2}\right)|n\rangle$$

*Mathematically it looks like $D = 2|+\rangle^{\otimes n}\langle+|^{\otimes n} - \mathbb{1}$. Track me down if you want to see the math.

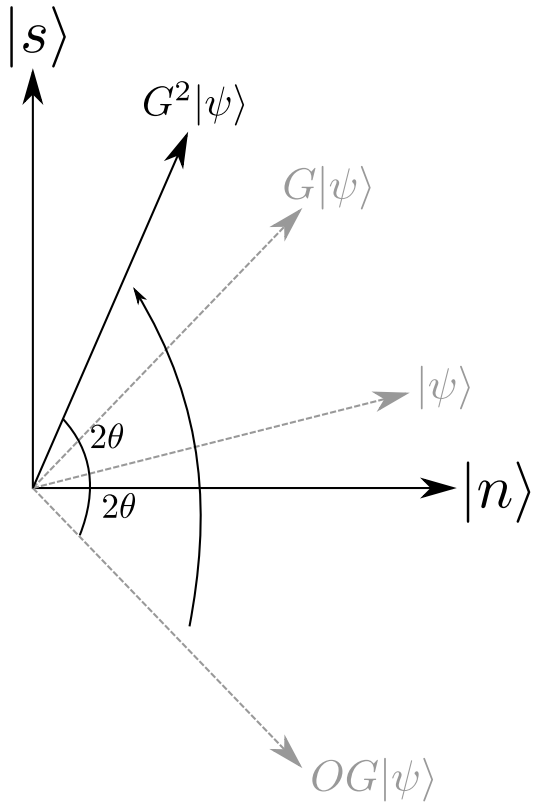
Grover's algorithm visually



Now we repeat this...

Apply the oracle and reflect about the non-special elements.

Grover's algorithm visually



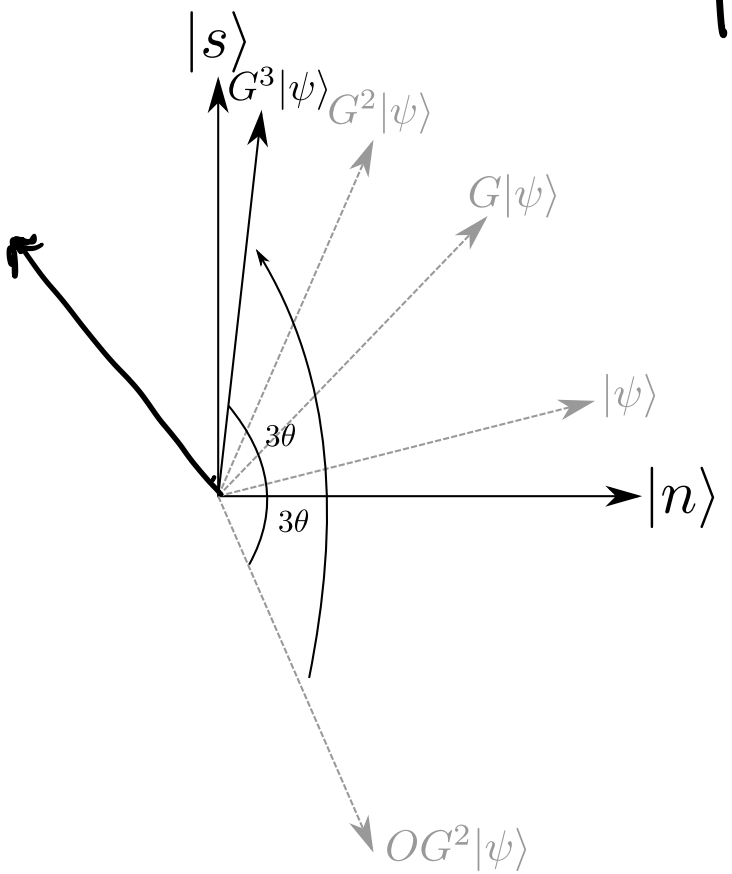
Apply the diffusion operator and reflect about the uniform superposition to boost the amplitude of the special state.

$$\sin\left(\frac{5\theta}{2}\right)|s\rangle + \cos\left(\frac{5\theta}{2}\right)|n\rangle$$

Grover's algorithm visually

After k Grover iterations we will have the state

$$G^k|\psi\rangle = \sin\left(\frac{(2k+1)\theta}{2}\right)|s\rangle + \cos\left(\frac{(2k+1)\theta}{2}\right)|n\rangle$$



It *is* possible to over-rotate! We can differentiate to find the optimal k:

$$k \leq \left\lceil \frac{\pi}{4}\sqrt{2^n} \right\rceil \qquad (15)$$

After k operations we will be most likely to obtain the special state as our measurement outcome.

Performance of Grover's algorithm

Recall that to solve the classical problem we needed to make 2^n oracle queries.

Using Grover's algorithm, we only need to make on the order of $\sqrt{2^n}$ queries!

Grover's algorithm provides a *polynomial speedup*, or *square-root* speedup over classical algorithms for searching unsorted spaces.

Quantum computing in practice: Quantum advantage

What can we do with a quantum computer?

There are many potential uses for quantum hardware other than Grover search:

- Use the quantum Fourier transform to solve problems in discrete mathematics (discrete logarithm, order-finding, factoring)
- Random number generation
- Quantum key distribution and cryptography for secure quantum communication
- Speeding up linear algebra operations; as 'accelerators' for machine learning algorithms
- *Tons of things we haven't even thought of yet!* Designing quantum algorithms is really hard.

Will quantum computers be better at everything?

Not every quantum algorithm provides the desired 'superpolynomial' speedup over classical computers.

There will still be problems that even quantum computers can't solve efficiently.[2]

So, more important question: what kinds of problems are complex enough to make it worth pulling out a quantum computer in the first place?

[2] For those of you who are interested in computational complexity, check out the complete problems in the complexity class QMA `https://arxiv.org/abs/1212.6312`.

Quantum advantage

At what point will quantum computers be able to do something that is intractable for a classical computer? How large of a problem, or how many qubits, do we need before we see a *quantum advantage*?[3]

Need things that are exponentially hard on a classical computer but that quantum computers can solve efficiently.

This is a moving, problem-dependent target! A few years ago, people were estimating we would need around 50 qubits for this, but classical simulators have caught up.

[3] See: https://medium.com/@wjzeng/clarifying-quantum-supremacy-better-terms-for-milestones-in-quantum-computation-d15ccb53954f

Candidate problem: sampling random circuits

Given a random circuit, sample from the probability distribution given by its output.

Classical:
Do the matrix multiplication to work through the entire circuit, get the final amplitudes, then sample. Exponentially hard!

Quantum:
Run the circuit many times and keep track of the distribution of measured outputs.

How large a circuit do we need before we can no longer achieve this with classical computers?

Candidate problem: sampling random circuits

Quantum Physics

Quantum Supremacy Circuit Simulation on Sunway TaihuLight

Riling Li, Bujiao Wu, Mingsheng Ying, Xiaoming Sun, Guangwen Yang

(Submitted on 13 Apr 2018 (v1), last revised 13 Aug 2018 (this version, v3))

With the rapid progress made by industry and academia, quantum computers with dozens of qubits or even larger size are being realized. However, the fidelity of existing quantum computers often sharply decreases as the circuit depth increases. Thus, an ideal quantum circuit simulator on classical computers, especially on high-performance computers, is needed for benchmarking and validation. We design a large-scale simulator of universal random quantum circuits, often called 'quantum supremacy circuits', and implement it on Sunway TaihuLight. The simulator can be used to accomplish the following two tasks: 1) Computing a complete output state-vector; 2) Calculating one or a few amplitudes. We target the simulation of 49-qubit circuits. For task 1), we successfully simulate such a circuit of depth 39, and for task 2) we reach the 55-depth level. To the best of our knowledge, both of the simulation results reach the largest depth for 49-qubit quantum supremacy circuits.

(When this paper was written, the TaihuLight was the top supercomputer in the world.)

Candidate problem: sampling random circuits

Quantum Physics

Massively parallel quantum computer simulator, eleven years later

Hans De Raedt, Fengping Jin, Dennis Willsch, Madita Nocon, Naoki Yoshioka, Nobuyasu Ito, Shengjun Yuan, Kristel Michielsen

(Submitted on 12 May 2018 (v1), last revised 11 Dec 2018 (this version, v2))

A revised version of the massively parallel simulator of a universal quantum computer, described in this journal eleven years ago, is used to benchmark various gate-based quantum algorithms on some of the most powerful supercomputers that exist today. Adaptive encoding of the wave function reduces the memory requirement by a factor of eight, making it possible to simulate universal quantum computers with up to 48 qubits on the Sunway TaihuLight and on the K computer. The simulator exhibits close-to-ideal weak-scaling behavior on the Sunway TaihuLight,on the K computer, on an IBM Blue Gene/Q, and on Intel Xeon based clusters, implying that the combination of parallelization and hardware can track the exponential scaling due to the increasing number of qubits. Results of executing simple quantum circuits and Shor's factorization algorithm on quantum computers containing up to 48 qubits are presented.

Candidate problem: sampling random circuits

Quantum Physics

Quantum Supremacy Is Both Closer and Farther than It Appears

Igor L. Markov, Aneeqa Fatima, Sergei V. Isakov, Sergio Boixo

(Submitted on 27 Jul 2018 (v1), last revised 26 Sep 2018 (this version, v3))

As quantum computers improve in the number of qubits and fidelity, the question of when they surpass state-of-the-art classical computation for a well-defined computational task is attracting much attention. The leading candidate task for this milestone entails sampling from the output distribution defined by a random quantum circuit. We develop a massively-parallel simulation tool Rollright that does not require inter-process communication (IPC) or proprietary hardware. We also develop two ways to trade circuit fidelity for computational speedups, so as to match the fidelity of a given quantum computer --- a task previously thought impossible. We report massive speedups for the sampling task over prior software from Microsoft, IBM, Alibaba and Google, as well as supercomputer and GPU-based simulations. By using publicly available Google Cloud Computing, we price such simulations and enable comparisons by total cost across hardware platforms. We simulate approximate sampling from the output of a circuit with 7x8 qubits and depth 1+40+1 by producing one million bitstring probabilities with fidelity 0.5%, at an estimated cost of $35184. The simulation costs scale linearly with fidelity, and using this scaling we estimate that extending circuit depth to 1+48+1 increases costs to one million dollars. Scaling the simulation to 10M bitstring probabilities needed for sampling 1M bitstrings helps comparing simulation to quantum computers. We describe refinements in benchmarks that slow down leading simulators, halving the circuit depth that can be simulated within the same time.

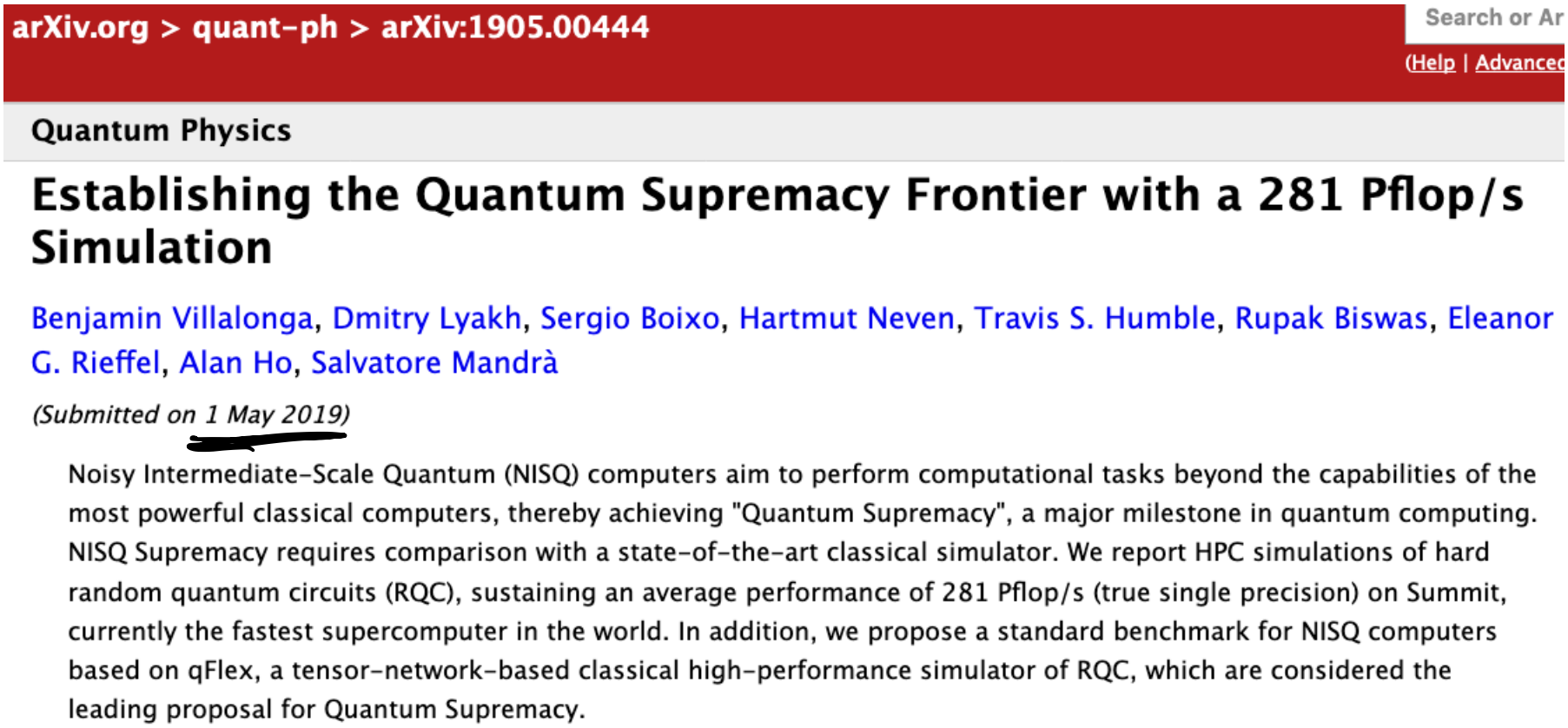
Quantum advantage

There has yet to be a concrete demonstration of quantum advantage. How close are we?

Google and NASA signed an agreement last July saying they would prove it in 12 months...

Quantum simulation on Summit

Instead they have just pushed forward the frontier.



arXiv.org > quant-ph > arXiv:1905.00444

Search or Ar

(Help | Advanced

Quantum Physics

Establishing the Quantum Supremacy Frontier with a 281 Pflop/s Simulation

Benjamin Villalonga, Dmitry Lyakh, Sergio Boixo, Hartmut Neven, Travis S. Humble, Rupak Biswas, Eleanor G. Rieffel, Alan Ho, Salvatore Mandrà

(Submitted on 1 May 2019)

Noisy Intermediate-Scale Quantum (NISQ) computers aim to perform computational tasks beyond the capabilities of the most powerful classical computers, thereby achieving "Quantum Supremacy", a major milestone in quantum computing. NISQ Supremacy requires comparison with a state-of-the-art classical simulator. We report HPC simulations of hard random quantum circuits (RQC), sustaining an average performance of 281 Pflop/s (true single precision) on Summit, currently the fastest supercomputer in the world. In addition, we propose a standard benchmark for NISQ computers based on qFlex, a tensor-network-based classical high-performance simulator of RQC, which are considered the leading proposal for Quantum Supremacy.

Summit is currently the top supercomputer....

Quantum simulation on Summit

They used the whole thing!!!

49 ; depth 42

Circuit Size	Nodes Used	Runtime (h)	PFlop/s		Efficiency (%)	
			Peak	Sust.	Peak	Sust.
$7 \times 7 \times (1 + 40 + 1)$	2300	4.84	191	142	92.0	68.5
$7 \times 7 \times (1 + 40 + 1)$	4600	2.44	381	281	92.1	68.0
$11 \times 11 \times (1 + 24 + 1)$	4550	0.278	368	261	89.8	63.7

TABLE I: Performance of the simulation of our three runs on Summit. For the

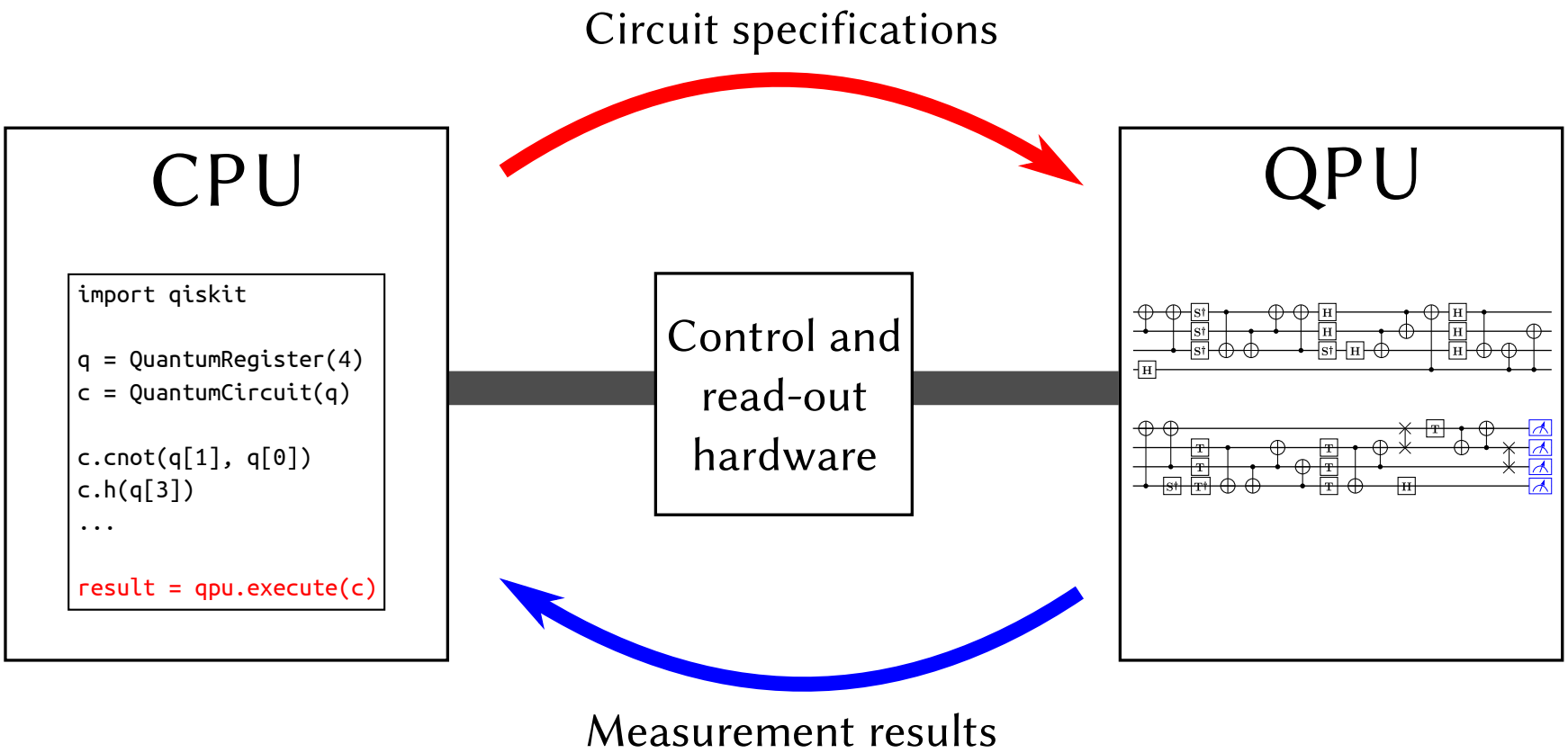
Simulated circuits for qubits in a 7×7 grid with depth 42, and 11×11 grid with depth 26.

Image credit: http://arxiv.org/abs/1905.00444

Quantum computing in practice: Hardware and the NISQ era

How do I use a quantum computer?

Now, and likely in the future, quantum computers are being used like special-purpose *accelerators* (think how GPUs are used today).



How do we build a *quantum processing unit*, or QPU?

The DiVincenzo criteria

1. A *scalable* physical system with well-characterized qubits.
2. The ability to *initialize* the state of the qubits to a simple fiducial state.
3. Long relevant decoherence times, much longer than the gate operation times.
4. A *universal* set of quantum gates.
5. A qubit-specific *measurement* capability.

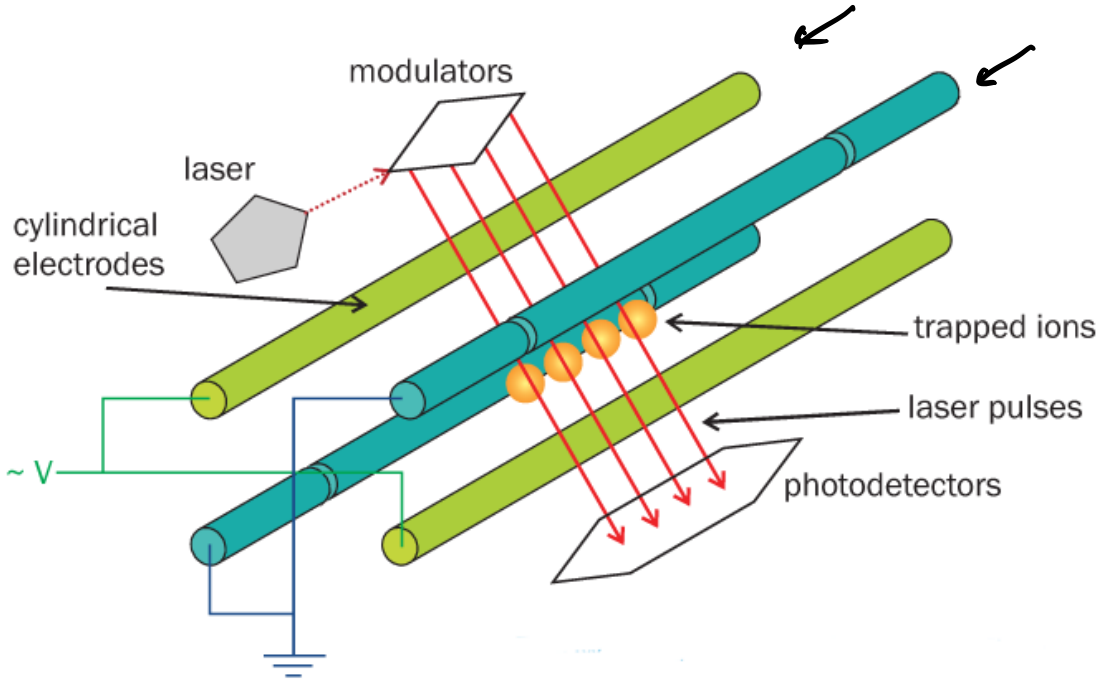
Candidate systems for qubits

- Superconducting qubits
- Trapped ions
- Spin qubits (neutral atoms, diamond NV centres, silicon spins qubits)
- Photons – quantum communication
- Topological qubits
- ... many more.

Superconducting qubits and trapped ions are currently the most well-developed.

Candidate systems for qubits: Trapped ions

Qubits are ground state hyperfine levels of ions suspended between electrodes.



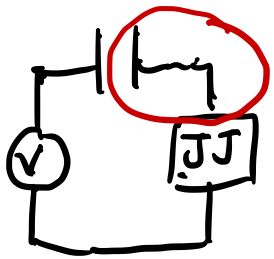
Qubits are individually addressable by applying EM fields of varying frequency and duration. Entangling gates are applied by coupling with a common vibrational mode of the whole chain.

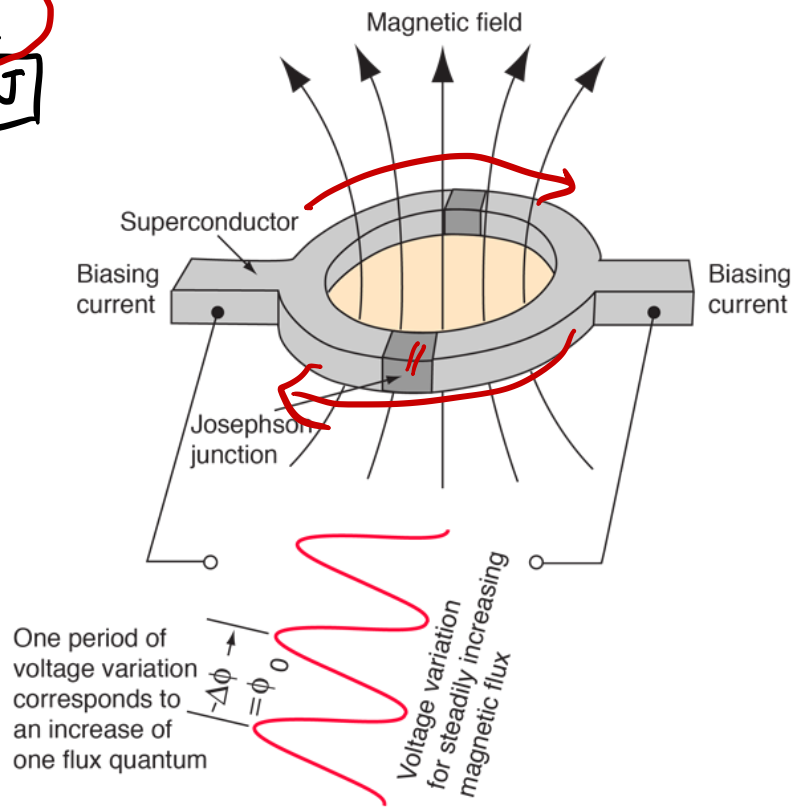
Image credit: Wikimedia commons

Candidate systems for qubits: superconducting qubits

Multiple ways to make qubits:

- Charge qubits: uses the number of Cooper pairs sitting between a capacitor and a Josephson junction in an electrical circuit
- Flux qubits: uses the direction of magnetic flux induced by current in a superconducting loop



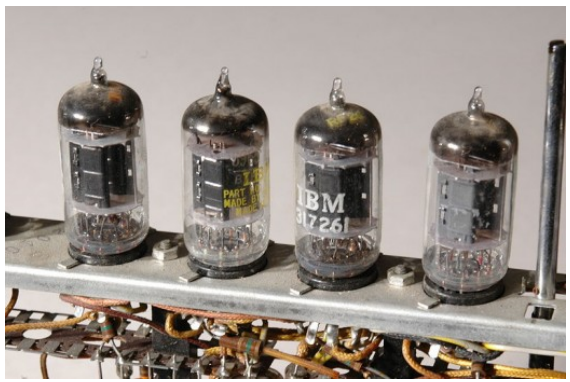


Qubits are individually addressable by applying microwave pulses; qubits are coupled via electric circuit connections.

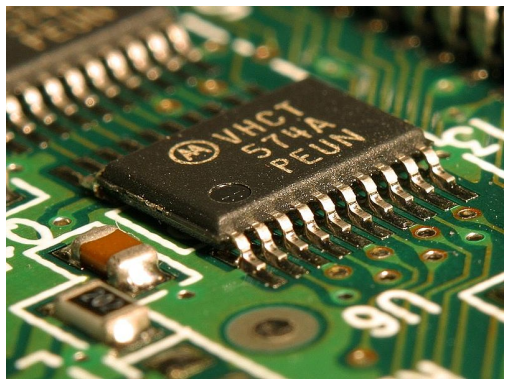
Image credit: `http://hyperphysics.phy-astr.gsu.edu/hbase/Solids/Squid.htm`

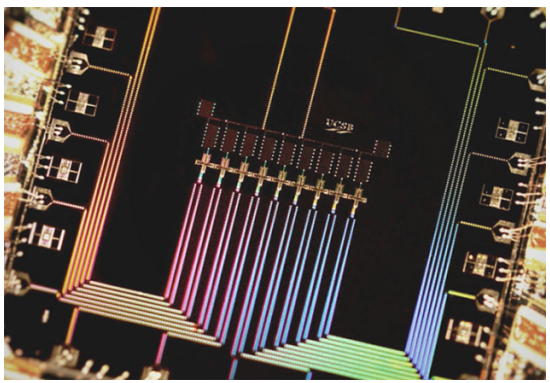
Candidate systems for qubits

However, it's important to remember that the "real" qubit of the future might not have even been invented yet!



vs.





vs.

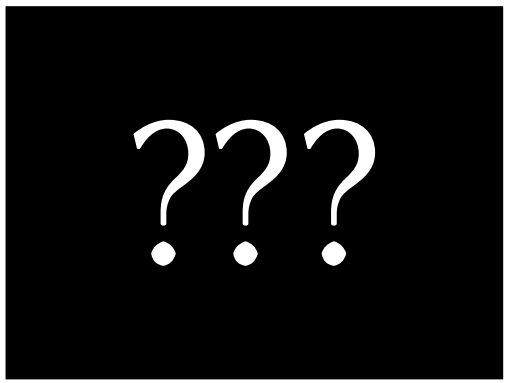


Image credits:
https://hubpages.com/business/What-Is-a-Transistor-and-Why-is-it-Important
https://en.wikipedia.org/wiki/Solid-state_electronics
https://physicsworld.com/a/google-gains-new-ground-on-universal-quantum-computer/

NISQ-era quantum computing

We are in the era of 'Noisy, intermediate-scale quantum', or **NISQ** devices: medium-sized machines, but they have a number of limitations.

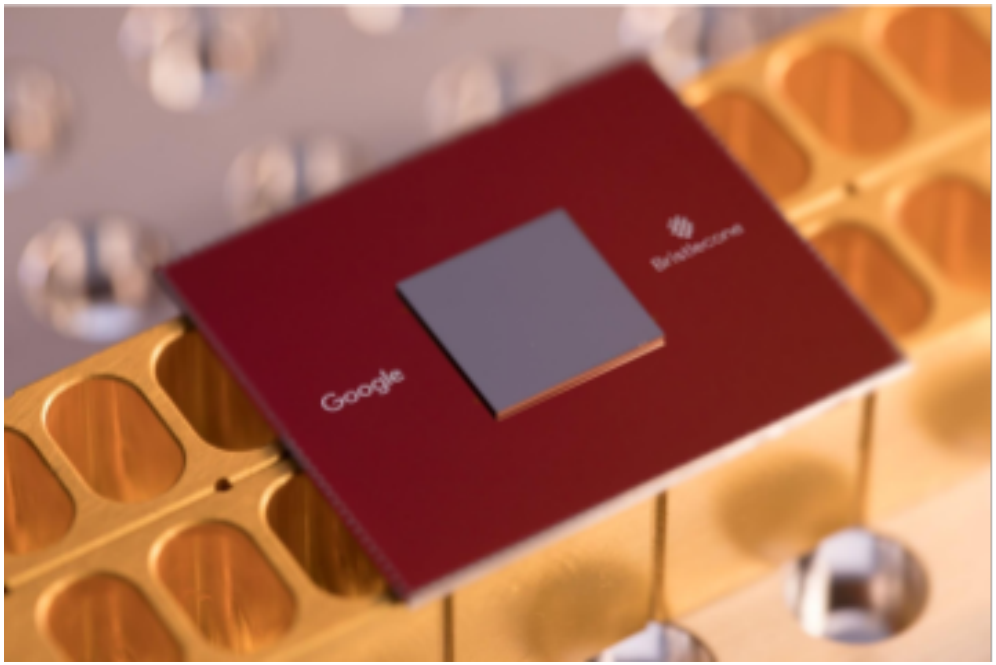
- Qubit count
- Qubit connectivity
- Error rates, coherence times

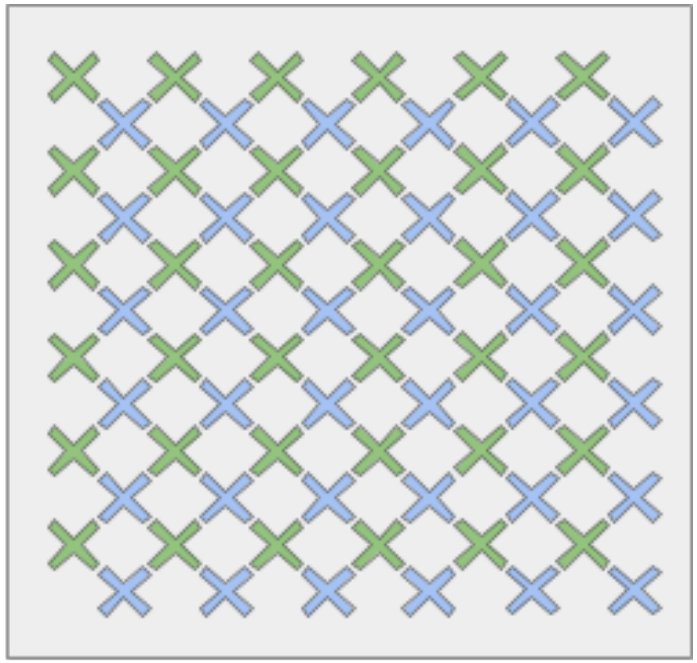
Useful site:
`https://quantumcomputingreport.com/scorecards/qubit-count/`

Important note: things are changing fast; no guarantees any of these numbers will be the same a month from now.

Google

Currently Google's *Bristlecone* is the largest device. It has 72 qubits arranged in a grid pattern.

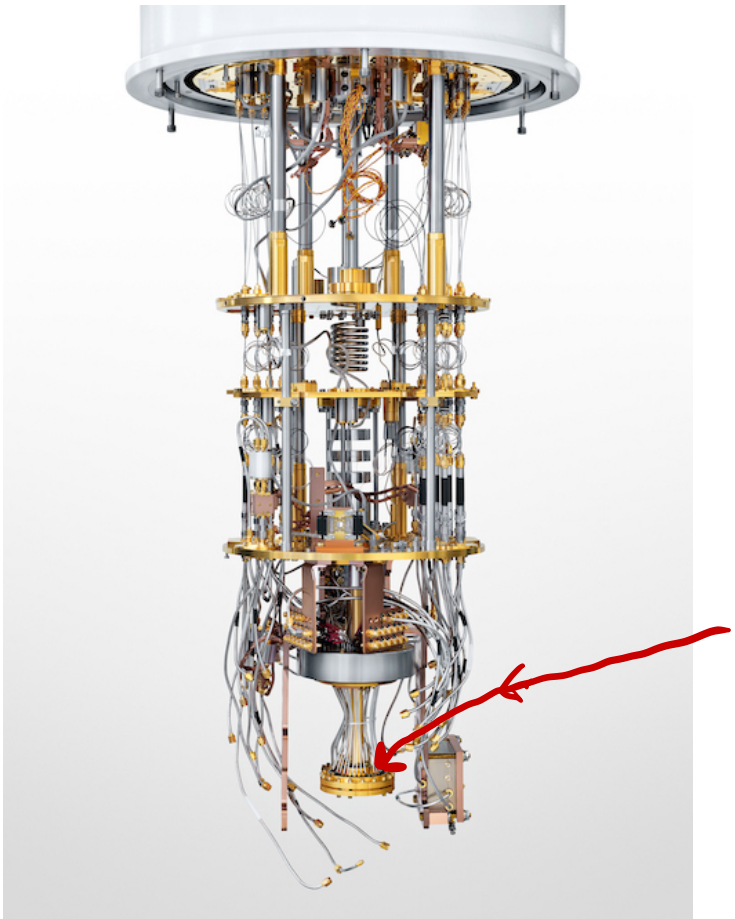




https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html

Rigetti

A California-based startup. Full-stack (hardware + software), accessible via the cloud.

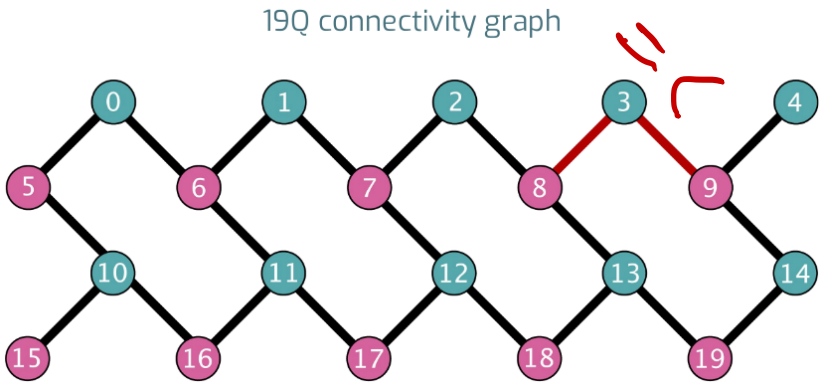


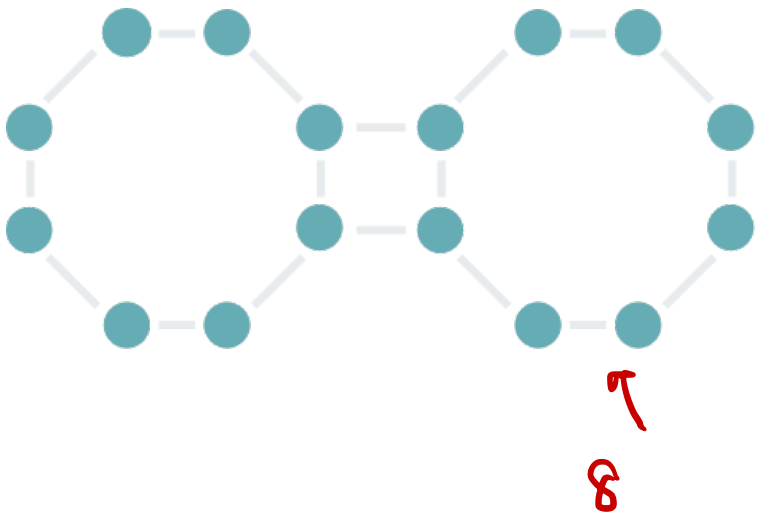
"Image courtesy Rigetti Computing. Photo by Justin Fantl."

Hardware graph - Rigetti

19-qubit chip in use (20 on hardware, but one is 'out of spec')

128-qubit chip in development; 8 x 16 of the unit cell below.





https://medium.com/rigetti/the-rigetti-128-qubit-chip-and-what-it-means-for-quantum-df757d1b71ea
https://hpcuserforum.com/presentations/tuscon2018/QCOverview_Rigetti_UFTucson2018.pdf

IBM

Full-stack, available in the cloud.

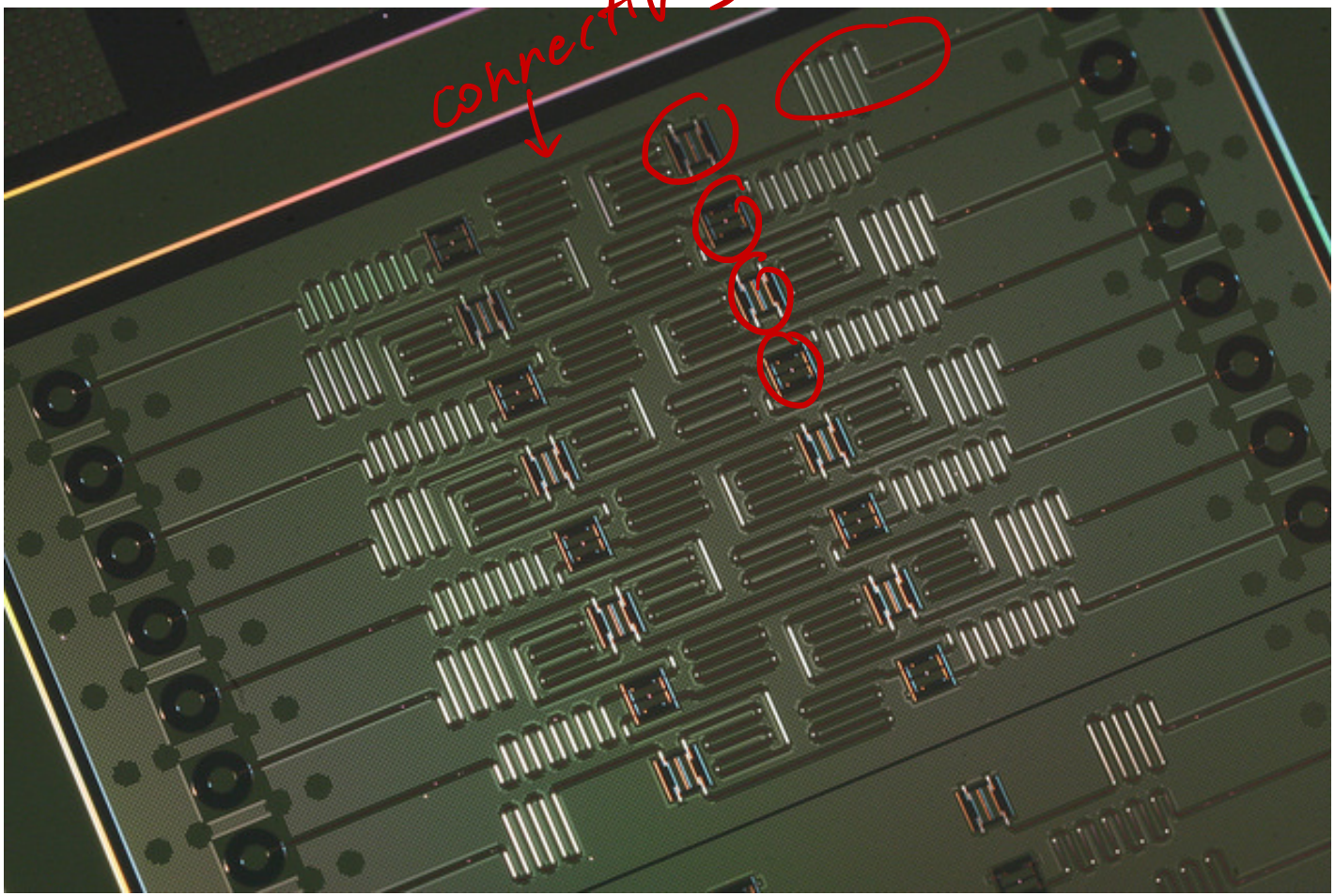
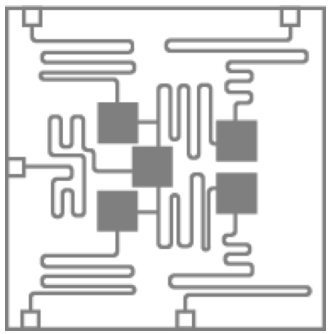
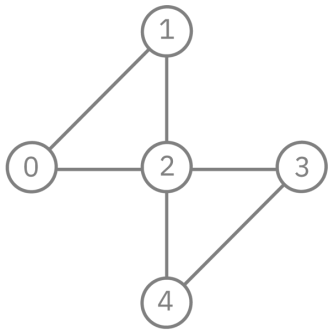


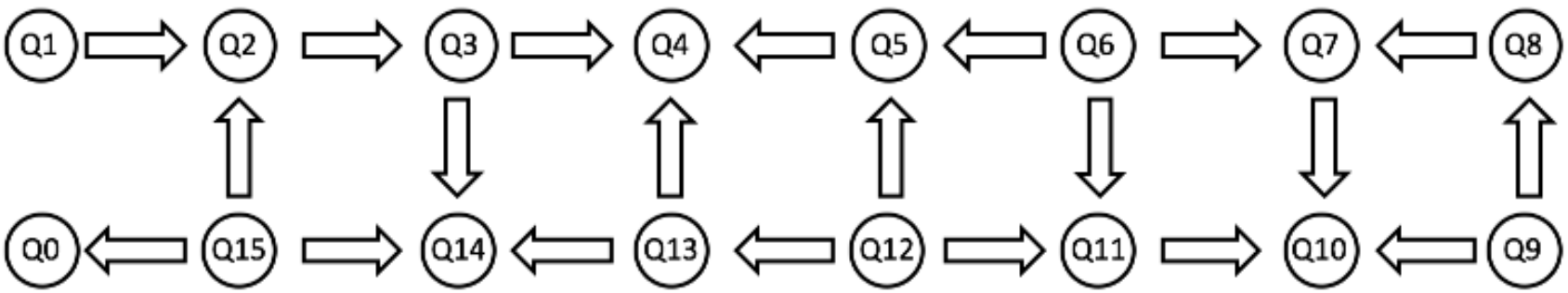
Image credit: `https://www.ibm.com/blogs/research/2018/09/next-gen-qubits/`

Hardware graph - IBM

Small machines





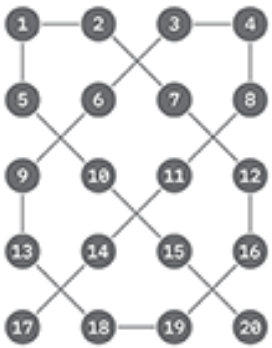


https://www.research.ibm.com/ibm-q/technology/devices/

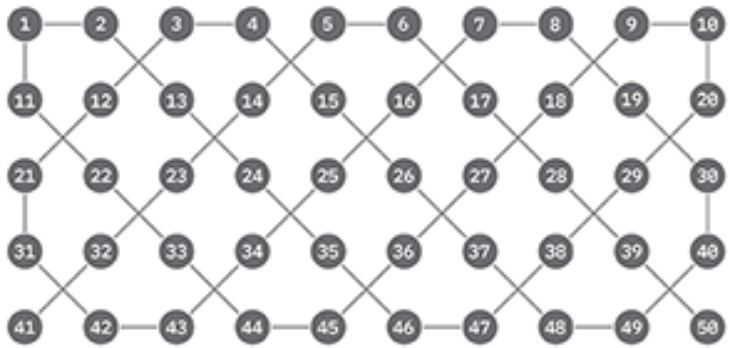
Hardware graph - IBM

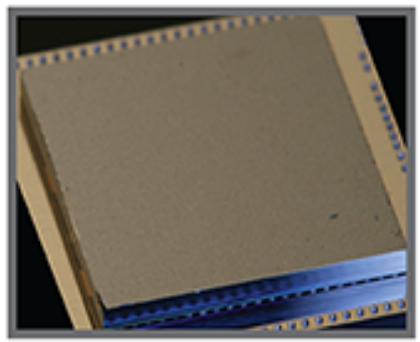
Larger machines

20



50



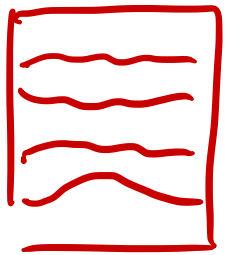


https://www.ibm.com/blogs/research/2017/11/the-future-is-quantum/

Qubit counts

Other players (non-exhaustive):

- Superconducting: Intel (49), U. of Science and Technology China (10), Alibaba (11)
- Ion traps: IonQ (11)[4], Inst. for Quantum Optics and Quantum Information (20)
- Spin qubits[5]: Intel (26)
- Neutral atoms: U. Wisconsin (49)
- Photonic computing: Xanadu, PsiQuantum
- Topological qubits: Microsoft

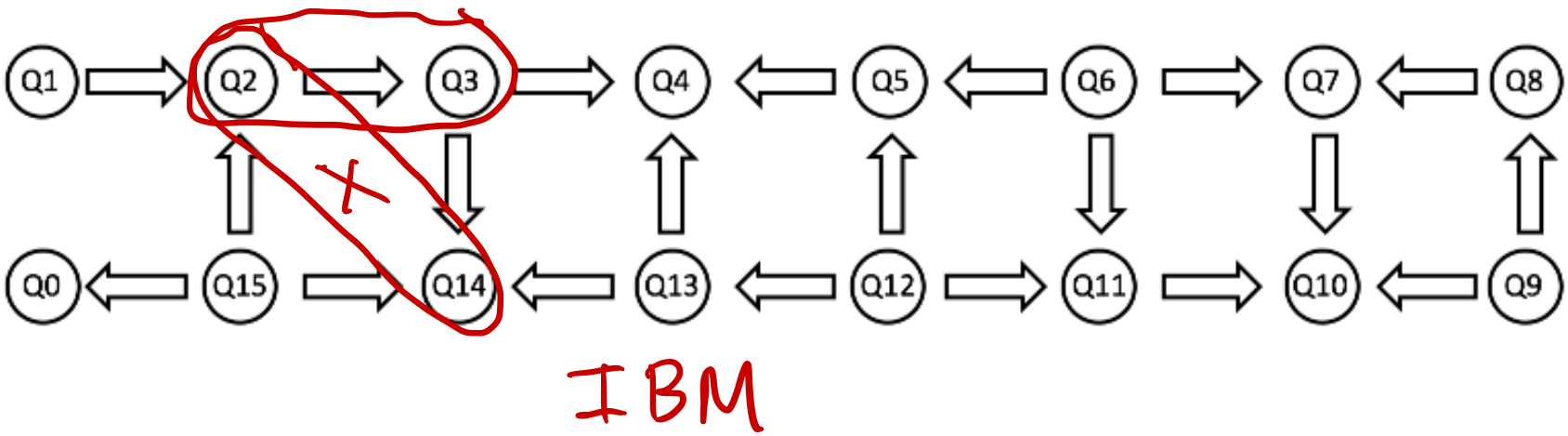


[4] 160 atoms, single-qubit gates on 79, two-qubit gates on all pairs of 11

[5] Being worked on at UBC and SFU!

Qubit connectivity

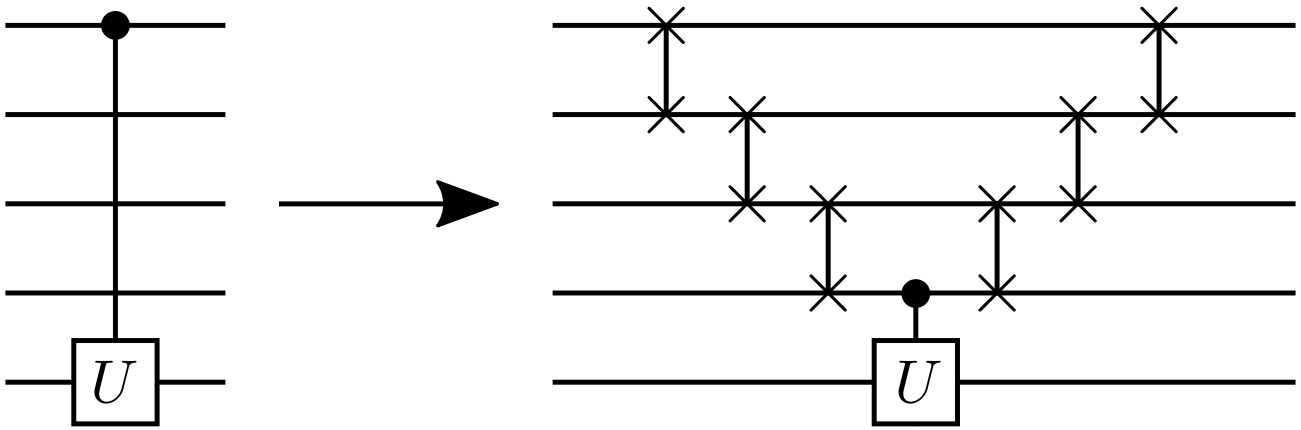
Look closely at the structure of this graph:



The arrows indicate where, and in what direction we can perform CNOT gates. What happens when we need to apply a CNOT on qubits that are far apart?

Qubit connectivity

Naive solution: when you need to operate on two non-adjacent qubits, perform SWAP gates until they are adjacent, perform the operation, then undo all the SWAPs.



Better solution: Heuristic techniques for compiling down to circuits over the universal gate set that fit the topology. For example, 're-allocation' of qubit indexing to minimize the number of SWAPs.

Qubit allocation

Noise

This is the 20-qubit IBM Q System One. It's in a dilution refrigerator cooled to $\sim 10 - 20$mK, suspended in a 9-foot cube.

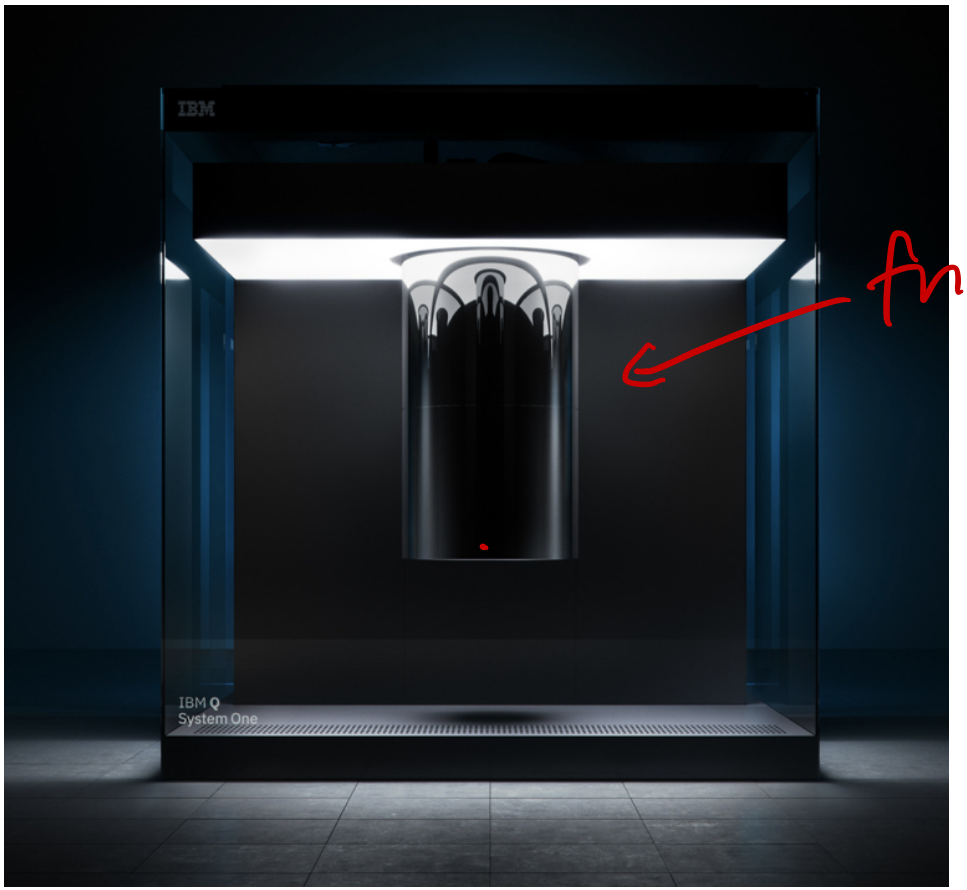


Image credit:
`https://www.hpcwire.com/2019/01/10/ibm-quantum-update-q-system-one-launch-new-collaborators-and-qc-center-plans/`

Quantifying noise in quantum computing

Why so cold? Energy levels are very close together; need thermal excitations to be small enough that they don't cause transition between the qubit states.

Quantifying noise in quantum computing

Why so cold? Energy levels are very close together; need thermal excitations to be small enough that they don't cause transition between the qubit states.

Any amount of interaction with the outside world can cause *decoherence* of our quantum system.

Quantifying noise in quantum computing

Why so cold? Energy levels are very close together; need thermal excitations to be small enough that they don't cause transition between the qubit states.

Any amount of interaction with the outside world can cause *decoherence* of our quantum system.

We judge the quality of our qubits using a variety of metrics, for example:

- Gate fidelity (1- and 2-qubit gates)
- Spin relaxation time (time it takes for $|1\rangle$ to 'relax' to $|0\rangle$)
- Decoherence time

See the 'bonus' slides for more details, and an example calculation of fidelity.

fidelity

NISQ-device qubit quality

'Live' machines:

Org.	Machine	1-qubit fid.	2-qubit fid.
IBM	Q System One	99.96	98.31
Rigetti	19Q Acorn	98.63	87.50

Other news:

- 2018: Silicon qubits with gate fidelity 99.96 (`https://arxiv.org/abs/1807.09500`)
- 2018: IonQ reports single-qubit fidelities of $> 99\%$, two-qubit of $> 98\%$ (`https://ionq.co/news/december-11-2018#appendix`)
- APS March Meeting 2019: Google and Rigetti claim $> 99\%$ fidelity two-qubit gates

For relatively up-to-date numbers:
`https://quantumcomputingreport.com/scorecards/qubit-quality/`

NISQ-device coherence times

All times are averages:

Org.	**Machine**	$\mathbf{T_1}$ (μ s)	$\mathbf{T_2}$ (μ s)
IBM	Q System One	73.9	69.1
Rigetti	19Q Acorn	20.3	10.9
IonQ	11 Qubit	$> 10^{10}$	$\sim 3 \cdot 10^6$

T_1 is the relaxation time, T_2 is the decoherence time.

Other news:

- 2018: Silicon qubits have seen $T_2 \approx 9.4ms$
- 2018: IonQ reports $T_1 \approx 10^{10}\mu$s, $T_2 \approx 10^6\mu$s
- 2019: IBM reports lifetimes up to 500μs lifetime for individual superconducting qubits (`https://www.ibm.com/blogs/research/2019/03/power-quantum-device/`)
- 3 days ago: TU Delft reports lifetimes up to 63s in diamond NV-center qubit `http://arxiv.org/abs/1905.02094`

Prospects for NISQ devices

We're getting there.

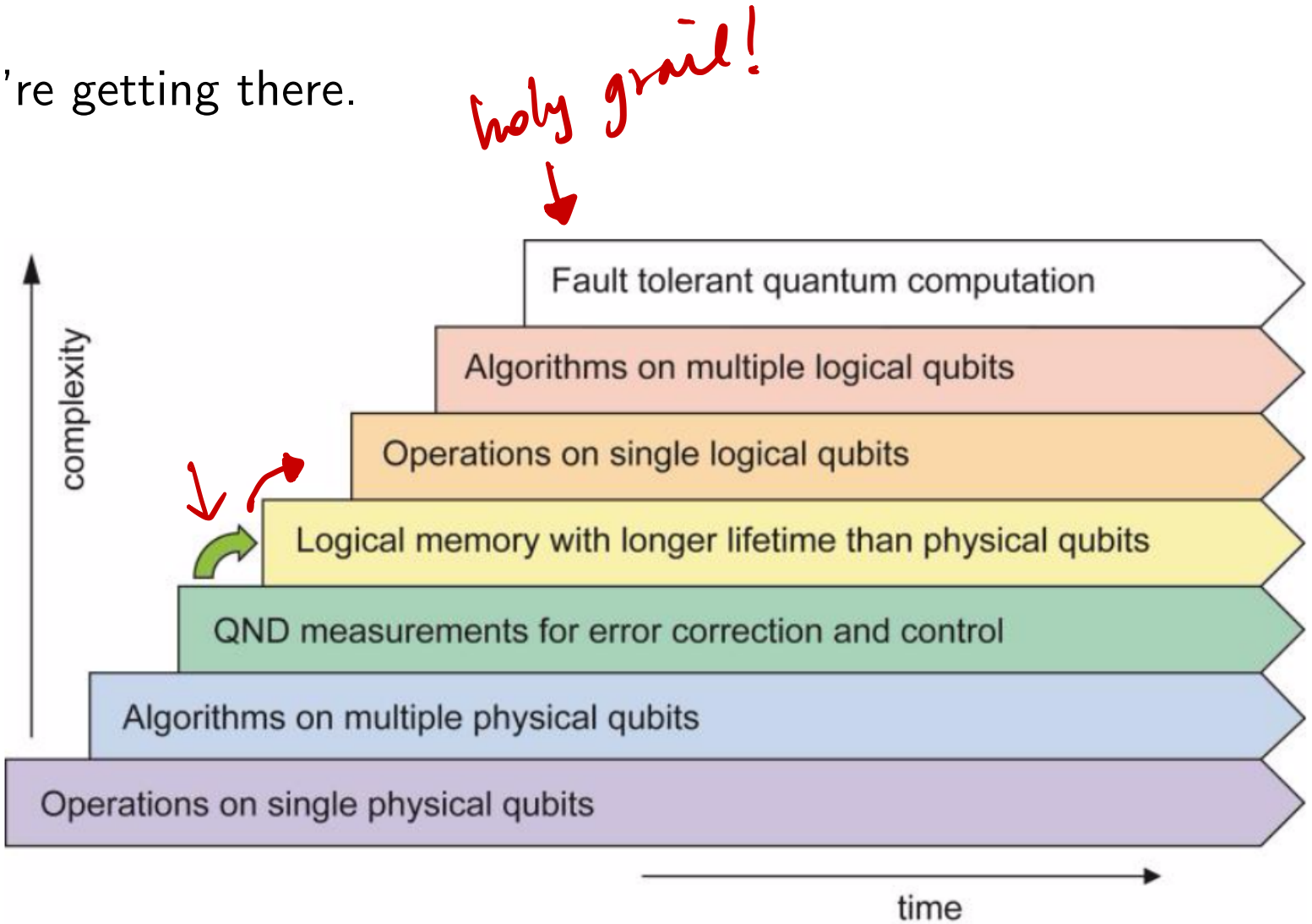


Image credit: 'Superconducting circuits for quantum information: An Outlook', Devoret and Schoelkopf, 2013

Prospects for NISQ devices

Today's qubits are too noisy to run algorithms 'at scale'.

However, there is a well-developed theory of *quantum error correction* and *fault-tolerance*. Multiple physical qubits can 'work together' in an error-correcting code as a single 'logical qubit'.

The problem is that there is typically an overhead of *thousands* of physical qubits required to make a single logical qubit. We don't have that many qubits!

This is beyond the scope of these lectures. For a cute, intuitive description: `https://www.youtube.com/watch?v=0U9_mrxL13g`

Quantum computing in practice: Applications

Applications of NISQ devices

... can we still do interesting things with NISQ devices?

Yes! There is a steady stream of proof-of-concept applications and small/toy problem instances.

A few things that have been done so far:

- Calculations of molecular energies; largest to date has been water on the IonQ platform
  http://arxiv.org/abs/1902.10171
- Supervised classification (ML) on a toy dataset
  https://www.nature.com/articles/s41586-019-0980-2
- Simulation of lattice gauge theories
  http://stacks.iop.org/1367-2630/19/i=10/a=103020

Prospects for quantum computing in HEP

Some of the major applications in HEP are:

- Quantum simulation: atoms/molecules, particle collisions, field/gauge theories
- Analysis of particle collision / track data using (quantum) machine learning (more on this tomorrow)

Two specific applications to show you today:

- Simulation of neutrino oscillations
- Calculation of the ground state energy of a deuteron

Disclaimer

I am **not** an expert in:

- Particle or nuclear physics
- Quantum chemistry
- Quantum field theory
- Hamiltonian simulation

I am learning about it only since I arrived! So if I say something wrong or unclear, and you can explain it better, please do!

(I could also use some help deciphering papers in the area, as there are many new developments I don't have the background for. Journal club?)

Neutrino oscillations

The (very strange) phenomena that explains how the flavour of a neutrino can *oscillate* as it propagates.

Consider for now the case of two neutrinos, ν_e and ν_μ.

Since they can convert between flavours, neither of these alone can be an eigenstate of their Hamiltonian; the eigenstates must be linear combinations of the two.

Let ν_e, ν_μ be the *flavour states*. We construct the *mass states*

$$\nu_1 = \cos\theta\,\nu_e + \sin\theta\,\nu_\mu$$
$$\nu_2 = -\sin\theta\,\nu_e + \cos\theta\,\nu_\mu$$

If these are the eigenstates of the Hamiltonian, then by the Schrodinger equation their time-evolution must look like this:

$$\nu_1(t) = \nu_1(0)\, e^{-iE_1 t/\hbar}$$
$$\nu_2(t) = \nu_2(0)\, e^{-iE_2 t/\hbar}$$

where E_1, E_2 are the energy eigenvalues.

Let's assume the particle begins in state ν_e:

$$\nu_e(0) = 1 \quad \nu_\mu(0) = 0$$

Then

$$\nu_1(0) = \cos\theta \quad \nu_2(0) = -\sin\theta$$

Plugging in the initial coefficients,

$$\nu_1(t) = \cos\theta\, e^{-iE_1 t/\hbar}$$
$$\nu_2(t) = -\sin\theta\, e^{-iE_2 t/\hbar}$$

We can use the eigenstates to solve for the wavefunction $\nu_\mu(t)$:

$$\nu_\mu(t) = \sin\theta\, \nu_1(t) + \cos\theta\, \nu_2(t)$$

This enables us to calculate the probability that at time t the electron neutrino has converted into a muon neutrino:

$$P_{\nu_e \to \nu_\mu} = |\nu_\mu(t)|^2 = \left[\sin(2\theta)\sin\left(\frac{E_2 - E_1}{2\hbar}t\right)\right]^2$$

Why am I showing you this?

We have

$\nu_e, \nu_\mu \qquad \nu_1, \nu_2$

$$\nu_e(t) = \cos\theta\nu_1(t) - \sin\theta\nu_2(t) \tag{20}$$
$$\nu_\mu(t) = \sin\theta\nu_1(t) + \cos\theta\nu_2(t) \tag{21}$$

Let's look specifically at $t = 0$...

$$\nu_e(0) = \cos\theta\nu_1(0) - \sin\theta\nu_2(0) \tag{22}$$
$$\nu_\mu(0) = \sin\theta\nu_1(0) + \cos\theta\nu_2(0) \tag{23}$$

We have a 2-state system. Let's make the following association:

$\nu_1(0) \to |0\rangle \quad \nu_2(0) = |1\rangle$

Why am I showing you this?

Then we can express the flavour states as

$$|\nu_e(0)\rangle = \cos\theta\,|0\rangle - \sin\theta\,|1\rangle \tag{25}$$

$$|\nu_\mu(0)\rangle = \sin\theta\,|0\rangle + \cos\theta\,|1\rangle \tag{26}$$

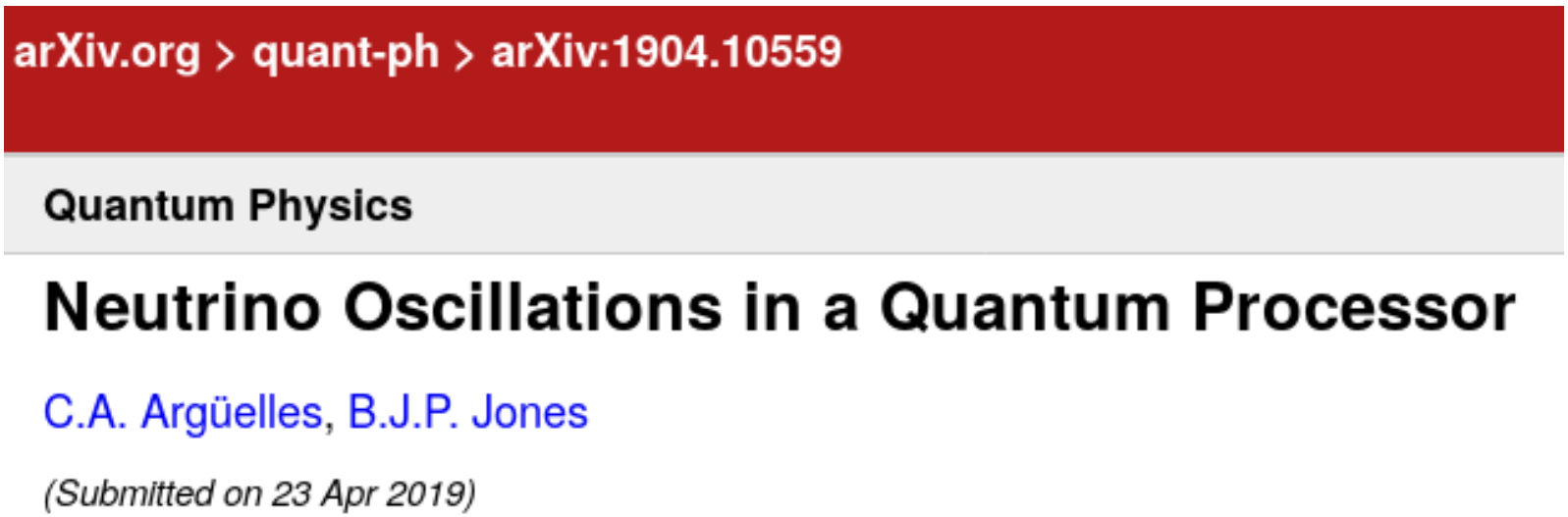
This is a unitary operation!

$$|\nu_e(0)\rangle = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} |0\rangle = U\,|\nu_1(0)\rangle$$

$$|\nu_\mu(0)\rangle = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} |1\rangle = U\,|\nu_2(0)\rangle$$

This looks suspiciously similar to performing unitary operations on single-qubit computational basis states...

We can do this!



We can use a quantum computer as an *analog* simulator of neutrino oscillations.

Qubit as an analog neutrino simulator

We saw how a unitary can convert between the mass and flavour bases; but neutrino oscillations are occuring as the particle propagates, i.e. as a function of time. To represent time evolution, we apply

$$H(t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \tag{29}$$

where $\phi = \Delta m^2 t / 2E\hbar$ contains the time dependence and can be derived from the transition probability from the previous slide. Here E is the neutrino energy.

Instead of time, this can be re-expressed in terms of a propagation distance L: ↙ prop dist.

$$\phi = \frac{\Delta m^2}{2\hbar c} \frac{L}{E} \tag{30}$$

Qubit as an analog neutrino simulator

Let's look closely at the sequence of operations here:

1. Begin by applying U to prepare a neutrino in state $|\nu_e\rangle$,

$$|\nu_e(0)\rangle = U|0\rangle = \cos\theta|0\rangle - \sin\theta|1\rangle \tag{31}$$

2. Perform Hamiltonian evolution for some time t:

$$|\nu_e(t)\rangle = H(t)|\nu_e(0)\rangle = \cos\theta|0\rangle - e^{i\phi}\sin\theta|1\rangle \tag{32}$$

   We changed the relative phase here, which means we changed the measurement statistics in some other bases.

3. Perform a measurement in the flavour basis.

With this, we can calculate the oscillation probabilities for any t just by running the quantum circuit!

Results: 2-flavour case

Plot electron 'survival' probability for a fixed propagation distance.

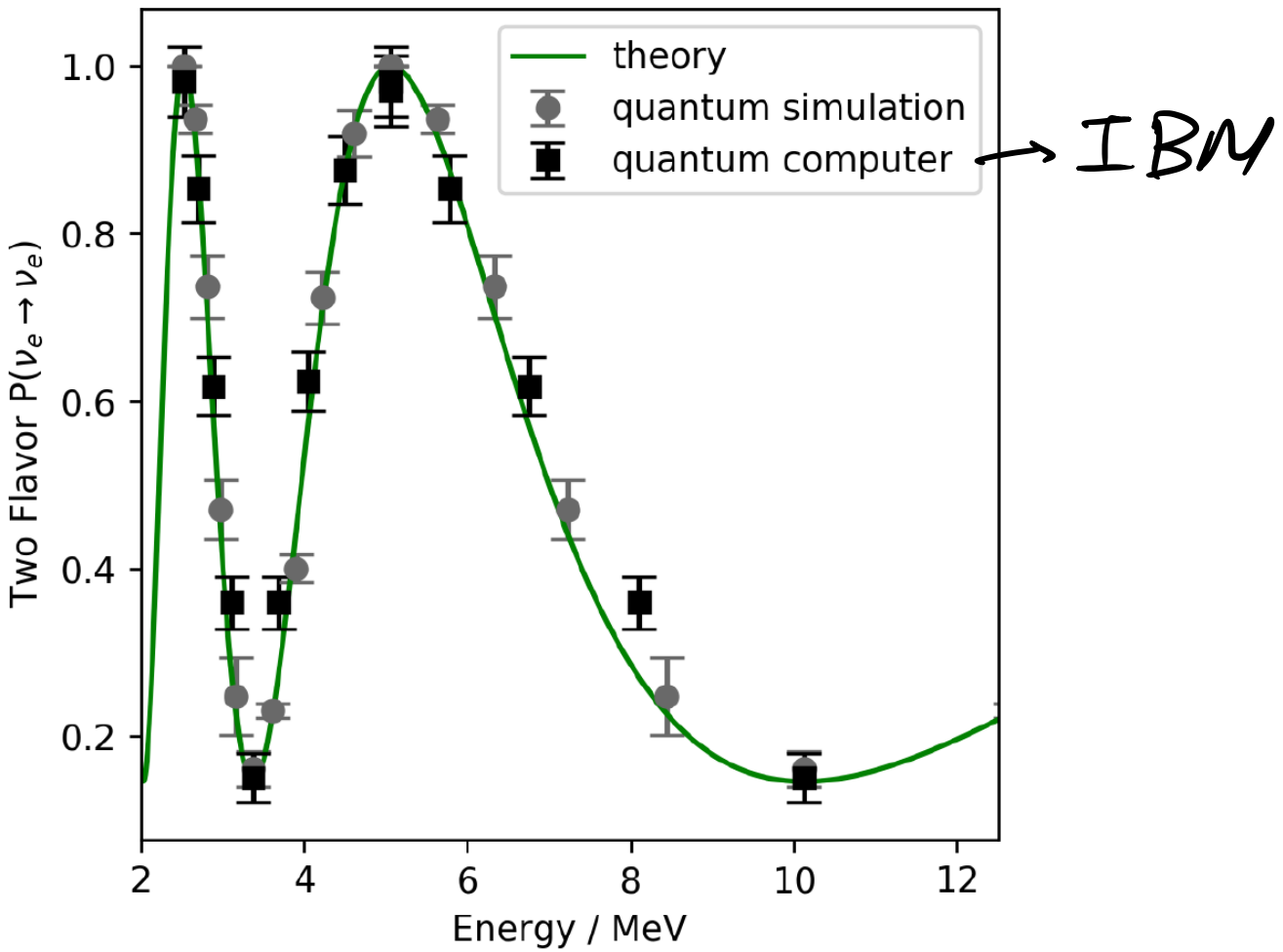
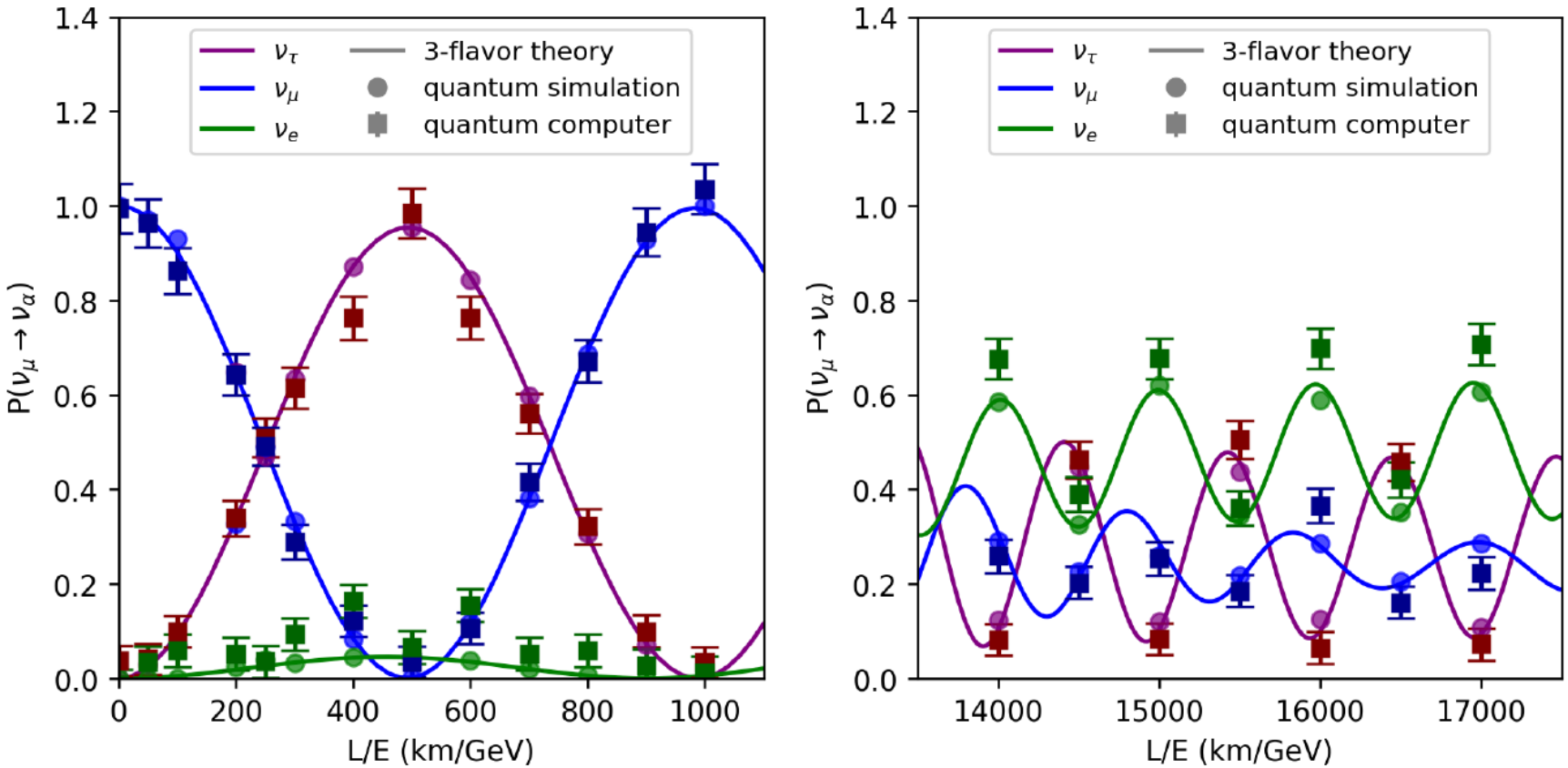


Image credit: `http://arxiv.org/abs/1904.10559`

Results: 3-flavour case



Need 2 qubits; 3 of the 4 basis states represent ν_e, ν_μ, ν_τ neutrinos, the remaining one is either redundant or can be used to explore beyond-standard-model physics like 'sterile neutrinos'.

Image credit: `http://arxiv.org/abs/1904.10559`

Hamiltonian simulation

Performing neutrino oscillation simulations on a quantum computer is a simple example of *Hamiltonian simulation*.

Main idea: we have some Hamiltonian H that the system evolves according to the Schrodinger equation

$$i\hbar \frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle \tag{33}$$

Hamiltonian simulation

Simulate the evolution of the system by finding a unitary operator U that acts on a quantum state and evolves it in the same way as H. For Hermitian (time-independent) H, use

$$U(t) = e^{-iHt} \tag{34}$$

The hard part: expressing U as a sequence of operations on qubits.

The harder part: doing this using the native gate set of the hardware architecture and making the circuits efficient.

Hamiltonian simulation: the Trotter-Suzuki formula

To simulate evolving the system, we can break up the simulation into N time slices and the Hamiltonian into chunks

$$H = \sum_k H_k \tag{35}$$

where each H_k is a smaller Hamiltonian.

Then we can approximate the evolution like this:

$$U = e^{-iHt} \approx \left(\prod_k e^{-iH_k t/N} \right)^N \tag{36}$$

Trotterization

Hamiltonian simulation: the Trotter-Suzuki formula

Suppose we want to decompose the Hamiltonian into the form

$$H = \sum_k H_k \tag{37}$$

What is a suitable choice for H_k? We want to choose them so

$$U_k = e^{-iH_k t} \tag{38}$$

is easy to implement on a quantum computer.

It turns out the *Pauli operators* are a convenient choice. Recall that we can always do this since the Paulis form a basis for the Hermitian operators.

Hamiltonian simulation: the Trotter-Suzuki formula

Why Paulis? Paulis can be diagonalized using combinations of H, S, and CNOT gates [6] i.e. for any Pauli P, we can find

$$C_i^\dagger P C_i = D \tag{39}$$

where D is diagonal, and C_i is build from H, S and CNOT.

H, S, and CNOT are typically native gates on our quantum hardware, so we can easily implement:

$$e^{-iPt} = e^{-i\left(C_i D C_i^\dagger\right)t} = C_i\left(e^{-iDt}\right)C_i^\dagger \tag{40}$$

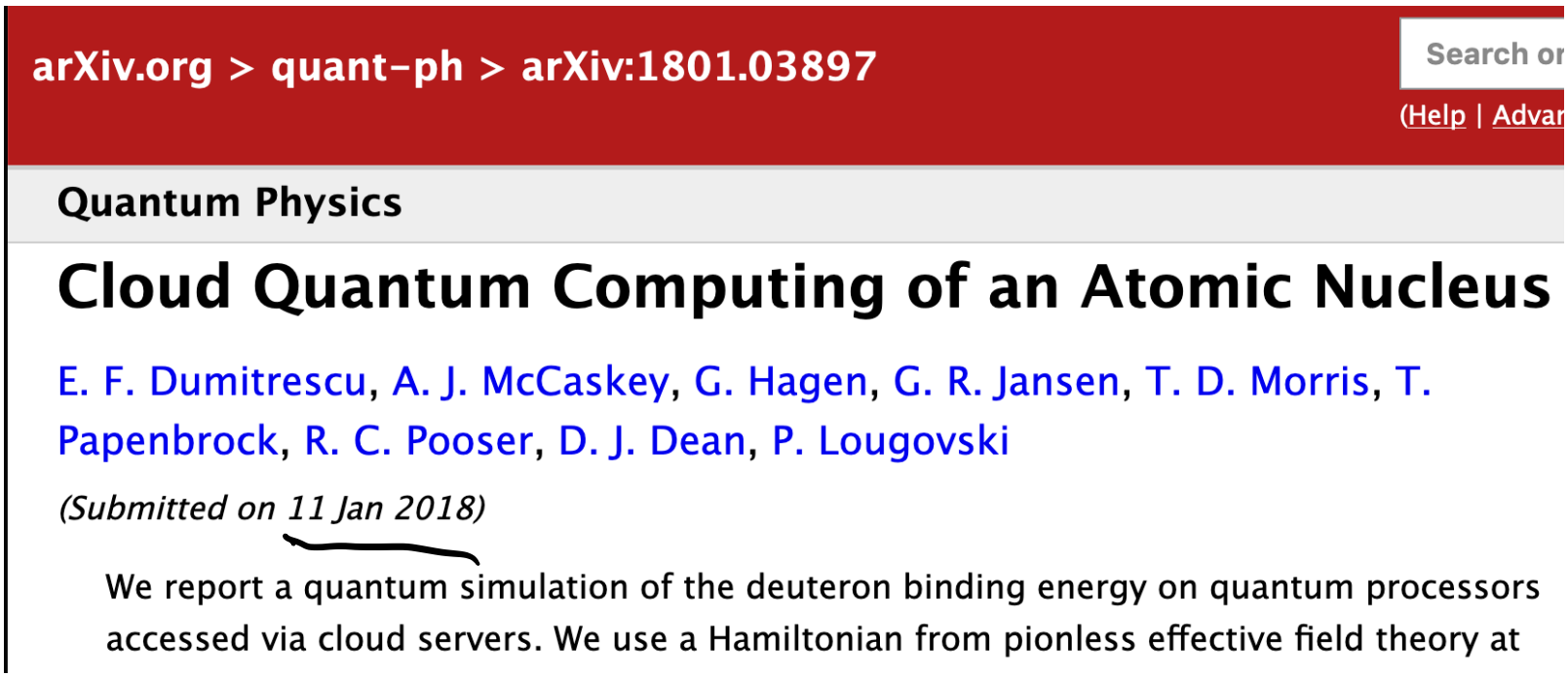
where e^{-iDt} can be implemented (up to a global phase) using only products of Zs.

Paulis = good!

[6] Together, these gates form what is known as the *Clifford group*. It is the normalizer of the Pauli group, i.e. sends Paulis to Paulis

Estimating ground state energies on a quantum computer

Hamiltonian simulation is what we use to perform quantum chemistry calculations.



Variational quantum eigensolver

This work uses a special *hybrid algorithm* called the *variational quantum eigensolver* to calculate the ground state energy of a deuteron.

The variational quantum eigensolver implements short sequences of parameterized gates that are native to the hardware, to work with NISQ hardware.

Variational quantum eigensolver

This work uses a special *hybrid algorithm* called the *variational quantum eigensolver* to calculate the ground state energy of a deuteron.

The variational quantum eigensolver implements short sequences of parameterized gates that are native to the hardware, to work with NISQ hardware.

The variational principle says that for any Hamiltonian H with ground state energy E_g, the expectation value of any other state $|\varphi\rangle$ must be greater:

$$\langle\varphi| H|\varphi\rangle \geq E_g \tag{41}$$

Variational quantum eigensolver?

In the VQE, we will express $|\varphi\rangle$ in terms of some parameters and solve an optimization problem:

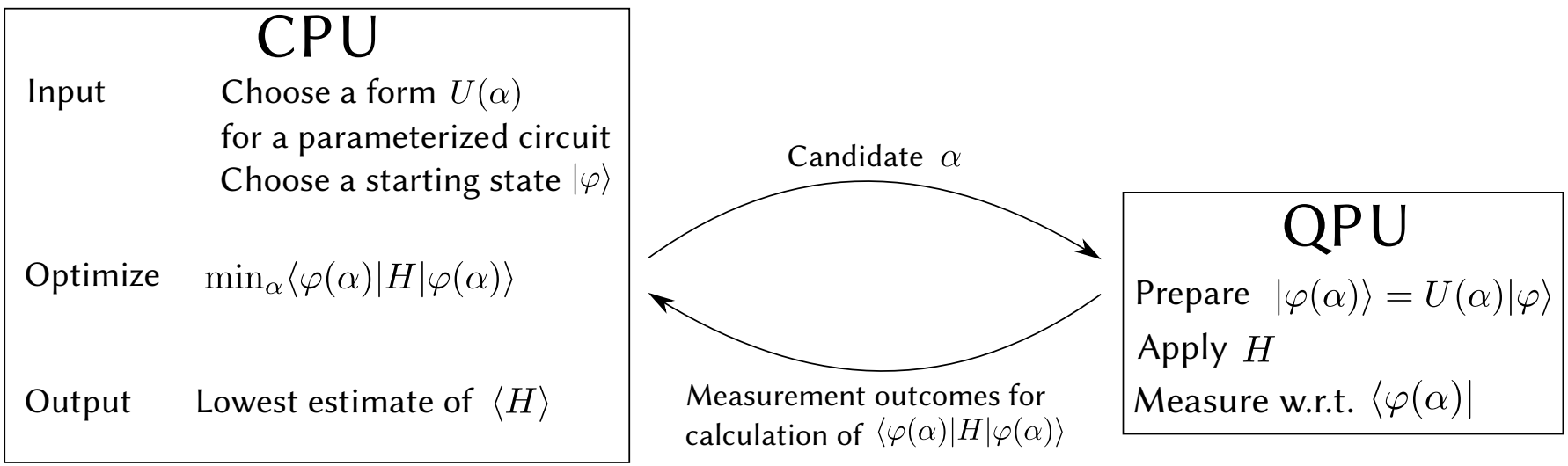
$$\text{Find } \alpha \text{ such that } \langle\varphi(\alpha)|\, H|\varphi(\alpha)\rangle \text{ is minimized} \tag{42}$$

Equivalently, we can start with a fixed, easy-to-prepare state, and parameterize some unitary operation $U(\alpha)$. Then, the optimization problem becomes:

$$\text{Find } \alpha \text{ such that } \langle\varphi|\, U^\dagger(\alpha) \cdot H \cdot U(\alpha)|\varphi\rangle \text{ is minimized} \tag{43}$$

Variational quantum eigensolver?

The classical computer will perform the optimization. It gets input from the QPU which will run the circuits and take the measurements we need to compute the expectation values.



Ground state energy of the deuteron

Write out the Fermionic Hamiltonian for the deuteron:

$$H_N = \sum_{n,n'=0}^{N-1} \langle n'|(T+V)|n\rangle a_{n'}^\dagger a_n \tag{44}$$

where a_n are creation/annihilation operators that create/annihilate a deuteron in a harmonic oscillator wave state $|n\rangle$, and T, V are the kinetic/potential energies of each state.

Ideally would take $N \to \infty$; they go up to $N = 2$ and $N = 3$.

But... how do we turn this Hamiltonian into one that acts on qubits?

Use the Jordan-Wigner transformation:

$$a_n^\dagger \rightarrow \frac{1}{2}\left[\prod_{j=0}^{n-1} -Z_j\right](X_n - iY_n)$$

$$a_n \rightarrow \frac{1}{2}\left[\prod_{j=0}^{n-1} -Z_j\right](X_n + iY_n)$$

Qubit n corresponds to the state $|n\rangle$; if it is $|\uparrow\rangle$ there is a deuteron in that state, if it is $|\downarrow\rangle$ there is no deuteron.

Allows you to re-express

$$H_N \to \sum_i h_i P_i \tag{45}$$

where P_i are Pauli operators.

For example, their Hamiltonian for $N = 2$ looks like:

$$H_2 = 5.906709 \cdot \mathbb{1}\mathbb{1} + 0.218291 \cdot Z\mathbb{1} - 6.125 \cdot \mathbb{1}Z - 2.143304 \cdot (XX + YY)$$

where I have removed the tensor products for brevity.

Recall from earlier that Pauli Hamiltonians are easy to implement.
Recall also that expectation values can sum:

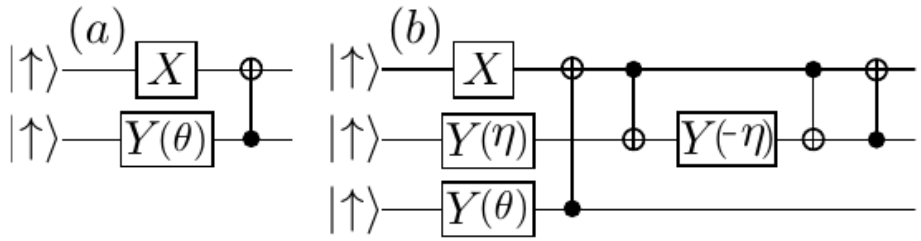
$$E_N(\alpha) = \langle\varphi(\alpha)|H_N|\varphi(\alpha)\rangle = \sum_i h_i \langle\varphi(\alpha)|P_i|\varphi(\alpha)\rangle$$

So on the quantum processor we can execute the circuit for each Pauli individually, and then sum to get the final result!

Form of $U(\alpha)$?

Choosing the form of the $U(\alpha)$ is a complicated question; they must be low-depth. It often involves alternating between rotations and entangling gates that are native to the architecture.

Circuits for H_2 and H_3 were based on the *Unitary Coupled Cluster ansatz*.



$$U(\theta) \equiv e^{\theta\left(a_0^\dagger a_1 - a_1^\dagger a_0\right)} = e^{i\frac{\theta}{2}(X_0Y_1 - X_1Y_0)},$$

$$\begin{aligned} U(\eta,\theta) &\equiv e^{\eta\left(a_0^\dagger a_1 - a_1^\dagger a_0\right) + \theta\left(a_0^\dagger a_2 - a_2^\dagger a_0\right)} \\ &\approx e^{i\frac{\eta}{2}(X_0Y_1 - X_1Y_0)} e^{i\frac{\theta}{2}(X_0Z_1Y_2 - X_2Z_1Y_0)} \end{aligned}$$

Image credit: `http://arxiv.org/abs/1801.03897`

Deuteron results

The group ran on both the IBM QX5 (16 qubits) and Rigetti 19Q.

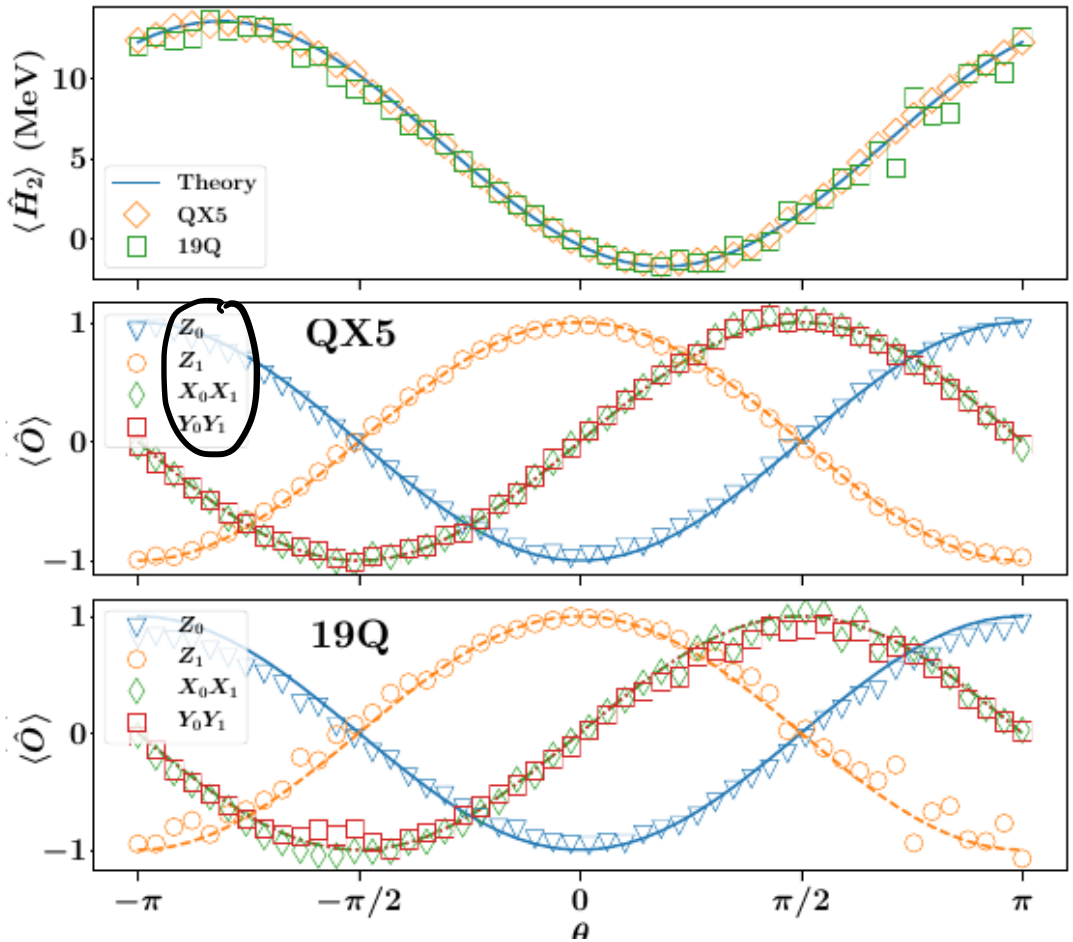


Image credit: `http://arxiv.org/abs/1801.03897`

Deuteron results

They got results to within a few percent!

E from exact diagonalization				
N	E_N	$\mathcal{O}(e^{-2kL})$	$\mathcal{O}(kLe^{-4kL})$	$\mathcal{O}(e^{-4kL})$
2	-1.749	-2.39	-2.19	
3	-2.046	-2.33	-2.20	-2.21
E from quantum computing				
N	E_N	$\mathcal{O}(e^{-2kL})$	$\mathcal{O}(kLe^{-4kL})$	$\mathcal{O}(e^{-4kL})$
2	$-1.74(3)$	$-2.38(4)$	$-2.18(3)$	
3	$-2.08(3)$	$-2.35(2)$	$-2.21(3)$	$-2.28(3)$

TABLE I. Ground-state energies of the deuteron (in MeV) from finite-basis calculations (E_N) and extrapolations to infinite basis size at a given order of the extrapolation formula (6). The upper part shows results from exact diagonalizations in Hilbert spaces with N single-particle states, and the lower part the results from quantum computing on N qubits. We have $E_1 = -0.436$ MeV. The fit at $\mathcal{O}(e^{-4kL})$ requires three parameters and is only possible for $N = 3$. The deuteron ground-state energy is -2.22 MeV.

Image credit: `http://arxiv.org/abs/1801.03897`

Summary

Key points:

- Types of quantum algorithms
- Quantum advantage
- Major players and state of available hardware
- Example applications in HEP (really just getting started)

Next week

Shifting gears to talk about quantum annealing

- Adiabatic quantum computing
- The 2-D Ising model and quantum annealing
- D-Wave hardware
- Converting problems to QUBOs
- Two particle physics applications

Homework

1. Extend Grover's algorithm to the case where there are M valid solutions instead of just 1. How do the initial subspaces have to change? How does the final number of iterations depend on M?
2. In the last 3-5 years there has been a massive increase in the number of quantum computing related start-up companies. Find 3 such start-up companies, hardware or software, that do not have the letter 'Q' in their name (and are not D-Wave, Xanadu, or Rigetti). Where are they based, and what do they work on?

Homework

3. Challenge problem: walk through the implementation of the QFT on 3 qubits, and estimate the eigenvalue of a simple Hamiltonian. I've provided a notebook to help you through this, and some bonus slides about the implementation of the QFT.
4. Challenge problem: Code up the two-neutrino oscillation circuit (`http://arxiv.org/abs/1904.10559`) and try to reproduce the plot in Figure 2 using the Qiskit simulator. The course materials contain a notebook to get you started.