Computing with Quantum Mechanics

Raghav G. Jha
Jefferson Lab

Quantum Computing Bootcamp
June 20, 2023

Welcome!

- Ask questions anonymously in the Google doc link on #general channel of Slack.
- Feel free to also contact for any non-academic questions.
- raghav@jlab.org , Office# A 227
- Especially contact for research questions.

THE LIMITS OF Quantum

By Scott Aaronson

Quantum computers would be exceptionally fast at a few specific tasks, but it appears that for most problems they would outclass today's computers only modestly. This realization may lead to a new fundamental physical principle

"Haggar Physicists Develop 'Quantum Slacks,' " read a headline in the satirical weekly the *Onion*. By exploiting a bizarre "Schrödinger's Pants" duality, the article explained, these non-Newtonian pants could paradoxically behave like formal wear and casual wear at the same time. *Onion* writers were apparently spoofing the breathless articles about quantum computing that have filled the popular science press for a decade.

A common mistake—see for instance the February 15, 2007, issue of the *Economist*—is to claim that, in principle, quantum computers could rapidly solve a particularly difficult set of mathematical challenges called NP-complete problems, which even the best existing computers cannot solve quickly (so far as anyone knows). Quantum computers would supposedly achieve this feat not by being formal and casual at the same time but by having hardware capable of processing every possible answer simultaneously.

If we really could build a magic computer capable of solving an NP-complete problem in a snap, the world would be a very different place: we could ask our magic computer to look for whatever patterns might exist in stock-market data or in recordings of the weather or brain activity. Unlike with today's computers, finding these patterns would be completely routine and require no detailed understanding of the subject of the problem. The magic computer could also automate mathematical creativ-

ILLUSTRATIONS BY DUŠAN PETRIČIĆ

© 2008 SCIENTIFIC AMERICAN, INC.

NP-hard

Bridges connecting islands

You are given a list of islands connected by bridges and you want a tour that visits each island exactly once. Finding a solution is hard as n increases. In modern language, this problem due to Euler eventually now known as 'traveling salesman' problem. This is NP hard problem.

However, if given a solution to this problem, it can be verified in polynomial time. A problem is said to be NP-hard if everything in NP can be transformed in polynomial time. A problem is said to be NP-complete if it is in both NP and NP-hard. These are the hardest problems for any Turing machine. We call it complete, since solution to any problem [in polynomial time] would imply solution to any other problem.

So, what happens if a problem is NP-hard?

Paradigm shift

Necessity is the mother of invention

- We need to formally revise the definition of the computer itself! But, what can it be? Nature is governed either by classical or quantum mechanics. So, can it be a quantum computer instead of classical computer? We don't have any other version of information theory!

- Originated in the 1970s! Made popular by Feynman in 1980s. Quantum mechanics was developed in the 1920s. *Why did it take over fifty years to think about quantum mechanical computers?*

Can we solve all problems?

Unfortunately no!

- There is a quantum version of NP known as QMA. There are many problems in Physics (even in lower dimensions!) that belong to this class. Most likely won't be solved even by a quantum computer. For example - ground state of Fermi Hubbard model.

- There is a complexity class which is probabilistic version of deterministic P-class. This is called BPP [bounded probabilistic polynomial]. The quantum version of BPP is known as BQP.

- BQP — bounded-error quantum polynomial time is the class of decision problems solvable by a quantum computer in polynomial time to some error ϵ!

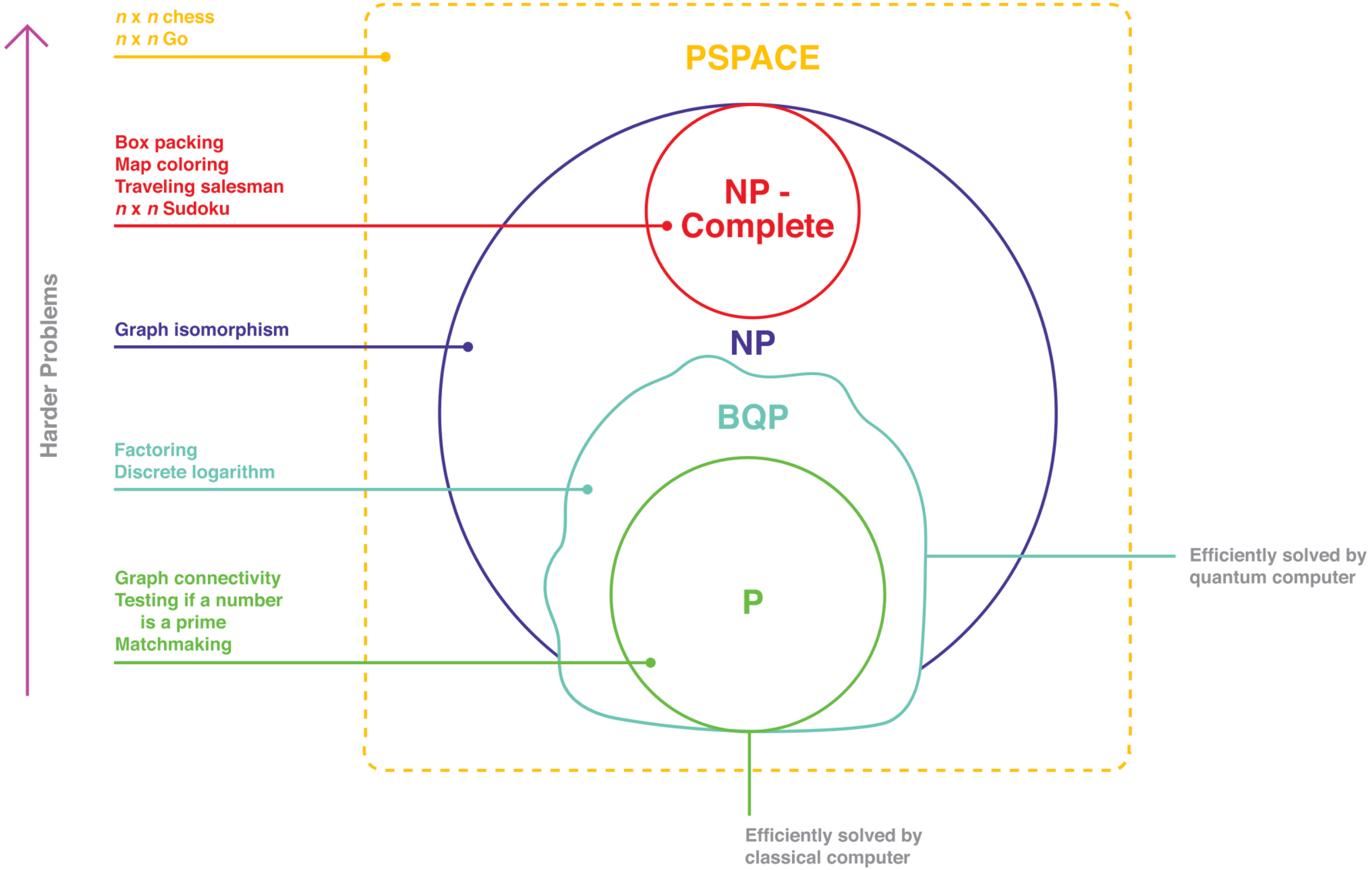
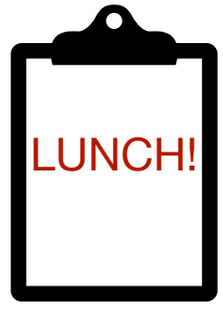


Figure: Simons Foundation Website

Rough outline

- Qubits, qudits, qumodes, Bloch sphere representation.
- Quantum gates, Entangled States, No-Cloning theorem, Fidelity
- Quantum Fourier transform (QFT)
- Universality and gate counts methods
- Variational Quantum Eigensolver (VQE) for anharmonic oscillator



- $O(3)$ model in $1+1$-dimensions with qubits
- Simple time evolution circuits with quantum gates and Pauli decomposition
- Two general Hamiltonian simulation methods for approximation $\exp(-iHt)$
- Alternate approach to QC - Using quantum harmonic oscillators (qumodes)
- Bose-Hubbard model qumodes with Strawberry Fields simulator.

State-of-the-art — Tensor Networks

- Hamiltonian version: Approximate the ground state i.e., $|\psi\rangle = \sum_{i_1,\cdots,i_N} C_{i_1,\cdots,i_N} |i_1,\cdots,i_N\rangle$ of a model with local Hamiltonian of N spins in fewer coefficients than 2^N, O(N).

- Lagrangian version: Approximate the partition function using tensor networks considering decomposition of Boltzmann weight (truncate if necessary) and then coarse-graining by performing successive iterations using singular-value-decomposition (SVD).

Scalability is a `major' problem!

- Higher-dimensions are harder. Less progress. There is no known (classical) efficient idea similar to MPS in 3+1 dimensions. Time evolution of QFTs (almost impossible) to study in these cases.

- MPS can only faithfully represent ground state of local Hamiltonians for 1d quantum systems. There are networks that can represented states of critical spin chains too. But not in higher dimensions.

- The ingenious algorithms MPS etc can only go so far. Limitation goes back to quote by Feynman, 1982. 'Nature is quantum-mechanical, we cannot simulate it classically in an efficient manner'.

Quantum Computation

Quantum Mechanical Computers

By Richard P. Feynman

Introduction

This work is a part of an effort to analyze the physical limitations of computers due to the laws of physics. For example, Bennett[1] has made a careful study of the free energy dissipation that must accompany computation. He found it to be virtually zero. He suggested to me the question of the limitations due to quantum mechanics and the uncertainty principle. I have found that, aside from the obvious limitation to size if the working parts are to be made of atoms, there is no fundamental limit from these sources either.

We are here considering ideal machines; the effects of small imperfections will be considered later. This study is one of principle; our aim is to exhibit some Hamiltonian for a system which could serve as a computer. We are not concerned with whether we have the most efficient system, nor how we could best implement it.

Since the laws of quantum physics are reversible in time, we shall have to consider computing engines which obey such reversible laws. This problem already occurred to Bennett[1], and to Fredkin and Toffoli[2], and a great deal of thought has been given to it. Since it may not be familiar to you here, I shall review this, and in doing so, take the opportunity to review, very briefly, the conclusions of Bennett[2], for we shall confirm them all when we analyze our quantum system.

It is a result of computer science that a universal computer can be made by a suitably complex network of interconnected primitive elements. Following the usual classical analysis we can imagine the interconnections to be ideal wires carrying one of two standard voltages representing the local 1 and 0. We can take the primitive elements to be just two, NOT and AND (actually just the one element NAND = NOT AND suffices, for if one input is set at 1 the output is the NOT of the other input). They are symbolized in Fig. 1, with the logical values resulting on the outgoing wires, resulting from different combinations of input wires.

From a logical point of view, we must consider the wires in detail, for in other systems, and our quantum system in particular, we may not have wires as such. We see we really have two more logical primitives, FAN OUT when two wires are connected to one, and EXCHANGE, when wires are crossed. In the usual computer the NOT and NAND primitives are implemented by transistors, possibly as in Fig. 2.

What is the minimum free energy that must be expended to operate an ideal computer made of such primitives? Since, for example, when the AND operates the output line, c' is being determined to be one of two values no matter what it was before the entropy change is $ln(2)$ units. This represents a heat generation of $kT\,ln(2)$ at temperature T. For many years it was thought that this represented an absolute minimum to the quantity of heat per primitive step that had to be dissipated in making a calculation.

The question is academic at this time. In actual machines we are quite concerned with the heat dissipation question, but the transistor system used actually dissipates about $10^{10}kT$! As Bennett[3] has pointed out, this arises because to change a wire's voltage we dump it to ground through a resistance; and to build it up again we feed charge, again through a resistance, to the wire. It could be greatly reduced if energy could be stored in an inductance, or other reactive element.

Richard P. Feynman is a professor of theoretical physics at California Institute of Technology. This article is based on his plenary talk presented at the CLEO/IQEC Meeting in 1984.

However, it is apparently very difficult to make inductive elements on silicon wafers with present techniques. Even Nature, in her DNA copying machine, dissipates about 100 kT per bit copied. Being, at present, so very far from this $kT\,ln(2)$ figure, it seems ridiculous to argue that even this is too high and the minimum is really essentially zero. But, we are going to be even more ridiculous later and consider bits written on one atom instead of the present 10^{11} atoms. Such nonsense is very entertaining to professors like me. I hope you will find it interesting and entertaining also.

What Bennett pointed out was that this former limit was wrong because it is not necessary to use irreversible primitives. Calculations can be done with reversible machines containing only reversible primitives. If this is done the minimum free energy required is independent of the complexity or number of logical steps in the calculation. If anything, it is kT per bit of the output answer.

But even this, which might be considered the free energy needed to clear the computer for further use, might also be considered as part of what you are going to do with the answer—the information in the result if you transmit it to another point. This is a limit only achieved ideally if you compute with a reversible computer at infinitesimal speed.

Computation with a reversible machine

We will now describe three reversible primitives that could be used to make a universal machine (Toffoli[4]). The first is the NOT which evidently loses no information, and is reversible, being reversed by acting again with NOT. Because the conventional symbol is not symmetrical we shall use an X on the wire instead (see Fig. 3a).

Next is what we shall call the CONTROLLED NOT (see Fig. 3b). There are two entering lines, a and b and two exiting lines, a' and b'. The a' is always the same as a, which is the control line. If the control is activated $a = 1$ then the out b' is the NOT of b. Otherwise b is unchanged, $b' = b$. The table of values

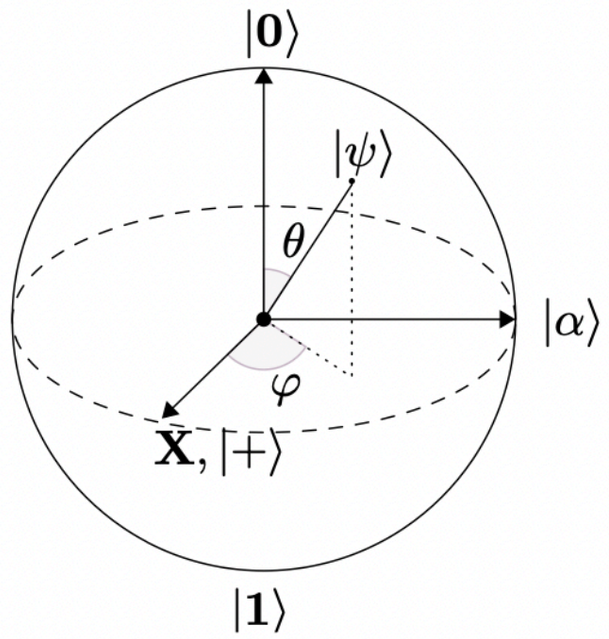
Approaches

- **Discrete-variable quantum computing**: Use qubits to perform computations. There are three steps in general: 1) Initial state-preparation, 2) Implementing unitary evolution using O(1) qubit gates, 3) Measurements.

- **Continuous-variable quantum computing**: Use of qumodes (harmonic oscillator) to carry out state preparation with some cutoff, time evolution using CV gates, and measurements.

States

- Qubits: $d = 2, \quad |0\rangle, |1\rangle$ [spin-1/2]
- Qudits: $d > 2, \quad \text{say} \quad |\phi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$ [spin-1]
- Qumodes: $d = \infty$



QUESTION: How many (c)bits are in a qubit?

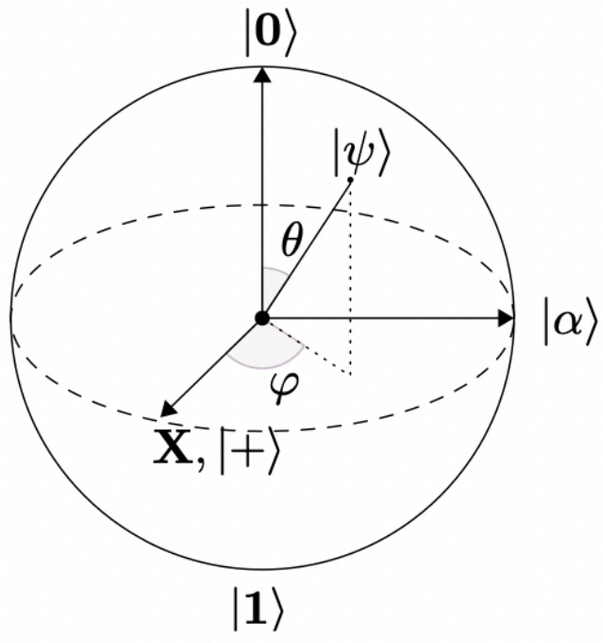
General state

- We can represent an arbitrary state on the Bloch sphere by:

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle$$

- We can construct the density matrix and show that:

$$|\psi\rangle\langle\psi| = \frac{1}{2}\left(\mathbb{I} + \vec{n}\,.\,\vec{\sigma}\right)$$



QUESTION: Why do we have $\theta/2$ rather than θ?

Reminder: Little endian vs. Big endian

- Textbooks use Big endian i.e., |00>, |01>, |10>, |11>. This is big-endian notation. The most significant bit is entered first from left.

- QISKIT and CPU architectures uses little endian notation i.e., |00>, |10>, |01>, |11>. The least significant bit is left.

Quantum gates (unitary!)

$$-\boxed{H}- = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \quad |0\rangle -\boxed{H}- |+\rangle \qquad |+\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big)$$

$$-\boxed{X}- = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, \quad |0\rangle -\boxed{X}- |1\rangle$$

$$-\boxed{Z}- = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}, \quad |1\rangle -\boxed{Z}- -|1\rangle$$

$$-\boxed{Y}- = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$$

$$-\boxed{P}- = \begin{bmatrix}1 & 0\\ 0 & e^{i\phi}\end{bmatrix}$$

$$-\boxed{R_z(\theta)}- = \begin{bmatrix}e^{-i\theta/2} & 0\\ 0 & e^{i\theta/2}\end{bmatrix}$$

$$-\boxed{S}- = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$$

$$-\boxed{T}- = \begin{bmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{bmatrix} = e^{\frac{i\pi}{8}}\begin{bmatrix}e^{\frac{-i\pi}{8}} & 0\\ 0 & e^{\frac{i\pi}{8}}\end{bmatrix}$$

Classical vs. Quantum

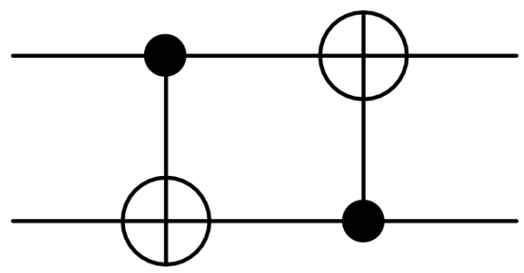
A	B	AND $(A \cdot B)$	OR $(A + B)$	XOR$(A \oplus B)$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Classical: Boolean Algebra

$\|A\rangle$	$\|B\rangle$	$\|A\rangle$	$\|A \oplus B\rangle$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

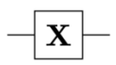
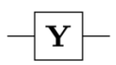
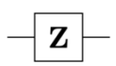
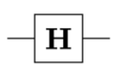
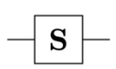
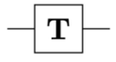
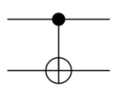
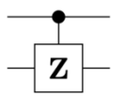
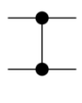
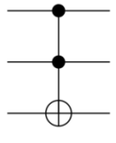
CNOT gate (CX gate)

Combination of 2 CNOT gates



$$CX = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X \qquad CZ = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes Z$$

Other gates

Operator	Gate(s)		Matrix
Pauli-X (X)			$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)			$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)			$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)			$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)			$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)			$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

``Special" features of QM

- Superposition [Hadamard] - Most of the algorithms applies this. It is quantum version of `fair coin toss'.
- Entanglement [One example is — Hadamard followed by CX]
- Arbitrary states cannot be cloned (copied). No-cloning theorem.

Cloning: $U|\psi\rangle|0\rangle \rightarrow |\psi\rangle|\psi\rangle$

Proof: $U|10\rangle = |11\rangle$

$U|00\rangle = |00\rangle$

By linearity, $U\left(\frac{|00\rangle+|10\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}\left(|11\rangle+|00\rangle\right)$

Equivalent to

$$U\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|0\rangle = \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)$$

$$= \frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}$$

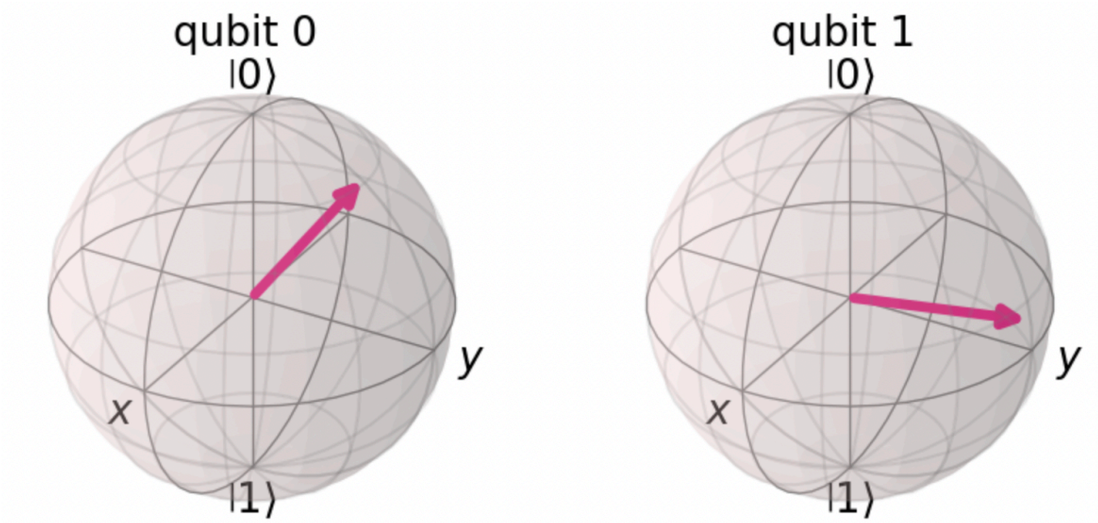
Not equal

↳ Contradiction

QISKIT implementation

- Open-source SDK developed by IBM which acts as a simulator. For our purposes, QISKIT is good enough. You can play around on ibm_q hardware by creating account etc.
- We will use Google Colab (and Mathematica couple of times) to code.

Create a state, visualise, and measure it - QISKIT Demonstration



A random two-qubit state

GHZ (Greenberger–Horne–Zeilinger) state

One of the two ways to entangled three-qubits. Given by:

$$\frac{1}{\sqrt{2}}\Big(|000\rangle + |111\rangle\Big)$$

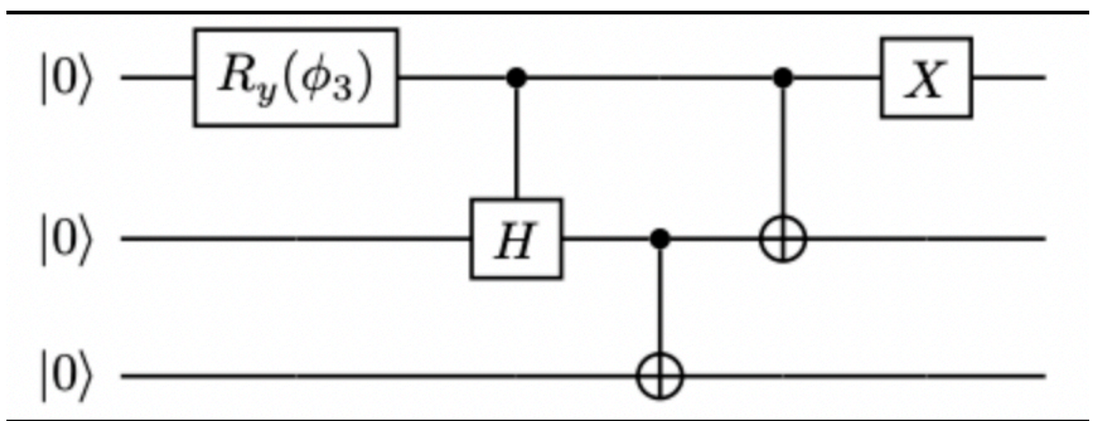
We will soon implement this in QISKIT.

Another possibility: W-state

Given by:

$$\frac{1}{\sqrt{3}}\Big(|001\rangle + |010\rangle + |100\rangle \Big)$$

We will construct this state as:



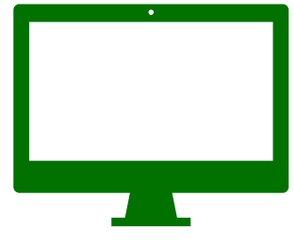
$$\phi_3 = 1.91063$$

Fidelity

- Measure of how close two quantum states are. What do you think is the fidelity between GHZ and W-state?
- Fidelity for two density operators are defined as:

$$F(\rho, \sigma) = \mathrm{Tr}(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}$$

- This simplifies to overlap^2 if they are pure states. For ex: consider two pure states, $|0\rangle, \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. What is the fidelity?



Quantum FT

- The most famous quantum algorithm is probably Shor's factoring algorithm. It has an exponential speedup over the best known classical algorithm. We now discuss QFT which forms an integral part of the factoring algorithm.

Discrete Fourier Transform

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \, e^{2\pi i j k / N}$$

Quantum Fourier transform is same but different notation.

Quantum Fourier Transform

Coppersmith '94 $\longrightarrow$ Shor factoring algorithm

$$QFT|x\rangle = \frac{1}{\sqrt{N}} \left(\sum_{y=0}^{N-1} e^{2\pi i xy/N} |y\rangle \right)$$

Let's take $N=2$ i.e one qubit.

$$QFT|x_1\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i x \cdot 1/2} |1\rangle \right)$$

$e^{2\pi i [0.x_1]}$

$$= \frac{1}{\sqrt{2}} \left(|0\rangle + e^{i\pi x_1} |1\rangle \right)$$

$$[0.x_1 \cdots x_m] = \sum_{k=1}^{m} x_k / 2^k$$

$$QFT\,|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle$$

$$QFT\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$$

General form:

$$|j_1, \ldots j_n\rangle \rightarrow \frac{\left(|0\rangle + e^{2\pi i\, 0.j_n}|1\rangle\right) \otimes \ldots \ldots}{\sqrt{N}} \otimes \left(|0\rangle + e^{2\pi i\, 0.j_1 \ldots j_n}|1\rangle\right)$$

We will implement this in Q'SKIT.

Gate complexity of QFT

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$P = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

$\longrightarrow$ H followed by

$(n-1)$ controlled P

$\longrightarrow$ n gates

$\longrightarrow$ H followed by

$(n-2)$ controlled P

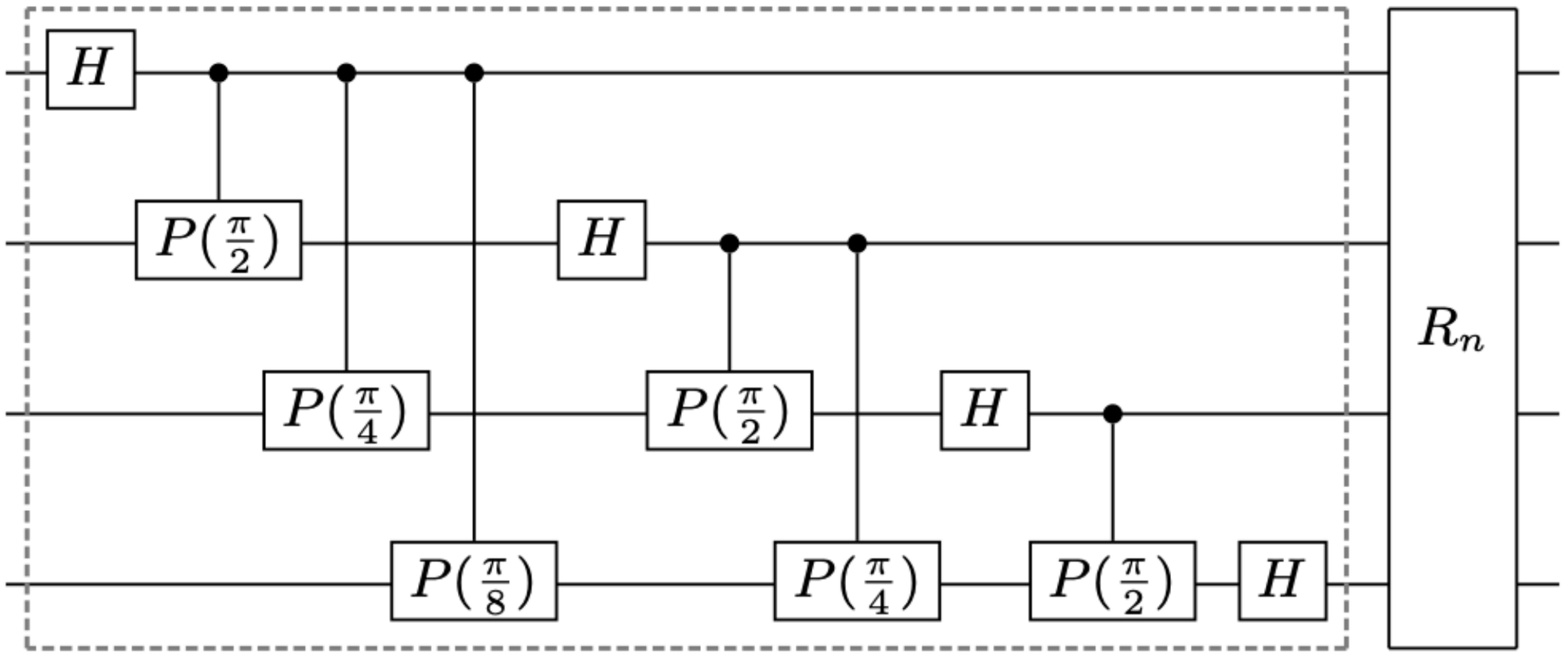
on 2^m $\longrightarrow$ $(n-1)$ gates

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2} + \text{SWAP gates}$$

$$= O(n^2).$$

The best classical algorithms such as FFT use $O(n\,2^n)$ gates. → EXPONENTIAL SPEED UP. in # of operations

QUESTION: Does QFT increase or decrease entanglement ??



More on gate costs and universality

- One needs to know which gate set are universal for quantum computation. On our Intel chips, NAND and NOR gates are universal. What is the quantum analog?

- To satisfy the universality for quantum gates, we have to satisfy few conditions. 1) Create superposition, 2) Create entanglement, 3) Cannot just be real gates, 4) Belongs to Clifford group i.e., {CNOT, Hadamard, P} since we know by Gottesmann-Knill theorem that it can be simulated efficiently in polynomial time on classical computer.

- What are our options then? It is clear that we need CNOT can help us get past (2). We now desire one-qubit gates. There is a mathematical result which says that any 2x2 unitary matrix can be written in rotation gates. Rx, Ry, Rz. So, CNOT+ seems like a good option and it indeed is. IBM hardware also has this gate set!

More on gate costs and universality

- We also know that error-rate of one qubit gates are much less than two-qubit gates. So in any practical calculation, the dominate cost/error will come from CNOT gates. Hence, we often use this as metric to compare the costs.

- So, now the goal is to write any unitary matrix in terms of CNOT, and rotation gates and count how many CNOTs are needed for n qubits.

- This is a well-researched problem. To understand this, we have to understand — Quantum Shannon Decomposition (QSD) which was introduced in the paper Shende et al. quant-ph/0406176

Definitions needed for QSD

For 1 qubit: No CNOT needed. We have what is called ZYZ decomposition.

Circuit form of $U = e^{i\Phi} R_z(\alpha) R_y(\beta) R_z(\gamma)$

Before moving to n-qubits with $n > 1$, we need some definitions.

↳ QUANTUM MULTIPLEXORS.

We denote it by □. It is of block-diagonal form.

$$U = \begin{pmatrix} U_0 & & \\ & \ddots & \\ & & U_1 \end{pmatrix}$$

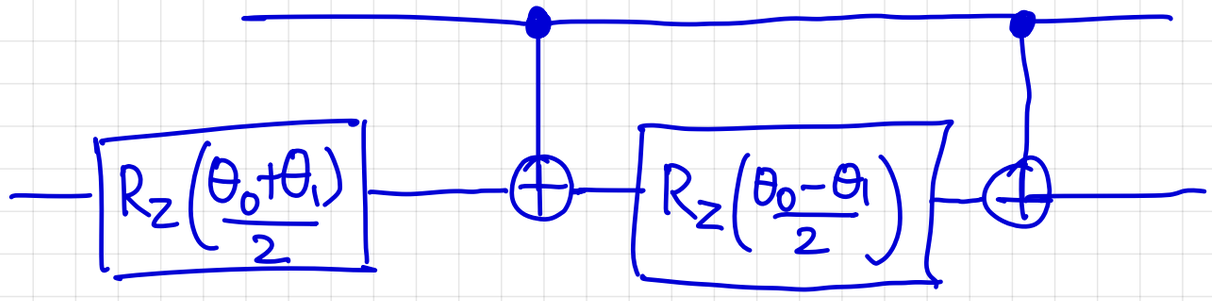
Simplest example is Quantum Multiplexor on two-qubits. Any guesses ??

Definitions needed for QSD

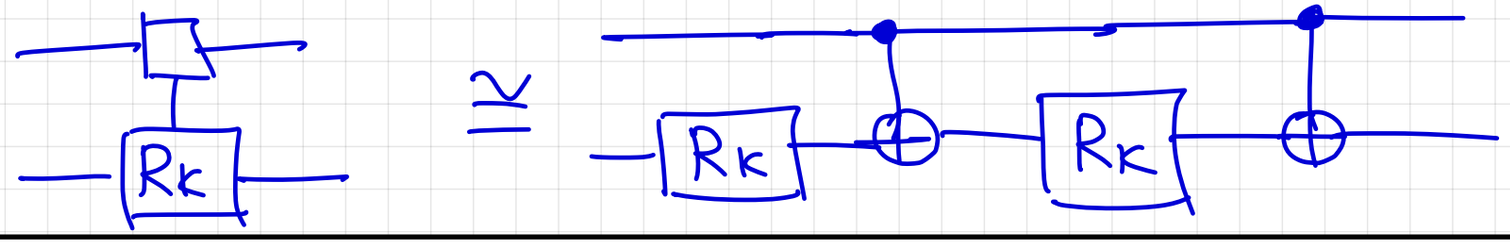
$$CNOT = \mathbb{1} \oplus \sigma_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Flips the second (data) bit if the first (SELECT) bit is $|1\rangle$.

How to implement $R_z(\theta_0) \oplus R_z(\theta_1)$ by:



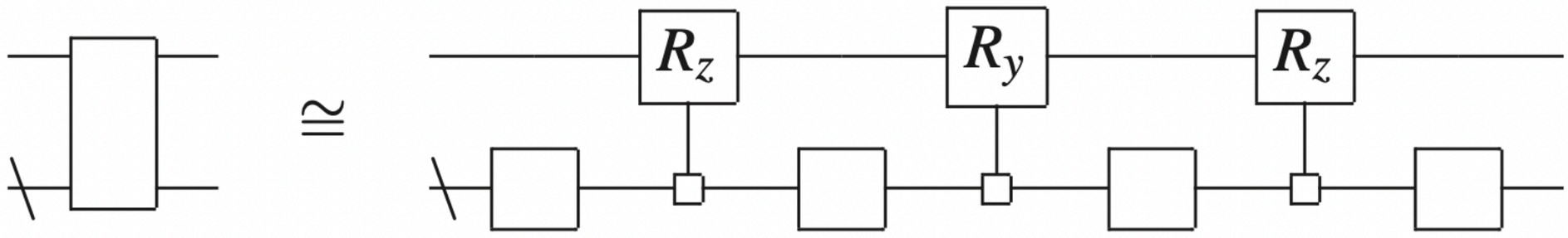
Notation:



QSD

quant-ph/0406176

- Any n qubit gate can be decomposed into three multiplexed rotations and four generic $(n-1)$ qubit operators. Though this looks simple, this is pretty messy for $n > 2$. The goal is to count the number of CNOTs needed.



QSD

quant-ph/0406176

- There is a theoretical lower-bound on # CNOT gates needed which is given by:

$$\frac{1}{4}(4^n - 3n - 1)$$

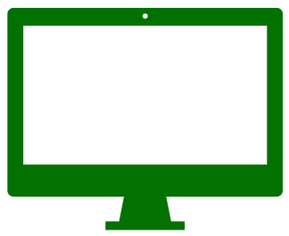
- In the paper referenced above, it was shown that optimised QSD they implemented costs roughly :

$$\frac{23}{48}4^n - \frac{3}{2}2^n$$

- Lower bound is about half of the best implementation we have at the moment. *If we have time at the end of the day: we will try to do this exercise in QISKIT.*

Quantum Fourier transform - QISKIT Demonstration

- But let's go back to QFT and see how this works in QISKIT. We will also check that it is unitary.



Quantum Algorithms cont.

- A very popular NISQ-era hybrid algorithm is variational quantum eigensolver (VQE). The steps are:

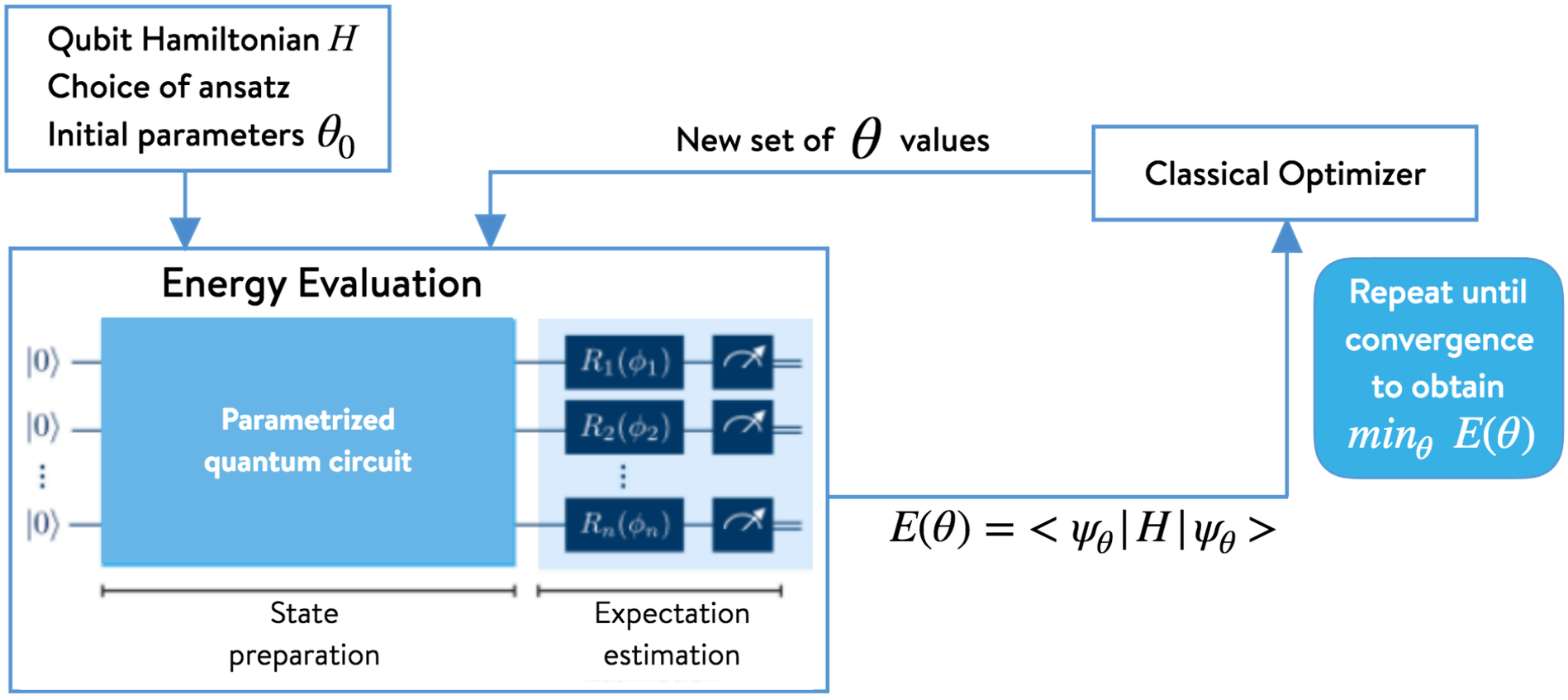
  1. Prepare initial state on QC i.e., $|0\rangle$

  2. Obtain good ansatz by acting with some $U(\Theta)$i.e., $|\psi\rangle = U|0\rangle$

  3. Measure energy on QC and optimise the parameters Θ using classical optimisers

  4. Repeat until convergence.

VQE representation



Qubit H, VQE etc.

- Pauli matrices are special — $\{X, Y, Z, I\}$ forms a complete basis. Though, not the most efficient basis. Usually Hamiltonians for VQE are represented by Pauli strings. Either through some transformation like JW or BK or matrix decomposition method which we will soon see. VQE has been extensively used to study various molecules in quantum chemistry like H_2, BeH_2 etc.

PHYSICAL REVIEW X

Highlights Recent Subjects Accepted Collections Authors Referees Search Press About Editorial Team

Open Access

Scalable Quantum Simulation of Molecular Energies

P. J. J. O'Malley *et al.*
Phys. Rev. X **6**, 031007 – Published 18 July 2016

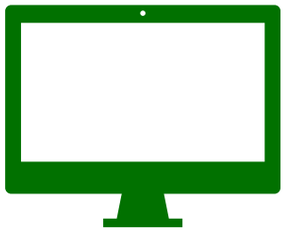
151

More

Article References Citing Articles (464) PDF HTML Export Citation

Anharmonic oscillator - Set up

```
SetDirectory[NotebookDirectory[]];
nbits = 3; (*number of qubits*)
n = 2^nbits;
g = 0.02;
A1 = Table[If[(j - i) == 1, Sqrt[i], 0], {i, n}, {j, n}];  (*Annihilation operator*)
X1 = Sqrt[0.5] (A1\[ConjugateTranspose] + A1); (*Position operator in E basis*)
H1A[A_, X_] := A\[ConjugateTranspose] . A + IdentityMatrix[n]/2 -
   g MatrixPower[X, 3]; (*Cubic anharmonic Hamiltonian in E basis*)
H = H1A[A1, X1]; (*Make Hamiltonian to export*)
hamName = "HO"; (*Set Hamiltonian name for file*)
Export["ham_" <> hamName <> ".txt", H,
  "Table"]; (*Export Hamiltonian to be read by our QISKIT program!*)
```



If you don’t have Mathematica, copy the matrix to a local file from **https://shorturl.at/btwFG**

Link expires: 26 June, 2023

Decompose matrix

Assume we don't know the expression just that it forms a complete basis. Let's build the formula for decomposing any Hermitian matrix H in terms of Paulis.

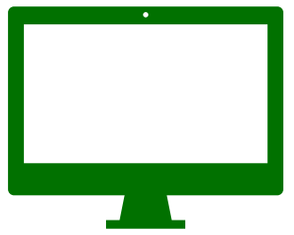
```
CT = KroneckerProduct;

X = PauliMatrix[1];

Y = PauliMatrix[2];

Z = PauliMatrix[3];

H = 0.23 CT[X, CT[Z, Y]] + 0.45 CT[Z, CT[Y, X]]
```



Anharmonic oscillator

QISKIT Demonstration.

O(3) model in 1+1

arXiv: 2210.03679 [quant-ph]

- The Hamiltonian is given by ($\beta = 1/g^2$) :

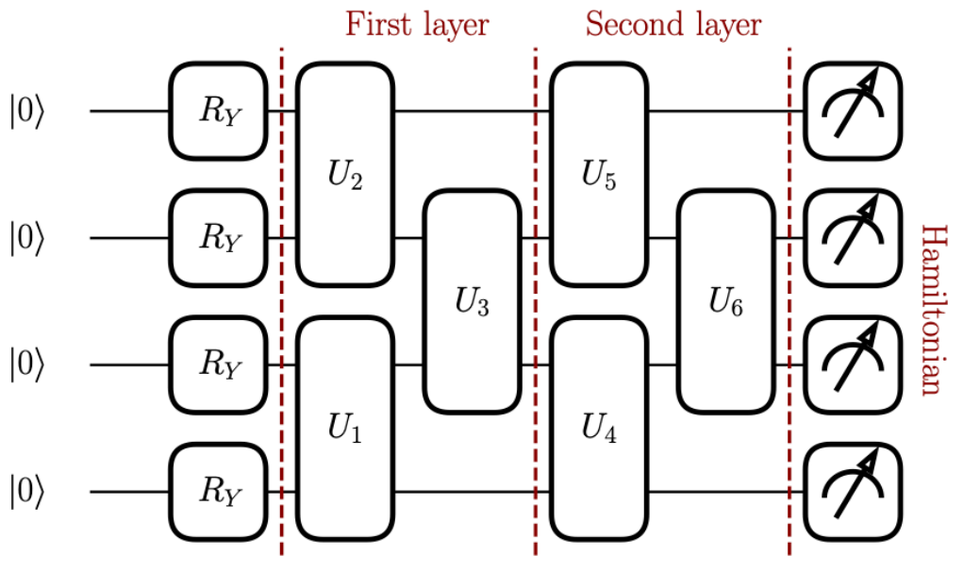
$$\hat{H} = \frac{1}{2\beta} \sum_i \mathsf{L}_i^2 - \beta \sum_{\langle i,j \rangle} \mathsf{n}_i \cdot \mathsf{n}_j,$$

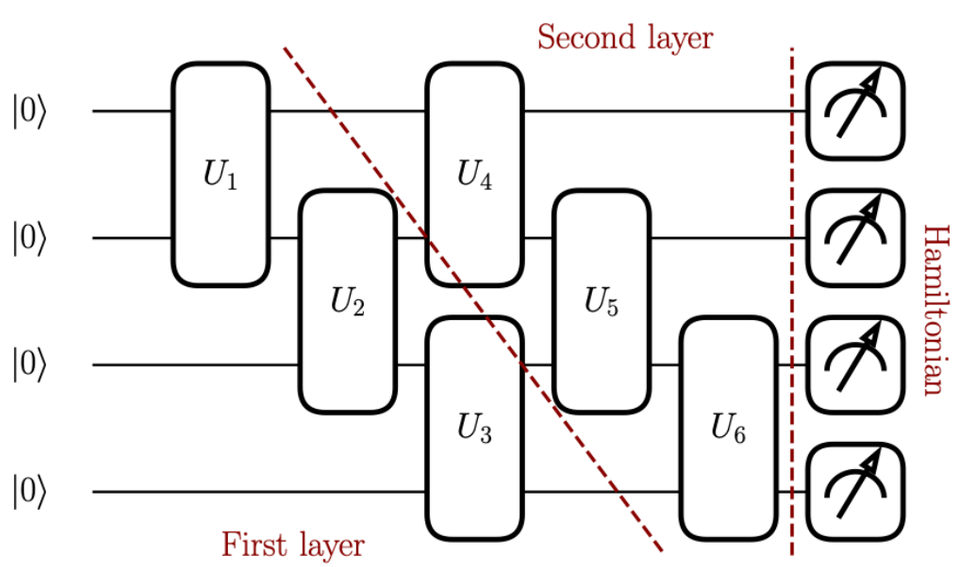
- We can construct this matrix for some fixed value of $l_{\text{max.}}$

  The Hamiltonian for a N-site lattice is a $(l_{\text{max.}} + 1)^{2N} \times (l_{\text{max.}} + 1)^{2N}$ matrix. We can consider model with or without a θ-term. As we saw before, we need to express the H in terms of qubits which is often done use Jordan-Wigner or Bravyi-Kitaev transformations.

O(3) model

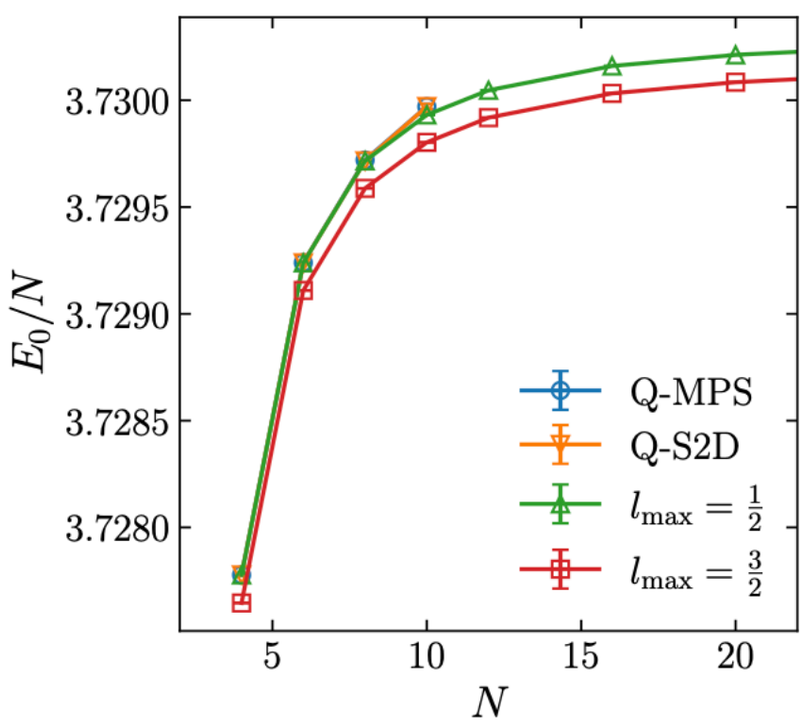
arXiv: 2210.03679 [quant-ph]



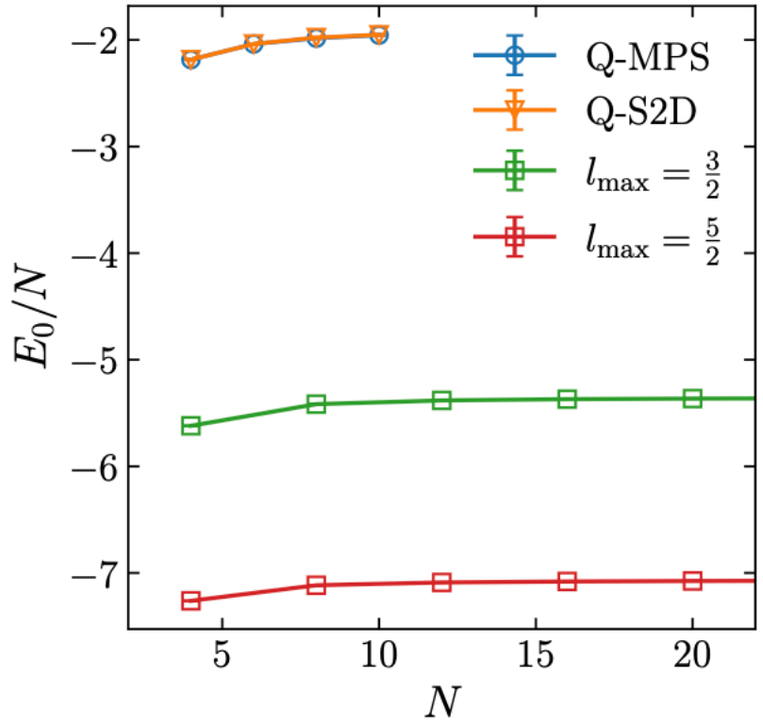


Results

- At the moment, VQE is at times, no better than exact diagonalization. But, there are various improvements and it will improve in future.



$\beta = 0.1$



$\beta = 10$

Hamiltonian Simulation

- The most important application of quantum computers for Physics will be to carry out time evolution. The problem of implementing $\exp(-iHt)$ is known as HS problem. We say that a HS is efficient if N-qubit H can be simulated within some error ϵ using $\mathcal{O}(N)$ operations/gates. There has been more than two decades of intense research on how to do this efficiently and various quantum algorithms have been developed. We will hardly touch the surface. You will hear more about them during the bootcamp.

Time evolution of quantum systems

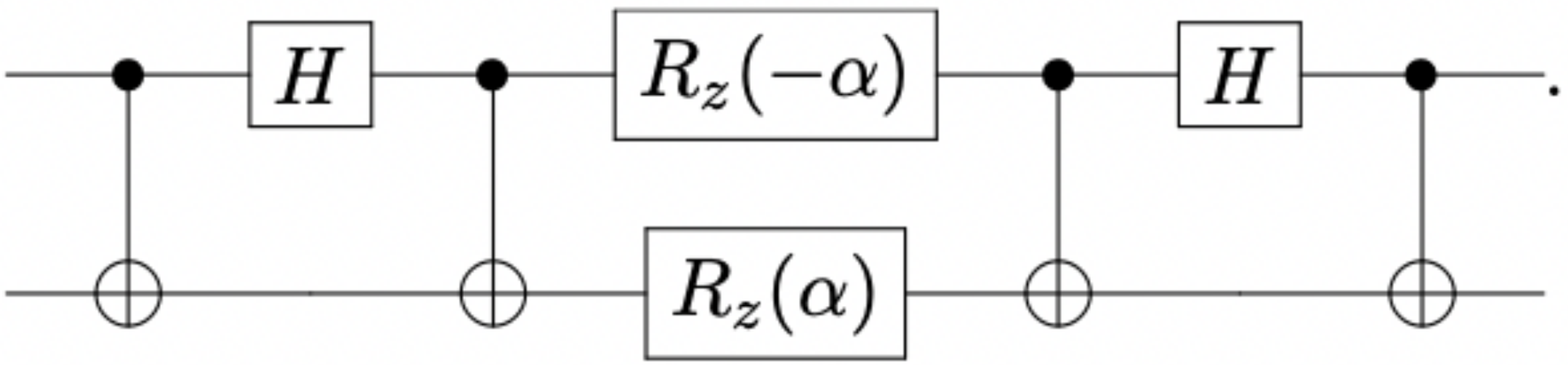
- Let us start with a very simple system. Suppose, we have spin-1/2 particle each on two sites with some H below, we would need two qubits to initialise the state say, $|00\rangle$. Now suppose the 4x4 Hamiltonian of this two-site model is given by:

$$H = (X \otimes X) + (Y \otimes Y)$$

We want to do time evolution of this system i.e., $\exp(-iHt)$. We have to represent this unitary operator with quantum (unitary) gates. Note than any Hamiltonian can be written entirely in terms of Pauli strings.

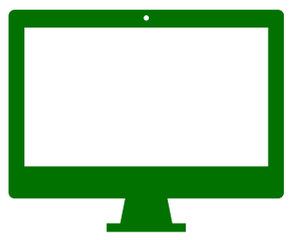
Time evolution of quantum systems

- Note that we have to keep dt sufficiently small, so we have to repeat the circuit below N times where $N = t/dt$. As we can see, we need about $8N$ unitary gates (4 one-qubit, and 4 two-qubit) for this simple Hamiltonian and two sites!

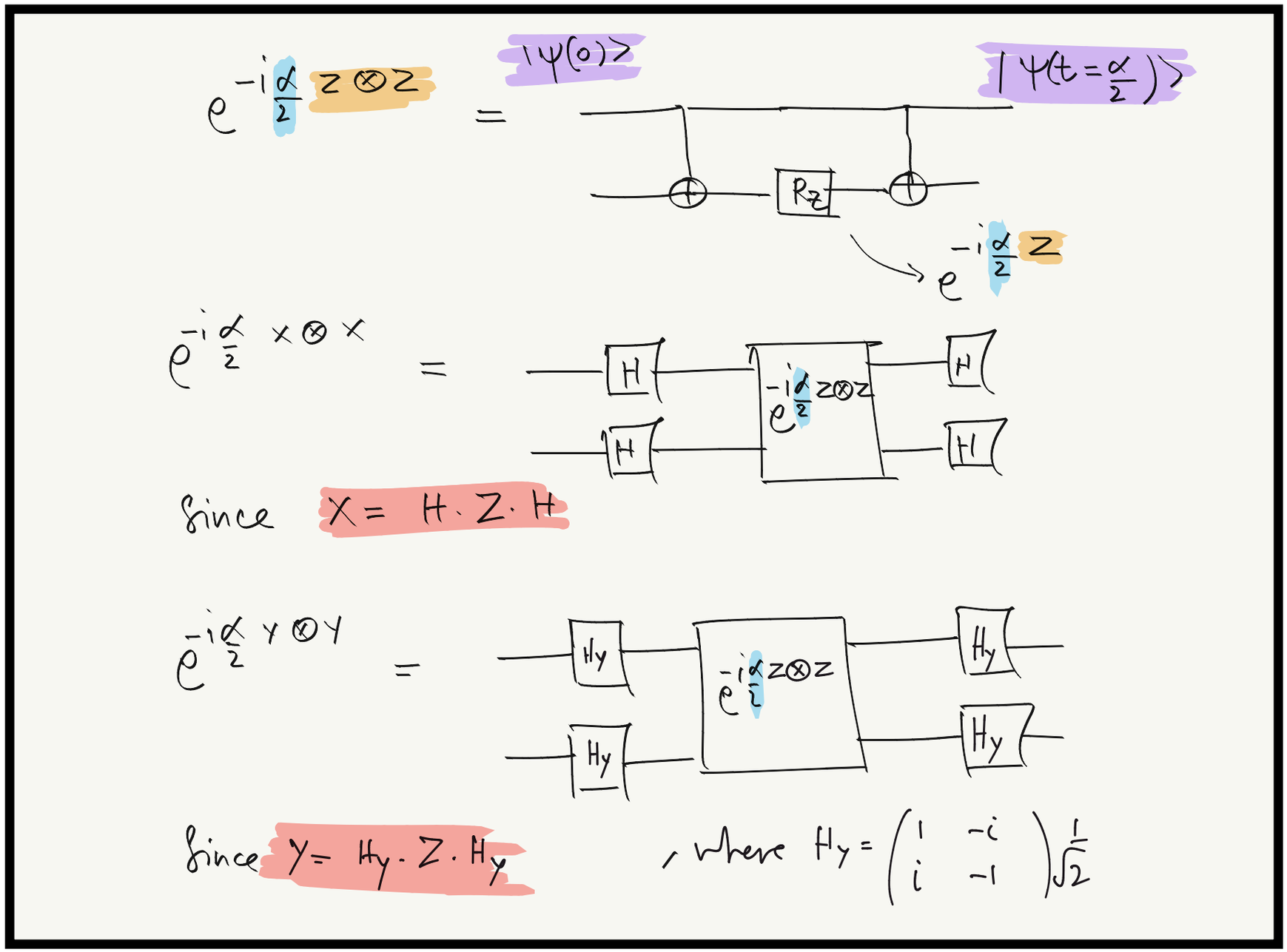


Quick check!

```
CT = KroneckerProduct;
X = PauliMatrix[1];
Y = PauliMatrix[2];
Z = PauliMatrix[3];
CNOT = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}};
HAD = {{1/\[Sqrt]2, 1/\[Sqrt]2}, {1/\[Sqrt]2, -(1/\[Sqrt]2)}};
HY = {{1/\[Sqrt]2, -I/\[Sqrt]2}, {I/\[Sqrt]2, -(1/\[Sqrt]2)}};
XXYYv1 = CNOT . CT[HAD, IdentityMatrix[2]] . CNOT . CT[MatrixExp[I (\[Alpha]/
2) Z], MatrixExp[I (-\[Alpha]/2) Z]] . CNOT . CT[HAD, IdentityMatrix[2]] . CNOT //
FullSimplify;
XXYYv2 = (CT[HAD, HAD] . CNOT . CT[IdentityMatrix[2], MatrixExp[I (\[Alpha]/
2) Z]] . CNOT . CT[HAD, HAD]) . (CT[HY, HY] . CNOT . CT[IdentityMatrix[2],
MatrixExp[I (\[Alpha]/2) Z]] . CNOT . CT[HY, HY]) // FullSimplify;
XXYYv1 == XXYYv2 == MatrixExp[I (KroneckerProduct[X, X] +
KroneckerProduct[Y, Y]) \[Alpha]/2]
```



Basic idea


$e^{-i\frac{\alpha}{2} Z\otimes Z}$ = $|\psi(0)\rangle$ $|\psi(t=\frac{\alpha}{2})\rangle$
R_Z
$e^{-i\frac{\alpha}{2} Z}$
$e^{-i\frac{\alpha}{2} X\otimes X}$ =
H
H
H
H
$e^{-i\frac{\alpha}{2} Z\otimes Z}$
since $X = H \cdot Z \cdot H$
$e^{-i\frac{\alpha}{2} Y\otimes Y}$ =
H_y
H_y
H_y
H_y
$e^{-i\frac{\alpha}{2} Z\otimes Z}$
since $Y = H_y \cdot Z \cdot H_y$, where $H_y = \begin{pmatrix} 1 & -i \\ i & -1 \end{pmatrix}\frac{1}{\sqrt{2}}$

QUESTIONS?

Not efficient!

- It might seem that the basic idea is straightforward and we can carry on with this. However, it is easy to check that if N increases, the decomposition in terms of Paulis is not efficient. It scales terribly with N.

- One thing to note about Pauli matrices and strings is that they are 1-sparse.

- A matrix is said to be d-sparse if it has at most d non-zero entries in any given row/column. A 2-sparse matrix looks like:

Sparse-row-computable (SRC)

- Wide class of Hamiltonians are SRC. It means the matrix is d-sparse and there exists efficient algorithms such that $f(x, j)$ = the column number of j^{th} non-zero in row x. For matrix on earlier slide, $f(1,0) = 0$ and $f(1,1) = 2$ [this tells where the non-zero is]. We also need to know the element i.e., H_{xy} efficiently i.e., with only $\mathcal{O}$(poly (n)). We can define the action of two oracles as:

$$O_f(j)|x\rangle|0\rangle = |x\rangle|f(x,j)\rangle$$

$$O_H(j)|x\rangle|y\rangle|0\rangle = |x\rangle|y\rangle|H_{xy}\rangle$$

Efficient decomposition

- We do not want to do: $H = \sum_j a_j P_j$ but would rather like to do $H = \sum_{j=1}^{N} H_j$ where each H_j is 1-sparse. Note that 1-sparse matrices are broader class. Every P_j is 1-sparse, but not every 1-sparse can be written in terms of Pauli matrices.

- Side note: There are two ways to deal with the unitary operator. One is Lie-Suzuki-Trotter product formula which splits into product of unitary or we can use Taylor series and truncate it, and use oracles to simulate the time evolution. We won't discuss the latter and will only briefly discthe Trotter method.

Lie - Product formula:

$$e^{t(A+B)} = \lim_{N\to\infty} \left(e^{\frac{At}{N}} e^{\frac{Bt}{N}}\right)^N$$

or more useful form:

$$e^{-it(A+B)} = e^{-itA} e^{-itB} + O(t^2).$$ In general for p^{th} order, the error grows like $O(t^{p+1})$.

Generally,

$$e^{-i\sum_{j=1}^{m} H_j t} = \prod_{j=1}^{m} e^{-iH_j t} + O(m^2 t^2)$$ for $t \ll 1$.

What happens if $t \gg 1$??

For that, split into "r" steps. Lloyd '96

$$e^{-i\sum H_j t} = \left(\prod e^{-iH_j t/r}\right)^r + O\left(\frac{m^2t^2}{r}\right)$$

If we want to bound the error by ϵ i.e

$$\left\| U(t) - \left(\prod e^{-iH_j t/r}\right)^r \right\| \lesssim \epsilon$$

↳ spectral norm
↳ Schatten ∞-norm

we require $r \sim m^2t^2/\epsilon$

2^{nd} order formula $\rightarrow$ $O\left(\frac{m^3t^3}{r^2}\right)$

↳ $r \sim \frac{(mt)^{3/2}}{\sqrt{\epsilon}}$

Taylor Series:

$$U(t) = e^{-iHt} = \sum_{k=0}^{\infty} \frac{(-itH)^k}{k!}$$

Truncate to "p" terms i.e.

$$\tilde{U}(t) = \sum_{K=0}^{p} \frac{(-itH)^k}{k!} + \text{error.}$$

↳ Berry, Childs, Cleve, Kothari, Somma 1312.1414

QUESTIONS?

The career of a young theoretical physicist consists of treating the harmonic oscillator in ever-increasing levels of abstraction.

- Quote by Sidney Coleman

Alternate approach

Encode information in QHO/photons

- Can represent qubits and qudits effectively since the Hilbert space is “theoretically” infinite-dimensional. It has been shown to be useful for constructing error-correcting codes. There are different prescriptions to encode a qubit in an oscillator.

- One such method is to represent qubits as single photon, this is sometimes useful since practically all the required components (for a single photon quantum computer) are already available. In addition, noise is a well-understood problem in quantum optics. This is known as `dual-rail’ representation.

$$|0\rangle_L := |01\rangle \qquad\qquad |1\rangle_L := |10\rangle.$$

CV vs. Qubit

- But for the moment let's focus on the infinite-dimensional bosonic operators. There are now simulators for qumodes (photonic simulator) like Bosonic QISKIT, Strawberry Fields etc.

	CV	Qubit
Basic element	Qumodes	Qubits
Relevant operators	Quadrature operators $\hat{x}, \hat{p}$ Mode operators $\hat{a}, \hat{a}^\dagger$	Pauli operators $\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z$
Common states	Coherent states $\vert\alpha\rangle$ Squeezed states $\vert z\rangle$ Number states $\vert n\rangle$	Pauli eigenstates $\vert 0/1\rangle$, $\vert\pm\rangle$, $\vert\pm i\rangle$
Common gates	Rotation, Displacement, Squeezing, Beamsplitter, Cubic Phase	Phase Shift, Hadamard, CNOT, T Gate

Bosons/Fermions

Canonical mode operators: $\hat{a}, \hat{a}^\dagger$ satisfying $[a, a^\dagger] = \mathbb{I}$. One crucial thing to note is that for single fermionic mode, the analogous relation is $\{a, a^\dagger\} = 1$.

$$aa^\dagger + a^\dagger a = 1$$

also $aa = a^\dagger a^\dagger = 0$

Therefore, using $\hat{n} = a^\dagger a$, we have:

$$aa^\dagger = 1 - \hat{n}$$

$$\Rightarrow \hat{n}(1-\hat{n}) = (a^\dagger a)(aa^\dagger)$$

$$= 0 \quad \text{since } aa = 0$$

$n = 0$ or 1. → Pauli Exclusion principle

Not true for bosons. ∞-dimensional

How to see this ?? Take trace of $[a, a^\dagger] = \mathbb{1}$ on both sides.

$$\hat{x} = \sqrt{\frac{\hbar}{2}}\,(\hat{a} + \hat{a}^\dagger)$$

$$\hat{p} = \frac{-i}{\sqrt{2}}\,(\hat{a} - \hat{a}^\dagger)$$

} Quadrature ops.

CV states

Coherent States : States of light (photon) whose expectation value corresponds to classical EM wave.

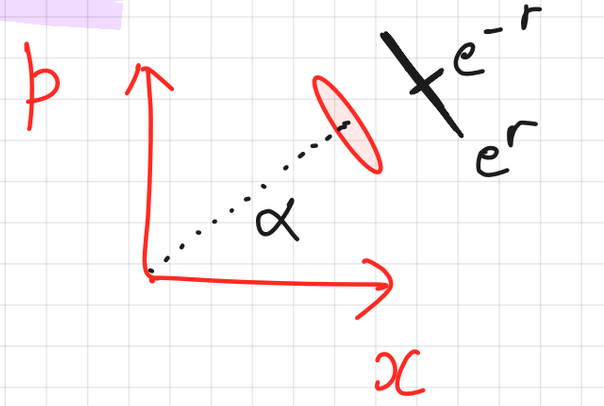
$$|\alpha\rangle = e^{-\frac{1}{2}|\alpha|^2} \sum_n \frac{\alpha^n}{\sqrt{n!}} |n\rangle$$

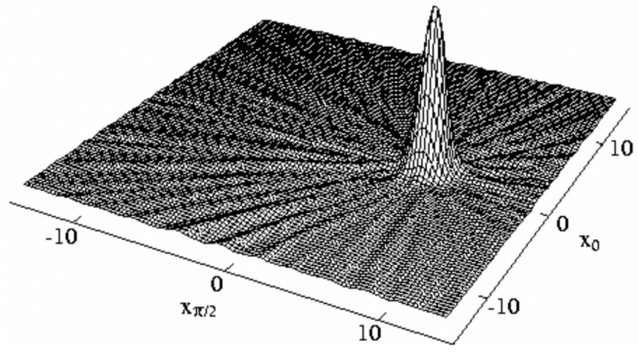
α is the displacement.

$$z = re^{i\phi}$$

↳ Sq. parameter

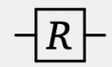
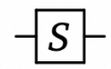
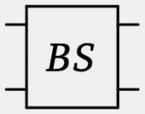
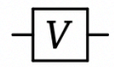
$|z\rangle \rightarrow$ squeezed state





arXiv: 1804.3159

State family	Displacement	Squeezing
Vacuum state $\|0\rangle$	$\alpha = 0$	$z = 0$
Coherent states $\|\alpha\rangle$	$\alpha \in \mathbb{C}$	$z = 0$
Squeezed states $\|z\rangle$	$\alpha = 0$	$z \in \mathbb{C}$
Displaced squeezed states $\|\alpha, z\rangle$	$\alpha \in \mathbb{C}$	$z \in \mathbb{C}$
$\hat{x}$ eigenstates $\|x\rangle$	$\alpha \in \mathbb{C}$, $x = 2\sqrt{\frac{\hbar}{2}}\mathrm{Re}(\alpha)$	$\phi = 0$, $r \to \infty$
$\hat{p}$ eigenstates $\|p\rangle$	$\alpha \in \mathbb{C}$, $p = 2\sqrt{\frac{\hbar}{2}}\mathrm{Im}(\alpha)$	$\phi = \pi$, $r \to \infty$

Gate	Unitary	Symbol
Displacement	$D_i(\alpha) = \exp(\alpha \hat{a}_i^\dagger - \alpha^* \hat{a}_i)$	
Rotation	$R_i(\phi) = \exp(i\phi \hat{n}_i)$	
Squeezing	$S_i(z) = \exp(\frac{1}{2}(z^* \hat{a}_i^2 - z\hat{a}_i^{\dagger 2}))$	
Beamsplitter	$BS_{ij}(\theta, \phi) = \exp(\theta(e^{i\phi}\hat{a}_i\hat{a}_j^\dagger - e^{-i\phi}\hat{a}_i^\dagger\hat{a}_j))$	
Cubic phase	$V_i(\gamma) = \exp\left(i\frac{\gamma}{3\hbar}\hat{x}_i^3\right)$	

Bose Hubbard Model with CVs
(arXiv:1801.06565)

- For fermionic systems, like Ising model, the qubit approach is generally preferred but for models with bosonic degrees of freedom (where the local Hilbert space dimension is infinite), the more natural setting is one of oscillator (qumodes). Suppose, we consider the famous Bose-Hubbard model where the H is given by:

$$H = J\sum_{\langle ij\rangle} a_i^\dagger a_j + \frac{1}{2}U\sum_i \hat{n}_i(\hat{n}_i - 1)$$

where we have used create /annihilation operators and the number operators. The first term denotes the hopping of bosons between neighbouring sites and second term is the on-site potential term.

Two-site model

$$H = J\left(\hat{a}_1^\dagger \hat{a}_2 + \hat{a}_2^\dagger \hat{a}_1\right) + \frac{U}{2}\left(\hat{n}_1^2 - \hat{n}_1 + \hat{n}_2^2 - \hat{n}_2\right)$$

hopping; on-site

Use Lie-Product formula:

$$e^{A+B} = \lim_{N\to\infty}\left(e^{A/N} e^{B/N}\right)^N$$

We can write

$$e^{-iHt} = \left[e^{\frac{-iJt}{K}\left(a_1^\dagger a_2 + a_2^\dagger a_1\right)} e^{\frac{-iUt}{2K}\hat{n}_1^2} - ( ) - ( ) - ( )\right]^K + O\left(t^2/K\right)$$

$e^{\frac{-iJt}{K}\left(a_1^\dagger a_2 + a_2^\dagger a_1\right)}$: BS_{12}; $e^{\frac{-iUt}{2K}\hat{n}_1^2}$: K_1; K_2; R_1, R_2

- We can write the time-evolution operator as:

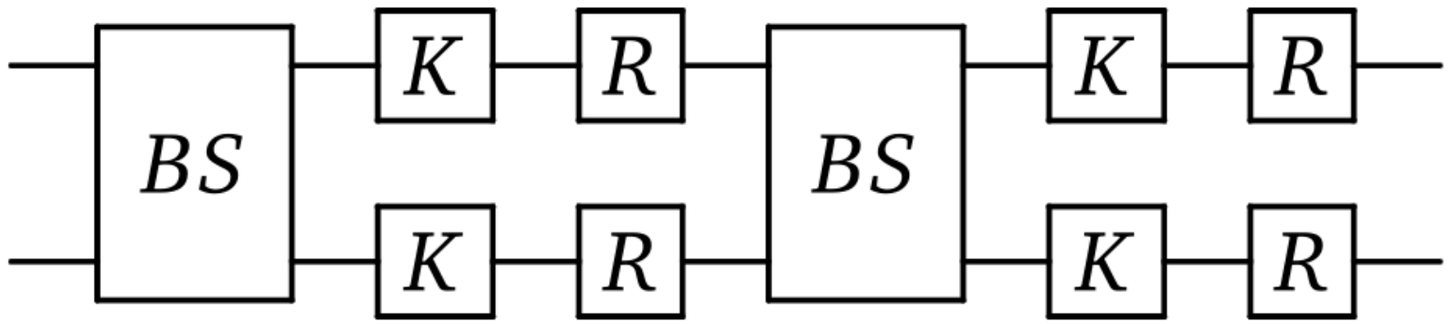
$$e^{iHt} = \left[BS\left(\theta,\phi\right)\left(K(r)R(-r)\otimes K(r)R(-r)\right)\right]^N + \mathcal{O}\left(t^2/N\right) \quad ;$$

$$\theta = -Jt/N, \phi = \pi/2,\ r = -Ut/2N$$

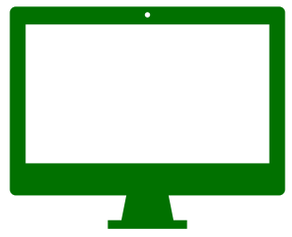
where BS is the beam-splitter gate, K is the Kerr gate (non-Gaussian), and R is the rotation gate. These gates are qumodes equivalent of the qubit gates we saw before. For example, $K(\kappa) = \exp(i\kappa\hat{n}^2)$. Constructing these gates are major undertaking in quantum photonics labs where the photon is modelled as an oscillator.

Time evolution

- Two steps of evolution can be achieved by the following circuit.



- Let’s try it out using Xanadu’s Strawberry Fields photonics simulator.



Not just 1!

- In the example we just considered, we had one set of creation and annihilation operator i.e., $a, a^\dagger$ at each lattice site. The subscript labelled the site they belonged to.

- But we can have more than one QHO at each site. Consider that there are two sets of Bose operators at each site. Let us denote them by a, b respectively. In this case, we then have $\hat{n} = a^\dagger a + b^\dagger b$. It turns out that we can define operators such as

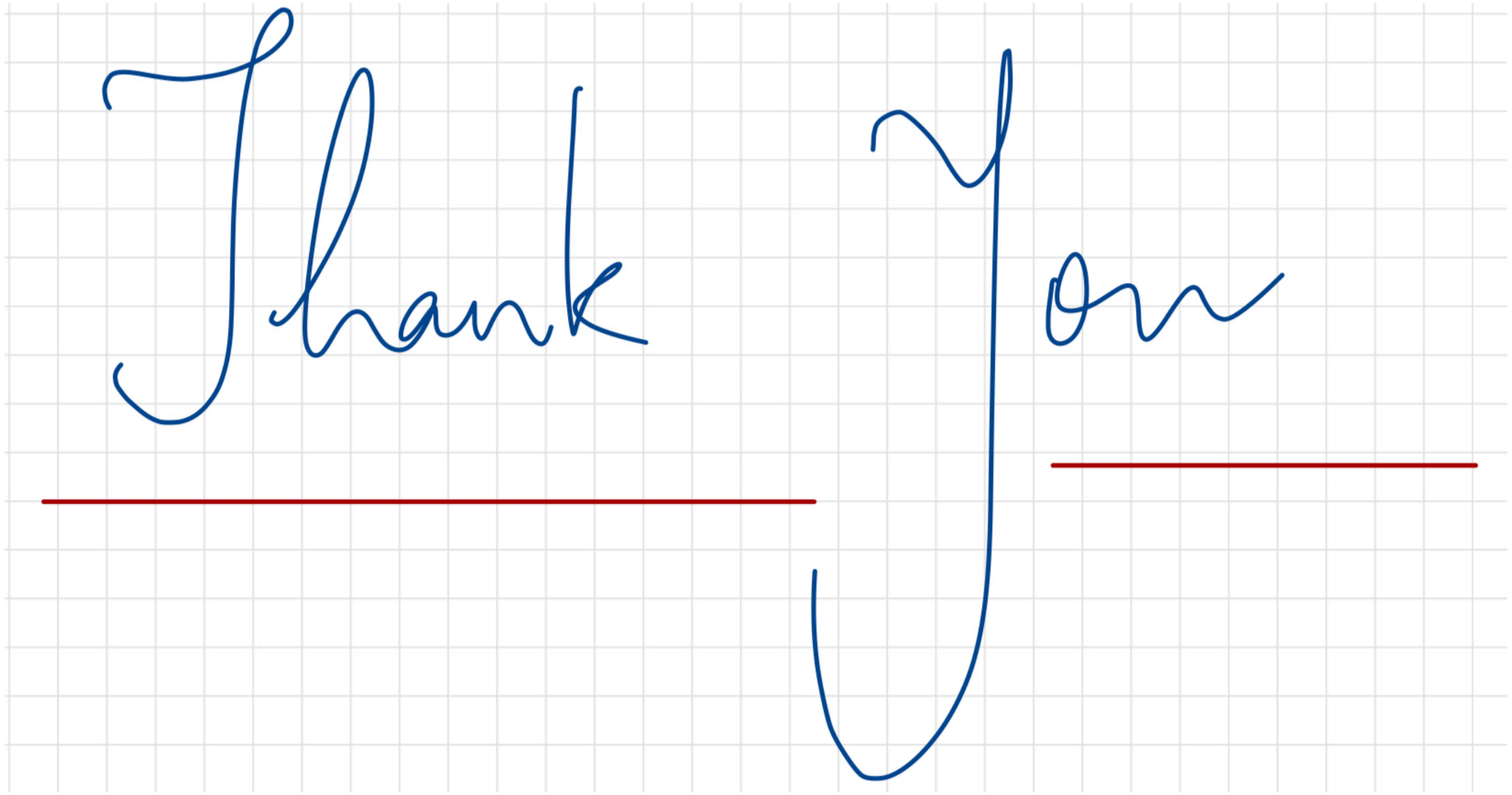
  $K_+ := a^\dagger b^\dagger$, $K_- := ba$, $K_3 := \frac{1}{2}(n \hat{+} \mathbb{I})$ and some of the interesting physical Hamiltonians can be written these operators. These operators form representation theory of $\mathfrak{su}(1,1)$ algebra. These might provide better ways to approach quantum computing of models with bosonic d.o.f and this is also equally interesting to pursue.

- In addition, there are also hybrid based approach which makes use of both qubits and continuous variables.

Conclusions

- We are entering a new era where as quantum computers become more capable, we can start solving ‘some’ problems not possible with current computers. However, this is not anytime soon. Since, QM is quite restrictive unlike classical computing, the progress might not be smooth.

- For now, VQE+variants is sort of state-of-the-art. This will improve in coming decade with more qubits (with error-correction) and better algorithms.


Thank You

Backup: Holevo bound

Consequently, although a quantum state of n qubits can be thought to represent a large amount of information, in the sense that the state is specified by $2^n - 1$ complex numbers, in fact, such a state can communicate at most n bits of decodable information.