Can quantum computing help us to better understand quantum fields?

Raghav G. Jha

Jefferson Lab

Sorry for the title.

- Most likely I won’t say anything about the title but we will make necessary steps towards it.

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

Class NP

Bridges connecting islands

You are given a list of islands connected by bridges and you want a tour that visits each island exactly once. The famous "seven bridges of Konigsberg" deemed unsolvable by Euler. However, if given a solution to this problem, it can be verified in polynomial time. This belongs to NP. A problem is said to be NP-hard if everything in NP can be transformed in polynomial time into it even though it may not be in NP.

A problem is said to be NP-complete if it is in both NP and NP-hard.

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

Paradigm shift

- We need to formally revise the definition of computer. 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 which uses classical information theory?

- Originated in the 1970s! Made popular by Feynman in 1980s.

Can we solve all problems?

- 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.

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

Quantum Physics

[Submitted on 23 Jan 2023]

Notes on Quantum Computation and Information

Raghav G. Jha

We discuss fundamentals of quantum computing and information - quantum gates, circuits, algorithms, theorems, error correction, and provide collection of QISKIT programs and exercises for the interested reader.

Comments: Suggestions, comments, and corrections are very welcome!
Subjects: **Quantum Physics (quant-ph)**; High Energy Physics - Lattice (hep-lat); Computational Physics (physics.comp-ph)
Cite as: arXiv:2301.09679 **[quant-ph]**
(or arXiv:2301.09679v1 **[quant-ph]** for this version)
https://doi.org/10.48550/arXiv.2301.09679

Submission history

From: Raghav Govind Jha [view email]
[v1] Mon, 23 Jan 2023 19:18:43 UTC (619 KB)

Outline

- Why quantum computation and which problems can it solve?
- Definitions: qubits, qudits, and qumodes
- Quantum gates with QISKIT demonstration
- Variational Quantum Eigensolver (VQE) for anharmonic oscillator
- Scalar field theory
- O(3) non-linear sigma model with qubits
- Time evolution circuits with quantum gates
- Qumodes - Bose-Hubbard model using SF simulator
- Conclusions

Tensor Networks

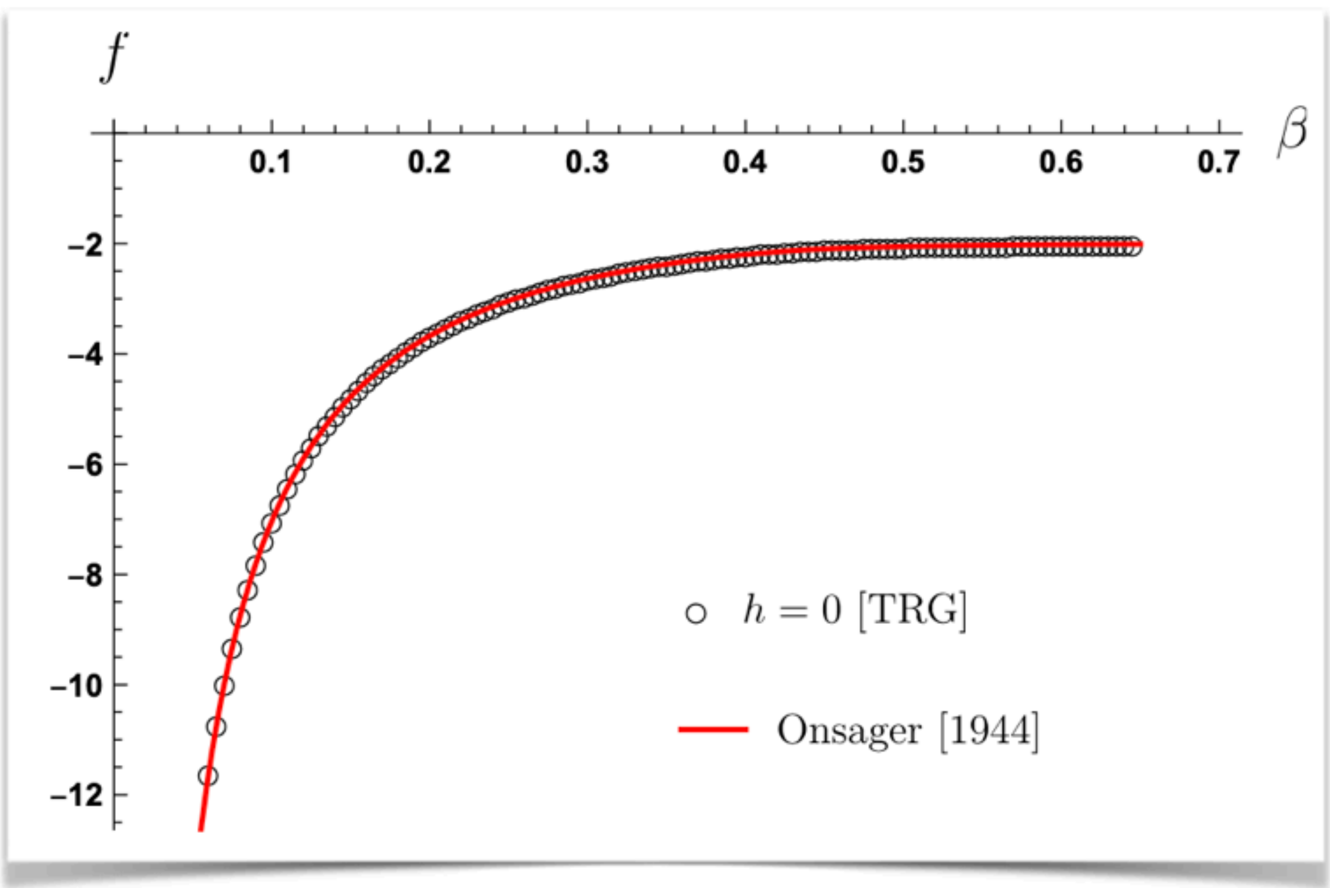
- Hamiltonian version: Approximate the ground state i.e., $|\psi\rangle = \sum_{i_1,\cdots,i_N} C_{i_1,\cdots,i_N} | i_1, \cdots, i_N\rangle$ of a model with local Hamiltonian of N spins in fewer coefficients than 2^N, O(N).

- 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).

Ising Model

- Reproduce the interesting Physics in less than a minute of computer time using tensor methods. Find code if interested here: https://github.com/rgjha

$$f(\beta) = -\frac{1}{\beta}\left(\ln(2) + \frac{1}{8\pi^2}\int_0^{2\pi}\int_0^{2\pi} \ln\left[2\cosh^2(2\beta) - \sinh(2\beta)\cos(\phi_1) - \sinh(2\beta)\cos(\phi_2)\right] d\phi_1 d\phi_2 \right)$$



Scalability is a 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.

- Go 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

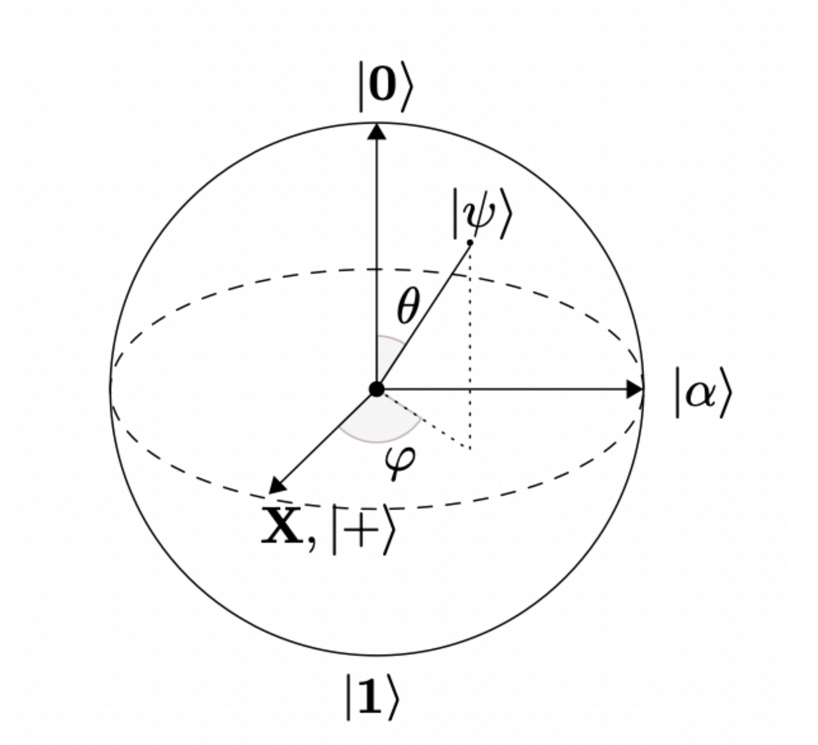
Approaches

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

- **Continuous-variable quantum computing**: Use of continuous variables (harmonic oscillator) to carry out state preparation, time evolution, and measurements (sort of Analog version!)

States

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



Quantum gates (unitary!)

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

$$|+\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big)$$

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

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

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

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

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

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

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

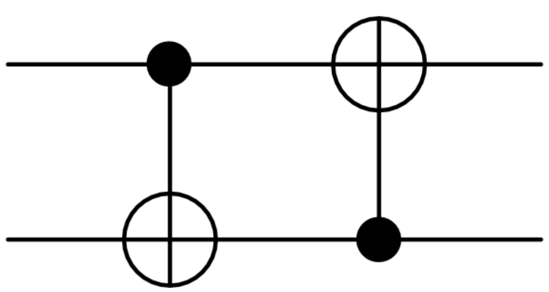
Classical vs. Quantum

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

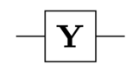
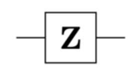
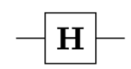
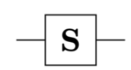
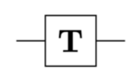
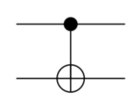
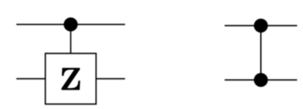
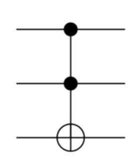
$\|A\rangle$	$\|B\rangle$	$\|A\rangle$	$\|A \oplus B\rangle$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

CNOT gate (aka CX gate)



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

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)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Two “special” features of QM

- Superposition
- Entanglement (Example of two and three qubits)

QISKIT implementation

- Open-source SDK developed by IBM which acts as a simulator. Think of this as how airline pilots train in a simulator before real flights. The programs can be sent to IBM devices if you have an account. For learning purposes, QISKIT is good enough.

- You can install on your laptop and play around. I will show you some examples. Many working codes can be found in the arXiv article.

GHZ state

```
#!pip install qiskit ipywidgets
# We will now construct GHZ state
from qiskit import *
from qiskit.quantum_info import *
from qiskit.visualization import *
qn = 3 # three-qubit GHZ

circ = QuantumCircuit(qn)
circ.h(0)
circ.cx(0, 1)
circ.cx(0, 2)
circ.draw()

ghz = Statevector.from_instruction(circ)
display(array_to_latex(ghz, prefix="\\text{Statevector} = "))
```

Bell state cont.

```
# Creating a circuit with 4 quantum bits and 2 classical bits
qc = QuantumCircuit(4,2)
```

```
qc.h(0)
qc.cx(0,1)
```

Example of measurement

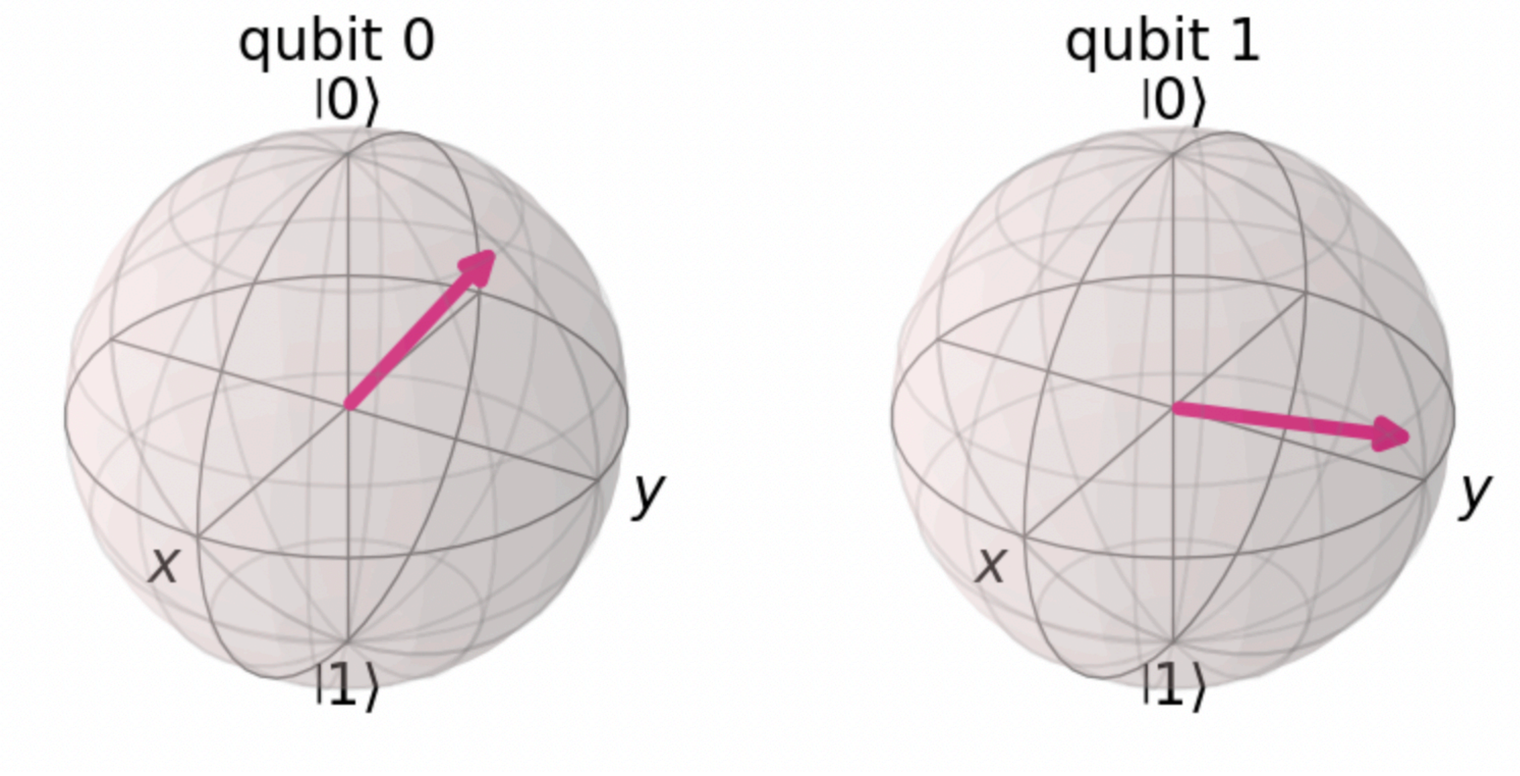
```
qc = QuantumCircuit(2,2)
qc.h(0)
qc.cx(0,1)

qc.measure(0,0)
qc.measure(1,1)

qc.draw()

#counts = execute(qc,
Aer.get_backend('qasm_simulator'),
shots=1024).result().get_counts()
#plot_histogram(counts)
```

Visualize!



A random two-qubit state

Quantum Algorithms

- One of the most popular algorithms in the NISQ-era is the variational quantum eigensolver (VQE). This is actually a hybrid quantum/classical algorithm.

- The steps are:

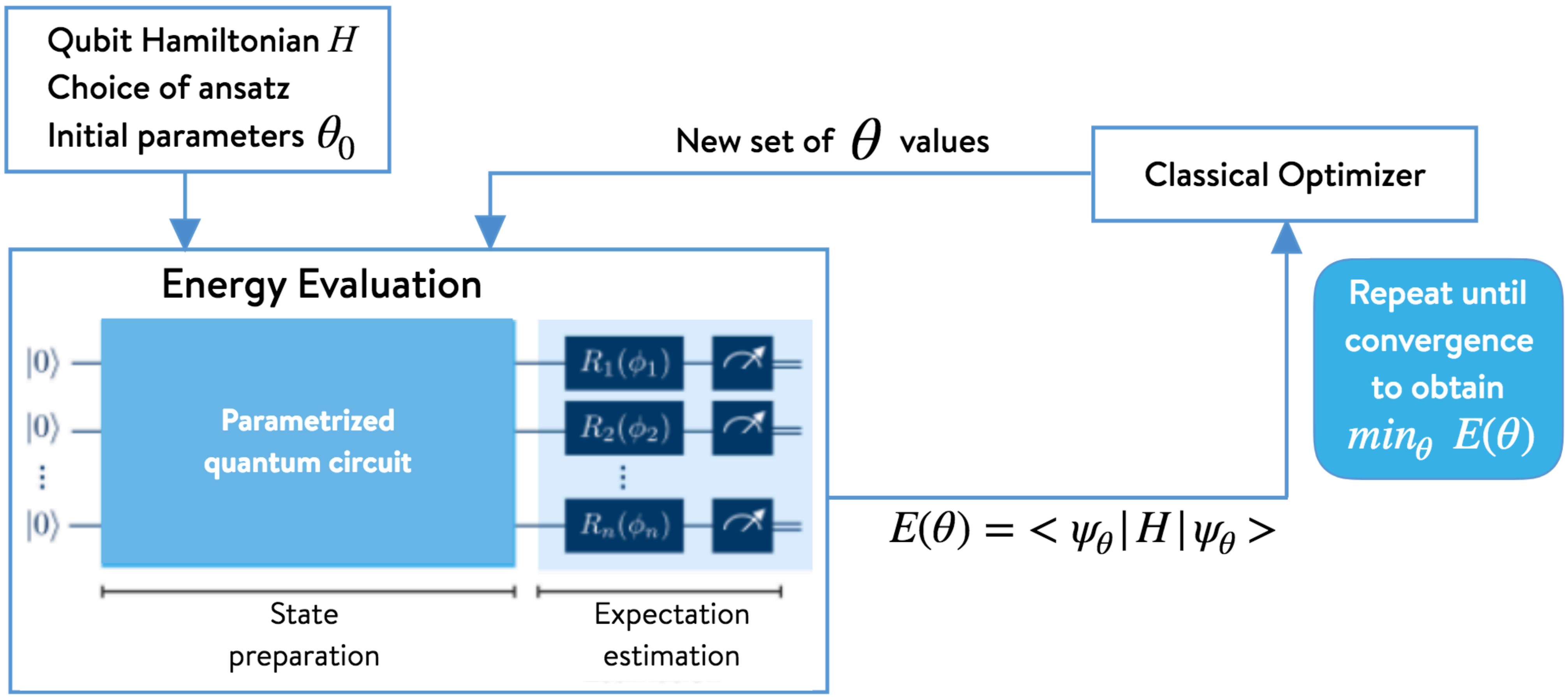
  1. Prepare initial state on QC i.e., $|0\rangle$

  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

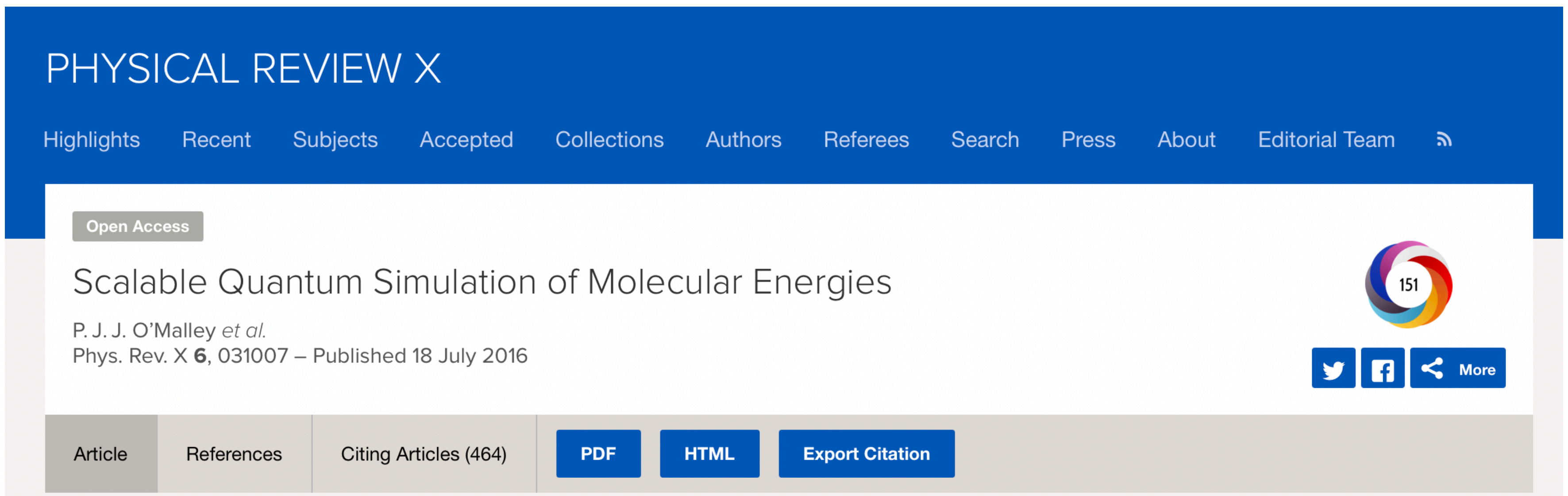


Complete basis set

- Pauli matrices are special.

VQE continued ..

- We can write the Hamiltonian as a $2^n \times 2^n$ matrix using $a, a^\dagger$. Then using the fact that every such matrix can be written entirely in terms of X, Y, Z, I, we decompose it in Pauli strings [strings of Paulis]. This method has been extensively used to study various molecules in quantum chemistry like H_2, BeH_2 etc.



- With some fixed coupling of cubic term, we get our AHO Hamiltonian as:

$$\begin{aligned}\hat{H}' =& 4\mathbb{1}_8 - 0.152956\mathbb{1}_4 X - 0.5\mathbb{1}_4 Z - 0.12289 * \mathbb{1}XX - 0.0629948\mathbb{1}YY - 1\mathbb{1}Z\mathbb{1} + 0.0237627\mathbb{1}ZX - 0.0280252X\mathbb{1}X \\ &- 0.0561195XXX + 0.0287333XYY + 0.0107047XZX - 0.0280252Y\mathbb{1}Y - 0.0287333YXY - 0.0561195YYX \\ &+ 0.0107047YZY - 2Z\mathbb{1}_4 + 0.0872346Z\mathbb{1}X + 0.0842295ZXX + 0.041655ZYY + 0.0207442ZZX\end{aligned}$$

Anharmonic oscillator - Set up

```
import numpy as np
import math
import time
import warnings
import itertools
from qiskit import Aer
from qiskit.algorithms import VQE
from qiskit.opflow import MatrixOp
from qiskit.opflow import X, Y, Z, I
from qiskit.algorithms.optimizers import SLSQP
from qiskit.algorithms.optimizers import COBYLA
from qiskit.circuit.library import EfficientSU2
from qiskit.utils import algorithm_globals, QuantumInstance
from qiskit.visualization.array import array_to_latex
```

```
Hps = (4 * I ^ I ^ I)+(-0.152955 * I ^ I ^ X)+(-0.5 * I ^ I ^ Z)+(-0.12289 * I ^ X ^
    X)+(-0.0629948 * I ^ Y ^ Y)+(-1 * I ^ Z ^ I)+(0.0237627 * I ^ Z ^ X)+(-0.0280252 * X ^ I
    ^ X)+(-0.0561195 * X ^ X ^ X)+(0.0287333 * X ^ Y ^ Y)+(0.0107047 * X ^ Z ^ X)+(-0.0280252
    * Y ^ I ^ Y)+(-0.0287333 * Y ^ X ^ Y)+(-0.0561195 * Y ^ Y ^ X)+(0.0107047 * Y ^ Z ^
    Y)+(-2 * Z ^ I ^ I)+(0.0872346 * Z ^ I ^ X)+(0.0842295 * Z ^ X ^ X)+(0.041655 * Z ^ Y ^
    Y)+(0.0207442 * Z ^ Z ^ X)
```

Anharmonic oscillator

```
start_time = time.time()
nbits =3
var_form = EfficientSU2(nbits, su2_gates=['ry'], entanglement="full", reps=1)
rngseed = 5
warnings.filterwarnings("ignore")
backend = Aer.get_backend("statevector_simulator")
q_instance = QuantumInstance(backend, seed_transpiler=rngseed,
seed_simulator=rngseed)
#optimizer = SLSQP(maxiter=600)
optimizer = COBYLA(maxiter=600)

# Run the VQE
vqe = VQE(ansatz=var_form,optimizer=optimizer,quantum_instance=q_instance)
ret = vqe.compute_minimum_eigenvalue(Hps)
vqe_result = np.real(ret.eigenvalue)
print("VQE Result:", vqe_result)
exact = 0.5 - (11/8)*g*g -(465/32)*g*g*g*g
print ("Exact result for cubic oscillator upto O(g^4) is", exact)
print ("Error is", np.round(abs((exact-vqe_result)/exact)*100,10), "percent")
end_time = time.time()
runtime = end_time-start_time
print('Program runtime: ',runtime, "seconds")
```

•

O(3) model in 1+1

arXiv: 2210.03679 [quant-ph]

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

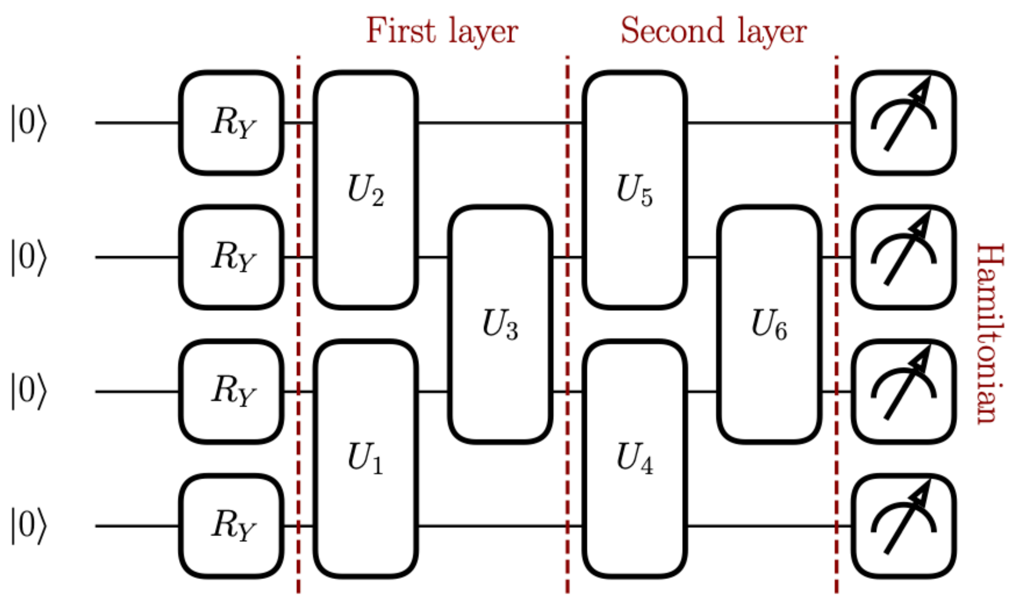
$$\hat{H} = \frac{1}{2\beta} \sum_i \mathsf{L}_i^2 - \beta \sum_{\langle i,j \rangle} \mathsf{n}_i \cdot \mathsf{n}_j,$$

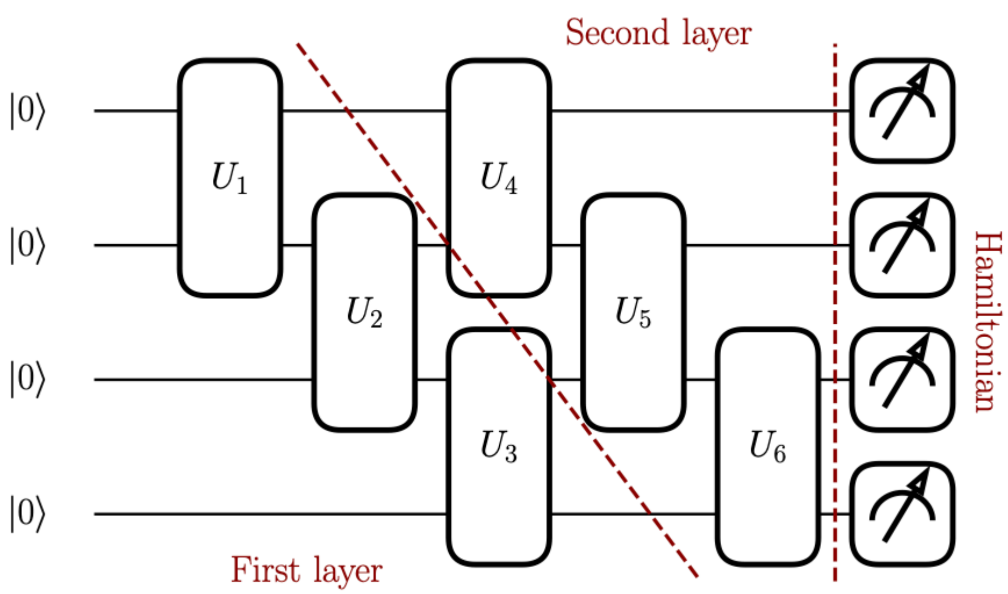
- We can construct this matrix for some fixed value of $l_{\text{max.}}$

  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

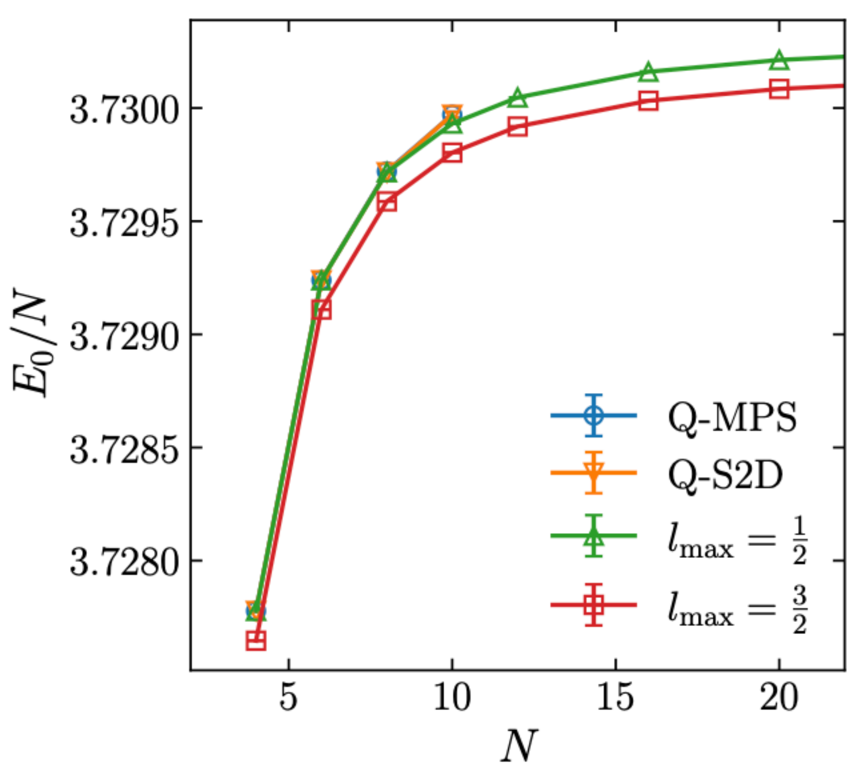
arXiv: 2210.03679 [quant-ph]



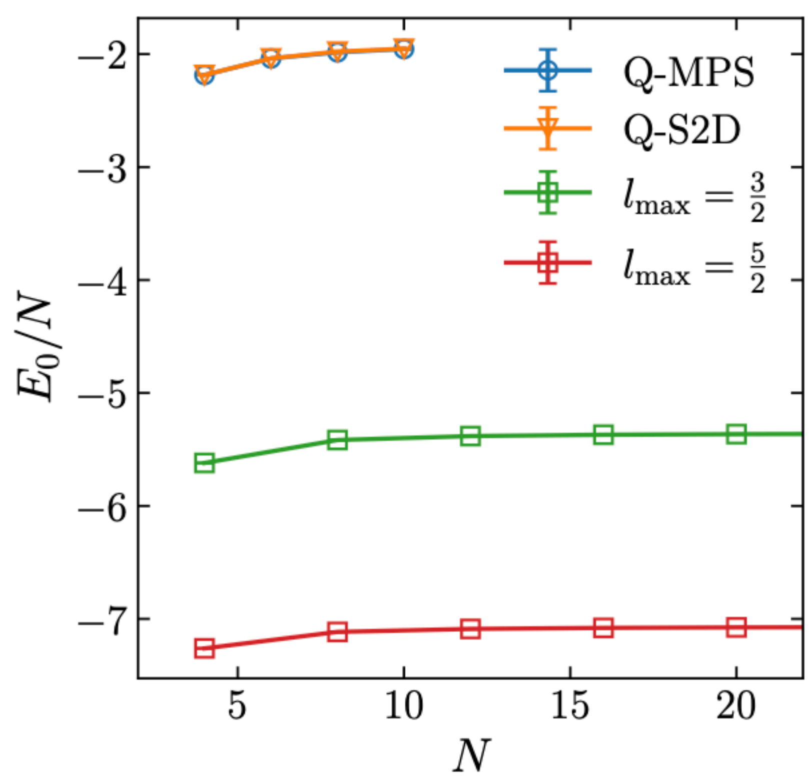


Results

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



$\beta = 0.1$



$\beta = 10$

Time evolution of quantum systems

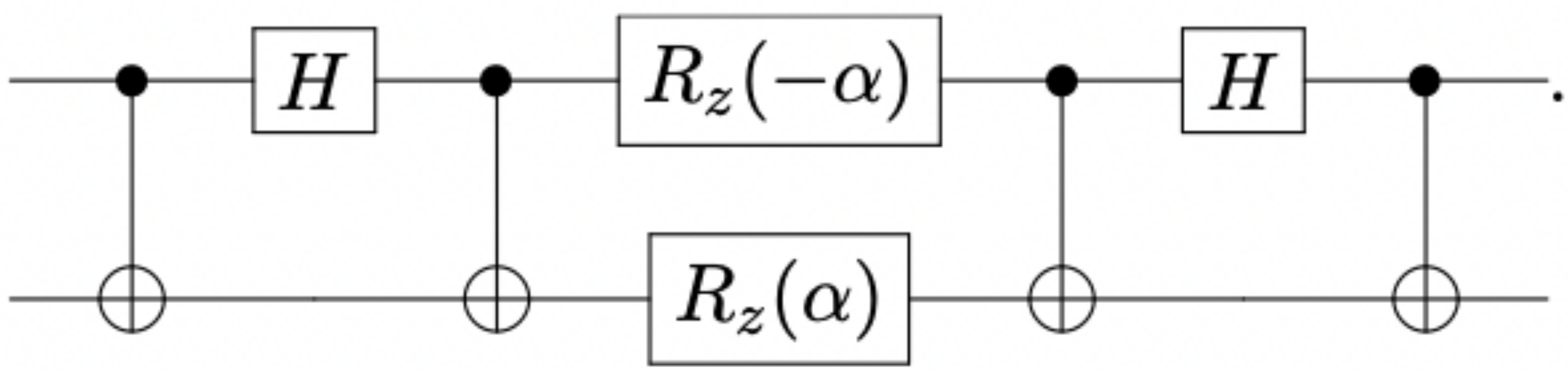
- One of the problems where theoretical physicists would like to apply QC is to understand the time-evolution of some complicated quantum many-body system. Suppose, we have spin-1/2 particle each on two sites with some H below, we would need two qubits to initialise the state say, $|00\rangle$. Now suppose 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.

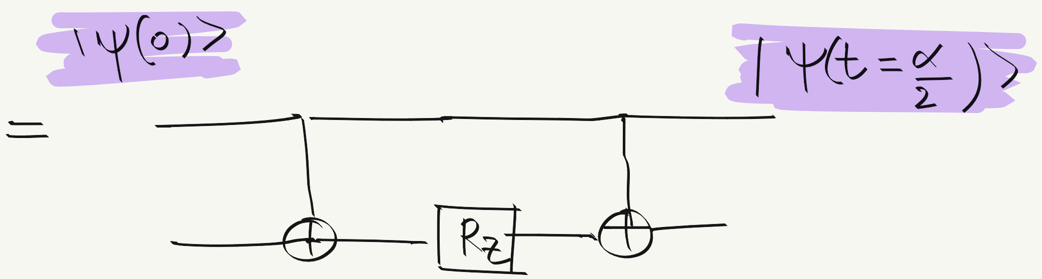
Time evolution of quantum systems

- 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!



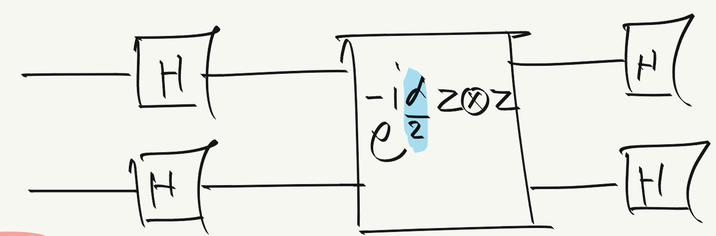
Basic idea

$$e^{-i\frac{\alpha}{2} Z\otimes Z} =$$



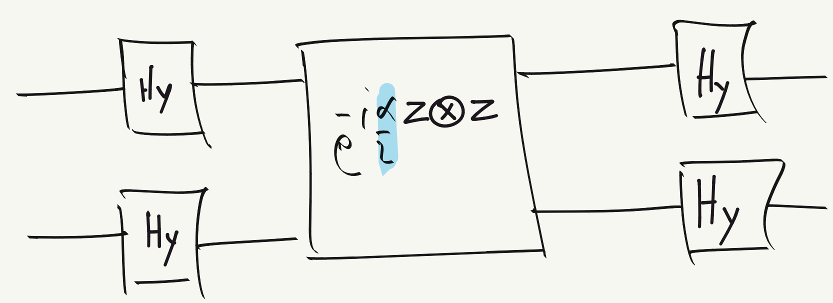
$$e^{-i\frac{\alpha}{2} Z}$$

$$e^{-i\frac{\alpha}{2} X\otimes X} =$$



Since $X = H\cdot Z\cdot H$

$$e^{-i\frac{\alpha}{2} Y\otimes Y} =$$



Since $Y = H_y\cdot Z\cdot H_y$, where $H_y = \begin{pmatrix} 1 & -i \\ i & -1 \end{pmatrix}\frac{1}{\sqrt{2}}$

Simple demonstration

```
circ = QuantumCircuit(2,2)
t = 0.1
circ.h(0)
circ.h(1)
circ.cx(0,1)
circ.rz(t,1)
circ.cx(0,1)
circ.h(0)
circ.h(1)
circ.draw()

# Now since we see that we are happy with the circuit. Let’s measure!


circ.measure(0,0)
circ.measure(1,1)

counts = execute(circ, Aer.get_backend('qasm_simulator'),
shots=1024).result().get_counts()
plot_histogram(counts)
```

Using qumodes

- As mentioned before, there are other ways to approach quantum computing not just 2-state (or qubit) methods. We can also use a quantum mechanical HO. There are now simulators for qumodes (or continuous variables as well) 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 $\lvert\alpha\rangle$ Squeezed states $\lvert z\rangle$ Number states $\lvert n\rangle$	Pauli eigenstates $\lvert 0/1\rangle, \lvert\pm\rangle, \lvert\pm i\rangle$
Common gates	Rotation, Displacement, Squeezing, Beamsplitter, Cubic Phase	Phase Shift, Hadamard, CNOT, T Gate

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

$$H = J\left(\overbrace{\hat a_1^{+} a_2 + a_2^{+} a_1}^{\text{hopping}}\right) + \frac{1}{2}U\left(\hat n_1^2 \underbrace{- \hat n_1 + \hat n_2^2 -}_{\text{on-site}} \hat n_2\right)$$

Using Lie product formula:

$$e^{A+B} = \mathop{\mathrm{Lt}}_{N\to\infty}\left(e^{A/N} e^{B/N}\right)^N$$

We can write

$$e^{-iHt} = \Big[\underbrace{e^{-\frac{iJt}{K}(a_1^{+}a_2 + a_2^{+}a_1)}}_{BS_{1,2}} \underbrace{e^{-\frac{iUt}{2K}\hat n_1^2}}_{K_1} \underbrace{e^{-(\ )}}_{K_2} \underbrace{e^{(\ )}}_{R_1} \underbrace{e^{(\ )}}_{R_2}\Big]^K + \mathcal{O}\left(\frac{t^2}{K}\right)$$

- We can write the time-evolution operator as:

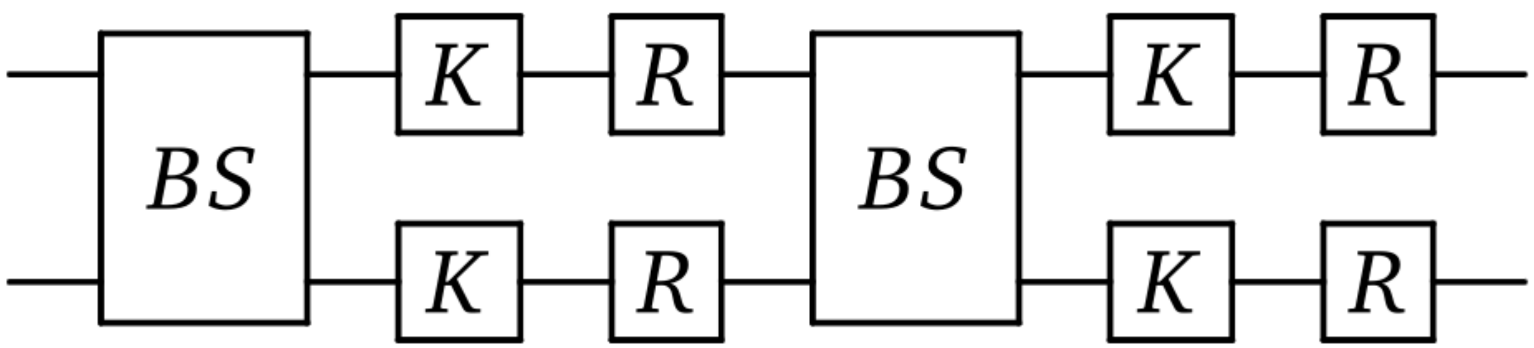
$$e^{iHt} = \left[BS(\theta,\phi)\left(K(r)R(-r) \otimes K(r)R(-r)\right)\right]^N + \mathcal{O}\left(t^2/N\right) \quad ;$$

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

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.

```python
#!pip install strawberryfields
import numpy as np
np.random.seed(11)
import strawberryfields as sf
from strawberryfields.ops import *

ham_simulation = sf.Program(2)
# Set the Hamiltonian parameters

J = 1               # hopping transition
U = 1.5             # on-site interaction
k = 30              # Lie product decomposition terms
t = 0.0             # timestep
theta = -J*t/k
r = -U*t/(2*k)

with ham_simulation.context as q:

    # Prepare the initial state
    Fock(2) | q[0]

    # Two node Hamiltonian simulation
    for i in range(k):
        BSgate(theta, np.pi/2) | (q[0], q[1])
        Kgate(r)  | q[0]
        Rgate(-r) | q[0]
        Kgate(r)  | q[1]
        Rgate(-r) | q[1]

eng = sf.Engine(backend="fock", backend_options={"cutoff_dim": 3})
results = eng.run(ham_simulation)
state = results.state
print (state)
print("P(|0, 2>) = ", state.fock_prob([0, 2]))
print("P(|1, 1>) = ", state.fock_prob([1, 1]))
print("P(|2, 0>) = ", state.fock_prob([2, 0]))

result = [state.fock_prob([0,2]), state.fock_prob([1, 1]), state.fock_prob([2, 0])]
print(np.sum(result))
```

Conclusions

- We are entering a new era (similar to lattice gauge theories on classical computers in the 1970s) 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

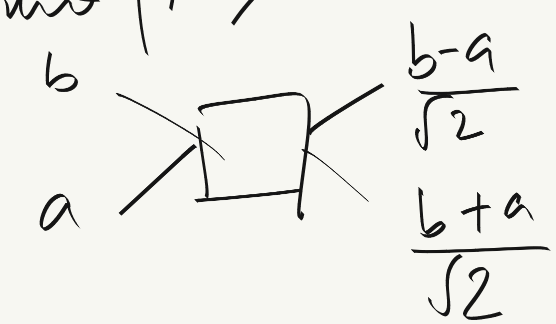
$$B\big|_{\phi=0} = e^{\theta\,(a^\dagger b - a b^\dagger)} \qquad e^{\theta(e^{i\phi}a^\dagger b - e^{-i\phi} a b^\dagger)}$$

$$B a B^\dagger = a\cos\theta + b\sin\theta$$

Note that $B|00\rangle = |00\rangle$ (no photon in either input mode = no photon in o/p modes)

$$
\begin{aligned}
B|01\rangle &= B a^\dagger |00\rangle \\
&= B a^\dagger B^\dagger B |00\rangle \\
&= a^\dagger \cos\theta + b^\dagger \sin\theta\, |00\rangle \\
&= \cos\theta\, |01\rangle + \sin\theta\, |10\rangle
\end{aligned}
$$

$\theta = \pi/4$ 50-50 BS



Other CV gates

Displacement : $D_i(\alpha) = e^{(\alpha a_i^\dagger - \alpha^* a_i)}$

Rotation : $R_i(\varphi) = e^{i\varphi \hat{n}_i}$; $\hat{n} = a^\dagger a$

Squeezing : $S_i(z) = e^{\frac{1}{2}(z^* a_i^2 - z a_i^{\dagger 2})}$

Coherent states $\equiv |\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle$

and $D(\alpha)|0\rangle$

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