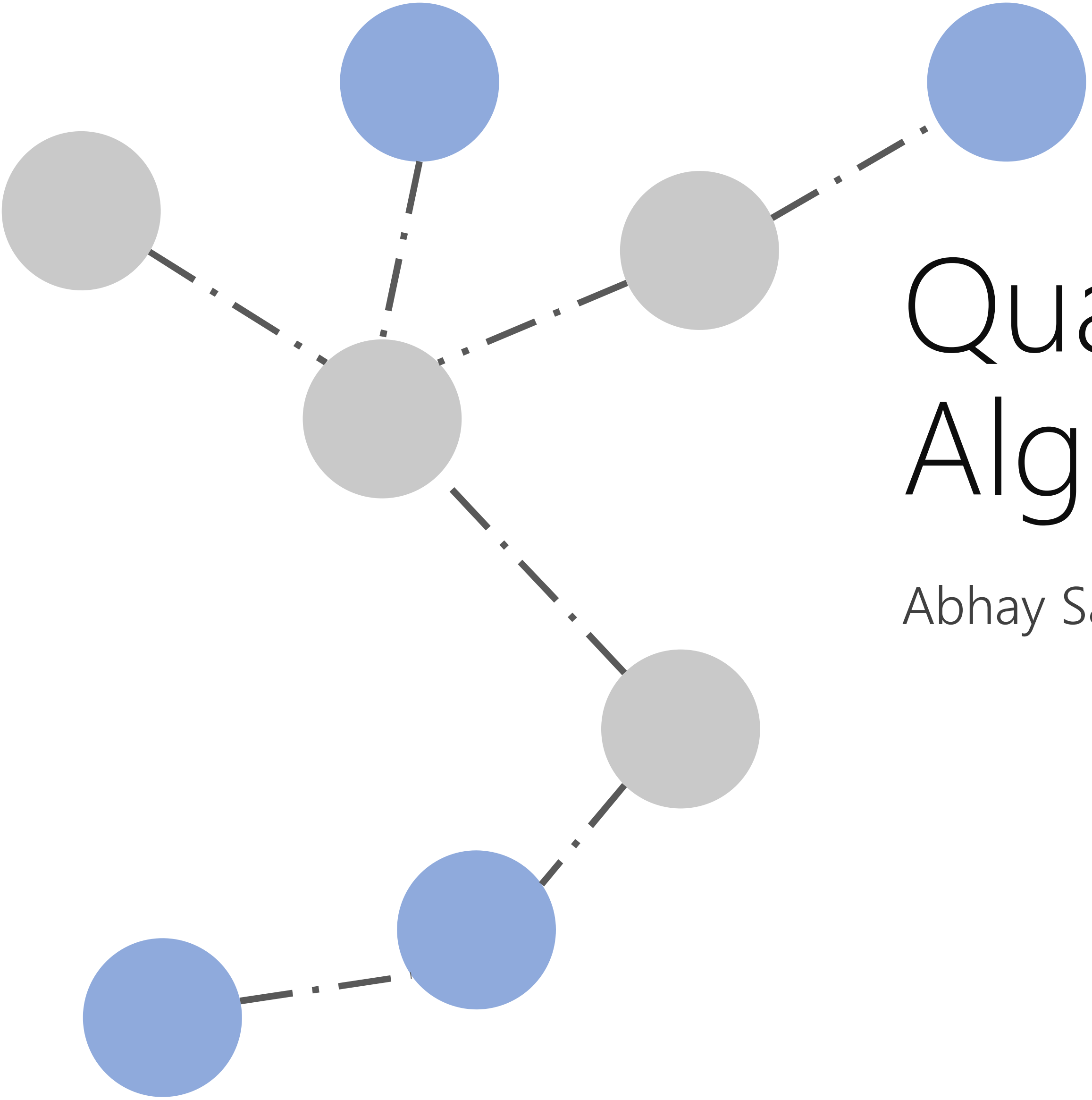


Quantum Algorithms

Abhay Saxena

classic: $O\left(\exp\left(\sqrt[3]{\frac{64}{9}n(\log n)^2}\right)\right)$

quantum: $O(n^2 \log n \log \log n)$

Shor’s Algorithm

Modular Exponentiation Function

Quantum Fourier Transformation

GCD calculation and trial to find factors

Protocol

N=pq   (find p and q, N is odd)

1. Pick a number ‘a’ that is coprime with N
2. Find the ‘order’ r of the function $a^r (mod\ N)$

$$\equiv smallest\ r\ st.\ a^r \equiv 1\ (mod\ N)$$

3. if r is even:

$$x = \ a^{r/2}\ (mod\ N)$$

if x+1 $\not\equiv$ 0 (mod N):

{p,q} contained in set {gcd(x+1,N), gcd(x-1,N)}

polynomial complexly computable by classic computer

else: find another ‘a’

N divides: $a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$

N=15

N=pq   (find p and q (3 and 5), N is odd)

1. Pick a number ‘a’ that is coprime with 15 say a=7
2. Find the ‘order’ r of the function $7^r(mod\ 15)$

$$smallest\ r\ st.\ 7^4\ (mod\ 15) \equiv 1 : \mathrm{r} = 4$$

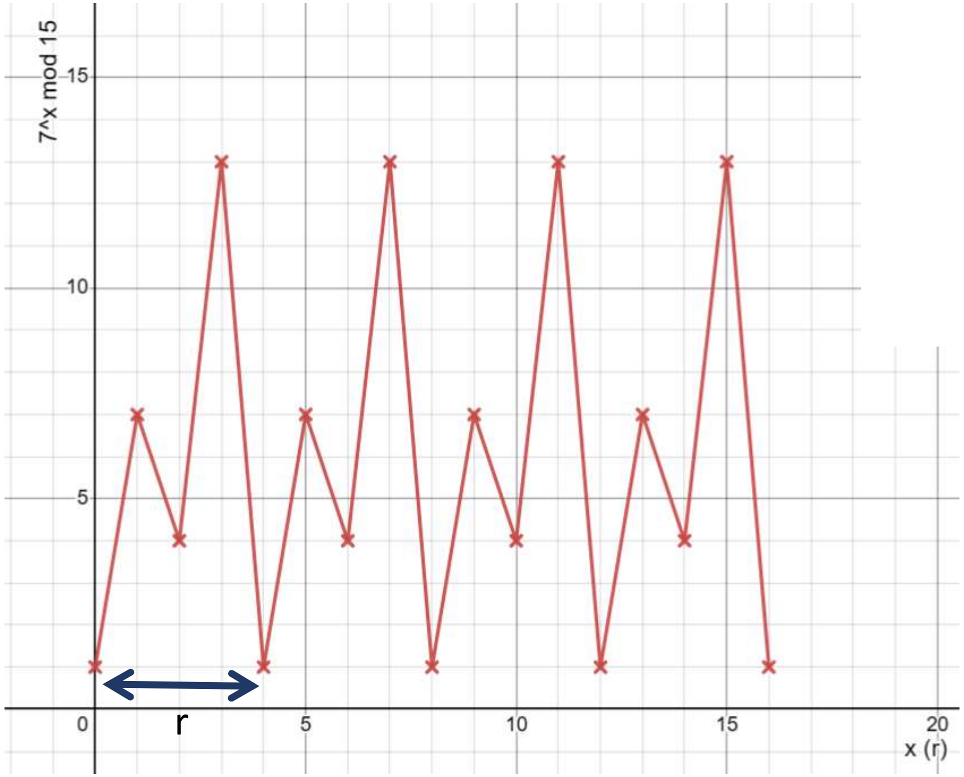
$$(\ 7^4 - 1)\ mod\ 15\ \ \equiv 0$$
$$(7^{4/2} - 1)(7^{4/2} + 1) \equiv 0$$

so 15 divides 48*50

i.e. {p,q} contained in set {gcd(x+1,N), gcd(x-1,N)}

Modular Exponentiation Function

So far the algorithm provides no advantage over classic method of simply finding the period as the problem is reduced to simple Modular Exponentiation Function and can be solved using discrete fourier Transformation. Thus, for quantum advantage we introduce Quantum Phase Estimation(QPE) using Quantum Fourier Transformation(QFT).



Quantum Fourier Transformation

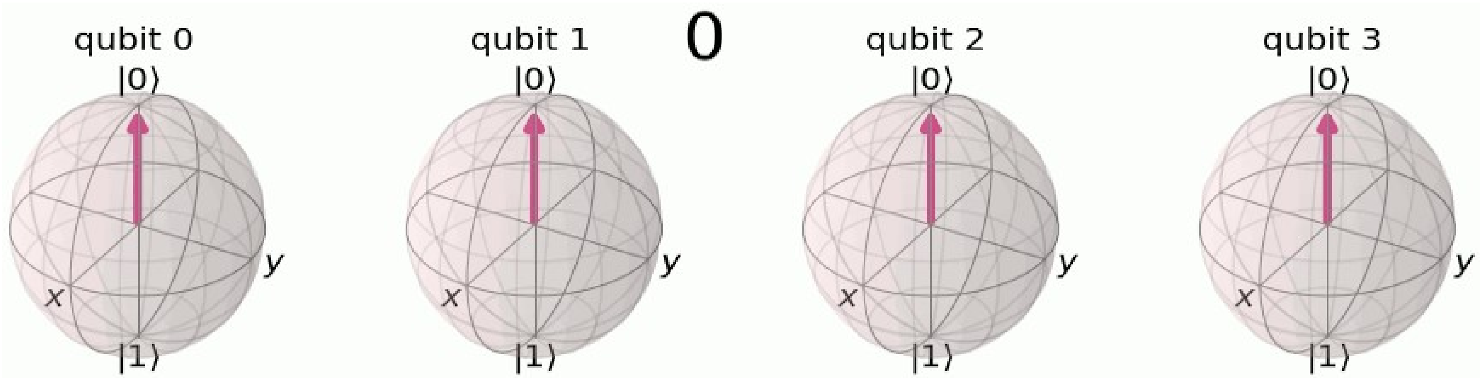
Descrete Fourier Transformation acts on a vector (x$_0$, x$_1$,…x$_n$) and maps it to the vector (x$_0$, x$_1$,…x$_n$)

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk}$$

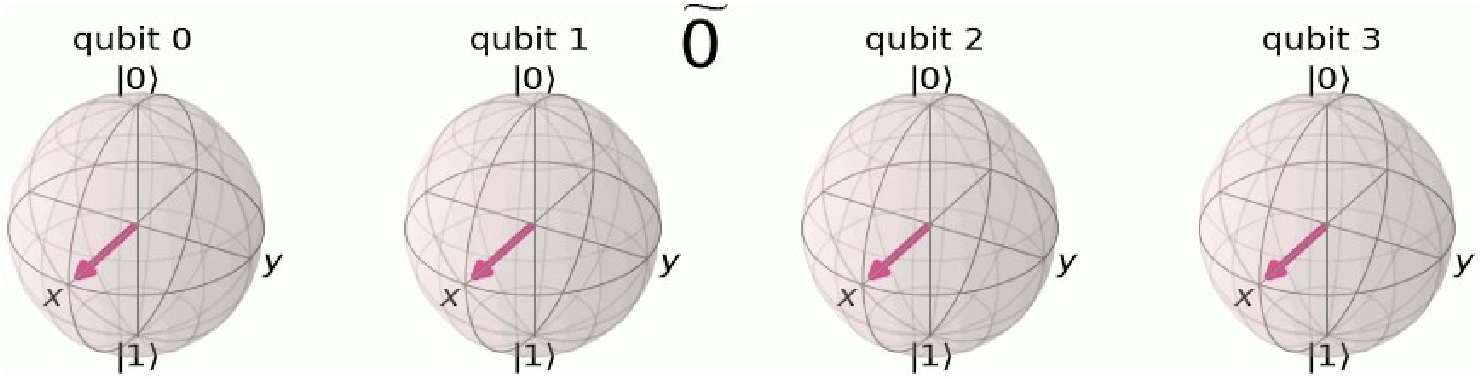
Quantum Fourier transform acts on a quantum state $|X\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$ and maps it to $|Y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$ according to formula

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk}$$

where $\omega_N^{jk} = e^{2\pi i \frac{jk}{N}}$



Computational Basis (in 0 and 1)



gif from qiskit textbook

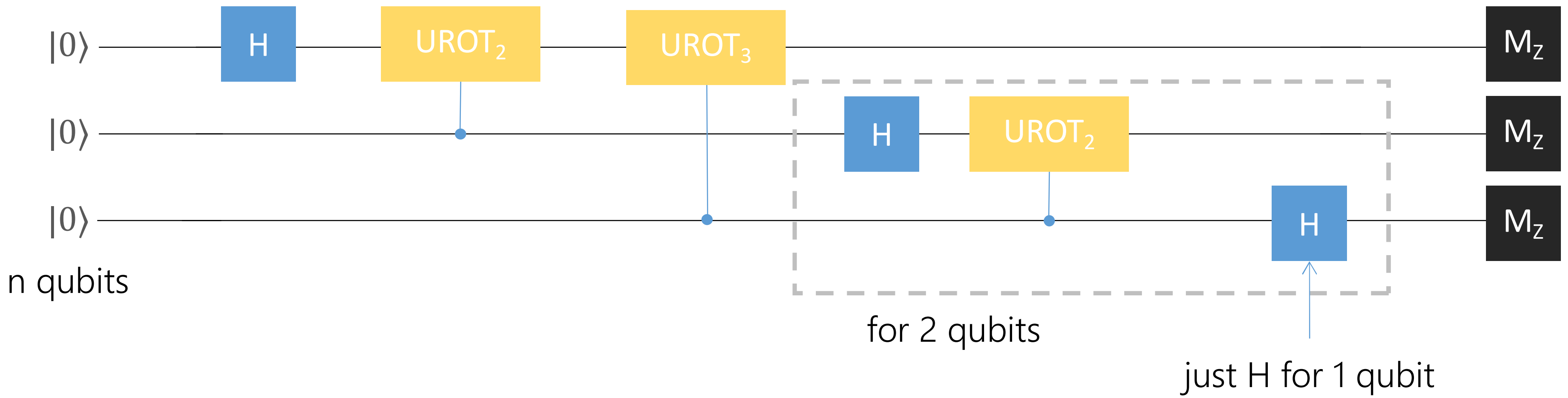
Fourier Basis (in 0 and 1)

$$for\ kth\ Fourier\ basis\ leftmost\ qubit\ is\ rotated\ by\ \frac{k}{2^n} \times 2\pi\ radians$$

Circuit for QFT

$$UROT_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

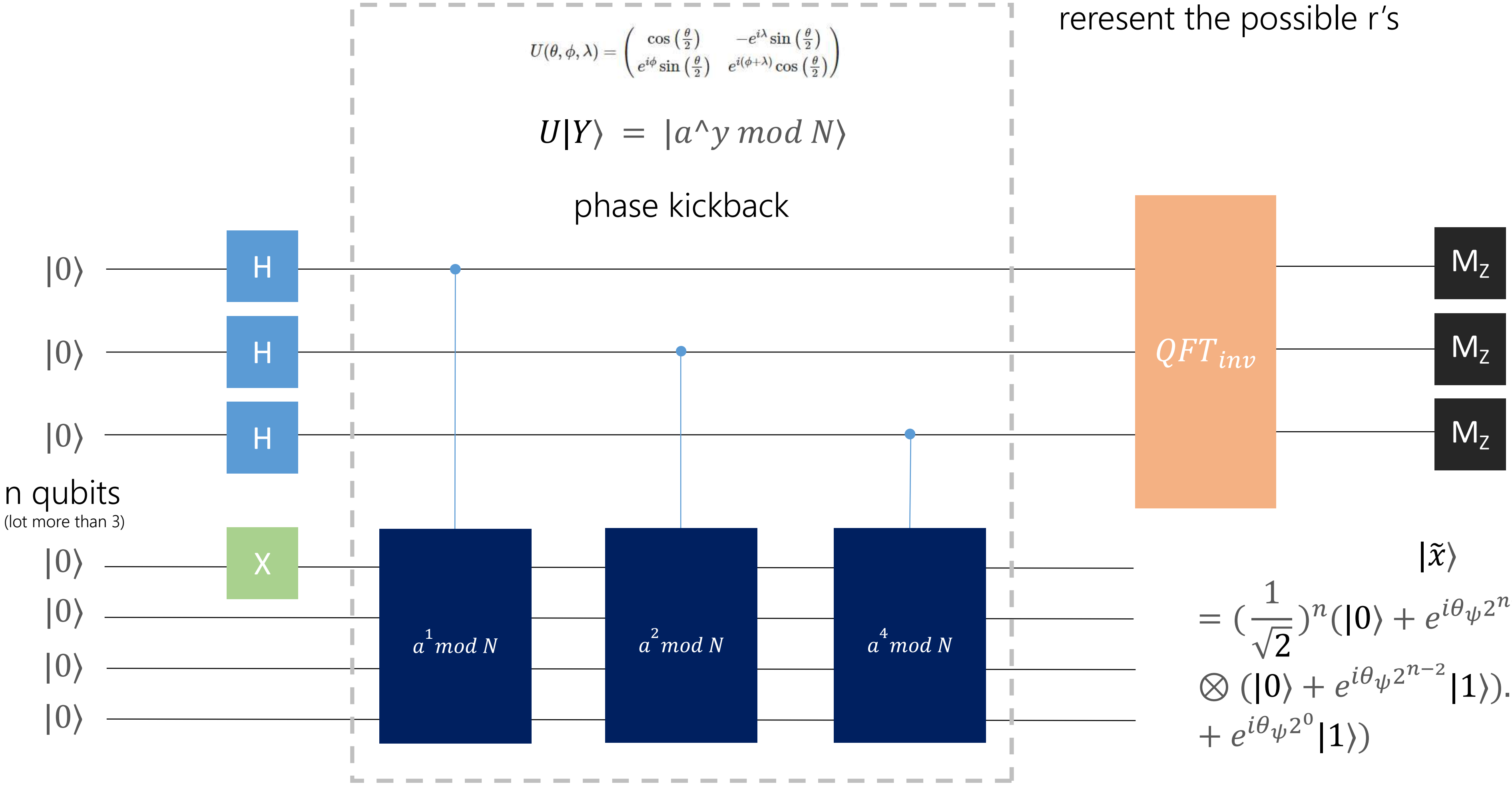
$$\omega_N^{jk} = e^{2\pi i \frac{jk}{N}}$$



$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{\frac{2\pi i x}{2^1}}|1\rangle) \otimes \frac{1}{\sqrt{N}}(|0\rangle + e^{\frac{2\pi i x}{2^2}}|1\rangle) \ldots\ldots \otimes \frac{1}{\sqrt{N}}(|0\rangle + e^{\frac{2\pi i x}{2^n}}|1\rangle)$$

Circuit for QPE

the returned output divided by 2^n will give resp. fractions denominators of which should reresent the possible r's

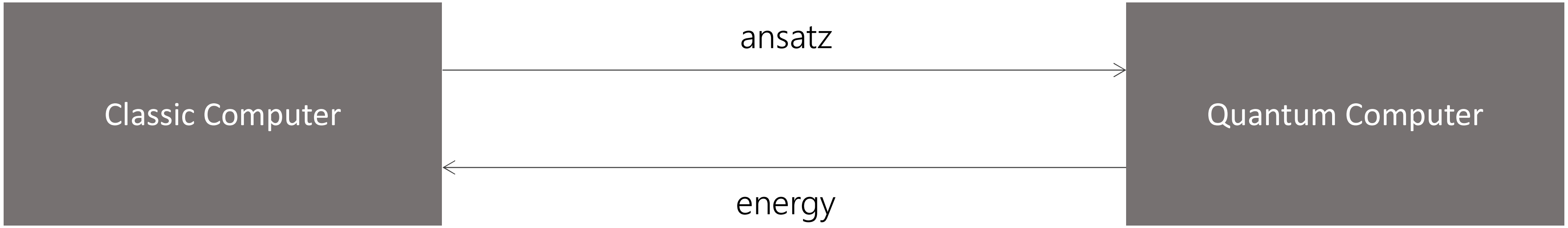


$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$U|Y\rangle = |a\hat{\ }y \; mod \; N\rangle$$

$$|\tilde{x}\rangle = \left(\frac{1}{\sqrt{2}}\right)^n \left(|0\rangle + e^{i\theta_\psi 2^{n-1}}|1\rangle\right) \otimes \left(|0\rangle + e^{i\theta_\psi 2^{n-2}}|1\rangle\right) \ldots\ldots \otimes \left(|0\rangle + e^{i\theta_\psi 2^{0}}|1\rangle\right)$$

Variational Quantum Eigensolver



Classic Computer

ansatz

energy

Quantum Computer

Variational Method

- we know that and eigenvector $|\psi_i\rangle$ of a matrix A, does not vary under transformation upto eigenvalue. $A|\psi_i\rangle = \lambda_i|\psi_i\rangle$.
- Eigenvalue of any Hermitian matrix has property $\lambda_i = \lambda_i{}^*$
- $H = \sum_{i=1}^{N} \lambda_i \, |\psi_i\rangle\langle\psi_i|$
- $\langle H\rangle_\psi = \langle\psi|H|\psi\rangle = \langle\psi|(\sum_{i=1}^{N} \lambda_i \, |\psi_i\rangle\langle\psi_i|)|\psi\rangle \; = \sum_{i=1}^{N} \lambda_i \, \langle\psi|\psi_i\rangle\langle\psi_i|\psi\rangle$

$$= \sum_{i=1}^{N} \lambda_i \, |\langle\psi|\psi_i\rangle|^2$$

so it is clear $\lambda_{min} \leq \langle H\rangle_\psi = \langle\psi|H|\psi\rangle = \sum_{i=1}^{N} \lambda_i \, |\langle\psi|\psi_i\rangle|^2$

this eqn is known as variational method for $\langle H\rangle_{\psi_{min}} = \lambda_{min}$

Variational Forms

- we select some initial guess $|\psi\rangle$ (called ansatz) for $|\psi_{min}\rangle$, find it's expectation $\langle H\rangle_{\psi}$ to update to new guess.
- VQE varies these ansatz using parameterized circuit with a fixed form. Such Circuits are called Variational Forms.
- Let te action of Variational form be represented by linear transformation $U(\theta)$
- $U(\theta)|\psi\rangle = |\psi(\theta)\rangle$ is the optimized output state. Through iterative optimization, it aims to yield expectation close to $\langle H\rangle_{\psi_{min}} = \lambda_{min}$.
- Closeness of a state to E_{gs} is measured through manhattan distance.
- n Qubit variational form would be able to generate any possible state $|\psi\rangle$

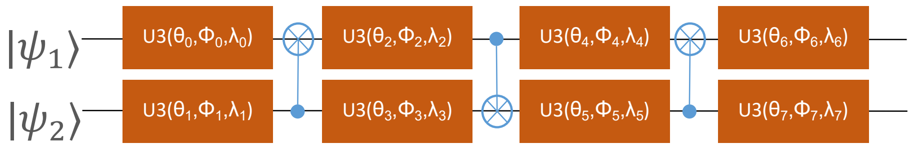
where $|\psi\rangle \in C^N \, and \, N = 2^n$.

Variational forms

- for n=1, U3 gate is a variational form capable of generating any possible state

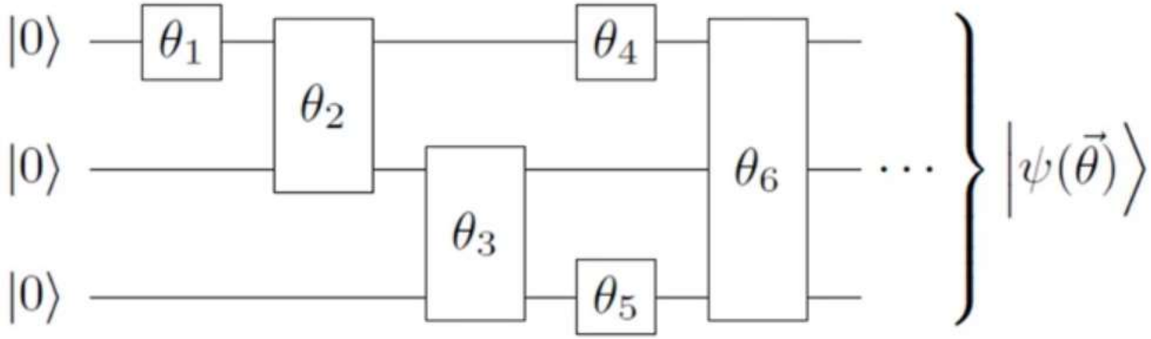
- for n=2, where two body Interactions happen, entanglement, must be considered to achieve universality.

$$U3(\theta,\phi,\lambda) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -e^{i\lambda}\sin(\frac{\theta}{2}) \\ e^{i\phi}\sin(\frac{\theta}{2}) & e^{i\lambda+i\phi}\cos(\frac{\theta}{2}) \end{pmatrix}$$



NOTE:during the optimization process, the variational form does not limit the set of attainable states over which the expectation value of Hamiltonian.This ensures that the minimum expectation value is limited only by the capabilities of the classical optimizer.

General parametrized state



Chosing Variational forms

- quantum hardware has various types of noise and so objective function evaluation (energy calculation) may not necessarily reflect the true objective function.
- Appropriate optimizer should be selected by considering the requirements of an application.
- gradient descent: each parameter is updated in the direction yielding the largest local change in energy (often gets stuck at poor local optima, high number of circuit evaluations)
- Simultaneous Perturbation Stochastic Approximation optimizer (SPSA): approximates the gradient of the objective function with only two measurements. Concurrently perturbing all the parameters in a random fashion.
- If no noise is present (Perfect simulator for VQE):
  - Sequential Least Squares Programming optimizer (SLSQP): use if objective function and the constraints are twice continuously differentiable.
  - Constrained Optimization by Linear Approximation optimizer (COBYLA): only performs one objective function evaluation per optimization iteration (and thus the number of evaluations is independent of the parameter set's cardinality).

Gradient Descent for 1 qubit sustem

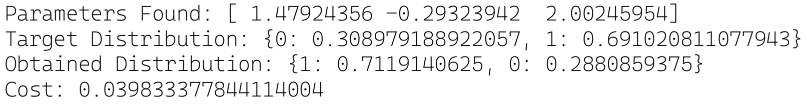
```
import numpy as np
np.random.seed(999999)
p0 = np.random.random()
target_distr = {0: p0, 1: 1 - p0}
```

```
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
from qiskit.circuit import Parameter

p = [Parameter("theta"), Parameter("phi"), Parameter("lam")]

def form(p):
    qr = QuantumRegister(1, name="q")
    cr = ClassicalRegister(1, name="c")
    qc = QuantumCircuit(qr, cr)
    qc.u(p[0], p[1], p[2], qr[0])
    qc.measure(qr, cr[0])
    return qc

qc = form(p)
```



```
Parameters Found: [ 1.47924356 -0.29323942  2.00245954]
Target Distribution: {0: 0.308979188922057, 1: 0.691020811077943}
Obtained Distribution: {1: 0.7119140625, 0: 0.2880859375}
Cost: 0.039833377844114004
```

```
from qiskit_aer.primitives import Sampler, Estimator

sampler = Sampler()
def objective_function(p):
    result = sampler.run(circuits=qc, parameter_values=p).result()
    output_distr = result.quasi_dists[0]
    cost = sum(
        abs(target_distr.get(i, 0) - output_distr.get(i, 0))
        for i in range(2**qc.num_qubits)
    )
    return cost
```

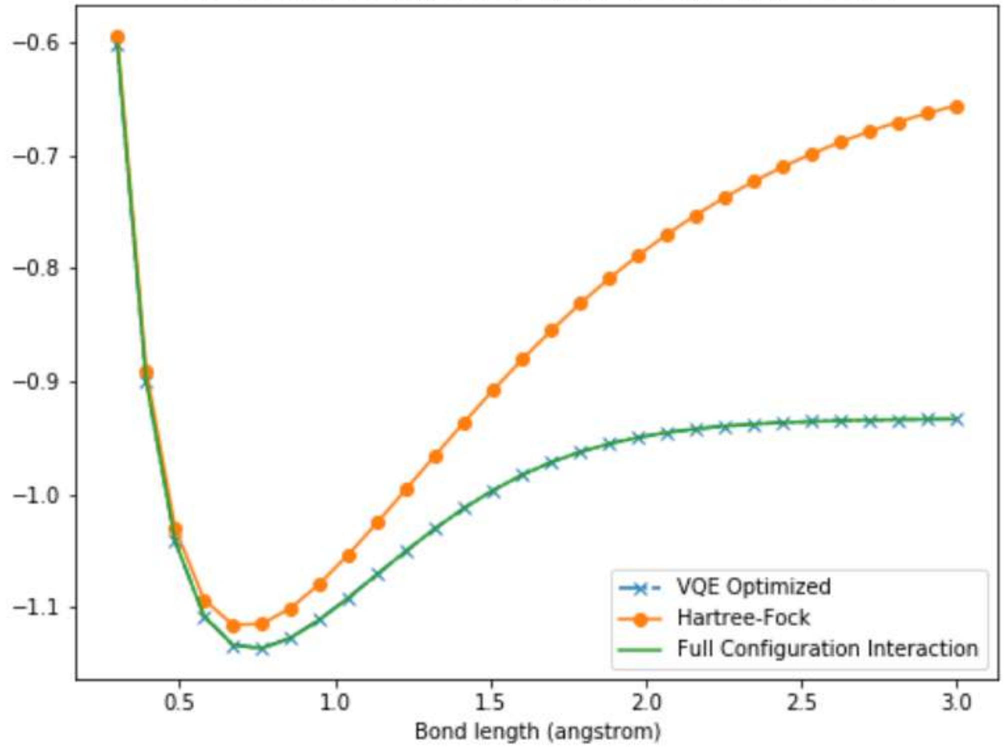
```
from qiskit.algorithms.optimizers import SPSA, SLSQP, COBYLA
optimizer = COBYLA(maxiter=500, tol=0.0001)
initial_point = np.random.rand(3)
result = optimizer.minimize(fun=objective_function, x0=initial_point)
output_distr = (
    sampler.run(circuits=qc, parameter_values=result.x).result().quasi_dists[0]
)
print("Parameters:", result.x)
print("Target DIST:", target_distr)
print("Obtained DIST:", output_distr)
print("Cost:", objective_function(result.x))
```

Brief summary as to how to make VQE work

1. Map the problem that you want to solve to finding the ground state energy of a Hamiltonian (ie, a molecule problem or some cost function)
2. Prepare a trial state with some collection of parameters
3. Run that through the parameterized quantum circuit
4. Measure expectation values of Hamiltonian (by measuring each of the Pauli strings or the ‘observables’ of the Hamiltonian)
5. Calculate the ‘energy’ corresponding to the trial state by summing up all the measurements in step #4
6. Update all the parameters via some optimization algorithm (ie, some form of gradient descent)
7. Now you have the first iteration of the trial state with some new parameters to feedback into step 2 and just repeat everything all over again until you get the lowest possible energy.

$$\begin{aligned}\epsilon_i\psi_i(\mathbf{r}) &= \left(-\frac{1}{2}\nabla^2 + V_{ion}(\mathbf{r})\right)\psi_i(\mathbf{r}) + \sum_j \int d\mathbf{r}' \frac{|\psi_j(\mathbf{r}')|^2}{|\mathbf{r}-\mathbf{r}'|}\psi_i(\mathbf{r}) \\ &- \sum_j \delta_{\sigma_i\sigma_j}\int d\mathbf{r}'\frac{\psi_j^*(\mathbf{r}')\psi_i(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\psi_j(\mathbf{r}) \quad .\end{aligned}$$

Ground state energy vs bond length of H_2 using UCCGSD



Topics Covered

- Quantum Circuit Basics with Qiskit
- Postulates of QM
- Qubits
- Quantum Gates
- No-Cloning Theorem
- Quantum Teleportation
- QFT, QPE
- Super Dense Coding
- Density Matrix
- Measurement Postulates
- Grover’s Search Algorithm
- Shor’s Algorithm
- VQE