Quantum Computing for Computational Fluid Dynamics

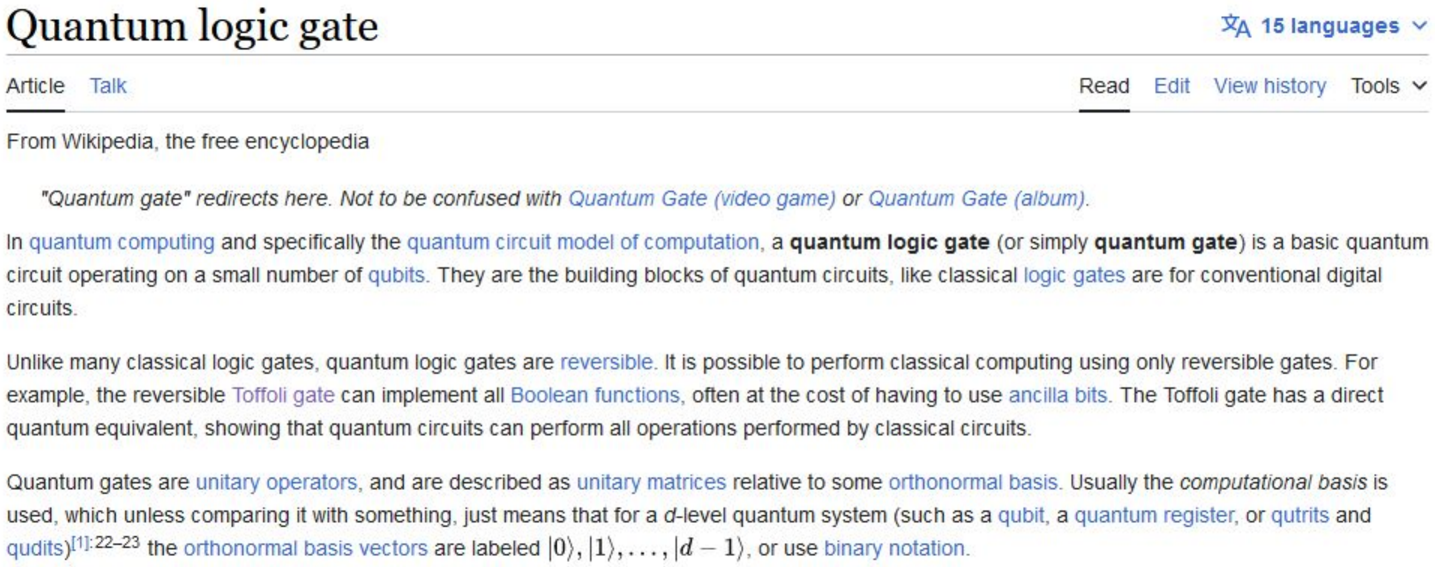
Poisson Equation

- Discretized Poisson equation in one dimension with Dirichlet boundary conditions is expressed in the following matrix.
- We decompose this matrix into two different ways, one expressed in the Pauli basis, and the other expressed with high-entanglement gates
    - The decomposition in the Pauli basis requires an exponential number of sub-matrices with respect to system size.
    - The high-entanglement decomposition is constant with respect to system size, but one of the sub-matrices scales polynomially with respect to the number of qubits.

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

Bra-ket Notation and Quantum Gates

$$\langle A|B\rangle \doteq A_1^*B_1 + A_2^*B_2 + \cdots + A_N^*B_N = \begin{pmatrix} A_1^* & A_2^* & \cdots & A_N^* \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{pmatrix}$$



Quantum logic gate

15 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

"Quantum gate" redirects here. Not to be confused with Quantum Gate (video game) or Quantum Gate (album).

In quantum computing and specifically the quantum circuit model of computation, a **quantum logic gate** (or simply **quantum gate**) is a basic quantum circuit operating on a small number of qubits. They are the building blocks of quantum circuits, like classical logic gates are for conventional digital circuits.

Unlike many classical logic gates, quantum logic gates are reversible. It is possible to perform classical computing using only reversible gates. For example, the reversible Toffoli gate can implement all Boolean functions, often at the cost of having to use ancilla bits. The Toffoli gate has a direct quantum equivalent, showing that quantum circuits can perform all operations performed by classical circuits.

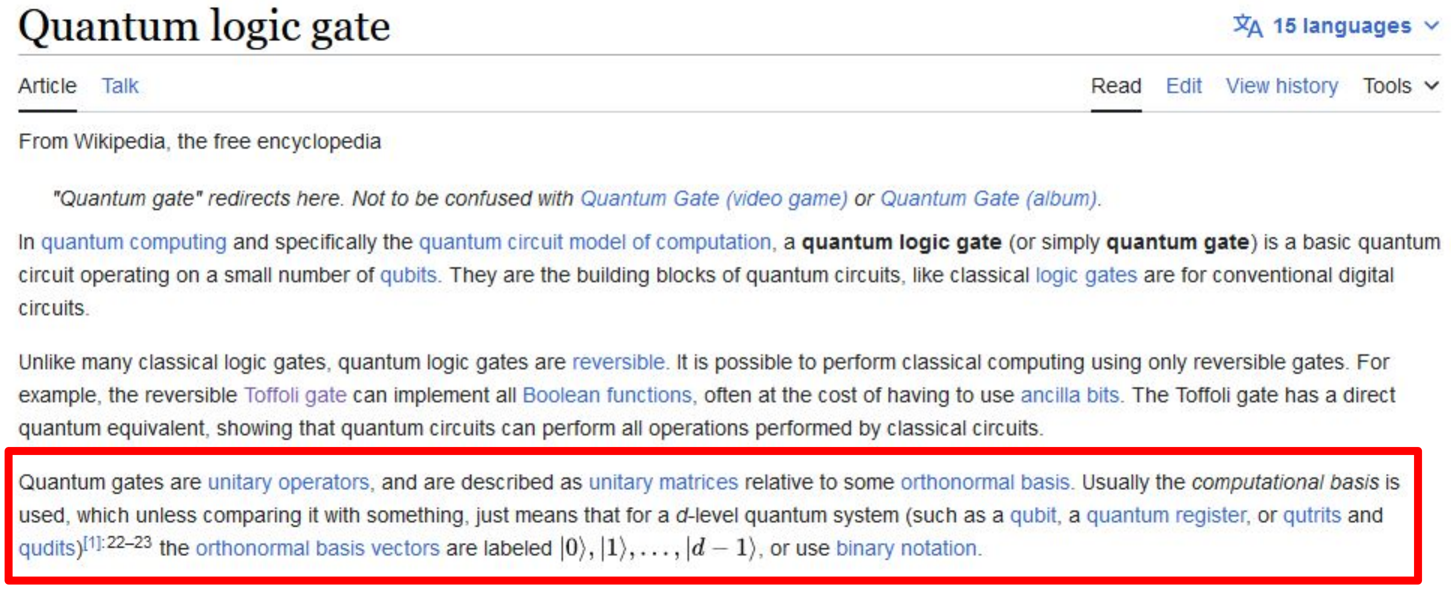
Quantum gates are unitary operators, and are described as unitary matrices relative to some orthonormal basis. Usually the *computational basis* is used, which unless comparing it with something, just means that for a *d*-level quantum system (such as a qubit, a quantum register, or qutrits and qudits)[1]:22–23 the orthonormal basis vectors are labeled $|0\rangle, |1\rangle, \dots, |d-1\rangle$, or use binary notation.

https://en.wikipedia.org/wiki/Quantum_logic_gate

Bra-ket Notation and Quantum Gates[1]

$$\langle A|B\rangle \doteq A_1^* B_1 + A_2^* B_2 + \cdots + A_N^* B_N = \begin{pmatrix} A_1^* & A_2^* & \cdots & A_N^* \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{pmatrix}$$

|B> represents a vector containing probabilities of all outcomes occurring.
<A| is the conjugate transpose of such a vector



Quantum logic gate

From Wikipedia, the free encyclopedia

"Quantum gate" redirects here. Not to be confused with Quantum Gate (video game) or Quantum Gate (album).

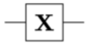
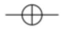
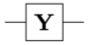
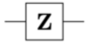
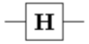
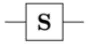
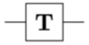
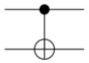
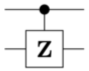
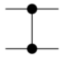
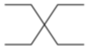
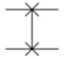
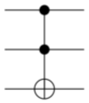
In quantum computing and specifically the quantum circuit model of computation, a **quantum logic gate** (or simply **quantum gate**) is a basic quantum circuit operating on a small number of qubits. They are the building blocks of quantum circuits, like classical logic gates are for conventional digital circuits.

Unlike many classical logic gates, quantum logic gates are reversible. It is possible to perform classical computing using only reversible gates. For example, the reversible Toffoli gate can implement all Boolean functions, often at the cost of having to use ancilla bits. The Toffoli gate has a direct quantum equivalent, showing that quantum circuits can perform all operations performed by classical circuits.

Quantum gates are unitary operators, and are described as unitary matrices relative to some orthonormal basis. Usually the *computational basis* is used, which unless comparing it with something, just means that for a *d*-level quantum system (such as a qubit, a quantum register, or qutrits and qudits)[1]:22–23 the orthonormal basis vectors are labeled $|0\rangle, |1\rangle, \ldots, |d-1\rangle$, or use binary notation.

[1]https://en.wikipedia.org/wiki/Quantum_logic_gate

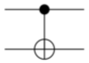
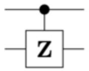
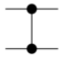
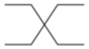
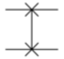
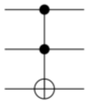
Quantum Logic Gates[1]

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

[1]https://en.wikipedia.org/wiki/Quantum_logic_gate

Quantum Logic Gates[1]

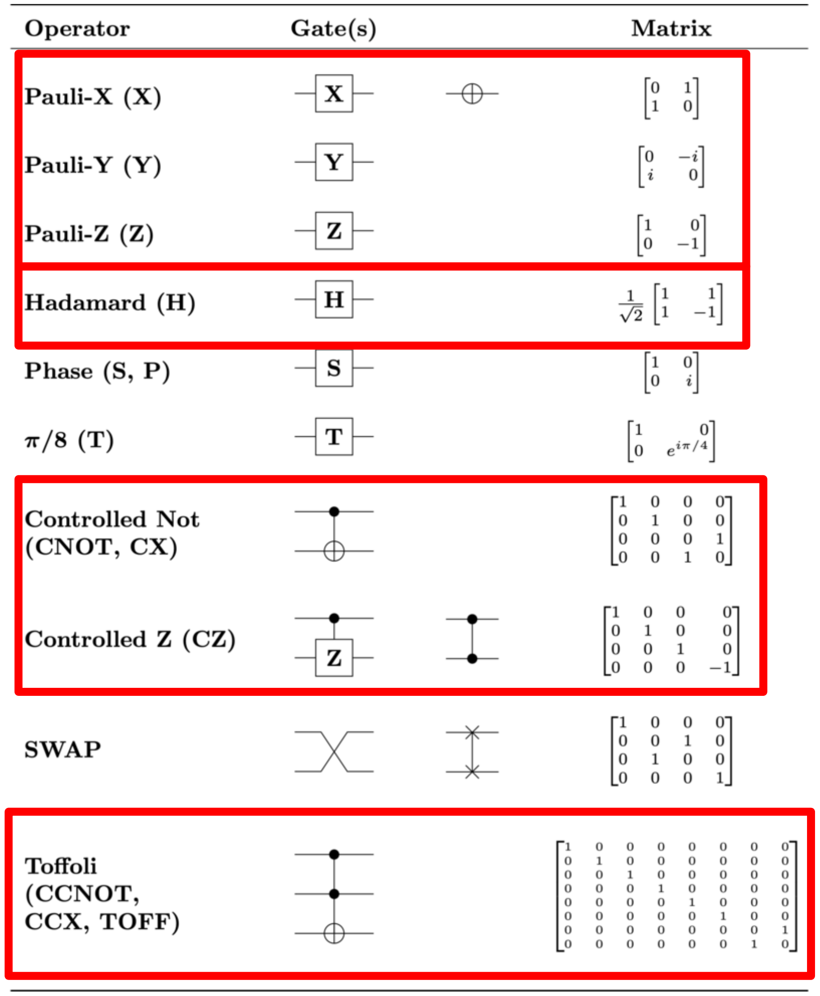
these gates, along with the Identity (no-op) gate, compose the Pauli basis

Operator	Gate(s)		Matrix
Pauli-X (X)	$\boxed{X}$	$\oplus$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	$\boxed{Y}$		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	$\boxed{Z}$		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	$\boxed{H}$		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	$\boxed{S}$		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	$\boxed{T}$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
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)			$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

[1]https://en.wikipedia.org/wiki/Quantum_logic_gate

Quantum Logic Gates[1]

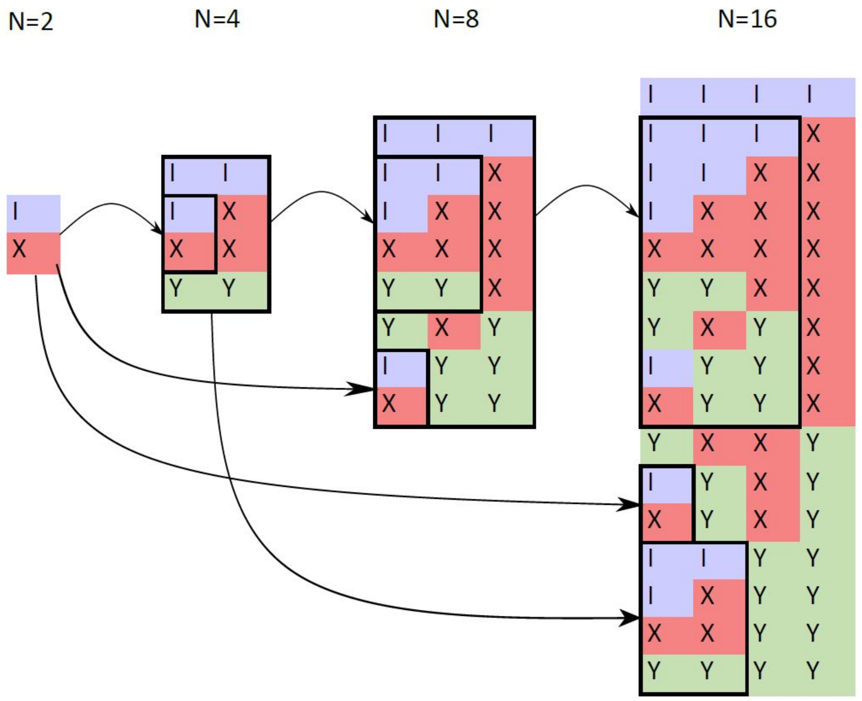
these gates, along with the Identity (no-op) gate, compose the Pauli basis



Operator	Gate(s)		Matrix
Pauli-X (X)	$\boxed{X}$	$\oplus$	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	$\boxed{Y}$		$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)	$\boxed{Z}$		$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	$\boxed{H}$		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)	$\boxed{S}$		$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)	$\boxed{T}$		$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
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)	$\boxed{Z}$		$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & -1\end{bmatrix}$
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)			$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

[1]https://en.wikipedia.org/wiki/Quantum_logic_gate

Poisson Equation - Pauli Basis Decomposition[2]



N=4:

$$A = 2II - IX - 0.5XX - 0.5YY$$

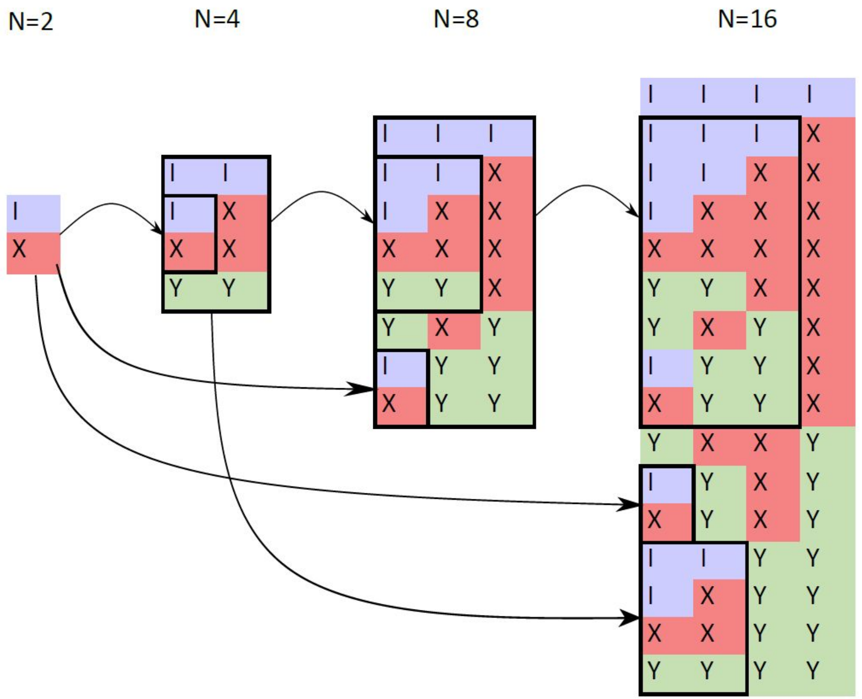
N=8:

$$A = 2III - IIX - 0.5IXX - 0.25XXX$$
$$-0.25YYX - 0.25YXY - 0.5IYY + 0.25XYY$$

$$D_n = 2I + \frac{1}{2}(D_{n-1} - 4I_{n-1}) \otimes X - \sum_{k=0}^{n-2} \frac{1}{2^{k+2}} D_{n-2-k} \otimes Y \bigotimes^{k} X \otimes Y$$

[2]https://repository.tudelft.nl/islandora/object/uuid\%3Adeba389d-f30f-406c-ad7b-babb1b298d87

Poisson Equation - Pauli Basis Decomposition[2]



N=4:

$$A = 2II - IX - 0.5XX - 0.5YY$$

N=8:

$$A = 2III - IIX - 0.5IXX - 0.25XXX$$
$$-0.25YYX - 0.25YXY - 0.5IYY + 0.25XYY$$

$$D_n = 2I + \frac{1}{2}(D_{n-1} - 4I_{n-1}) \otimes X - \sum_{k=0}^{n-2} \frac{1}{2^{k+2}} D_{n-2-k} \otimes Y \bigotimes^{k} X \otimes Y$$

exponential scaling

[2]https://repository.tudelft.nl/islandora/object/uuid\%3Adeba389d-f30f-406c-ad7b-babb1b298d87

Poisson Equation - High-Entanglement Decomposition

$$A = 2.5I - L_1 - L_2 - 0.5L_3$$

$$L_1 = I^{\otimes n} \otimes X$$

$$L_2 = C_1 \otimes C_2 \otimes \ldots \otimes C_{n-1}$$

$$C_i = CX^i_{i-1} \otimes CX^i_{i-2} \cdots CX^i_0 \otimes mCX_i^{0,1,\ldots,i-1} \otimes CX^i_0 \cdots CX^i_{i-2} \otimes CX^i_{i-1}$$

$$L_3 = mCZ^{0,1,\ldots n-1}\, X\, mCZ^{0,1,\ldots n-1}\, X$$

$$L_1 = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & & & \\ & & \ddots & & \\ & & & 0 & 1 \\ & & & 1 & 0 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1 & & & & & & \\ & 0 & 1 & & & & \\ & 1 & 0 & & & & \\ & & & \ddots & & & \\ & & & & 0 & 1 & \\ & & & & 1 & 0 & \\ & & & & & & 1 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} -1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & -1 \end{bmatrix}$$

Poisson Equation - High-Entanglement Decomposition

$$\boxed{A = 2.5I - L_1 - L_2 - 0.5L_3}$$

constant scaling of the number of sub-matrices!

$$L_1 = I^{\otimes n} \otimes X$$

$$L_2 = C_1 \otimes C_2 \otimes \ldots \otimes C_{n-1}$$

$$C_i = CX^i_{i-1} \otimes CX^i_{i-2} \cdots CX^i_0 \otimes mCX_i^{0,1,\ldots,i-1} \otimes CX^i_0 \cdots CX^i_{i-2} \otimes CX^i_{i-1}$$

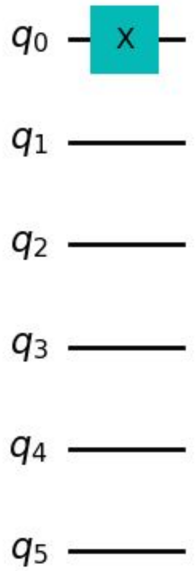
$$L_3 = mCZ^{0,1,\ldots n-1}\, X\, mCZ^{0,1,\ldots n-1}\, X$$

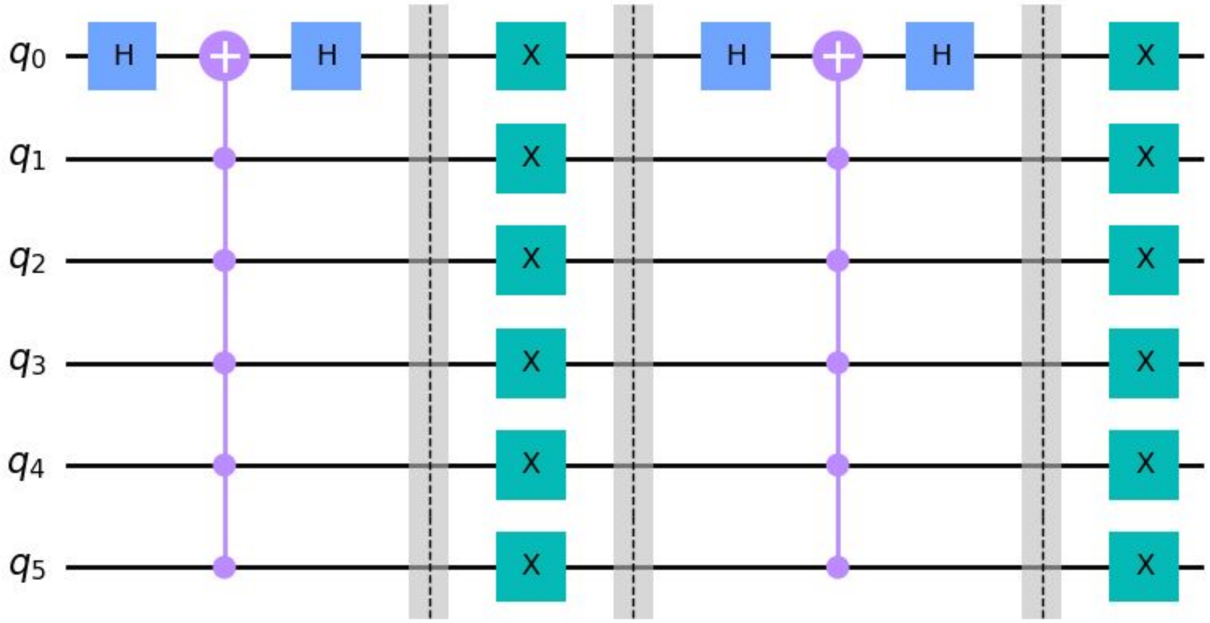
$$L_1 = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & & & \\ & & \ddots & & \\ & & & 0 & 1 \\ & & & 1 & 0 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1 & & & & & & \\ & 0 & 1 & & & & \\ & 1 & 0 & & & & \\ & & & \ddots & & & \\ & & & & 0 & 1 & \\ & & & & 1 & 0 & \\ & & & & & & 1 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} -1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & -1 \end{bmatrix}$$

L1 and L3 Implementations





L2 Implementation

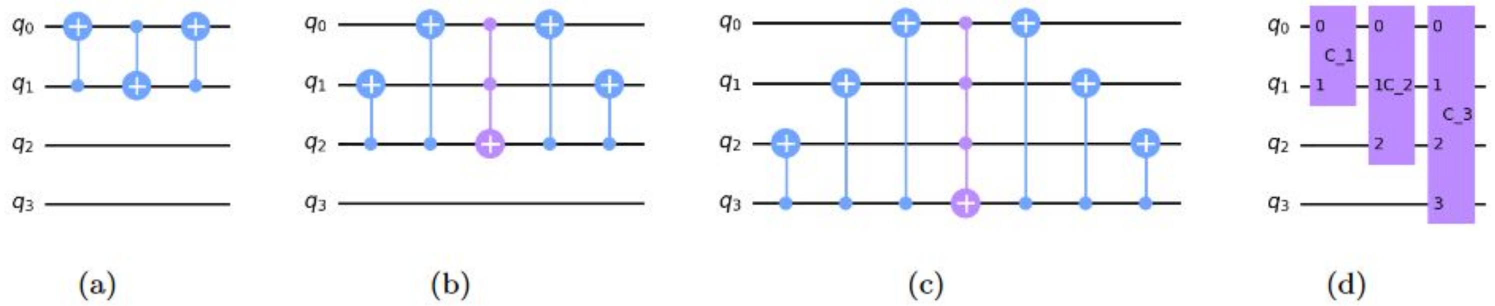


Figure 2: C_1, C_2, and C_3 from left to right. Finally, (d) depicts the overall implementation of the L_2 matrix for $n = 4$

L2 Implementation

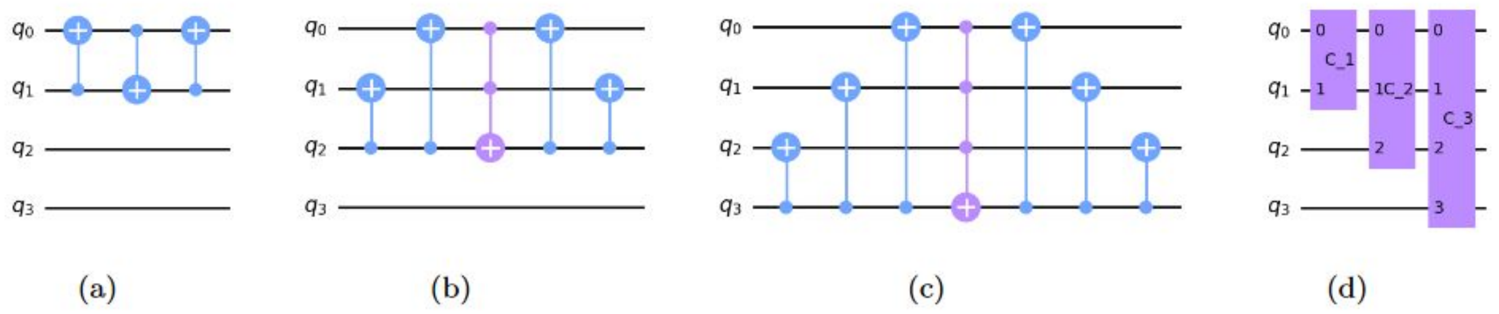
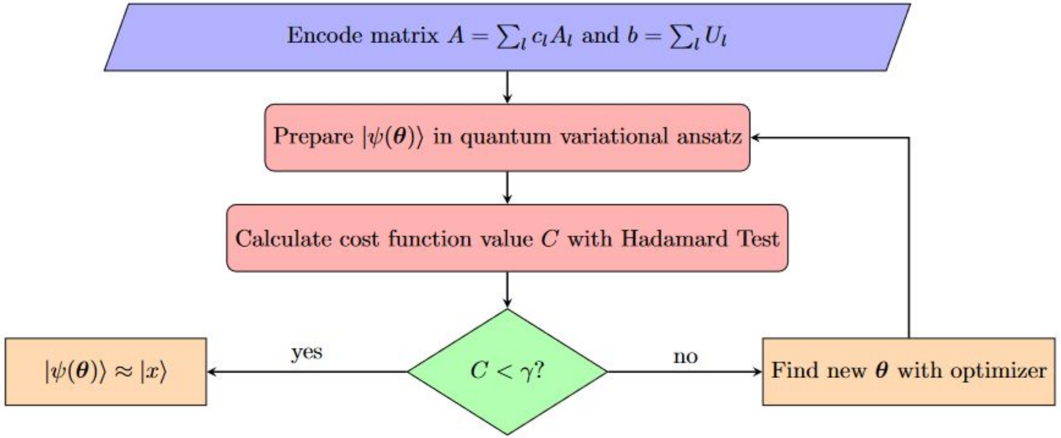


Figure 2: C_1, C_2, and C_3 from left to right. Finally, (d) depicts the overall implementation of the L_2 matrix for $n = 4$

only expense of this decomposition is polynomial scaling of circuit depth of this sub-matrix

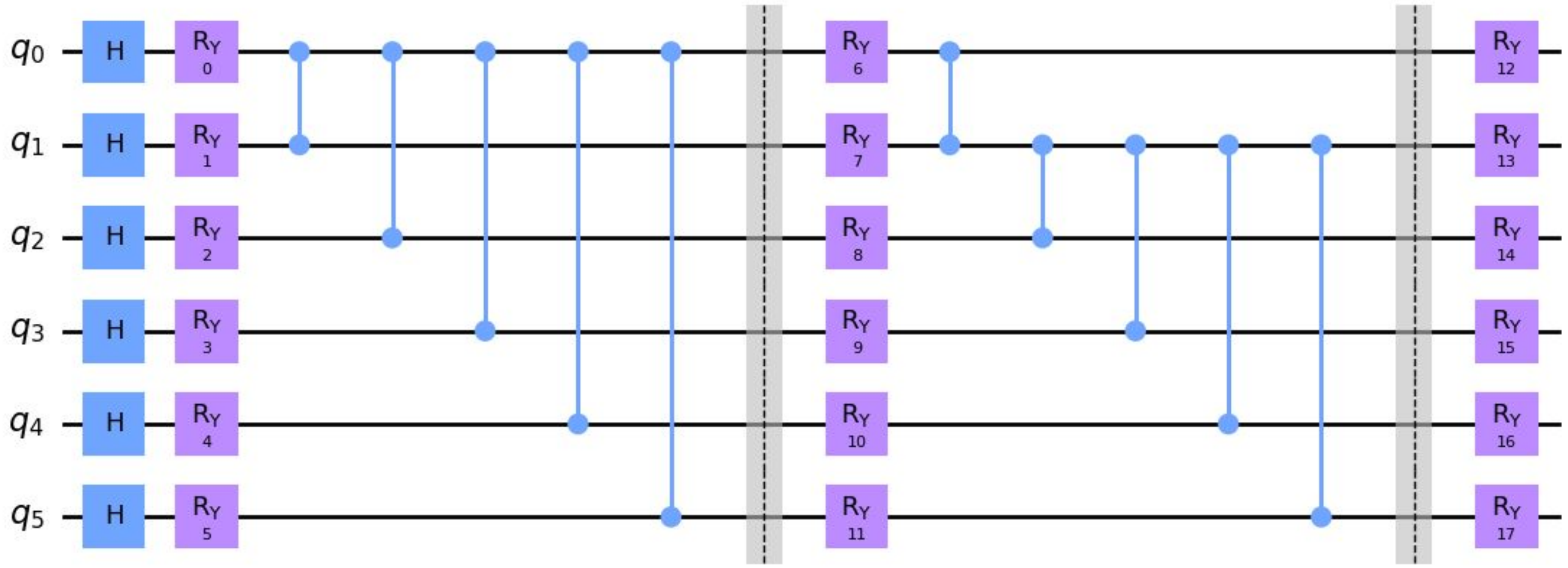
Variational Quantum Linear Solver (VQLS)

- Utilize a heuristic algorithm to find a parameterized circuit that encodes a probability vector proportional to the solution of the matrix.
- Use a classical optimizer that determines the cost via evaluations of the quantum circuit.
- Develop cost function that avoids a vanishing gradient and requires minimal number of quantum circuit evaluations.
- Use some optimal ansatz circuit that has a manageable number of parameters while also capable of expressing many states in the Hilbert space



Encode matrix $A = \sum_l c_l A_l$ and $b = \sum_l U_l$
Prepare $|\psi(\boldsymbol{\theta})\rangle$ in quantum variational ansatz
Calculate cost function value C with Hadamard Test
$C < \gamma$?
yes
no
$|\psi(\boldsymbol{\theta})\rangle \approx |x\rangle$
Find new $\boldsymbol{\theta}$ with optimizer

Global Entangling Variational Ansatz – $V(\theta)|0\rangle$



Cost Function - Normalized Global

$$\hat{C}_G = 1 - \frac{|\langle b|\Phi\rangle|^2}{\langle\Phi|\Phi\rangle}$$

$$\langle b| = (U|0\rangle)^\dagger = \langle 0| U^\dagger$$
$$|\Phi\rangle = A|\psi(\boldsymbol{\theta})\rangle = AV(\boldsymbol{\theta})|0\rangle$$

$$A = \sum_l c_l A_l,$$

$$A = 2.5I - L_1 - L_2 - 0.5L_3$$

$$\hat{C}_G = \tag{28}$$
$$1 - \frac{\sum_{l,l'} c_l^* c_{l'} \langle 0| V(\boldsymbol{\theta})^\dagger A_l^\dagger U |0\rangle \langle 0| U^\dagger A_{l'} V(\boldsymbol{\theta}) |0\rangle}{\sum_{l,l'} c_l^* c_{l'} \langle 0| V(\boldsymbol{\theta})^\dagger A_l^\dagger A_{l'} V(\boldsymbol{\theta}) |0\rangle}$$

Cost Function - Normalized Local

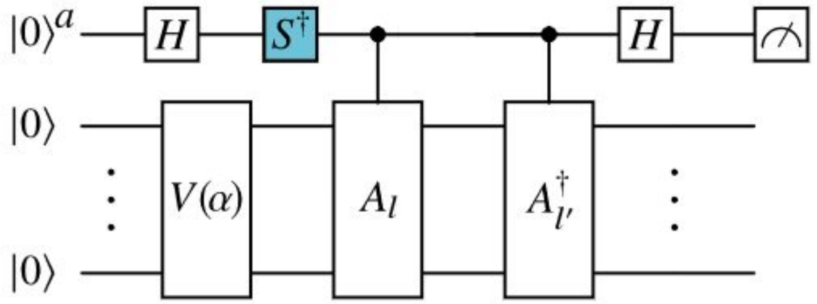
- Less barren plateau problems[3].
- The idea is to take information from qubits one at a time.

$$\hat{C}_L = \tag{30}$$
$$\frac{1}{2} - \frac{1}{2n} \frac{\sum_{l,l',j} c_l^* c_{l'} \langle 0 | V(\boldsymbol{\theta})^\dagger A_l^\dagger U Z_j U^\dagger A_{l'} V(\boldsymbol{\theta}) |0\rangle}{\sum_{l,l'} c_l^* c_{l'} \langle 0 | V(\boldsymbol{\theta})^\dagger A_l^\dagger A_{l'} V(\boldsymbol{\theta}) |0\rangle}$$

[3]https://arxiv.org/pdf/1909.05820.pdf

Hadamard Test

- In order to calculate expectation values from quantum circuit, we use the Hadamard Test - uses a phenomenon known as “phase kickback”

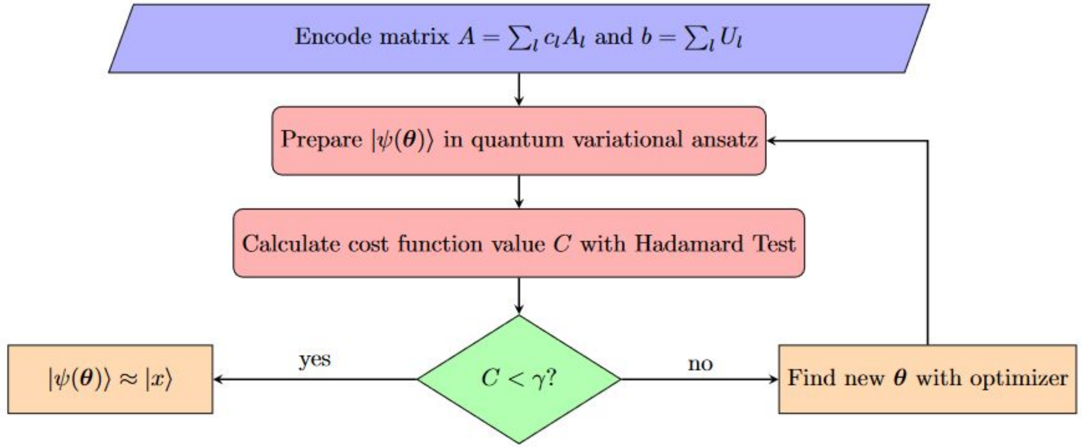


Classical Optimization

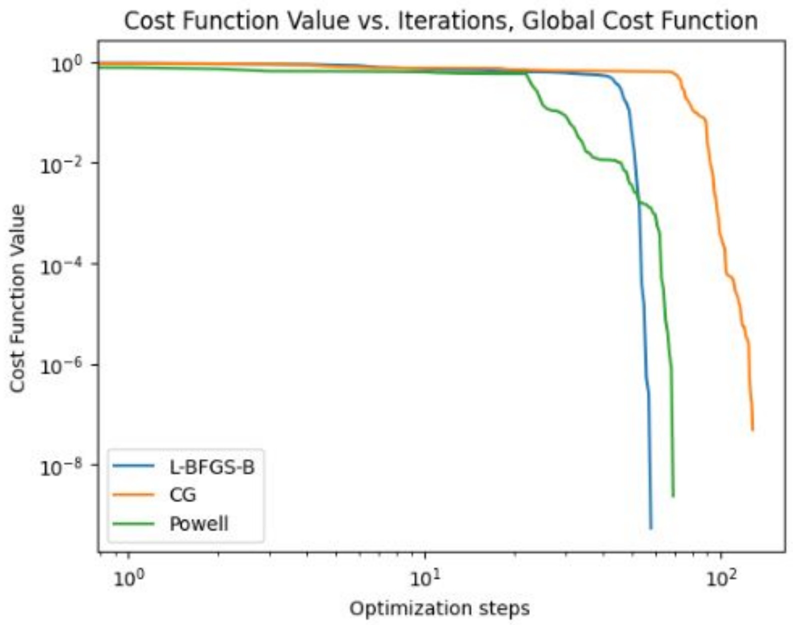
- Attempt to minimize cost function below some constraint.
- Mainly tested Powell, L-BFGS-B, and CG classical optimizers.

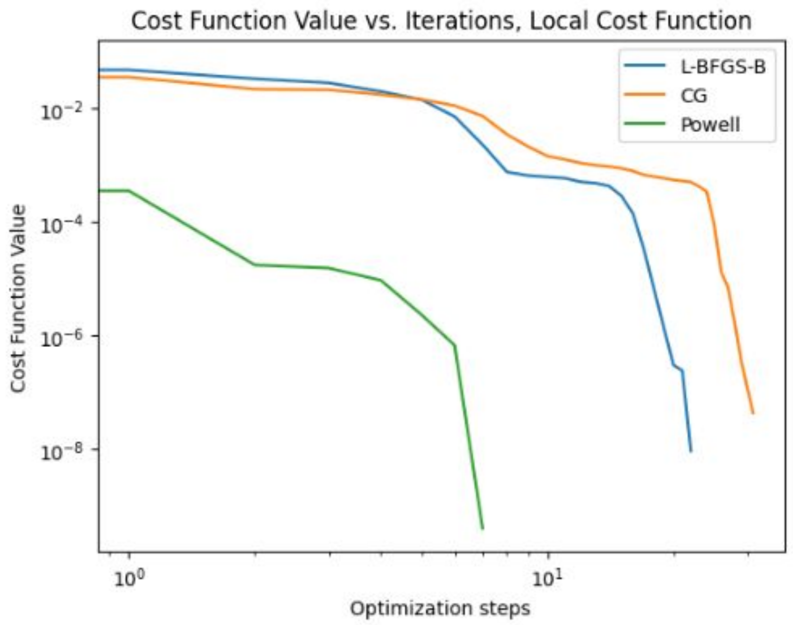
$$\widehat{C}_G \geqslant \frac{\epsilon^2}{\kappa^2}$$

$$\widehat{C}_L \geqslant \frac{1}{n}\frac{\epsilon^2}{\kappa^2}$$

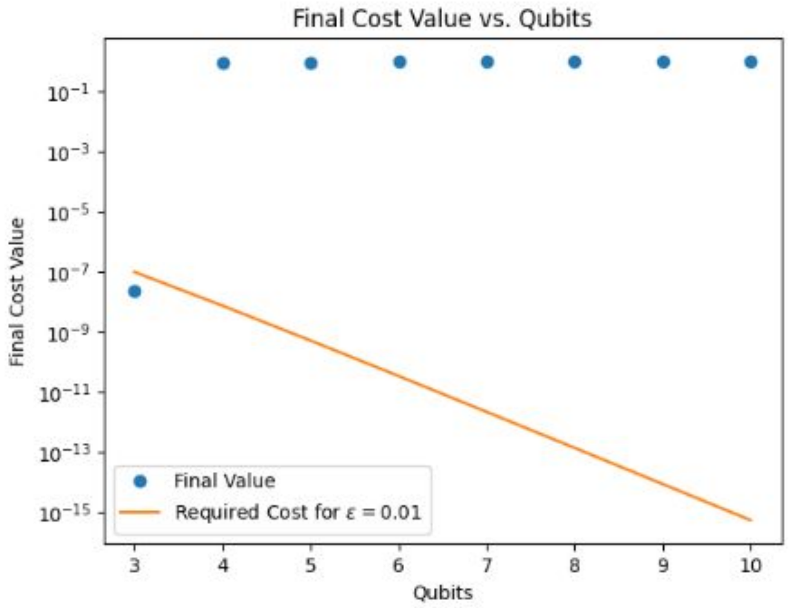

Encode matrix $A = \sum_l c_l A_l$ and $b = \sum_l U_l$
Prepare $|\psi(\boldsymbol{\theta})\rangle$ in quantum variational ansatz
Calculate cost function value C with Hadamard Test
$C < \gamma$?
yes
$|\psi(\boldsymbol{\theta})\rangle \approx |x\rangle$
no
Find new $\boldsymbol{\theta}$ with optimizer

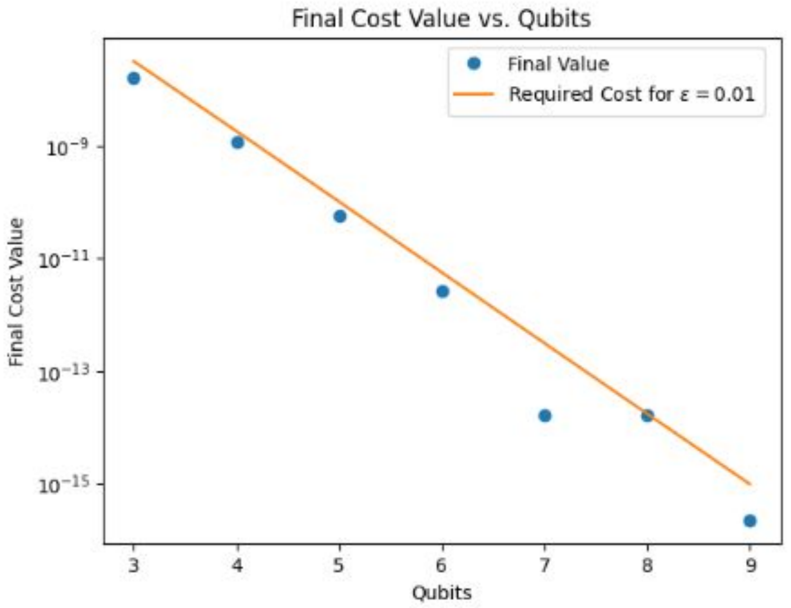
Numerical Simulations - 8x8 Matrix (3 qubits)



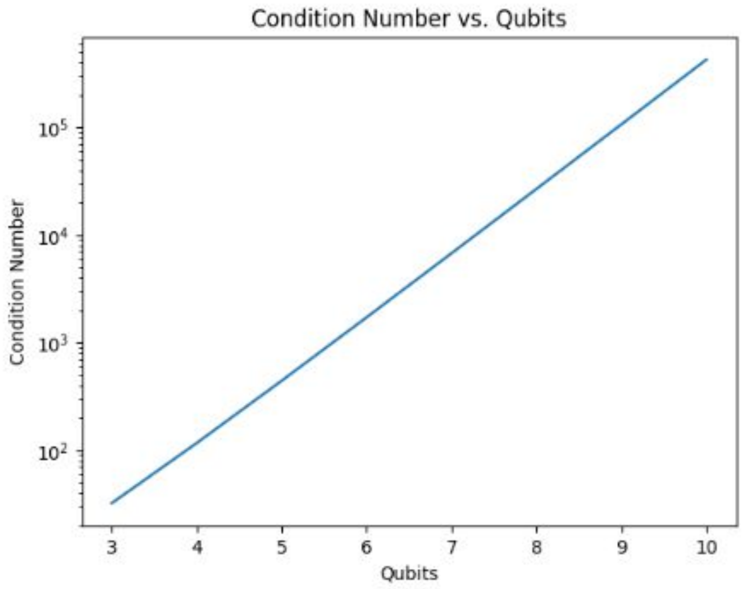


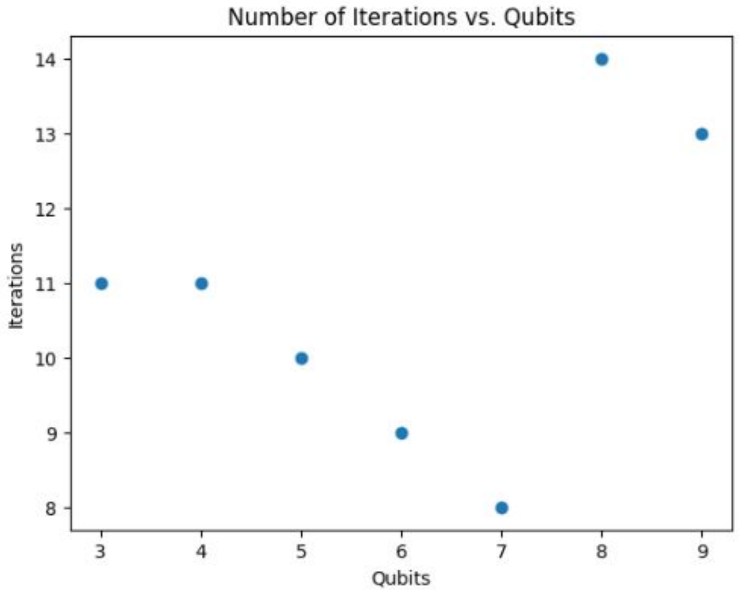
Numerical Simulations - Higher Qubit Systems





Numerical Results - General





Analysis on Error

- Numerically verified that the cost function requires ~8 digits of precision in order to have perfect results (indiscernible to statevector simulations)
  - Gamma constraint is normally 10^{-8} or less.
- Using Hadamard tests, our precision is determined by the number of shots used, which scales as $O(1/\varepsilon^2)$ [4].
  - To get our perfect precision, we would require 10^{16} shots.

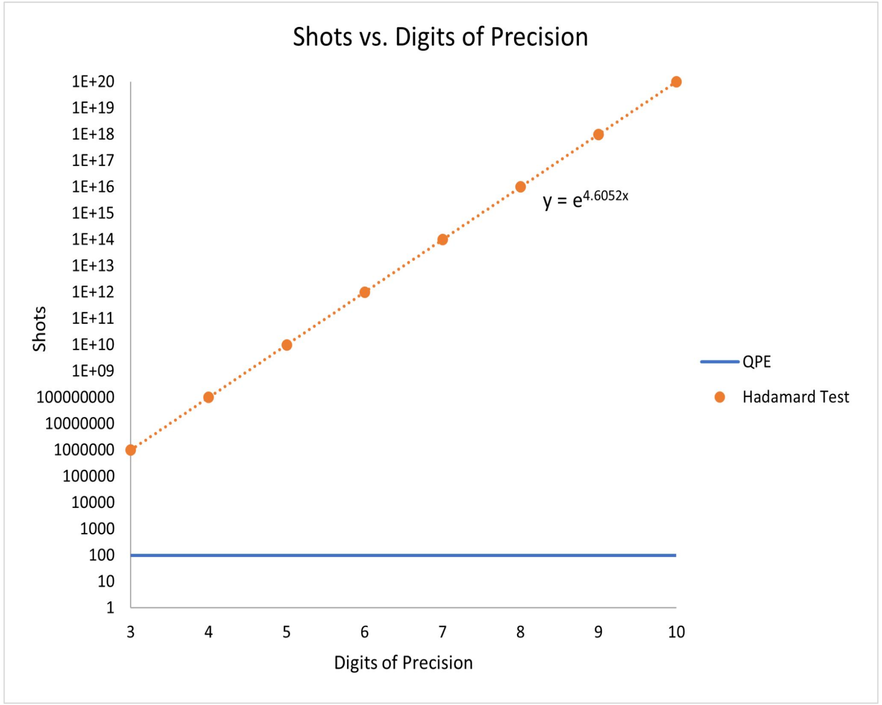
[4]https://en.wikipedia.org/wiki/Chebyshev%27s_inequality

Analysis on Error

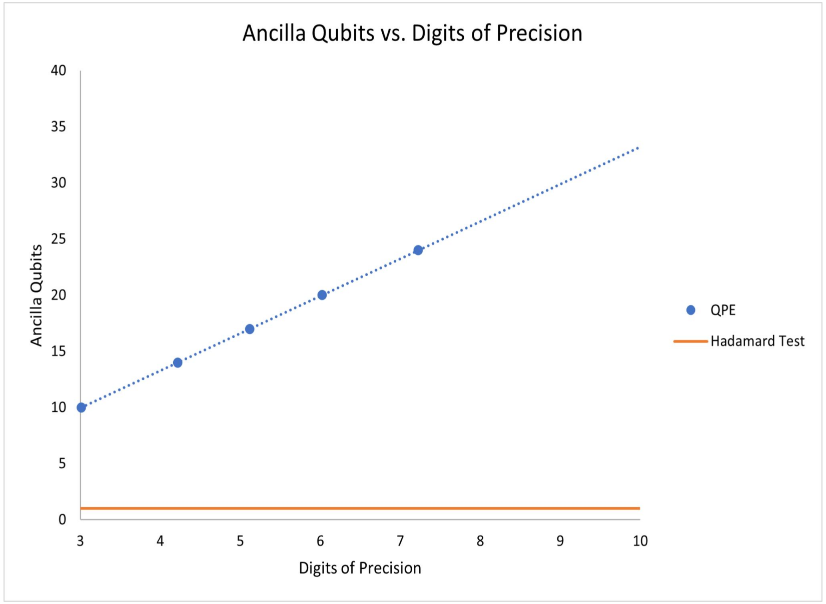
- Numerically verified that the cost function requires ~8 digits of precision in order to have perfect results (indiscernible to statevector simulations)
  - Gamma constraint is normally 10^{-8} or less.
- Using Hadamard tests, our precision is determined by the number of shots used, which scales as $O(1/\varepsilon^2)$.
  - To get our perfect precision, we would require 10^{16} shots (highly unreasonable)
- There exists a routine known as Quantum Phase Estimation[5] that carries a quadratic speedup of this precision scaling. However, it turns the exponential scaling of precision from shots to circuit depth.

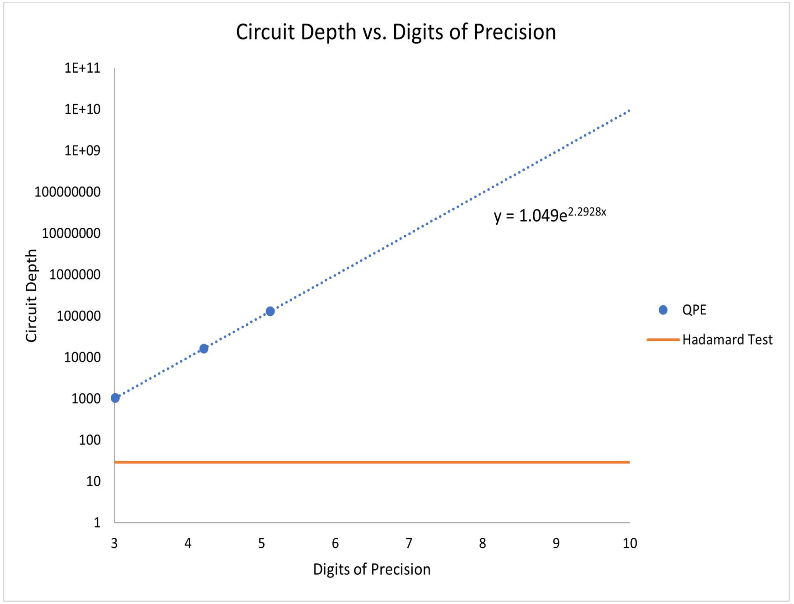
[5]https://arxiv.org/pdf/2102.04975.pdf

Analysis on Error - Hadamard Exponentials

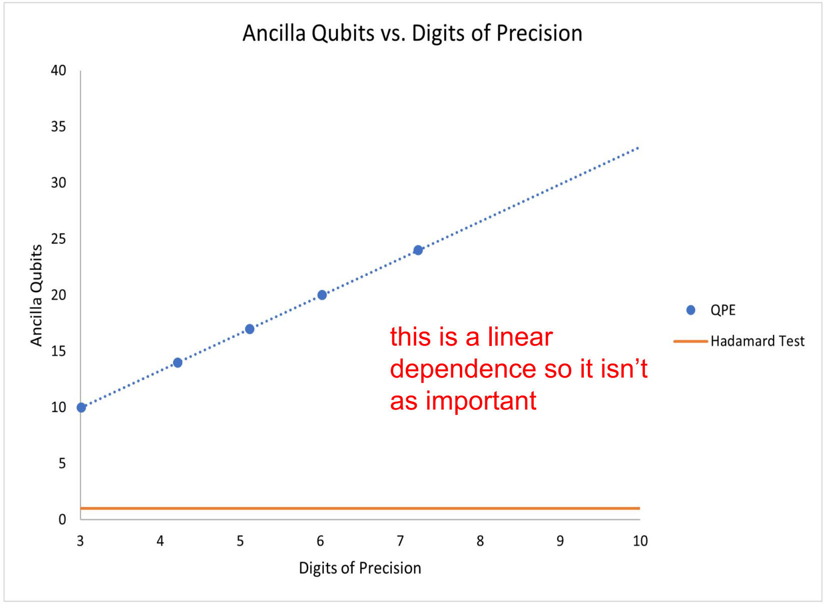


Analysis on Error - QPE Exponentials

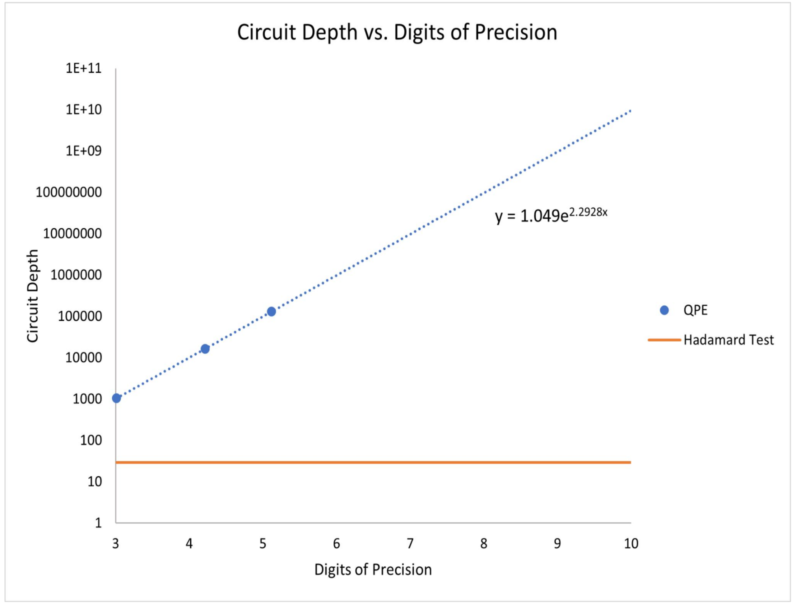




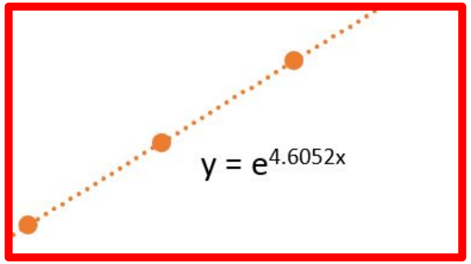
Analysis on Error - QPE Exponentials



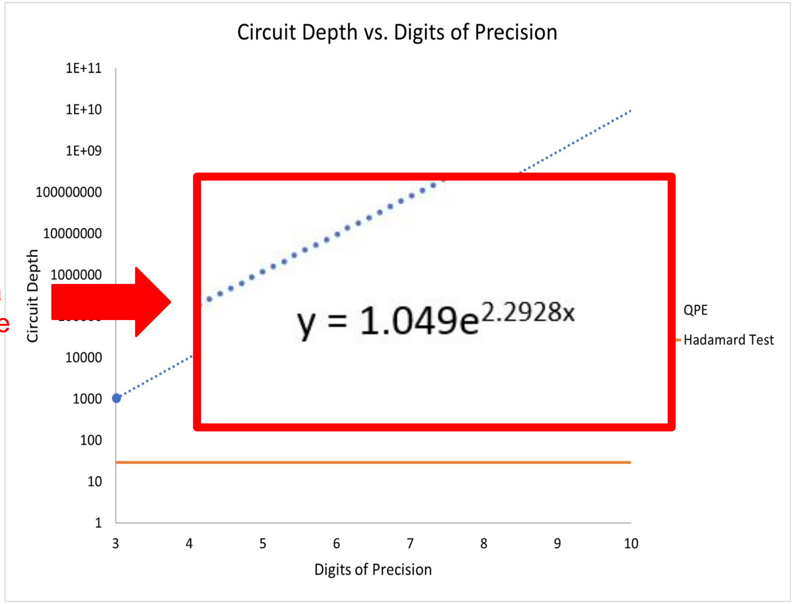
this is a linear dependence so it isn’t as important



Analysis on Error - QPE Exponentials



notice the difference by a factor of 2 in the exponential



Analysis on Error

- The number of qubits required to perform VQLS with sufficient precision is available.
- Unfortunately, this type of circuit depth is not possible on current quantum computers.
  - Mostly due to decoherence noise - qubits can only exist in quantum states for so long before unintentionally interacting with other matter around it, effectively destroying the quantum state.
- For now, we will simply try to maximize the number of shots we can perform in a reasonable time frame.
  - The VQLS can still perform reasonably well under the effects of quantum and shot noise.

Conclusion

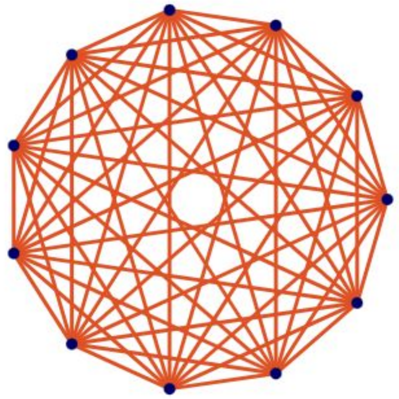
- Apply VQLS to solving the Poisson equation
- Used a high-entanglement ansatz that allows us to express a large portion of the Hilbert space with a minimal number of parameters.
- Used a high-entanglement decomposition for the Poisson that allows for a constant scaling of number of expectation values with respect to matrix size.

Conclusion

- Apply VQLS to solving the Poisson equation with a local cost function.
- Used a high-entanglement ansatz that allows us to express a large portion of the Hilbert space with a minimal number of parameters.
- Used a high-entanglement decomposition for the Poisson that allows for a constant scaling of number of expectation values with respect to matrix size.
- Why do we like high-entanglement schemes so much?

IonQ Quantum Computers

- IonQ boasts "total interconnectivity" of qubits. We would like to take advantage of this as much as possible, since this is not very common among quantum computers.

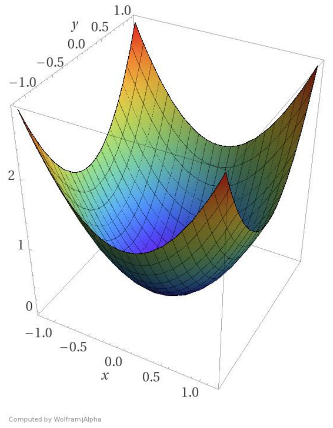


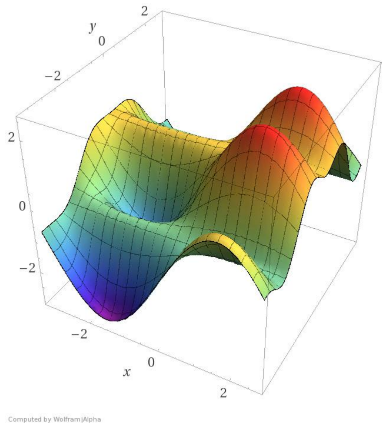
What’s next?

- Finish running tests of solving the Poisson on IonQ Quantum Computers for large qubit systems.
  - Currently experiencing some trouble with the local cost function.
  - Considering trying to use the Adder circuit to calculate cost function (also in link below)
- Different representations of the discretized Poisson equation - 2D, Neumann boundary condition, etc.
- Solving PDEs with time-dependence (eg. the Heat equation) using a Variational Quantum Algorithm similar to the VQLS.
  - https://youtu.be/aTwv2I0W7rE?si=GgDdaQ4j2IYLmplP

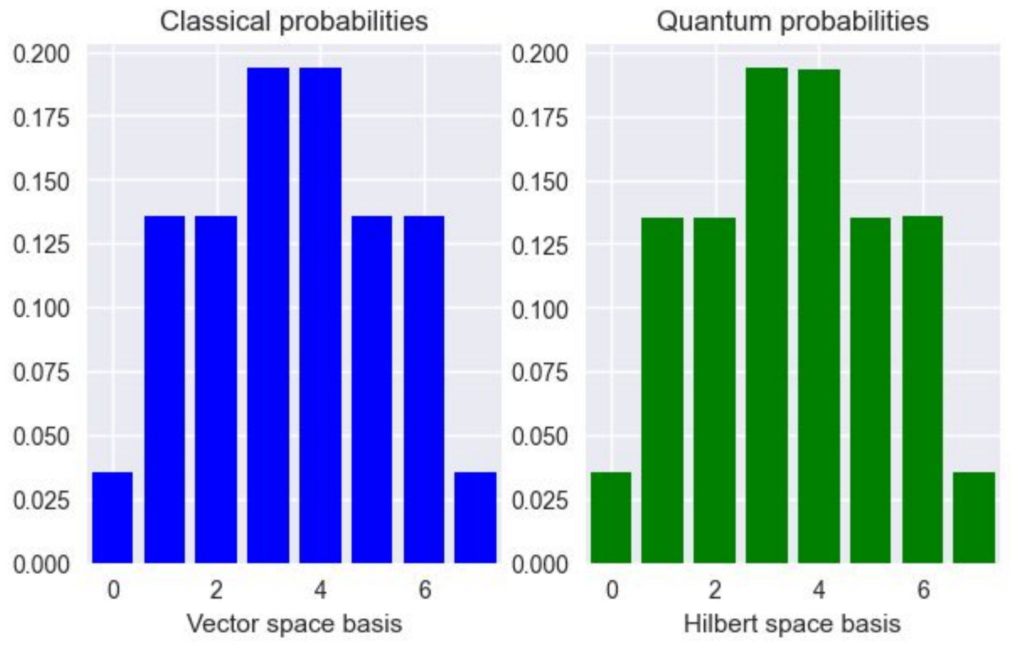
Local Cost Function with a Local Minima

- Recent verification of the results being outputted after minimizing the local cost function has shown that there is a local minima introduced in the optimization problem.
- It seems as though in the particular case of the Poisson equation, optimizing the local cost function is a non-convex problem.
- Additionally, the local minima seems to always be low enough to meet the threshold. Thus, the solutions are indistinguishable by cost.

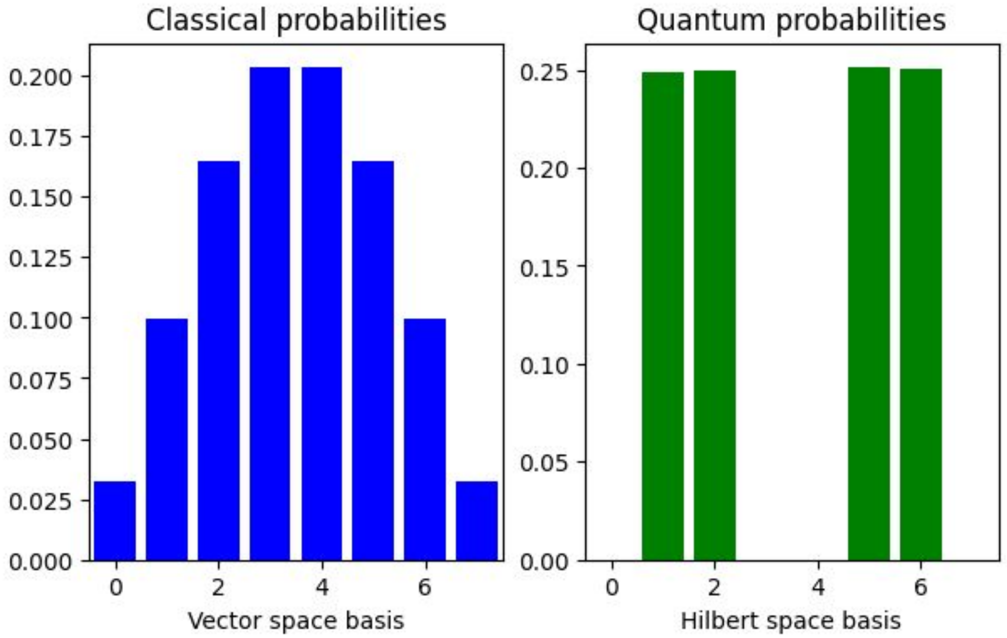




Local Cost Function - Results of Slightly Modified Poisson



Local Cost Function - Results of Poisson Equation

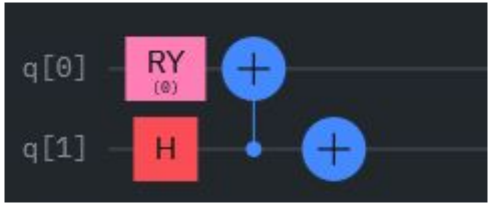


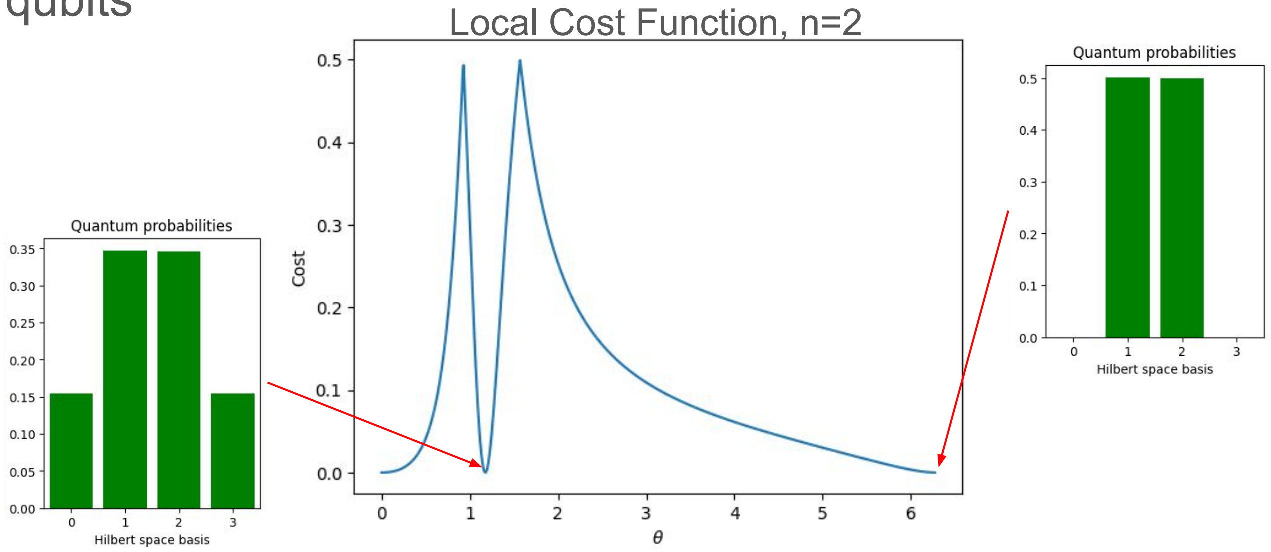
Local Cost Function - Local Minima Remedies

- Choose parameters that allow cost function to be greater than some target value, then let the optimizer start from there
  - This seems to work so far with the $n=3$ case using Conjugate Gradient, and an initial cost of 0.33
  - However, not much more luck with higher qubit cases
- Somehow modify the cost function that removes the local minima from the cost surface
  - Not entirely clear how to do this yet. Some ideas: using row norm (max element of probability vector) to penalize large element values in the solution.
  - LASSO doesn't help because the L_1 norm of all probability vectors are the same

Modified Ansatz

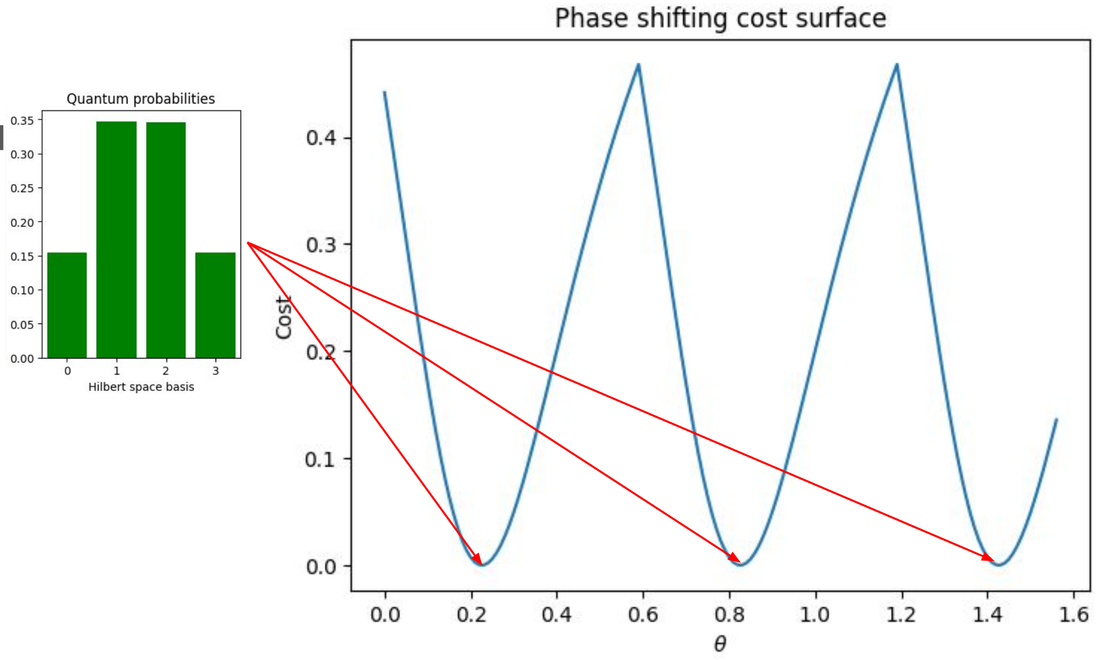
- "Problem-Specific" Ansatz: encodes the structure of the Poisson solution into ansatz to allow for a lot lower parameters per ansatz
- O(n) parameters for n qubits





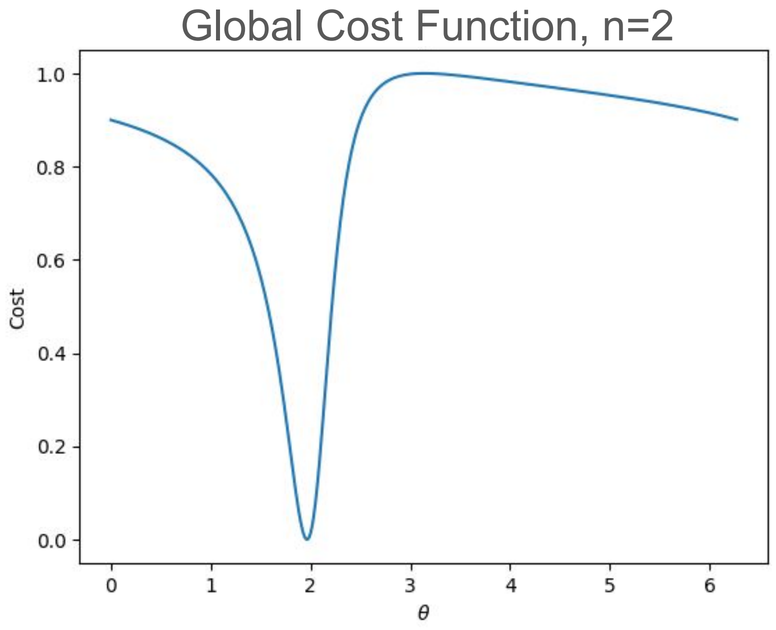
Modifying Cost Function

- With low-dimensional parameterized ansatz, we have more control over ou cost surface
- Phase shifting and taking the modulo of the rotation parameter to isolate the solution
- Doing this with higher qubit systems…



Global Cost Function

- Analysis of the global cost function shows the “double-minima” problem does not occur
- However, one can imagine how with higher qubit systems, this becomes a very flat barren plateau.
- We can try the same cost function modification trick.



Possible Paths Forward

- Try to return to global entanglement variational ansatz and use “dumb optimization” to minimize local cost function. Since the correct solution probably does exist in the cost surface, we just need to properly initialize the weights before minimizing.
- Try using a different kind of local cost function
    - Based on CNOT instead of CZ gates
    - Using Adder circuits
- Try to prepare a 2D Poisson decomposition (haven’t seen in literature)
- Work on outline of the talk for MATRX 2024 conference