QUANTUM COMPUTING

Anthony DiMaggio, Jessica Dong, Deepak Gopalakrishnan, Julia Granato, Bryant Har, Rohan Kulkarni, Michelle Liu, Craig Mulhern, Jr., Arianna Otoo, Ishika Patel, Ansh Sharma, Ian Viegas

Advisor: Daniel Kaplan
Assistant: Matheus Macena de Carvalho

ABSTRACT

The purpose of this work is to provide an overview of quantum computing and to describe and test some critical aspects of this technology with examples. All materials are based on the IBM Qiskit tool to develop quantum computing circuits. Quantum computers can be used to solve optimization problems much more efficiently than classical computers. Currently, real quantum computers experience decoherence errors. A linear relationship between the number of quantum gates and error was determined by varying the amount of CNOT and Hadamard gates separately. As of now, approaches to ameliorate these errors remain restricted due to the limited number of qubits on real quantum computers. With the VQE approach, a real-life application of quantum computers is being developed to solve a ground state analysis of simple molecules.

1. INTRODUCTION

1.1 Classical Computing

Classical computers operate using binary bits, meaning there are only two possible values—1 or 0. Bits can be thought of as true and false, or on and off. These bits are translated into electrical signals, where a high voltage represents a 1 and a low voltage represents a 0. Different programming languages have been developed to interpret these bits. ASCII, for example, translates bytes, which are strings of 8 bits, into English characters. Other higher level programming languages and compilers allow people to code and operate computers using languages closer to English. Classical computers can run an immense variety of algorithms using AND, OR, and NOT logic gates. Modern-day classical computing is very well suited for addressing many simple algorithms, but time constraints make it more difficult to rely on classical computers for complex algorithms. Classical computers process binary code one bit at a time. This means that when calculating probabilities, classical computers have to run each possibility separately. For instance, when trying to find the optimal combination in a scenario which is represented by three bits, there would be 8 possible bit combinations that would need to be processed separately. This doesn't present much of an obstacle in simple problems, but when it comes to immensely complex problems, like the stock market, classical computers are not always up to the task. Certain scenarios require computers to process an enormous number of variables, possible situations, and probabilities. It would take so long for classical computers to compute these problems that their usage would be almost impossible (1).

1.2 Qubits

A quantum state, the basis for quantum computing, allows computers to run calculations without being bound to the binary 1 or 0. Similar to how a classical computer's basic unit is a bit, a quantum computer's basic unit is a qubit which, like a classical bit, can be in a state in 1 or 0. However, a qubit can be in a superposition of 1 and 0 where it is simultaneously in a state of 1 and 0, with different probabilities of being either 1 or 0 (2). The idea of a superposition can be clearly explained by the thought experiment of Schrödinger's Cat. The premise is that there is a cat in a closed box that has a 50/50 chance of being either dead or alive, thus it is in a superposition of life and death. The act of observation, or measurement, by opening the box forces the state of superposition to collapse and the cat is found dead or alive. Like the cat, the qubit is simultaneously in a state of 1 and 0.

While powerful, a superpositioned qubit is fragile and any outside interaction, such as the walls of a container, results in an observation, collapsing the superposition and destroying it (3). Before measurement, qubits can represent numerous permutations of 1s and 0s at the same time. For instance, if there are three qubits, all 8 possibilities can be processed at the same time rather than in succession, and adding qubits exponentially increases a quantum machine's power. This means that with a thousand qubits, quantum computers would be able to outperform classical computers, opening a completely new range of situations that computers could explore. In the stock market, for example, a quantum computer would be able to efficiently examine numerous possibilities and could be a very effective tool (4).

1.3 Entanglement

Entanglement is a principle of quantum physics that is applied in quantum computers. Researchers can create entangled pairs of qubits, which exist in a related quantum state. They are entangled such that a measurement of one qubit will determine the measurement of the other qubit. Thus, measuring the state of one qubit allows the state of the other qubit to be known without collapsing it. Once entangled, qubits can be separated by any length and remain entangled until one or the other is measured or there is environmental interference. The mechanism behind how entanglement works is not completely understood, thus Albert Einstein referred to entanglement as "spooky action at a distance." For example, two particles can become entangled if they were formed from the decay of the same particle. They can only be separated and retain their entangled state if there is no environmental interference. External disturbances (discussed in detail in *3: Decoherence Issues with Current Technology)* cause a qubit to collapse from its state of superposition which leads to the break of the entangled state and the decay of the qubit (2). To prevent external disturbances, the particles have to be kept in a temperature of very cool temperatures, which are close to 0 Kelvin, and isolation from Earth's magnetic fields. While it is possible to approach a microkelvin and near absolute zero, any energy pulse could heat up the system to the point where the qubit is destroyed, which accounts for the error that is produced from actual quantum computers (5).

1.4 Quantum Teleportation

Quantum entanglement allows qubits to be teleported from one place to another. As mentioned in *1.3 Entanglement*, entangled qubits can be separated from each other by long distances and still be entangled. For example, there are two qubits that are entangled Q_A and Q_B

and are separated by 3 miles. The entangled qubits' properties are unknown. The purpose is to get an exact replica of Q_C ,which is currently with Q_A, to Q_B. By having Q_A and Q_C interact, both qubits are measured, then decay, and Q_C's properties are sent to Q_B. Operators at the location with Q_B now have the information needed to know how to change Q_B into Q_C without measuring the qubit. This application has a wild future that may lead to the teleportation of objects and humans, even though the complexity of human particles makes the theory unlikely. For instance, if every particle in a human was measured their particles' information would be sent to the new location and their original particles decayed, and changes would be made to the particles at the new location, which produces the same human in the new location. Although this would be groundbreaking technology there are moral questions about whether it is acceptable for a person's original self to be destroyed and a new one to be made (3).

1.5 Quantum Computers

Currently, IBM provides only 15 quantum computers that use qubits from a range of 5 to 53 for public use. Quantum computers like the IBM Q Experience are superconducting Josephson Junctions. No information and operations are stored in the computer itself since the qubit itself does not last long before it decays. Instead, information is passed through a cloud program to the quantum computer. Unlike classical computers, quantum computers do not have languages in which they are written in, but they use gates to change the rotation of the qubit. The user uses gates to control the rotation, and microwaves are shot at the qubit to change the rotation of the qubit. Quantum computers are still not perfect, since they produce slight errors in their calculation due to environmental interference. Many companies strive to achieve quantum supremacy where a quantum computer completes a calculation that is deemed impossible for a supercomputer to calculate (4). In 2019, Google's quantum computer, *Sycamore*, has completed a computation in a few minutes that would have taken a classical computer about 10,000 years to complete. In doing so, *Sycamore*, has become the first quantum computer reaching quantum supremacy. The current goal of quantum computers is to compute calculation not only faster than a classical computer, but also beneficial to society (6).

1.6 Qiskit

IBM has created a program called Qiskit for public use to execute code on quantum computers and simulators around the world. Qiskit provides a graphical interface that allows the user to submit circuits and code for the program to transpile it for the quantum computer or simulation. Qiskit's free quantum computers use a maximum of 5 qubits and its simulators use a maximum of 15 qubits. Quantum computers with more qubits require payment. The program is run thousands of times to produce accurate probabilities of the results. Each time the program is executed, the qubits' superposition collapses due to the act of measurement and new ones are used for the next execution. The simulations do not produce any errors, since they are simulated by classical computers and do not suffer environmental interference, but the actual quantum computers produce some errors that can be clearly seen in the results shown by a histogram. There are many types of gates that rotate the superposition of the qubit and some can change the likelihood of the outcome of a single qubit being measured. The Bloch sphere is a visual tool to help show the rotation of a single qubit. In the sphere, the z-axis faces upwards, the y-axis is horizontal, and the x-axis protrudes out of the paper. The Hadamard gate, or H gate, rotates the

qubit around the y axis. In simpler terms, if a qubit's original state is |0⟩, the H gate changes the qubit's superposition to be 50% a |1⟩ and 50% a |0⟩ when it is measured (|0⟩ + |1⟩). The X gate, or Pauli X gate, flips the state of the qubit. A qubit in state |1⟩ turns into |0⟩. The controlled-X gate or the CNOT gate applies the X gate on a second qubit if the first one is in a state of |1⟩, or it can entangle two qubits if the first one is in a superposition. The Rz gate rotates the qubit around the z-axis a set amount of radians. The cRz gate or controlled-Rz gate applies the Rz gate on a second qubit if the first one is in state |1⟩. The ccX gate or the Toffoli gate applies the X gate to a third qubit if the first two qubits are in a state of |1⟩. (7).

2. OPTIMIZATION ALGORITHMS

2.1 Overview

The applications of quantum computing can be seen with optimization algorithms. In this section, we will show the differences between the classical and quantum algorithms used to solve the following problem: what is the most optimal way to place Alice, Bob, and Chris into two taxis, Taxi #0 and Taxi #1, provided that Alice and Bob are friends but both Alice and Bob are enemies with Chris? The goal is to maximize the number of friend pairs and minimize the number of enemy pairs in a taxi.

2.2 Classical Algorithm

To understand the power of quantum computing, it is necessary to understand the limitations of a classical computer. In order to do this, the problem will be tackled with a classical algorithm using binary operations only. This algorithm is written in the Python computer language and is hardcoded for a three-person scenario. Let a *configuration* be an arrangement of Alice, Bob, and Chris, represented by three binary digits (bits). Each bit represents the person's *state*, or the taxi they choose.

The algorithm will first establish all possible configurations. Next, a system must be designed to indicate favorable configurations over unfavorable ones. A favorable configuration will maximize friend pairs (1) and minimize enemy pairs (0), while a bad case would deviate from this, with less friend pairs and more enemy pairs. The algorithm will assign a *score* to each configuration. The higher the score, the more favorable the configuration. After evaluating the score for all possible configurations, the algorithm will select the most optimal ones.

The algorithm is correct but note that this algorithm is designed only for a three-person scenario. Extrapolating to ten, a hundred, or a thousand people, this algorithm will still be correct, but will scale up exponentially, making it infeasible for a computer to calculate in a reasonable timeframe. This can be illustrated by printing each configuration's scores (Fig 1).

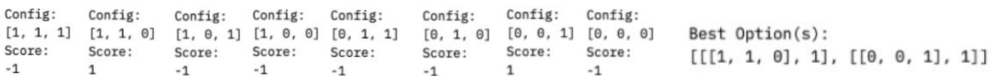


Figure 1: Classical Algorithm Results System output for each tested case

This shows that the classical algorithm is evaluating each possible scenario one-by-one, which can be computationally and economically expensive. For greater efficiency, a computational parallelism would be most desirable. This is where quantum computation is useful. The full code for this algorithm can be found in Appendix A.

2.3 Quantum Algorithms

We will work through three quantum algorithms to serve as a comparison to the classical computer, each one building upon the previous one until there is a viable solution for Algorithm 3. The initial algorithm is meant to mirror classical computing on a quantum computer, while the second algorithm begins to make use of quantum features for generalizing the solution to larger problems. The final algorithm will make use of a well-documented algorithm known as Grover's Algorithm to output a good state with high probability after just a single iteration, as opposed to running through all 8 combinations as seen in the classical algorithm. All complete circuits are available in the Appendix A.

2.3.1 Algorithm 1 - Bit Flip

The circuit will start with three qubits, with qubits q[0], q[1], and q[2] representing Alice, Bob, and Chris, respectively. The first step is to apply a Hadamard (H) gate to each of the qubits to put them into a superposition, as shown in Figure 2. Because each qubit can be either $|0\rangle$ or $|1\rangle$, all eight possible configurations are considered, with states $|0\rangle$ and $|1\rangle$ representing their respective taxi numbers. For example, the system can be represented in the state $|010\rangle$, signifying that Alice and Chris are in Taxi #0 and Bob in Taxi #1.

The algorithm will first identify the number of friend pairs. To do this, a Toffoli gate is applied so an additional qubit q[3] will be flipped if both q[0] and q[1] have a value of $|1\rangle$. This represents when Alice and Bob are in Taxi #1 together. However, applying the Toffoli gate alone will not flip q[3] if they are in Taxi #0 together, as the conditions for gate activation - two "true" values - will not be satisfied.

To address this possibility, two NOT (X) gates are applied after the Toffoli gate to flip q[0] and q[1]. Therefore, if they were in state $|0\rangle$, they will now be flipped to state $|1\rangle$. Another Toffoli gate is then applied to flip q[3]. In either case, q[3] will have a value of $|1\rangle$ if q[0] and q[1] are in the same state. Finally, two X gates are reapplied to q[0] and q[1] to reset them to their original states. This process is shown in the first segment of Figure 3.

The second task will be to identify the number of enemy pairs. To do this, the same strategy to identify friend pairs will be used with each of the possible enemy pairs. In the circuit below (Fig 3), if Bob and Chris are in a car together, q[4] will be flipped, while if Alice and Chris are together, q[5] will be flipped.

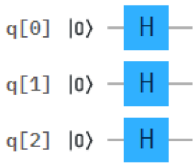


Figure 2: Hadamard Gates
3 Hadamard Gates to superimpose all 8 possible states

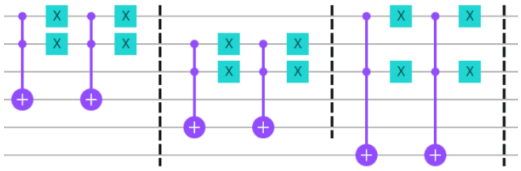


Figure 3: Pair Identification Circuits
Used to identify which pairs of people are in the same car

A half-adder mechanism as displayed below (Fig 4) will be used to add q[4] and q[5] to determine the number of enemy pairs, which will be represented by q[6] and q[7]. In the quantum circuit below (Fig 5), the XOR and AND gates are replicated with the two CNOT gates and the Toffoli gate, respectively.

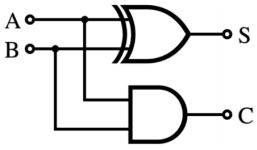


Figure 4: Classical Half-Adder
XOR and AND Gates

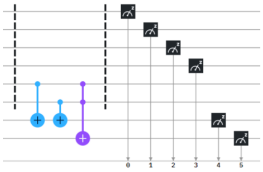


Figure 5: Quantum Half-Adder
2 CNOT and 1 Toffoli Gate

In the latter half of Figure 5, the algorithm will measure the first four qubits, in addition to q[6] and q[7]. From the possible configurations, the two optimal solutions can be selected based on the states of the final three qubits. The ideal result should be "001," as that will signify that the arrangement has no enemy pairs and one friend pair.

After running the quantum algorithm, the most optimal scenarios were when Alice and Bob were in one car and Chris in the other car, which aligns with the classical algorithm's results. Thus, with a quantum algorithm, all possibilities were able to be considered in parallel and the most optimal ones identified. Unfortunately, since no attempt was made to modify the probability of each outcome in this process, all possible outcomes have equal probability, as shown below (Fig 6). The variation observed is due to the random process built into the simulation. To do this, a phase shift approach with the qubits is needed.

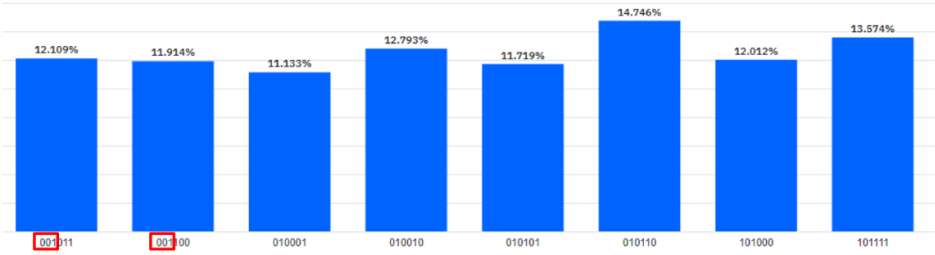


Figure 6: Bit Flip Results Similar Probabilities For All Configurations

2.3.2 Algorithm 2 - Phase Shift

The bit flip algorithm works well, but it is possible to write an algorithm that will exploit the power of a quantum computer even more. Quantum superposition makes it possible to implement a phase shift of the qubit's state in each dimension. This can be visualized on the

Bloch sphere as rotating the qubit's vector state about the *x*, *y*, and *z* axes. In this way, the qubit's probability of collapsing to a basic state, |0⟩ or |1⟩, can be manipulated.

This algorithm will utilize 5 qubits and 4 classical registers (classical bits that will store measured information). The first three qubits still correspond to the states of Alice, Bob, and Chris. The fourth qubit will be a *writing* qubit, meaning if the qubit's state is |1⟩, the qubit will perform an operation, or write, on another qubit. The fourth qubit is used as an internal working bit and will not be measured. Finally, the fifth qubit will be the *rotation* qubit. The probabilities of this qubit will be manipulated to indicate preference towards one configuration over another.

The first step is to prepare the qubits (Fig 7). Like the bit flip algorithm, the H gates are applied to q[0], q[1], and q[2], putting them into superposition. The algorithm must also prepare the rotation qubit in a state of superposition, but first we must establish the axis to rotate the qubit about. In this algorithm, we chose to rotate q[5] about the *z*-axis, allowing for use of the cRz gate. The fourth qubit will always remain in state |0⟩ until it needs to write, so it does not have to be prepared.

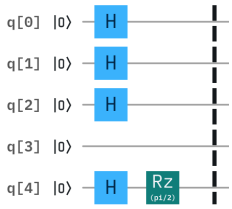


Figure 7: Qubit Preparation Initializing qubits using Hadamard Gates

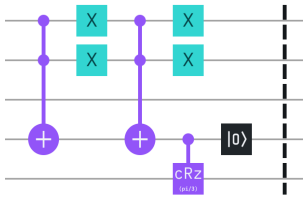


Figure 8: Friendly Phase Shift Raising the likelihood of the marker for friendly pairs

Now that all qubits have been prepared, the algorithm must indicate preferences for certain configurations. To represent this favorability factor, the rotation qubit will be manipulated. The algorithm will rotate the rotation qubit $+\pi/3$ radians about the *z*-axis if the qubit states are favorable and it will rotate the rotation qubit $-\pi/3$ radians about the *z*-axis if the qubit states are unfavorable. The favorable condition is shown in Figure 8.

The successive Toffoli and X gates perform an X on q[3] if Alice and Bob are in the same car, setting the writing qubit to |1⟩; it then rotates the rotation qubit $+\pi/3$ radians. Finally, the writing qubit is reset to |0⟩. The sub-algorithm to note unfavorable conditions is almost the same as that for the favorable condition, except that the rotation qubit is rotated $-\pi/3$ radians instead (Fig 9). The algorithm then prepares the rotation qubit for measurement by applying an H gate. Finally, all qubits except for the writing qubit are measured (Fig 10).

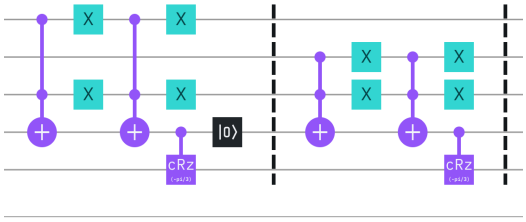


Figure 9: Enemy Phase Shift Reducing the likelihood of a marker on q[4] when an enemy pair is present

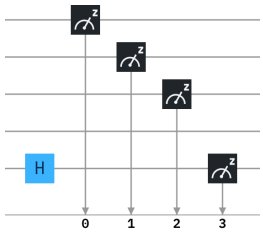


Figure 10: Measurement Collapsing the superposition

The effect of these rotations makes it such that the probability of a $|1\rangle$ is $\cos^2(\Theta)$ while the probability of a $|0\rangle$ is $1\text{-}\cos^2(\Theta)$, so the phase shift process increases the probability of a 1 for good pairings by decreasing theta, and decreases it for bad pairings by increasing theta. Our results are shown in Figure 11.

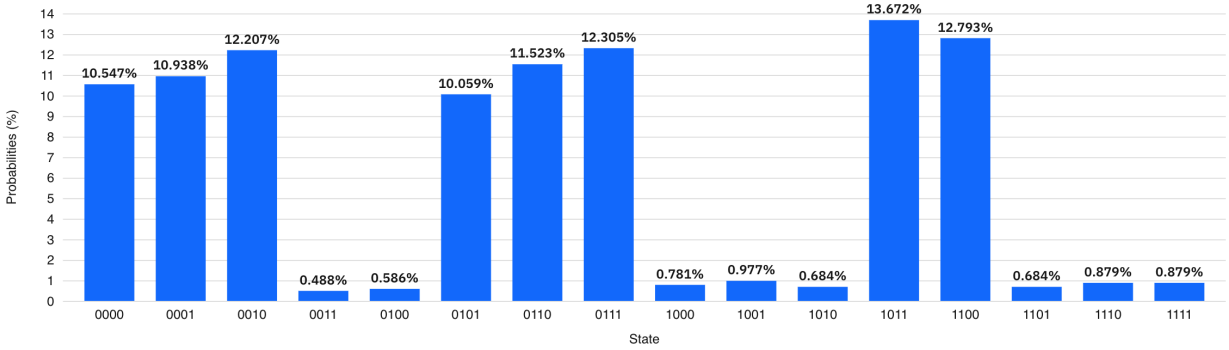


Figure 11: Phase Shift Results on a Quantum Simulator
Outcomes of 1024 iterations of the Phase Shift Circuit on a quantum simulator

Considering the results where the fifth qubit is $|1\rangle$ (amplifying favorable conditions), the optimal solutions, 011 and 100, are observed overwhelmingly over the other possible configurations. However, note that if the fifth qubit is $|0\rangle$ (amplifying unfavorable conditions), the optimal solutions are measured at exceptionally low levels, making the total probability of measuring a good solution on any given iteration still ⅛. The quantum algorithm has come closer to solving the problem, but there's still one more step to finish the optimization.

2.3.3 Algorithm 3 - Applying Grover's Algorithm

In the above algorithm, the user still had to deal with two final possibilities for the analysis by selecting an answer starting with 1. This can be done automatically with Grover's algorithm. The goal of Grover's algorithm is to run all possible cases simultaneously through superposition, mark the good solutions using a function called the oracle, and then finally apply amplitude amplification to increase the chance of collapsing the qubits into a good solution.

The Oracle Function

For this problem, an extension of the phase shift circuit will serve as the oracle function and mark the good cases in the superposition. Below, the phase shift circuit has been duplicated for convenience (Fig 12), with q[6] serving as a marker for good and bad cases.

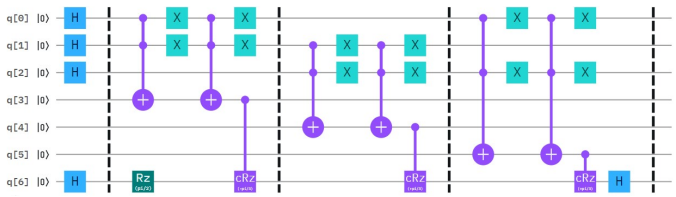


Figure 12: Phase Shift for the Oracle Circuit A duplication of the phase shift algorithm from section 2.3.2 to serve as a component of the oracle circuit

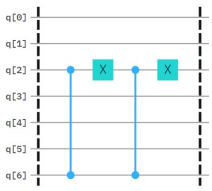


Figure 13: Amplitude Inverter A circuit with cZ gates to invert the amplitude of marked states

Following the setup for q[6], a controlled Z gate construction (Fig 13) is used to flip the amplitude of the good cases, effectively marking them to set up for the amplitude amplification step later on. The X gates are present in order to flip the sign of the amplitude even if q[2] is 0. The final step to complete the oracle is to reset the extra qubits used for computation as to remove their entanglement with the first three qubits. This can easily be done as all of the operations are reversible, allowing us to reset q[3], q[4], q[5], and q[6] back to $|0\rangle$ (Fig 14).

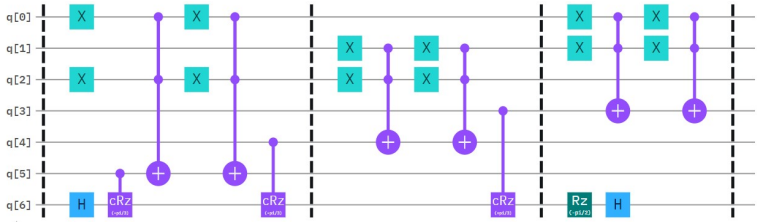


Figure 14: Qubit Reset Circuit Circuit used to reset calculation qubits to the $|0\rangle$ state

Now, assuming the oracle marks the correct cases, the qubits will be left in the following superposition, completing their preparation for the Amplitude Amplification function:

$$|000\rangle + |001\rangle + |010\rangle - |011\rangle - |100\rangle + |101\rangle + |110\rangle + |111\rangle$$

Amplitude Amplification Function

The last component of Grover's Algorithm is the Amplitude Amplification function. It effectively serves to increase the amplitude of the marked cases while uniformly decreasing the amplitudes of the unmarked cases by reflecting each amplitude across the mean amplitude. An example of how this works is shown below for an arbitrary function with a marker on 3.

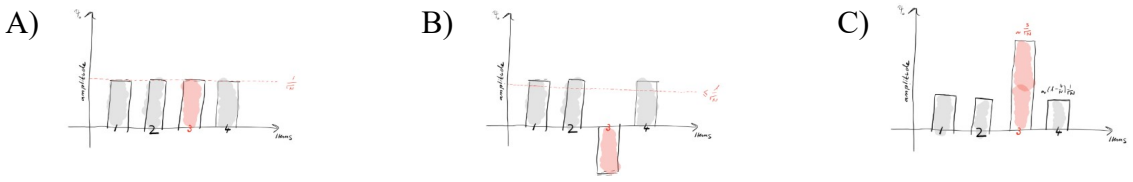


Figure 15: Grover's Algorithm Visualization IBM Q Visualization of Grover's Algorithm (8)

Initially, all outcomes have equal probability as shown in Figure 15a. The oracle function flips the sign for the target value, which is 3 in this case (Fig 15b), leading the overall mean, shown by a dashed line, to decrease. Finally, in Figure 15c, the amplitude amplification process

reflects each amplitude across the mean, causing the amplitude of unmarked cases to decrease slightly while the marked case increases significantly.

This process of using an oracle followed by an amplification circuit is repeated while the mean in the second step is still positive, which can be shown to scale to O(√N) iterations (8), a quadratic improvement over the classical O(N) iterations. Figure 16 shows an implementation of the amplification function which serves to amplify whichever states have a negative amplitude.

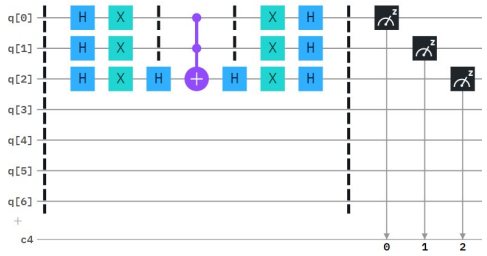


Figure 16: Amplitude Amplification An implementation of a single amplitude amplification

After running the full circuit on a noiseless quantum simulator for 8192 iterations, the histogram below (Fig 17) shows that the circuit has successfully discriminated for the marked cases, with approximately an 80% chance of getting a good solution on any given runthrough as opposed to the 25% chance that the classical and earlier quantum circuits demonstrated.

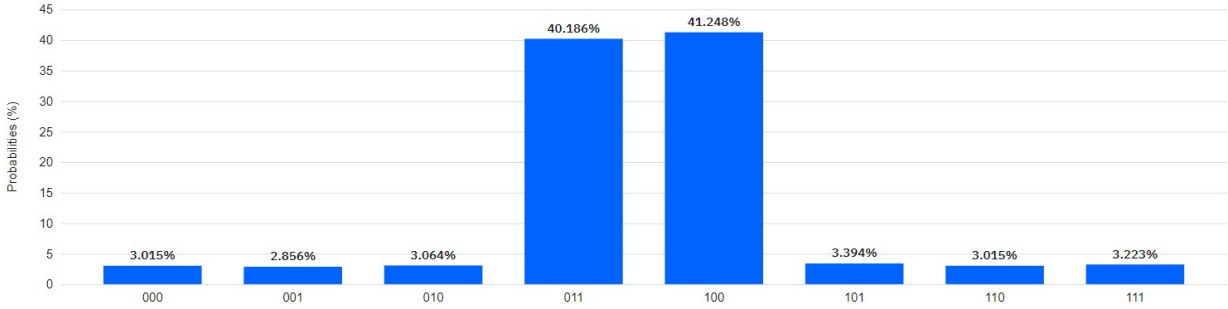


Figure 17: Grover's Algorithm Results on a Quantum Simulator
Final results for the Taxi Problem quantum circuit run on a noiseless quantum simulator

The 80% accuracy in this instance is due to the phase shift method used in the oracle function as it inherently makes it possible, albeit unlikely, for a case to be marked incorrectly. This percentage can be improved by choosing more extreme phase shifts for good and bad pairings, or by constructing an oracle that strictly outputs 1 for good cases and 0 for bad cases.

These quantum algorithms serve to demonstrate the power of superposition in solving a problem with brute force. While a classical computer requires an iteration for each possible configuration to brute force a solution, superposition allows quantum computers to reduce the number of iterations significantly by dealing with all possible outcomes in parallel, posing an immense advantage for brute forcing optimization problems with large solution sets.

3. DECOHERENCE ISSUES WITH CURRENT TECHNOLOGY

3.1 Overview

Decoherence, or noise, acts as one of the main obstacles to the reliability of quantum computing. Over time, the environment causes qubits to lose information, meaning that the quantum states will be randomly changed. Sources of decoherence in the environment include heat, vibrations, magnetic fields, and other various outside factors. As more gates or qubits are added, the computational time increases, and the error in the quantum computations worsens. Thus, qubits can only retain their superposition for a limited amount of time, known as the coherence length. Given that quantum supremacy entails a high number of qubits, the issue of quantum error will only grow in importance over time.

3.2 Effect of Gates on Decoherence

In order to determine the exact effect of the number of quantum gates on the noise, the numbers of CNOT gates and Hadamard gates were varied in two independent experiments. They were performed with both IBM's Ourense and London quantum systems to account for slight differences between quantum systems. Moreover, barriers were added between each gate to adjust the timing between processes so that specific events occur successively (Fig 18).

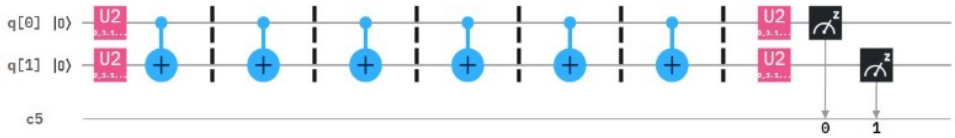


Figure 18: Arrangement of CNOT Circuit CNOT gates were increased to determine their effect on decoherence. The use of barriers prevented the transpiler from simplifying the circuit.

3.2.1 Effect of CNOT Gates

A simple circuit was designed in which the simulated outcome should be completely in the $|00\rangle$ state. Therefore, outcomes in any of the other three states — $|01\rangle$, $|10\rangle$, and $|11\rangle$ — could be considered definitively as error. The circuit used only even numbers of CNOT gates so that the effect of each CNOT gate would be canceled out by the other gate in its pair. However, the barriers forced the quantum computer to perform each gate individually, thus increasing the amount of time for the computation.

On IBM's Ourense quantum system, a direct relationship between the number of CNOT gates and percent error was observed (Fig 19a). Moreover, the correlation was closely linear with an R^2 value of 0.988. The same circuits were computed using IBM's London quantum system, in which the results corroborated the previous findings (Fig 19b). The number of CNOT gates versus the percent error still showed a linear relationship, with an R^2 value again of 0.988.

Comparison between the Ourense quantum system and the London quantum system showed similar results, but London had a greater percentage of errors as the same number of CNOT gates was increased. Though less significant at lower amounts of gates, the percent error of the London quantum system becomes more notable when twenty CNOT gates are used. While the Ourense quantum system has 21.46% error, the London quantum system displays nearly double at 38.95% error. Interestingly, the London quantum system has a quantum volume of 16 as compared to the Ourense quantum system's quantum volume of 8. The quantum volume is a

metric designed to take into account the quantum computer's performance capabilities and errors so that a higher quantum volume denotes a more effective quantum system.

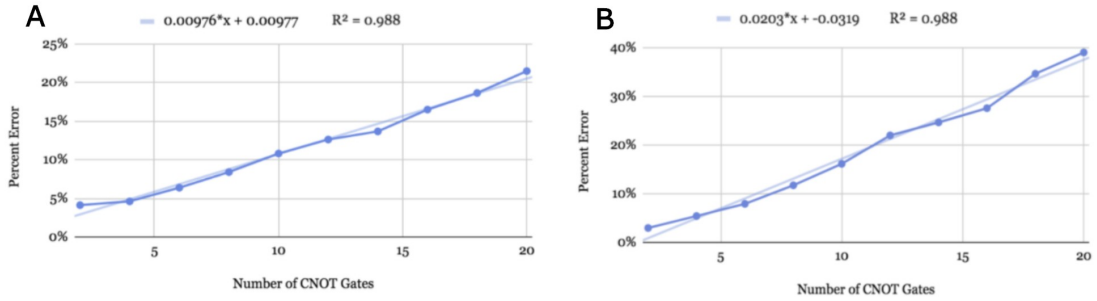


Figure 19: Effect of CNOT Gates on Percent Error Computations were performed using IBM's Ourense and London quantum systems **a.** The results from the Ourense system imply a direct linear relationship between the number of CNOT gates and percent error. **b.** The results from the London quantum system also confirm the direct linear relationship between the number of CNOT gates and percent error, as also found by the Ourense system.

3.2.2 Effect of Hadamard Gates

In creating the circuit to test the relationship between the number of Hadamard gates and the percent error produced by the IBM London and Ourense quantum systems, a base circuit was composed in which two Hadamard gates were each applied to two qubits, to ensure that the simulated response would be a |00⟩ state (Fig 20). This allowed easy error identification when running the circuit on each quantum computer. From this base circuit, the number of Hadamard gates was increased in sets of two, in order to ensure that the effects of the Hadamard gates would negate each other, but ensured that barriers were present between each gate operation so that the quantum computer would run each gate individually.

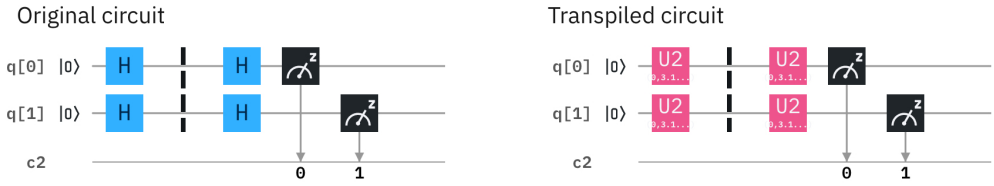


Figure 20: Arrangement of Base Hadamard Gate Circuit A base circuit consisting of two Hadamard gates on each qubit—barriers prevented simplification of the transpiled circuit.

Similar to the circuit composed of CNOT gates, the circuit made with H gates showed a linear upwards trend relating the number of H gates to the percent error. The R^2 value for results from the Ourense quantum system was 0.953, indicating a very close linear relationship between the two variables (Fig 21a). With an R^2 value of 0.961, the London quantum system also displayed this trend, demonstrating that increasing the number of H gates proportionately increases the percent error (Fig 21b). When the percent error, in comparison to the number of H gates, is examined between both Ourense and London, the percent error generally seems to be lower when the circuit is run on IBM's London quantum system. Though not consistent with the trend observed with the CNOT circuit, this observation seems more in line with the fact that the London quantum system has a larger quantum volume of 16, in comparison to Ourense's 8.

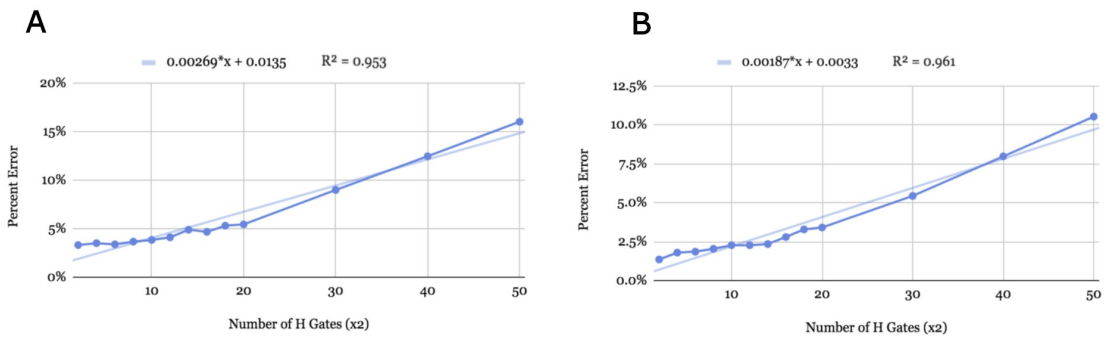


Figure 21: Effect of Hadamard Gates on Percent Error Computations were performed using IBM's Ourense and London quantum systems. **a.** The results from the Ourense system display a direct linear relationship between the amount of Hadamard gates and the percent error **b.** The results from the London quantum system again corroborate the direct linear relationship between the number of Hadamard gates and percent error, found by the Ourense quantum system.

3.3 Techniques for Reduction of Error

One possible solution to reduce the error inherent in quantum computers is to build circuits in order to replace individual qubits with a set of qubits so that the measurement would be the majority vote for the result. To reduce the error present in the CNOT quantum circuit (Fig 22), a circuit was constructed which would replace a qubit's value if it did not match up with its partnering qubit (Fig 23).

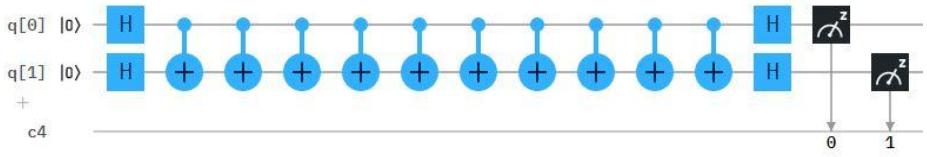


Figure 22: Original CNOT Quantum Circuit. This is a sample circuit used to simulate the quantum error. The circuit in Figure 23 attempts to resolve this problem using error correction.

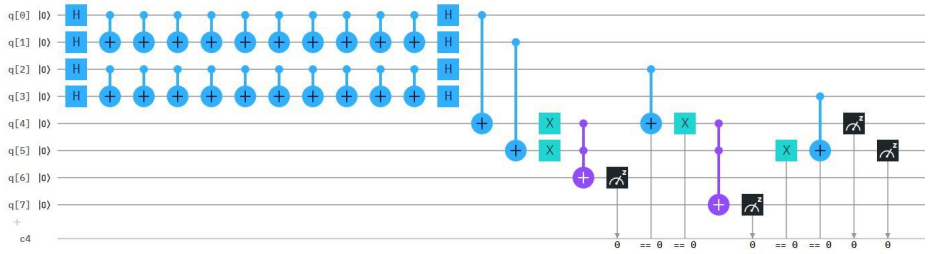


Figure 23: Sample Quantum Error Correction Machine This machine can only be run on the Melbourne quantum computer, which does not support if statements in its quantum circuits. It also requires more than triple the number of qubits than the circuit without any error correction.

The Quantum Error Correction Machine in Figure 23 highlights the limitations of quantum error correction at this time. The first issue is that the number of qubits required to perform error correction is many times higher than the original quantum circuit. This is seen as the error corrector requires seven qubits compared to the original two. Given the limited number

of qubits present in modern-day quantum computers—IBM's newest quantum computer has 53 qubits (9)—the type of error correction seen in the sample machine is simply not realistic for the much more complex circuits needed to compute intense mathematical problems.

Furthermore, the quantum corrector works off the assumption that the user has recognized what the results of the machine are supposed to be and that the two outputs must match in value. In the real world, quantum computers will be used to solve questions that nobody has the answer to, meaning that any correction device will need to be even more complex than what is represented in Figure 23 and only increasing the number of qubits necessary to run it. In addition, the Melbourne quantum computer is not able to run the circuit from Figure 23 given that it includes if statements relying upon the value of a classical register. This highlights how the limitations of quantum computers can affect the reliability of their results.

One possible solution to the problem of quantum error correction is to use a mix of both quantum and classical computing to run quantum circuits. For instance, in the circuit from Figure 22, the quantum computer will first run all of its shots. This will return the percent values for each output value. From there, a classical computer will determine that the value with the highest percent return will be the correct answer. Then, a second set of shots will be run. After each shot, the results will be reported to a classical computer. Next, the classical computer will decide whether or not to run a second trial and replace the values from trial 1 with trial 2. This will only work on circuits with limited error. If the error exceeds a point wherein a correct value cannot be ascertained from the outset, then this suggested error corrector will not work. However, the suggested model of combining quantum and classical computing to reduce error will still have benefits given that the user knows some characteristics that their correct answer will match. When quantum computing reaches a point wherein a computer can manipulate a high number of qubits, the digital computing aspect may be removed as any error correction can be achieved without measuring the result.

4. VARIATIONAL QUANTUM EIGENSOLVERS

4.1 Overview

The Variational Quantum Eigensolver (VQE) is a quantum computing algorithm that can be used to simulate combinations or states. The general structure of the algorithm involves making an ansatz (trial state) and slightly varying the parameters and variables within that state and measuring some inherent value of that new state (expectation value). This is repeated in a series of steps until a satisfactory state is found (10). In other words, the VQE algorithm is an iterative approximative method, similar to machine learning, that leverages quantum computing to model many-body systems. The purpose of this section is to provide a description of this process and to examine its efficacy.

4.1.1 Applications

Intuitively and correctly, VQE is especially equipped for dealing with classes of problems that involve many possible states. Notably, it can be used to model nature or, more generally, optimize solutions to combinatorial optimization problems. A particularly useful

application is that of simulating molecular ground state energies in computational chemistry. Finding the most stable state of a molecule, or the ground state, offers a better understanding of electron configurations and how molecules bond. Understanding molecular interactions has numerous practical applications, including engineering solar cells, filtering out poisonous gases in pollution, and calculating catalytic conversions, such as those of enzymes like nitrogenase, leading to new and better cures for humanity's worst diseases (11).

4.1.2 Mathematical Background

There are a few mathematical and scientific terms to become familiarized with before attempting to understand the mechanics of VQE, or quantum theory in general. These terms mostly involve linear algebra and complex-valued functions. Here are a few noteworthy terms:

1. *Eigenstates*, or states, are represented with vectors called eigenvectors. The ground state of a system is its lowest energy eigenstate.
2. *Eigenvalues* are scalars that scale the eigenvectors, which are vectors that represent the states of the given quantum system.
3. *Wave functions* are functions that describe how a particle's state and position evolve over time in a quantum system. Squaring them produces probability amplitudes, which represent electron densities in this context.
4. A *hartree* is a unit of energy equivalent to about 27.21 electron volts.
5. The *expectation value* is the expected result of a measurement, calculated as the weighted average of a measurement made in each possible state.
6. *Bra-ket notation* is used to elegantly represent eigenvectors (kets or $|\rangle$) that represent states and linear maps (bras or $\langle|$). Bras, kets, and operators represent a quantum state.
7. *Operators* are functions that map one space to another. They can act on bras and, in common notation, are written between the bras and kets when describing quantum states.
8. The *Hamiltonian* is a matrix operator that corresponds to the sum of kinetic and potential energies in a system. Applied to a quantum state, it is written as $\langle\Phi|H|\Psi\rangle$ (Fig 24).

$$\varepsilon[\Psi] = \frac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi \mid \Psi \rangle}$$

$\varepsilon[\Psi]$	Expectation Value or Energy for Wavefunction Ψ
$\mid \Psi\rangle$	Eigenvector in Bra-Ket Notation
$\langle \Psi \mid$	Linear Function that Maps Vectors to Complex Values in Bra-Ket Notation
$\hat{H}$	Hamiltonian, or Operator that Represents the Sum of Kinetic and Potential Energies

Figure 24: Calculating the Energy of a Trial State

4.2 The Hartree-Fock Method (HF)

Generally, when calculating the ground state energy of a molecule, the contributions of the nucleons (protons and neutrons) are neglected using the Born-Oppenheimer approximation. Much like how the Earth's movement through space is unnoticeable to those on its surface, so too are the nucleons' kinetic and potential energies to the energy of the isolated molecular system. With this approximation, ground state energy can be more readily calculated (12).

The Hartree-Fock Method offers an effective way to approximate the ground state energy of a molecule, which considers the *average* effects of each electron to simplify calculations. This generates the Hartree-Fock state, which accounts for 99% of the molecule's ground energy (13).

However, the Hartree-Fock method fails to deliver the last 1% of accuracy because it ignores the interactions between different electrons (known as "electronic correlation"). These interactions mean electrons can momentarily be excited into higher energy orbitals that still contribute to the ground state energy. Hartree-Fock, however, fails to account for the fact that these excited electrons may be in a superposition between various higher energy orbitals and instead "pigeonholes" each electron into the lowest energy orbital (13). To resolve this, we turn to quantum computing algorithms that can efficiently model these electronic interactions.

4.2.1 Coupled Cluster Singles and Doubles (CCSD)

CCSD resolves most of this 1% difference in total energy by considering electron interactions and excited states. It takes these excited states into account by using smaller excitations to approximate higher ones with an equation not unlike a Taylor series. Adding the energy of these excited states to the Hartree-Fock state allows for the approximation of the total expectation value of the molecule's ground energy state with the following equation (14).

$$\psi_{CCSD} = e^{T_1+T_2}\Phi = \Phi + T_1\Phi + \frac{T_1^2}{2}\Phi + T_2\Phi + \ldots$$

Φ = Hartree-Fock state
ψ = Expectation Value
T_1= Single Excitation Operator
T_2= Double Excitation Operator

CCSD only involves single and double excitations in order to be computationally affordable, but accuracy can be increased by accounting for more excitations. By optimizing the parameters T_1 and T_2, which represent the weight of each excited state, we can find the upper bound of the minimum expectation value for the molecule's ground state energy. To do so, the Hamiltonian, which corresponds to the total energy, must be minimized (Fig 24). By finding its minimum eigenvalue with VQE, we can find an upper bound for the ground state energy of a molecule (14).

4.3 Mechanics of VQE in Computational Chemistry

With the computational power of quantum computers, this calculation becomes much more efficient. The VQE combines quantum and classical computing to approach the upper bound of the minimum eigenvalue for the matrix representing the expectation value of the Hamiltonian using the variational method. Each qubit correlates to a molecular orbital in an encoding format known as the Jordan-Wigner transformation (15).

4.3.1 The Variational Method

VQE relies on the variational method for which it is named. In essence, the method consists of preparing an initial trial wave function, or ansatz, that depends on some parameters (variational parameters), calculating the energy (expectation value) of that state, modifying those parameters, and repeating the process until a satisfactorily low expectation value is reached (Fig 25). That resultant wave function, with its minimized energy, is a good approximation for the energy of the ground state wavefunction (12). The guiding principle behind this is the variational principle, which offers two critical assumptions:

1. The trial state is the ground state if and only if the wavefunctions are the same for both.
2. Any trial state is an upper bound of the ground state's energy.

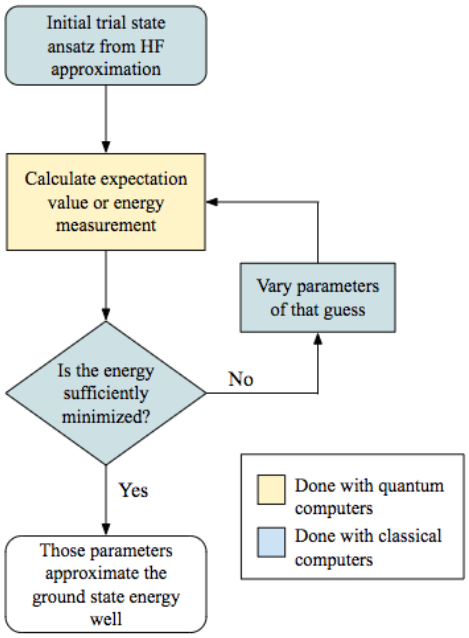


Figure 25: VQE Flowchart Steps of the variational method used to calculate ground state energy in VQE.

It is clear, then, that when minimizing the trial state energy, the resultant expectation value also approaches the ground state energy. In VQE, the variational method proceeds in a series of steps that progressively improves the approximation of the ground energy by altering the variational parameters T_1 and T_2 in the CCSD calculation. Its overarching process relies on gradually zeroing in on the ground energy state upper bound through slight tweaks to the CCSD parameters and consequent slight changes in measured energy. The calculation of the expectation value is completed via quantum computation while the optimization of the parameters is completed classically (Fig 25) (10).

4.3.2 Choosing an Ansatz

The choice of the ansatz, or starting guess, is critical to obtaining good approximations. As such, should a poor initial ansatz be chosen, the variational method may approach a local minimum rather than the global minimum of the expectation value (Fig 26). Thus, the Hartree-Fock state must first be obtained for use as an initial trial state (i.e. ansatz), since it is already a good approximation of the ground state. This is then fine-tuned using VQE (10).

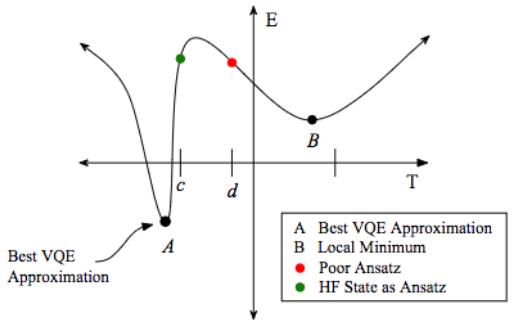


Figure 26: Choosing a Good Ansatz Using a poor ansatz may result in calculating a local minimum for the expectation value rather than the global minimum. In contrast, using the largely accurate HF state as an ansatz will allow for much more accurate approximations.

4.4 Implementation of the VQE Algorithm

For molecules with very few electrons, VQE can be readily simulated by classical computers. Specifically, the algorithm can be explored through IBM's Qiskit Python notebooks.

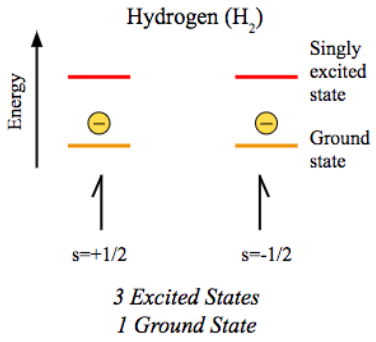


Figure 27: Spin orbitals of electrons in H_2
There is one permutation in which both electrons are in the ground state, and three permutations in which they are singly excited.

Initially, the types of atoms that constitute the molecule and the locations thereof in the xyz-space must be defined in order to initialize the electronic system. It is critical to note that electrons of a certain spin can only occupy spin orbitals of the same spin, even when in an excited state. These excited electronic configurations, whose number is dependent on the number of electrons and spin orbitals, each represent individual energy terms in the CCSD calculation whose weighting or probabilities comprise the variational parameters. For instance, computing the ground energy for hydrogen gas, with two electrons of opposite spin and two spin orbitals for each spin (four spin orbitals in total), requires three variational parameters for the three excited states and four qubits for the four spin orbitals (Fig 27). Evidently, the more excited configurations to consider, the more computationally intensive VQE becomes. All of this information is inputted into or calculated within the simulation.

```
driver = PySCFDriver(atom='O 0.0 0.0 0.0; H 0.757 0.586 0.0; H -0.757 0.586 0.0')
```

Each letter in the argument of the function is the elemental symbol for an atom and the numbers that follow are coordinates in the xyz-space. This way, the information is readily relayed to the quantum computer via an automatic transpiler. The transpiler converts an algorithm into a circuit composed of quantum logic gates. When comparing these VQE estimations to real values, the nuclear repulsion energy is notably used to apply a correction shift to the output energy. With this method and some computational shortcuts, ground state energy approximations were acquired for H_2, LiH, H_2O, and BeH_2 using code provided by Qiskit (16). Annotated code for the calculation of H_2O's ground state energy is provided in Appendix B2.

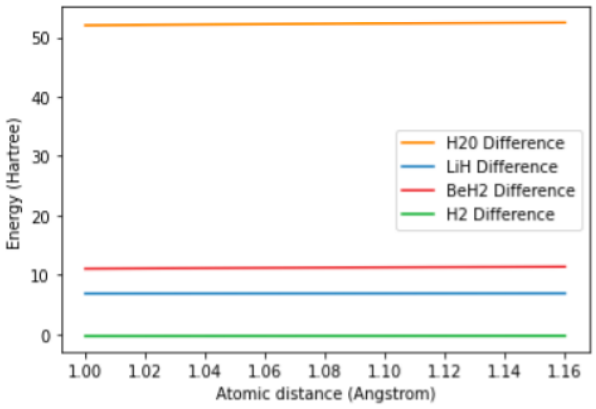


Figure 28: Difference between ground state energy calculated via quantum simulator and the actual value

The ideal VQE model performs well for short distances, with an error in the range of 10^{-8} Hartrees for these atomic distances when calculated on H_2 (Appendix B1). This magnitude of this error is acceptable for the purposes of chemical simulation, where error is generally. However, H_2 is the simplest molecule to simulate, and the approach does not scale well. With molecules like water, noisy error reaches many Hartrees.

A visualization of the VQE model's error over a range of atomic distances on these molecules reveals a few key trends (Fig 28). Critically, the difference between the VQE-approximated and ideal values is always positive, since the variational method always offers an upper bound when optimizing ground state energies per the variational principle. However, this

is not necessarily true for all the arbitrary classes of problems where VQE could be applied. Reductions and frozen core simplifications can allow for easier approximations. These methods generate a lower and less precise approximated ground energy by ignoring some unoccupied orbitals and neglecting the inner electrons respectively, but they reasonably offset this discrepancy by conserving computational power. Though these strategies skew the results, the trends remain (Fig 28).

4.4.1 Limitations

Simple molecules such as H_2 can be simulated fairly quickly. However, calculating more complex molecules quickly becomes a challenging task, since the number of spin orbitals and consequent qubits required (per Jordan-Wigner Encoding) rapidly increases (Table I). Worse still, additional qubits are necessary to correct errors and store supplementary values. This limits quantum computers to mostly light hydrides atoms. For reference, IBM's largest quantum computer has only 53 qubits, and currently, the most complex molecule simulated is BeH_2 (17).

Table I Complexity of Molecule and Qubits Required

Molecule	Electrons	Qubits Used (Spin Orbitals)	Variational Parameters*	Computational Shortcuts
H_2	2	4	3	None
LiH	4	12	92	R
BeH_2	6	14	204	FC, R
H_2O	10	14	140	FC, R
NaCl	28	36	4340	Not feasible

* Equal to the number of possible excited electron configurations

Note. *Freeze Core (FC) is an approximation technique where inner orbital electrons are ignored. Reductions (R) are when unoccupied molecular orbitals are discarded. This is done cautiously.*

Even more disastrous is the accumulation of noise or decoherence. With repeated use and increased numbers of qubits, noise worsens and error progressively increases (Appendix B3). Though ideal quantum computers were assumed here, the true approximations are far less precise due to engineering imperfections in modern quantum computers. Minimization of the number of qubits is critical to scale up to larger, more computationally intensive molecules.

REFERENCES

1. BBVA. [Internet]. [updated 2019 December 10] Quantum Computing: How it Differs from Classical Computing? Available from: https://www.bbva.com/en/quantum-computing-how-it-differs-from-classical-computing/
2. Giles, M. [Internet]. [updated 2019 January 29]. Explainer: What is a Quantum Computer. Available from: https://www.technologyreview.com/2019/01/29/66141/what-is-quantum-computing/
3. Bouwmeester, D., Pan, J., Mattle, K., Eibl, M., Weinfurter, H., Zeilinger, A. [1998]. Experimental Quantum Teleportation. Philosophical Transactions: Mathematical,

Physical and Engineering Sciences, 1733-1737. Accessed from: www.jstor.org/stable/55008
4. Linchfield, G. [Internet]. [updated 2020 February 26]. Available from: https://www.technologyreview.com/2020/02/26/916744/quantum-computer-race-ibm-google/
5. Horodecki, R., Horodecki, P., Horodecki, M., Horodecki, K.[written 2007]. Quantum Entanglement. Available from: http://eds.b.ebscohost.com/eds/detail/detail?vid=0&sid=3dfbf654-eed2-4045-aff6-c71ed2eaad80%40pdc-v-sessmgr06&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#db=edsarx&AN=edsarx.quant-ph%2F0702225
6. Satzinger, K. [Internet]. [updated 2020 March 3]. Quantum Supremacy: Benchmarking the Sycamore Processor. Available from: http://meetings.aps.org/Meeting/MAR20/Session/G68.1
7. IBM. [Internet] [updated 2020]. Available from: https://quantum-computing.ibm.com/docs/circ-comp/q-gates
8. Team, T. [Internet] (2020, July 29). Grover's Algorithm. Retrieved July 29, 2020, from https://qiskit.org/textbook/ch-algorithms/grover.html
9. Giles, M. [Internet]. [updated 2019 Sep 18]. MIT Technology Review; [cited 2020 Jul 29]. Available from: https://www.technologyreview.com/2019/09/18/132 956/ibms-new-53-qubit-quantum-computer-is-the-most-powerful-machine-you-can-use/
10. The Jupyter Book Community [Internet]. IBM Qiskit. Available from: https://qiskit.org/textbook/ch-applications/vqe-molecules.html
11. Mohseni, M., Read, P., Neven, H., Boixo, S., Denchev, V., Babbush, R., Fowler, A., Smelyanskiy, V., and Martinis, J. [Internet]. [updated 2017 March 03]. Commercialize quantum technologies in five years. Nature. Available from: http://www.nature.com/news/commercialize-quantum-technologies-in-five-years-1.21583
12. Piela, L. Two Fundamental Approximate Methods. Ideas of Quantum Chemistry. 2014
13. Ichimura A. S. [Internet]. [updated 2004]. California Institute of Technology. Available from: http://www.wag.caltech.edu/PASI/lectures/SFSU-ElectronicStructure-Lect-2.ppt
14. Thanthiriwatte K. S. [Internet]. [updated 2010]. Georgia Institute of Technology. Available from: http://vergil.chemistry.gatech.edu/notes/sahan-cc-2010.pdf
15. Steudner M. and Wehner S. Fermion-to-qubit mappings with varying resource requirements for quantum simulation. New Journal of Physics. 2018
16. Abraham H., Offei A., Akhalwaya I. Y., Aleksandrowicz G., Alexander T., Arbel E., Asfaw A., Asaustre C., Aziz Ngoueya, Barkoutsos P., and others [Internet]. Qiskit: An Open-source Framework for Quantum Computing. 2019. Available from:
17. Kandala, A., Mezzacapo, A., Temme, K., Takita, M., Brink, M., Chow, J. M., and Gambetta, J. M. Hardware-efficient Variational Quantum Eigensolver for Small Molecules and Quantum Magnets. Nature. 2017

APPENDICES

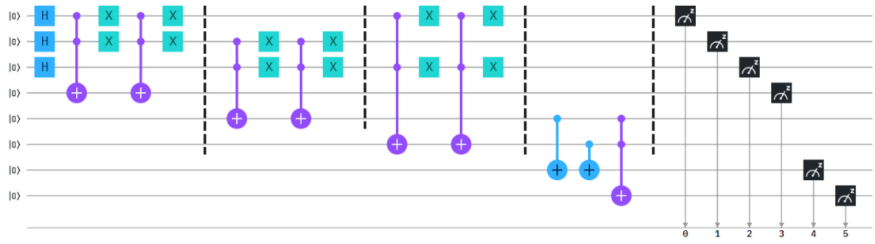
Appendix A

Classical Algorithm (Python)

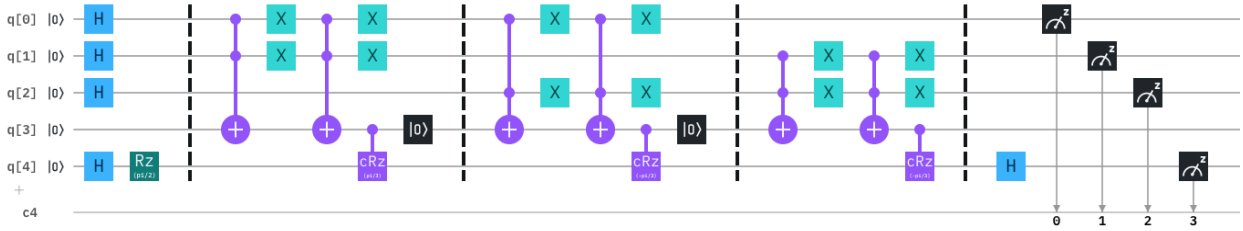
```python
scores = []

all_states = [[1,1,1], [1,1,0], [1,0,1], [1,0,0], [0,1,1], [0,1,0], [0,0,1], [0,0,0]]

for i in range(0,8):
    current_config = all_states[i]

    alice = current_config[0]
    bob = current_config[1]
    chris = current_config[2]

    current_score = 0

    if (alice == 0 and bob == 0) or (alice == 1 and bob == 1):
        current_score += 1
    if (alice == 0 and chris == 0) or (alice == 1 and chris == 1):
        current_score -= 1
    if (bob == 0 and chris == 0) or (bob == 1 and chris == 1):
        current_score -= 1

    insert = [current_config, current_score]
    scores.append(insert)

highest_score = scores[0][1]
best = []

for i in range(1, 8):
    if scores[i][1] >= highest_score:
        highest_score = scores[i][1]
        best.append(scores[i])

print('')
print('Best Option(s):')
print(best)

for i in range(0,8):
    print('Config:')
    print(scores[i][0])
    print('Score:')
    print(scores[i][1])
    print()

```

Quantum Algorithms

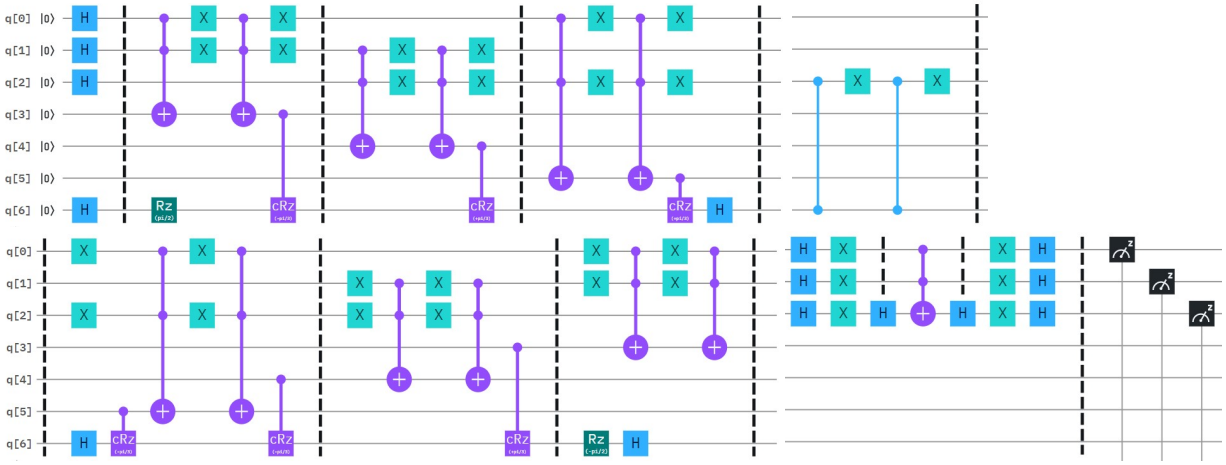
Bit Flip Circuit:



Phase Shift Circuit:



Grover's Algorithm Circuit:



Appendix B

Comparison between HF and VQE

When the ground state energy for an H_2 molecule is calculated at small distances, Hartree-Fock appears to serve as an adequate approximation (Fig 1). However, at greater distances, VQE provides a much more accurate estimate for this expectation value in comparison compared to Hartree-Fock (Fig 2). Only on the scale of 10^{-8} Hartrees does significant error begin to appear when using VQE (Fig 3).

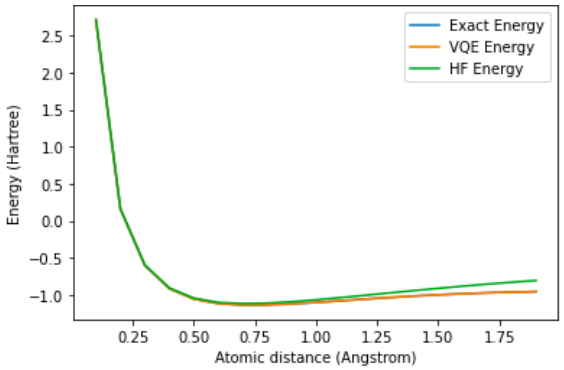


Fig 1 Calculated Energy vs. Atomic Distance

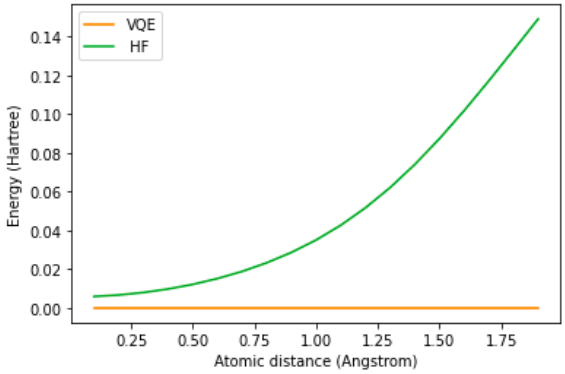


Fig 2 Difference between exact energy and calculated energy for HF and VQE

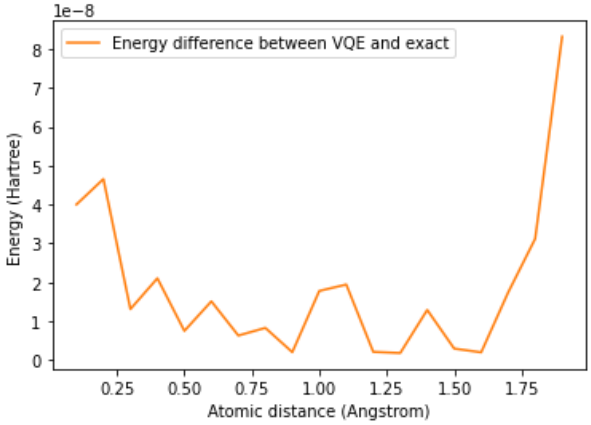


Fig 3 Energy difference between exact energy and VQE

Noise and Decoherence

All of the above graphs have been produced by ideal simulators of quantum computers. In reality, quantum computation suffers from the randomness of noise and decoherence. Whereas the ideal simulator is accurate to the scale of 10^{-8} Hartrees, the same computation performed on a noisy simulator already differs from the exact value on the much more significant scale of 10^{-3} Hartrees (Fig 4, Fig 5). A similar degree of diversion from the exact energy is visible when

calculated on an actual quantum machine (Fig 6). Thus, the physical limitations of quantum computers still present a significant limiting factor to the accuracy of VQE.

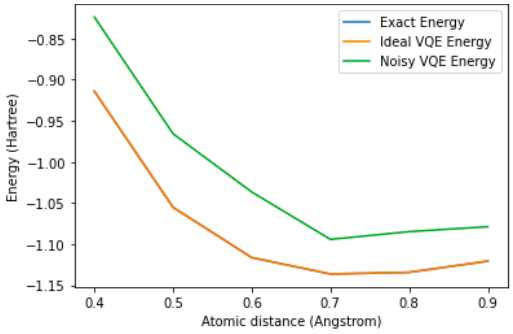


Fig 4 Comparing accuracy of calculation of dissociation curve of H_2 on ideal with noisy quantum computers

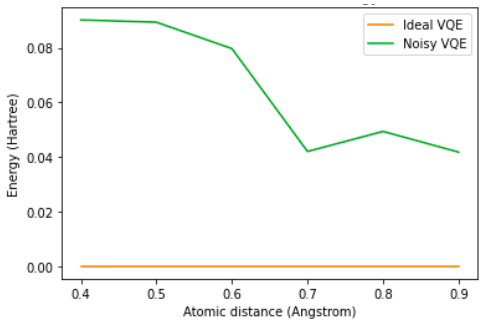


Fig 5 Difference between exact energy and calculated energy for ideal and noisy computers

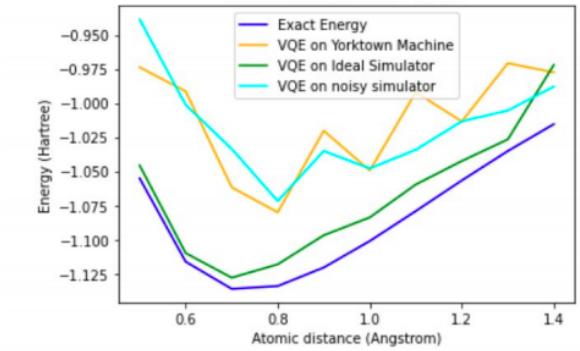


Fig 6 Comparing calculated dissociation curves of H_2 on various machines and simulators

Sample Code for Applying VQE to H_2O

```
#Initialize molecular driver
driver = PySCFDriver(atom='O 0.0 0.0 0.0; H 0.757 0.586 0.0; H -0.757 0.586 0.0',
                     unit=UnitsType.ANGSTROM, charge=0, spin=0, basis='sto3g')
molecule = driver.run()

#Print calculated values
print('Hartree-Fock energy: {}'.format(molecule.hf_energy))
print('Nuclear repulsion energy: {}'.format(molecule.nuclear_repulsion_energy))

#Initialize hamiltonian operator
core = Hamiltonian(transformation=TransformationType.FULL,
qubit_mapping=QubitMappingType.PARITY,
                   two_qubit_reduction=True, freeze_core=True)
qubit_op, aux_ops = core.run(molecule)
```

```
distances = [x * 0.01 + 1.00 for x in range(17)]
 vqe = VQE(qubit_op, var_form, optimizer)
 algo_result = vqe.run(quantum_instance)

energies = np.empty(len(distances))

#Loop over distances and calculate energy for each
for i, distance in enumerate(distances):
    driver = PySCFDriver(h2o.format(distance), basis='sto3g')
    qmolecule = driver.run()
    operator =  Hamiltonian(freeze_core=True)
    qubit_op, aux_ops = operator.run(qmolecule)
    result = NumPyMinimumEigensolver(qubit_op).run()
    result = operator.process_algorithm_result(result)
    energies[i] = result.energy
```