UVa

FacultaddeCiencias

ESDUVa
Escuela de Doctorado Universidad de Valladolid

Trabajo Fin de Máster

Máster en Física

Simulating Molecules with Variational Quantum Eigensolvers

Autora: Nonia Vaquero Sabater

Tutor: Juan Carlos García Escartín

Abstract

During this work, we have reviewed the theory on which variational quantum eigensolvers are based. Applying this type of algorithms, we have seen how to obtain the ground state of a molecule with a quantum computer. Using two programs, one we have developed and other provided by *Qiskit*, we have been able to study some of the properties of these algorithms.
Starting with the H_2 molecule, we have tested two types of ansätze based on quantum circuits. We have verified that for each program one of the ansätze gave a more accurate result. We have also tried one of the chemistry-inspired ansätze, the $UCCSD$, which has been proven to be very sensitive to noise. Afterwards, a simulation has been executed for each program and each type of ansatz but trying different mappings. We have come to the conclusion that the only one that generated good results is the Bravyi-Kitaev mapping with one of the ansatz here proposed.
Finally, we have focused on the study of bigger molecules using *Qiskit* packages to simulate the LiH molecule. We have found that fermionic based ansätze give better results than circuit based ansätze.

Abstract

En este trabajo se ha revisado la teoría en la que se basan los algoritmos cuánticos variacionales aplicados a la búsqueda de autoestados de un Hamiltoniano. Aplicando estos algoritmos, hemos visto cómo calcular el estado fundamental de una molécula usando un ordenador cuántico. Para ello, hemos utilizando un programa creado por nosotros y uno predeterminado proporcionado por *Qiskit*, que nos han permitido estudiar algunas de las propiedades de estos algoritmos.
Empezando con una molécula de H_2, hemos probado dos tipos de ansätze basados en circuitos cuánticos, comprobando que para cada programa, había uno más preciso que el otro. También hemos probado uno de los ansätze basados en la química, el $UCCSD$, que ha demostrado ser muy sensible al ruido. Después, se han ejecutado, para cada programa y cada tipo de ansatz, simulaciones para cada tipo de mapeado presentado en el trabajo. Hemos llegado a la conclusión de que solamente el mapeado de Bravyi-Kitaev con uno de los ansätze que estábamos probando nos proporciona buenos resultados.
Finalmente, nos hemos centrado en el estudio de moléculas de mayor tamaño, utilizando los paquetes de *Qiskit* para simular una molécula de LiH. Hemos encontrado que los ansätze basados en sistemas fermiónicos dan mejores resultados que aquellos basados en circuitos cuánticos.

Contents

Resumen

Uno de los campos en los que se espera obtener supremacía cuántica es en la química computacional. Simular grandes sistemas ha supuesto siempre un reto para la computación clásica ya que la cantidad de recursos necesitados escala exponencialmente con el tamaño del sistema. Aprovechando las propiedades de los ordenadores cuánticos, el objetivo es ser capaces de simular algún día grandes sistemas de manera eficaz.
Una de las maneras que se están estudiando actualmente para conseguir esto son los *Variational Quantum Eigensolvers algorithms*. Estos algoritmos son capaces de hallar el estado y energía fundamentales de moléculas. Su funcionamiento se basa en proponer un estado inicial de prueba (ansatz) expresado en función de ciertos parámetros y mediante un optimizador clásico hallar el valor de los parámetros que minimiza el valor de la energía. Para ello es necesario proponer una expresión del Hamiltoniano que sea ejecutable en un ordenador cuántico. La energía será el valor esperado de este Hamiltoniano. Vemos que la principal característica de estos algoritmos es la combinación de la parte cuántica, con el cálculo del valor esperado del Hamiltoniano, con la parte clásica, con la optimización del estado.

El principal objetivo de este trabajo era crear un programa que basándose en el ansatz propuesto en [1], fuera capaz de simular una mólecula de H_2, hallando la energía para distintas distancias entre átomos. Hallando así la distancia para la energía mínima que corresponde con el estado fundamental. Esto se consiguió y fuimos capaces de ejecutar el programa tanto en ordenadores clásicos con paquetes que simulan el comportamiento de un ordenador cuántico como directamente en ordenadores cuánticos. Esto nos ha permitido estudiar diferentes características de estos algortimos, como la infuencia de la elección del ansatz o del tipo de mapeado sobre la solución. La ejecución en ordenadores cuánticos se realizó gracias a *IBM Quantum*, una plataforma que permite lanzar programas tanto a simuladores de ordenadores cuánticos como a los ordenadores cuánticos reales que IBM tiene disponibles. Todos los programas han sido ejecutados en *Qiskit*, un software abierto creado para trabajar con ordenadores cuánticos y compatible con *IBM Quantum*.
Las primeras simulaciones lanzadas fueron las del programa que habíamos creado para distintos ansätze, siendo uno de ellos el propuesto por el paper citado anteriormente. Se lanzaron simulaciones suponiedo un comportamiento ideal del ordenador y simulaciones suponiendo ruido. Se encontró que el programa funcionaba adecuadamente con una precisión cercana a la química. Los mejores resultados hallados fueron los calculados con el ansatz del paper. Al ejecutar estos programas en el ordenador cuántico, se encontraron los mismos resultados, con la única diferencia de que esta vez los datos obtenidos con los dos tipos de ansätze eran bastante menos precisos.

También se aprovecharon los paquetes de *Qiskit* que proporcionan algoritmos cuánticos variacionales ya programados. De la misma manera, los ejecutamos en simulaciones no ruidosas, ruidosas y en ordenadores cuánticos. Se encontró que la precisión de el programa que nosotros hicimos es similar a la precisión del programa de *Qiskit* al realizar una simulación ruidosa. Sin embargo, al ejecutar estos programas en un ordenador cuánticos, se hallaron muy buenos resultados con uno de los ansätze utilizados, llamado ansatz universal. La precisión de este entra dentro de la precisión química, cosa que no pasaba con nuestro programa.

Como antes se mencionó, se necesita hacer un mapeado para pasar del Hamiltoniano cuantizado a qubits. Esto se puede hacer de diferentes maneras, hasta ahora hemos utilizado una forma en concreto de codificar la informacón en los qubits, llamado parity mapper. Hemos realizado todas estas simulaciones para los otros dos tipos que hemos estudiado, obteniendo que solo uno

de ellos daba buenos resultado y además solamente con un tipo de ansatz. Solo hemos mostrado estos resultados ya que eran los únicos coherentes. En este caso hemos comparado para el mismo ansatz, el funcionamiento de nuestro programa y del de *Qiskit*. Hemos visto que aunque nuestro programa no da malos resultados para la simulación sin ruido, aunque no lo suficientemente precisos, mientras que el programa de *Qiskit* alcanza la precisión química en este caso. Sin embargo, a la hora de ejecutarlos en un ordenador cuántico, la precisión de ambos no es muy buena.

Hoy en día, se han podido simular con eficiencia moléculas de mayor tamaño, estas son LiH, BeH_2 y H_6. Hemos utilizado los paquetes de *Qiskit* para comprobar la eficiencia de dos tipos de ansatz en el caso de simulación no ruidosa, comparando su precisión y tiempo de ejecución. Hemos concluido que en el caso de moléculas grandes los ansätze basados en circuitos cuánticos tienen dificulatdes para hallar el estado mínimo de energía, siendo los circuitos fermiónicos una mejor opción. Hemos comparado también cómo afecta la elección del optimizador clásico al programa cuando se está trabajando con ansätze basados en circuitos cuánticos, viendo que los de gradient descent no presentaban una gran ventaja y utilizando finalmente un optimizador clásico de aproximación lineal.

1 Introduction

It was in the early 1980s when the idea of quantum computing was heard for the first time. Yuri Manin was the first to mention this in the introduction of his book *Computable and Uncomputable*[2]. At the same time, Paul Benioff constructed a microscopic quantum mechanical Hamiltonian as a model of Turing machines [3]. Shortly after, in 1981, a conference on the Physics of Computation was organised by the MIT and IBM. It gathered several scientists that had been working in quantum computing. The talks that were given during this meeting were published the following year in the *International Journal of Theoretical Physics*. It is here where the famous Feynman conference [4], which has been credited for the birth of the quantum computation and simulation, can be found. From that time on we have witnessed the creation of the first quantum computers, together with the development of several quantum algorithms.
The main idea is to use a controllable quantum system to study another less controllable or accessible quantum system since simulating quantum mechanics with a classical computer is a really difficult computational problem. The applications of quantum computing can be found in different fields, such as physics, chemistry or biology. In this work we are going to focus on quantum computational chemistry.
Classical simulations of chemical systems have greatly helped us to understand the structural, physical, and chemical characteristics of molecules, but the systems soon become intractable on a conventional computer. When simulating molecules, the resources needed scale exponentially with the number of atoms, therefore an exact solution requires too many computational resources. Since quantum computers are able to simulate quantum systems, they are capable of solving this problem in a much more efficient and accurate way.
Solving the Schrödinger equation on a quantum computer is a well-studied and highly important challenge. Different algorithms have been created to tackle this problem, the one we are going to use is the Variational Quantum Eigensolver (VQE) [5], which allows us to calculate the eigenstates and eigenvalues of the Hamiltonian using an acceptable number of qubits for small molecules (this is an important aspect to take into account since the leading quantum processors contain around a few hundred qubits, and the errors in the results increase with the number of gates used during the simulation).

The first two sections, (2, 3), contain a brief introduction to quantum mechanics and quantum computing. Followed by an explanation, step by step, of how the VQE algorithm works (4). We are going to apply two different VQE algorithms to a H_2 molecule and a LiH molecule, obtaining their ground state for different configurations. We will analyse how changes in the type of ansatz, classical optimizer and type of mapping affect the results. The experiments will be carried out in a noisless simulator, a noisy simmulator, and finally, in a quantum computer. The results will be presented in section (5).

2 Introduction to Quantum Mechanics

The basic notions of quantum mechanics needed to understand the fundamentals of quantum computing presented in this work can be found in books such as Nielsen and Chuang [6] or Cohen-Tannoudji [7]. This section begins with an explanation of the notation that is going to be used, followed by a brief mention of the postulates of Quantum Mechanics and ending with the spectral theorem.

In quantum mechanics, the state of a system is given by a wave function. Since the probabilistic interpretation requires it to be square-integrable, the set of these functions have the structure of a Hilbert space (ε). A Hilbert space meets the conditions of a vector space. We are going to call each element of this space a ket and it is going to be represented by $|\psi\rangle$, where ψ is the wave function. In conclusion, each quantum state of a particle will be characterised by a state vector, belonging to a vector space ε, called the state space of a particle. The elements of the dual space of ε, symbolized by ε^*, are called bras and are denoted by $\langle\psi|$.

When dealing with a finite dimension n, kets and bras can be represented as matrices in an orthonormal basis of the space ε. The chosen basis must satisfy the following conditions:

- Orthonormalization relation. A set of kets $|\mathbf{u}_i\rangle$ is said to be orthonormal if they satisfy the relation
$$\langle\mathbf{u}_i|\mathbf{u}_j\rangle = \delta_{ij} \qquad i,j \in \{1,...,n\}\,, \tag{1}$$
where $\langle\phi|\psi\rangle$ denotes the scalar product between kets $|\phi\rangle$ and $|\psi\rangle$.
- Closure relation. The basis elements $|\mathbf{u}_i\rangle$ must follow the relation
$$\sum_{i=1}^{n} |\mathbf{u}_i\rangle\langle\mathbf{u}_i| = \mathbb{1} \qquad i,j \in \{1,...,n\}\,. \tag{2}$$

In the $|\mathbf{u}_i\rangle$ basis the ket $|\psi\rangle$ is represented by the set of its components ($\langle\mathbf{u}_i|\psi\rangle$), which can be arranged forming a one-column matrix as follows

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix} = \sum_{i=1}^{n} \psi_i |\mathbf{u}_i\rangle\,. \tag{3}$$

On the other hand, bras are represented by one-row matrices.

The last thing we need to define are linear operators. A linear operator A associates with every ket $|\psi\rangle \in \varepsilon$ another ket $|\psi'\rangle \in \varepsilon$, the correspondence being linear ($|\psi'\rangle = A|\psi\rangle$). In the $|\mathbf{u}_i\rangle$ basis they are represented by matrices whose structure is

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1j} & \dots \\ A_{21} & A_{22} & \dots & A_{2j} & \dots \\ \vdots & \vdots & & \vdots & \\ A_{i1} & A_{i2} & \dots & A_{ij} & \dots \\ \vdots & & \vdots & & \vdots \end{pmatrix}, \tag{4}$$

where $A_{ij} = \langle\mathbf{u}_i|A|\mathbf{u}_j\rangle$. Kets and bras are normalized, so in order to preserve the norm, all the matrices are unitary.

2.1 The Postulates of Quantum Mechanics

Once the notation is known we can briefly enunciate the postulates of quantum mechanics [7]. We are going to consider a finite discrete spectrum.

First Postulate: At a fixed time t_0, the state of a physical system is defined by specifying a ket $|\psi(t_0)\rangle$ belonging to the state space ε.

Second Postulate: Every measurable physical quantity $\mathcal{A}$ is described by an operator A acting in ε, this operator is an observable.

Third Postulate: The only possible result of the measurement of a physical quantity $\mathcal{A}$ is one of the eigenvalues of the corresponding observable A.

Fourth Postulate (case of a discrete spectrum): When the physical quantity $\mathcal{A}$ is measured on a system in the normalized state $|\psi\rangle$, the probability $\mathcal{P}(a_n)$ of obtaining the eigenvalue a_n of the corresponding observable A is

$$\mathcal{P}(a_n) = \sum_{i=1}^{g_n} |\langle \mathbf{u}_n^i | \psi \rangle|^2$$

where g_n is the degree of degeneracy of a_n and $\{u_n^i\}$ $(i = 1, 2, ..., g_n)$ is an orthonormal set of vectors which forms a basis in the eigensubspace ε_n associated with the eigenvalue a_n of A.

Fifth Postulate: If the measurement of the physical quantity $\mathcal{A}$ on the system in the state $|\psi\rangle$ gives the result a_n, the state of the system immediately after the measurement is the normalized projection, $\frac{P_n|\psi\rangle}{\sqrt{\langle\psi|P_n|\psi\rangle}}$, of $|\psi\rangle$ onto the eigensubspace associated with a_n. We can define the projector operator $P_n = |\mathbf{u}_n\rangle\langle\mathbf{u}_n|$ when referring to a discrete non-degenerate case. In the general case of a discrete spectrum the projector operator is defined by $P_n = \sum_{i=1}^{n} |\mathbf{u}_n^i\rangle\langle\mathbf{u}_n^i|$.

Sixth Postulate: The time evolution of the state vector $|\psi(t)\rangle$ is governed by the Schrödinger equation

$$i\hbar \frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle \,,$$

where H is the Hamiltonian operator of the system.

Notice that an observable is by definition an Hermitian operator. These operators are their own hermitian conjugate, which implies that their eigenvalues are always going to be real numbers. This last result makes sense considering that a physical variable must have real expectation values.

One last concept we should keep in mind is the definition of mean value. The mean value of the observable A in the state $|\psi\rangle$, which is denoted by $< A >_\psi$, is defined as the average of the results obtained when a large number N of measurements are performed on the state $|\psi\rangle$. If $|\psi\rangle$ is normalized, $< A >_\psi$ is given by the expression

$$\langle A \rangle_\psi = \langle \psi | A | \psi \rangle = \sum_{n=1}^{N} a_n \mathcal{P}(a_n) \,. \tag{5}$$

The last equality is true when $N \to \infty$, being N the number of experiments performed. $\mathcal{P}(a_n)$ represents the probability of getting the eigenvalue a_n when measuring the observable A.

2.2 The spectral decomposition

In this work we will only need to consider Hermitian matrices.

> *Spectral theorem for Hermitian matrices.* For a Hermitian matrix all eigenvalues are real, the eigenvectors corresponding to distinct eigenvalues are orthogonal and there exists an orthogonal basis of the whole space, consisting of eigenvectors.

This means that all Hermitian matrices are diagonalizable. Moreover, for every Hermitian matrix A, there exists a unitary matrix U such that

$$AU = U\Lambda\,,$$

where Λ is a real diagonal matrix. The diagonal entries of Λ are the eigenvalues of A, and the columns of U are eigenvectors of A. A Hermitian matrix $A \in \mathbb{C}^{n\times n}$ has n orthonormal eigenvectors.
This result makes it possible to express any Hermitian matrix as the sum of all the eigenvalues multiplied by the projector operator of each eigenvector. If $|\varphi_i\rangle$ is an eigenvector of A and a_i it's eigenvalue, we have

$$A = \sum_{i=1}^{n} a_i |\varphi_i\rangle\langle\varphi_i|\,, \tag{6}$$

being n the number of eigenvectors which corresponds to the dimension.

3 Introduction to Quantum Computing

The aim of this section is to introduce the basic concepts needed to understand how does the algorithm work. In order to do so, a brief definition of qubit, quantum state and quantum gate are going to be given. All these concepts can be found fully explained in [6].

3.1 What is a qubit?

In the same way that bits are the fundamental concept of classical computation, we have qubits in quantum computation. The major difference between them is that while a bit has only two possible states, 0 or 1, a qubit has $|0\rangle$, $|1\rangle$ and all their linear combinations. These states are called superpositions and can be written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \, , \tag{7}$$

where α and β are complex numbers that must obey $|\alpha|^2 + |\beta|^2 = 1$. When a qubit is measured, the result is either 0, with probability $|\alpha|^2$, or 1, with probability $|\beta|^2$. The states $|0\rangle$ and $|1\rangle$ are known as computational basis states and they form an orthonormal basis. Qubits can be represented by one-column matrices, where its elements are the coefficients of the state vectors of the basis, in this case, equation (7) is expressed in the computational basis, so its expression would be

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \, . \tag{8}$$

In order to fully understand the concept of qubit we can visualize it in 3 dimensions in what is called the Bloch sphere. This is done by rewriting equation (7) applying that α and β are complex numbers, so we have

$$|\psi\rangle = e^{i\gamma} \left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi} \sin\frac{\theta}{2}|1\rangle \right) \, . \tag{9}$$

Since the factor $e^{i\gamma}$ does not affect the general result, it can be removed from the equation, obtaining

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi} \sin\frac{\theta}{2}|1\rangle \, . \tag{10}$$

θ and φ can be understood as the coordinates of a point on the unit three-dimensional sphere, known as Bloch sphere, see figure (1). This is where the famous image of a qubit comes from

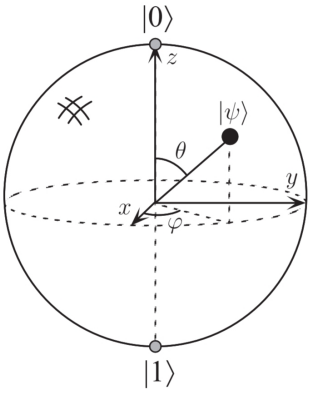

Figure 1: Representation of a qubit in the Bloch sphere, obtained from [6].

3.2 Quantum States

Let's suppose that we have more than one particle ($|\varphi_1\rangle \in \epsilon_1$ and $|\psi_2\rangle \in \epsilon_2$) and we want to describe their state. The state describing both of them will belong to the vector space ε, produced by the tensor product of ε_1 and ε_2,

$$\varepsilon = \varepsilon_1 \otimes \varepsilon_2 \,. \tag{11}$$

The vector belonging to ε describing this state of two particles is denoted by

$$|\varphi_1\rangle \otimes |\psi_2\rangle \,. \tag{12}$$

This is an example of a separable state. It is defined as a state belonging to ε that can be expressed as the tensor product of the individual states. Not all states belonging to ε can be written as a product state, these are called entangled states.
The most general state that can be written in this space ε is

$$|\phi\rangle = \sum_{i,j} \alpha_{ij} |\varphi_1\rangle \otimes |\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \,, \tag{13}$$

where $|00\rangle = |0\rangle \otimes |0\rangle$, $|01\rangle = |0\rangle \otimes |1\rangle$, etc. Now, applying the same argument as in the previous section, we have that the measurement result $x = (00, 01, 10 \,\text{or}\, 11)$ occurs with probability $|\alpha_x|^2$, with the state of the qubits after the measurement being $|x\rangle$. It is known that the sum of all the probabilities must be 1, this means $\sum_x |\alpha_x|^2 = 1$.
An important example of an entangled state is the Bell state

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} \,.$$

From equation (13) we can see that with n qubits, the basis of the Hilbert space of the n qubits is described by 2^n vector states. Our quantum computer will be able to operate with all the linear combinations of these 2^n states. This is the reason why quantum computing is so powerful, the exponential growth of the size of the Hilbert space with the number of particles of the system

is matched by the exponential growth of quantum computer's Hilbert space. A full quantum simulation would demand exponential resources on a classical computer. Notice however, we do not have access to the whole system, only to some global properties, such as expected values. This is enough in certain applications.

3.3 Quantum Gates

Just as classical computers circuits operate with logic gates, quantum computers use quantum logic gates. Quantum gates are linear operators, they associate with every ket $|\psi\rangle$, another ket $|\psi'\rangle$ ($|\psi'\rangle = A|\psi\rangle$). While qubits were represented as one-column matrices in the computational basis, quantum gates are described by unitary matrices ($U^\dagger U = I$). This idea is easier to comprehend by looking at a simple example, the NOT gate, and defining an analogous one for a qubit. In classical computing, the action of the NOT gate is defined by $0 \to 1$ and $1 \to 0$, so we need a quantum gate that takes the state $|0\rangle$ to $|1\rangle$ and vice versa. It can be proven that the following matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{14}$$

satisfies this requirement. If the initial state is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the action of this quantum gate will give as a result $|\psi'\rangle$, defined as

$$|\psi'\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} . \tag{15}$$

In other words, $|\psi'\rangle = \alpha|1\rangle + \beta|0\rangle$. We have succesfully changed $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$.
The previously used matrix is the Pauli matrix σ_x. Together with the other two Pauli matrices, σ_y and σ_z, they constitute the Pauli gates (X, Y, Z), which are of great value. Their structure is

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} , \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} , \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \tag{16}$$

Apart from these, there are two other gates that are going to be repeatedly used, the Hadamard gate (H) and the $S^\dagger-$gate ($S^\dagger$), represented by the following matrices in the computational basis

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{and} \qquad S^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} . \tag{17}$$

4 What are Variational Quantum Eigensolvers?

As mentioned in the introduction, computational chemistry is one of the fields that is expected to benefit the most from the creation of quantum computers. Due to the exponential growth in the size of the Hilbert space with increasing orbitals, a quantum computer with tens of qubits could potentially surpass classical algorithms. Different quantum algorithms have been developed in order to simulate molecules and at the same time reducing as far as possible the resources needed. Even though the runtime and resources they require are expected to grow polynomially with the size of the problem, we are still not able to reproduce big systems in an efficient way. This is due to the limitations on the hardware. On the one hand the biggest quantum computers that have been built have around one hundred qubits and, on the other hand, the more gates your algorithm has, the more noise you get in the results.

The first algorithm proposed to simulate molecules was the phase-estimation algorithm (PEA) [1], [8]. Since it can obtain eigenvalues of hermitian operators, it can be used to calculate the eigenvalues of a Hamiltonian. The major drawback when applying this method is the number of gates needed. This algorithm requires too deep circuits. All this makes PEA not workable in the current quantum computers. Variational quantum eigensolvers (VQE) are a good alternative proposed years later since they limit the circuit depth. This is why they can be implemented in real small scale quantum computers.
In this section the mechanism of variational quantum eigensolvers is going to be explained. First we give an overview of the general method and then we will discuss each part in detail.

4.1 Hamiltonian Ground State

Our main goal when simulating a molecule is to find its ground state, this is its minimum energy and the structure it has when this energy is reached. For example, in the case of H_2, when talking about the structure of the molecule, we are just referring to the distance between the two atoms. In order to do so, the Hamiltonian of the molecule is needed. The Hamiltonian is a Hermitian operator so according to the spectral decomposition theorem it can be expressed as the sum of all the eigenvalues multiplied by the projector operator of each eigenvector. These eigenvalues can be ordered from greater lo smaller, being the lowest of them the ground state energy. Any measurement of the energy in a certain state is an upper bound of the ground state energy.

The Hamiltonians that we are going to be able to reproduce and simulate in a quantum computer are those that can be written in terms of Pauli gates (a not very large number of Pauli gates).
Once we have the Hamiltonian expressed as a sum of several terms consisting of Pauli gates, measuring its mean value gives us the expectation value of the Hamiltonian. In order to find the ground state energy, the Hamiltonian has to be measured in the eigenstate that gives us the minimum eigenvalue.
The problem will then include obtaining the Hamiltonian written in terms of Pauli gates, finding the ground state and measuring the Hamiltonian expectation value in this state.

4.2 Introduction to VQE algorithms

Variational quantum algorithms are one of the primes candidates to obtain quantum advantage [9]. They combine classical and quantum computation in order to deal with different challenges

such as simulating quantum systems or solving large-scale linear algebra problems. Variational algorithms follow the next steps, first a function we want to solve because it describes a parameter of interest of a problem is defined . Then, an ansatz is proposed. This is a quantum operation depending on a set of parameters. These parameters will change and will produce different trial states until a minimum is reached in the energy of the Hamiltonian. This is how the ansatz is trained in a hybrid quantum-classical loop until the function is optimized.

Variational quantum algorithms applied to the search of the ground state of a given Hamiltonian are called variational quantum eigensolvers (VQE). They were first proposed as an alternative to PEA in 2014 by Peruzzo et al. [5] and then extended by McClean et al. [10]. The main idea is to construct a circuit that measures the energy of the Hamiltonian in a certain state, which will be given by the ansatz. As mentioned before, an ansatz is a parametrized circuit, in this case describing the wavefunctions of the system. Through a classical optimization its parameters are updated until a minimum in the value of the energy is reached. This mechanism can be visualized in the next image

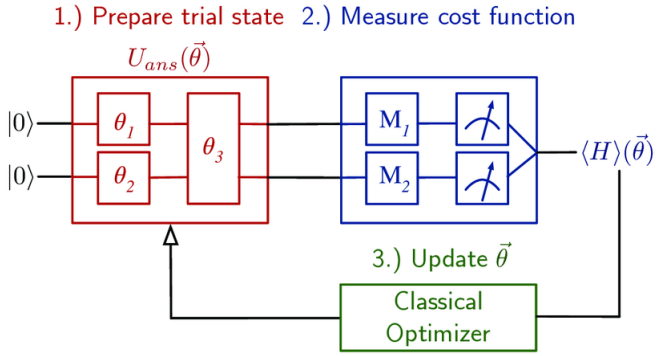

Figure 2: VQE mechanism [11].

Variational classical algorithms work in a similar way. First, a function that is believed to be similar to the ground state is proposed and then its parameters are changed until a minimum is reached. The advantage that quantum computing gives us is the ability to prepare the states more easily, we have access directly to quantum states.
Once the general mechanism of the VQE has been explained, we can focus on the small steps that need to be taken in order to develop the final algorithm. The main parts of the algorithm are

1. Find a qubit expression of the Hamiltonian of the system. We need a general method to translate the physical problem to the quantum computer. This means, as mentioned before, finding an expression of the Hamiltonian that can be simulated in the quantum computer. Starting with the molecular Hamiltonian, the next step is to apply the Born-Oppenheimer approximation so that its expression is simplified. Once the simplified, but still classical equation of the Hamiltonian is obtained, the next thing to do is to quantizate it. We have applied the second quantization. After that, the fermion to qubit mapping is carried out, this is the part where we take the physical problem and find a way to reproduce it in the quantum computer. This can be done in different ways, the most common ones are the Parity mapping, the Jordan-Wiegner mapping and the Bravyi-Kitaev mapping. When all

this steps have been completed, the result is the Hamiltonian of the system expressed in terms of Pauli gates. Finally, reduction methods are usually applied in order to minimize the number of qubits needed to solve the problem.

2. Prepare the ansatz. The next step is to create the ansatz, a parametrized quantum circuit in which the Hamiltonian will be measured. Imagine that the proposed ansatz is parametrized with $\vec{\theta}$, $|\varphi(\vec{\theta})\rangle$, then the variational principle holds that

$$\frac{\langle\varphi(\vec{\theta})|H|\varphi(\vec{\theta})\rangle}{\langle\varphi(\vec{\theta})|\varphi(\vec{\theta})\rangle} \geq E_0 \,, \tag{18}$$

where E_0 represents the ground state energy, which corresponds to the smallest eigenvalue of the Hamiltonian. The idea is to get as close as possible to E_0 by varying the value of $\vec{\theta}$. Through a classical optimization the parameters of the ansatz are varied until a minimum on the expectation value of the Hamiltonian is reached.
There exist different procedures for choosing the ansatz, the two main ones are the hardware-efficient ansatz and the chemistry-inspired ansatz, but both of them are too generic. Neither of them knows anything about the problem they are solving. The properties a good ansatz should have are; being as shallow as possible, not having too many optimization parameters and being able to span the space where the solutions are. Proposing an appropriate ansatz is critical to obtain a correct solution of the problem and it is not a simple task since you must find the equilibrium between spanning an adequate part of the Hilbert space so that it contains the solution and having a tractable amount of parameters.

3. Parameter optimization. Here can be found the classical part of the algorithm, with each iteration the parameters of the ansatz change and a new expectation value of the Hamiltonian is calculated. It must reach a value sufficiently close to the solution but at the same time accomplish this task in a reasonable number of iterations.

4.3 Minimum energy estimation

As mentioned before, we need to calculate the expectation values of Pauli gates in order to obtain the energy of the system. Suppose that we want to measure its energy in the state $|\psi\rangle$, $\langle\psi|H|\psi\rangle$, and that our Hamiltonian has the expression

$$H = \alpha II + \beta IZ + \gamma ZI + \zeta ZZ + \eta XX \,, \tag{19}$$

where $\alpha, \beta, \gamma, \zeta, \eta$ are coefficients whose value depend on the geometry of the molecule, and IZ denote

$$IZ = I \otimes Z \,. \tag{20}$$

With each simulation, just one term can be measured. Besides, all the individual terms have to be measured a sufficient amount of times to build up enough expectation value statistics. In other words, we would first run the simulation and measure the mean value of IZ an adequately amount of times, then, starting over, we would do the same for the observable ZI and the rest of the terms.
Due to the fact that measures are performed in the computational basis, the only expectation value that we are going to be able to obtain without any transformation needed is the one from the Z Pauli gate. Since the Pauli gates are Hermitian, the spectral theorem (2.2) tells that they can be diagonalized. The procedure is to apply the corresponding transformations in order to

make the X and Y Pauli gates diagonal in the computational basis and make the measurements of these new diagonal matrices. In other words, the idea is to rotate the qubit so that the x or y components can be accessed by measuring in the computational basis, rotating the standard basis frame to make it lie along the corresponding axis and obtain the correct results when measuring in the standard basis.

It can be proven that measuring the expectation value of the X Pauli gate in the basis given by its eigenvalues gives the same result as applying a Hadamard gate and then performing the measurement. This means that if we want to measure for example the expectation vale from XX, we need to build the following circuit:

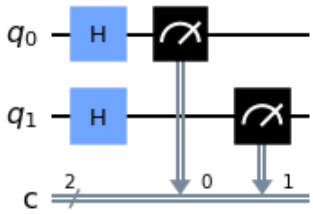

Figure 3: Measurement circuit for $\langle XX \rangle$.

The fact that measuring our state in the basis of the eigenvectors of X is equivalent to applying a Hadamard gate and measuring in the computational basis can be proven in the following way. Let $|+\rangle$ and $|-\rangle$ be the eigenvectors of X (with eigenvalues 1 and -1), with the following form

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,, \\ |-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\,, \end{aligned} \tag{21}$$

and the state that is going to be measured

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle\,. \tag{22}$$

The first procedure that is going to be used is measuring the previous state in the basis formed by the eigenvectorsof X. Applying the spectral decomposition theorem, we have the following expression for the X Pauli gate

$$\langle X \rangle = \langle q|X|q\rangle = \langle q|+\rangle\langle +|q\rangle - \langle q|-\rangle\langle -|q\rangle\,. \tag{23}$$

Knowing that:

$$\langle q|+\rangle = (\alpha^*\langle 0| + \beta^*\langle 1|)|+\rangle = (\alpha^*\langle 0| + \beta^*\langle 1|)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{\alpha^* + \beta^*}{\sqrt{2}} \tag{24}$$

and applying the same reasoning to the other eigenvector

$$\langle q|-\rangle = (\alpha^*\langle 0| + \beta^*\langle 1|)|-\rangle = (\alpha^*\langle 0| + \beta^*\langle 1|)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{\alpha^* - \beta^*}{\sqrt{2}}\,, \tag{25}$$

then we finally have

$$\langle X \rangle = |\langle q|+\rangle|^2 - |\langle q|-\rangle|^2 = \left|\frac{\alpha^* + \beta^*}{\sqrt{2}}\right|^2 - \left|\frac{\alpha^* - \beta^*}{\sqrt{2}}\right|^2 . \tag{26}$$

On the other hand, the other procedure makes us apply a Hadamard gate before measuring in the computational basis. This is

$$|q'\rangle = H|q\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle . \tag{27}$$

Measuring this state in the computational basis we get

$$\langle q'|0\rangle\langle 0|q'\rangle - \langle q'|1\rangle\langle 1|q'\rangle = \left|\frac{\alpha^* + \beta^*}{\sqrt{2}}\right|^2 - \left|\frac{\alpha^* - \beta^*}{\sqrt{2}}\right|^2 , \tag{28}$$

which matches the other result.

Once this circuit has been built, we only have to repeat the measure several times in order to get the expectation value.

The same reasoning applies when building the circuit of the Y Pauli gate. In this case, performing the measures in the basis formed by the eigenvalues of the Y Pauli matrix gives the same result as taking these measures of the gates $S^\dagger H$ in the computational basis. This can be proved in the same way as we did with the X gate. This means that if we want to measure the expectation value of YY, we need to built the following circuit.

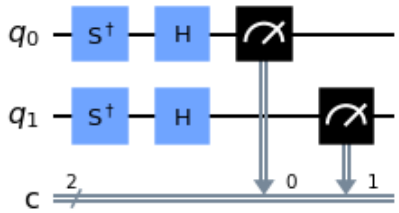

Figure 4: Measurement circuit for $\langle YY \rangle$.

4.4 Preparation of the molecular Hamiltonian

In order to obtain the Hamiltonian written in terms of Pauli gates, we need to apply several transformation to the original molecular one. The molecular Hamiltonian has the following expression [12]

$$H = -\sum_{i=1}^{N}\frac{1}{2}\nabla_{r_i}^2 - \sum_{A=1}^{M}\frac{1}{2M_A}\nabla_{R_A}^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|R_A - r_i|} +$$
$$+\sum_{j>i}\frac{1}{|r_i - r_j|} + \sum_{B>A}\frac{Z_A Z_B}{|R_A - R_B|} = T_e + T_n + V_{ne} + V_{ee} + V_{nn} , \tag{29}$$

where r_i is the position of the electrons, M_i its mass, R_A the nuclear positions, M_A its mass and Z_A the atomic number.

In order to simplify this expression the Born-Oppenheimer approximation is introduced. It is based on the fact that the nucleus is much heavier than the electrons and, consequently its velocity is much smaller. This makes possible to consider static nucleus surrounded by moving electrons. This implies that some of the terms of the previous Hamiltonian can be neglected, resulting in

$$H = -\sum_{i=1}^{N} \frac{1}{2}\nabla_{r_i}^2 - \sum_{i=1}^{N}\sum_{A=1}^{M} \frac{Z_A}{|R_A - r_i|} + \sum_{j>i} \frac{1}{|r_i - r_j|} = T_e + V_{ne} + V_{ee}\,. \tag{30}$$

Now we are just dealing with a problem of interacting electrons.

4.4.1 Second quantization

The next step is the quantization, as mentioned before, we are going to transform the molecular Hamiltonian to the second quantized formulation [12].

It is known that fermions are represented by anti-symmetric states. Slater determinants are used to write these states, for example in the case we have n fermions, we can write

$$|\psi\rangle_n = \frac{1}{\sqrt{n!}} \begin{vmatrix} |\chi_1\rangle_1 & \cdots & |\chi_1\rangle_n \\ \vdots & \vdots & \vdots \\ |\chi_n\rangle_1 & \cdots & |\chi_n\rangle_n \end{vmatrix}. \tag{31}$$

Where $|\chi_1\rangle_1$ represents the first particle being in the first state. In general, $|\chi_j\rangle_j$ denotes a spin-orbital. This can be written in a simpler way,

$$|\psi\rangle_n = |\chi_0 \cdots \chi_{N-1}\rangle\,, \tag{32}$$

where $\chi_j \in \{0, 1\}$ due to the Pauli exclusion principle, with 0 meaning that the orbital is unoccupied and 1 meaning that it is occupied. This is called the occupation number formalism.

The subspace spanned by these anti-symmetric states is known as the Fock space. In the case we have a Fock space of N orbitals, it will be spanned by 2^N states. It is noticeable that the number of states of N qubits is also 2^N, this is how we start to think about mapping this Fock space to the space of N qubits.
In order to do so, we define creation and annihilation operators in the Fock space. Their action is defined by the following equations,

$$a_j^\dagger = |\chi_0 \cdots \chi_{j-1}\, 0\, \chi_{j+1} \cdots \chi_{N-1}\rangle = (-1)^{\sum_{s=0}^{j-1} \chi_s} |\chi_0 \cdots \chi_{j-1}\, 1\, \chi_{j+1} \cdots \chi_{N-1}\rangle\,, \tag{33}$$

$$a_j^\dagger = |\chi_0 \cdots \chi_{j-1}\, 1\, \chi_{j+1} \cdots \chi_{N-1}\rangle = 0\,, \tag{34}$$

$$a_j = |\chi_0 \cdots \chi_{j-1}\, 1\, \chi_{j+1} \cdots \chi_{N-1}\rangle = (-1)^{\sum_{s=0}^{j-1} \chi_s} |\chi_0 \cdots \chi_{j-1}\, 0\, \chi_{j+1} \cdots \chi_{N-1}\rangle\,, \tag{35}$$

$$a_j = |\chi_0 \cdots \chi_{j-1}\, 0\, \chi_{j+1} \cdots \chi_{N-1}\rangle = 0\,, \tag{36}$$

following the anti-commutation relations

$$[a_j, a_i] = 0\,, \qquad [a_j^\dagger, a_i^\dagger] = 0\,, \qquad [a_j, a_i^\dagger] = \delta_{ji}\,. \tag{37}$$

In the second quantization, we enforce the anti-symmetry needed thorough the construction of the operators. This is why we are using annihilation and creation operators.
Any operator in our Fock space can be expressed as a linear combination of products of creation and annihilation operators, so since the Hamiltonian defined in equation (30) is an operator acting in the Fock space, we are going to be able to write it in terms of $a_j^\dagger$ and a_j. This is done in the following way,

$$H = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \,, \tag{38}$$

where h_{pq} and h_{pqrs} are one and two body integrals given by

$$h_{pq} = \langle \chi_p | T_e + V_{ne} | \chi_q \rangle = \int d\vec{x} \chi_p^*(\vec{x}) \left(-\frac{1}{2}\nabla^2 - \sum_A \frac{Z_A}{|R_A - r_i|} \right) \chi_q(\vec{x}) \,, \tag{39}$$

$$h_{pqrs} = \langle \chi_p \chi_q | V_{ee} | \chi_r \chi_s \rangle = \int d\vec{x}_1 d\vec{x}_2 \frac{\chi_p^*(\vec{x}_1)\chi_q^*(\vec{x}_2)\chi_r(\vec{x}_2)\chi_s(\vec{x}_1)}{|r_1 - r_2|} \,, \tag{40}$$

which are easily computed classically.

4.4.2 Orbitals removal

In order to lower the number of qubits needed to simulate the system, another approximation in applied. It is called the frozen core approximation. It consists in just considering the outer electrons and assuming that the rest do not change their state. In other words, it considers a core composed of the nucleus and the inner electrons, this is why it is called frozen core approximation. Once this approximation has been applied, there is one more manipulation that will reduce the number of qubits needed, the removal of some fermionic orbitals. With the lower energy orbitals (the ones which are filled with the inner electrons whose state does not change) removed, we just have the active space (this is the space where the outer electrons move) and the high energy orbitals. Due to the fact that we are looking for the state that gives us the ground energy, the high energy orbitals are not going to be used, so they can also be removed. So the only orbitals that are going to be simulated are the ones whose state may vary, affecting the solution.

4.4.3 Fermion to qubit mapping

Now we have to transform electronic states and operators to states and operations of qubits. Three of the most common techniques are going to be briefly described in this section, the Jordan-Wigner mapping, the parity mapping and the Bravyi-Kitaev mapping [13], [14], [15].

Jordan-Wigner mapping: This mapping is based on the occupation number formalism, the state of each qubit represents if an orbital is occupied or not. This can be visualized in the following way,

$$|\chi_{N-1} \cdots \chi_1 \chi_0\rangle \to |q_{N-1}\rangle \cdots \otimes |q_1\rangle \otimes |q_0\rangle \,, \tag{41}$$

where N is the number of fermionic states and $\chi_j = q_j \in \{0, 1\}$. For example, if the ith orbital is occupied ($\chi_i = 1$), then the ith qubit will be in the state $|1\rangle$ (the same happens if the orbital is unoccupied ($\chi_i = 0$), the state will then be $|0\rangle$). Once the states have been mapped, the only thing left is the action of the creation and annihilation operators. We need a set of operators that have the following effect on the qubits

$$Q^\dagger|0\rangle = |1\rangle \,, \qquad Q^\dagger|1\rangle = 0 \,, \qquad Q|1\rangle = |0\rangle \,, \qquad Q|0\rangle = 0 \,. \tag{42}$$

It can be proven that the following operators fulfill this requirements:

$$Q^+ = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = |1\rangle\langle 0| \,, \qquad Q^- = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = |0\rangle\langle 1| \,. \tag{43}$$

Comparing the action of these operators with the action of creation and annihilation operators in equations (35,33), we can see that the phase shift still remains to be added. This is solved by including strings of Z operators in the previous definitions of $Q^{\pm}$. The effect of the Z Pauli matrix is to multiply by -1 every time the occupation of the fermionic mode is 1 (if it is 0 it multiplies by 1), so an operator which has the same effect as the phase $(-1)^{\sum_{j<i} n_j}$ on the ith qubit is the product of Z operators in all the qubits that precede it. In conclusion, in Jordan-Wigner transformation the mapping from fermion operators to qubits is

$$a_i^\dagger \to \left(\prod_{j<i} Z_j\right) Q_i^+ = \left(\prod_{j<i} Z_j\right) \frac{1}{2}(X_i - iYi) \,, \tag{44}$$

$$a_i \to \left(\prod_{j<i} Z_j\right) Q_i^- = \left(\prod_{j<i} Z_j\right) \frac{1}{2}(X_i + iY_i) \,. \tag{45}$$

The major drawback of this transformation is that the number of Z gates needed scale linearly with the size of the system, it scales as $O(N)$.

The parity mapping: Whereas in the previous transformation the state of each qubit indicated if the orbital was occupied or unoccupied, now the information encoded by each qubit is different. We will have $|0\rangle_j$ if the number of orbitals up to and including the jth that are occupied is even and $|1\rangle_j$ if it is odd. This means that if we have the fermionic state $|\chi_{N-1} \dots \chi_1 \, \chi_0\rangle$, the mapping to qubits will be

$$\begin{pmatrix} q_{N-1} \\ q_{N-2} \\ \vdots \\ q_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} \chi_{N-1} \\ \chi_{N-2} \\ \vdots \\ \chi_0 \end{pmatrix} = \begin{pmatrix} \chi_{N-1} + \chi_{N-2} + \dots + \chi_0 \\ \chi_{N-2} + \dots + \chi_0 \\ \vdots \\ \chi_0 \end{pmatrix} , \tag{46}$$

so that the first qubit informs us of the parity of the total number of particles. Apart from the parity, this mapping also gives us information about the occupation of the orbitals. If an orbital is occupied the value of the qubit will change with respect to the previous one (that is if we have either $|0\rangle_{j-1}$ and $|1\rangle_j$ or $|1\rangle_{j-1}$ and $|0\rangle_j$). On the other hand, if it is unoccupied, it will remain the same. The values of the qubits, determined by equation (46), can be summarized in the following expression, being the fermionic state $|\chi_{N-1} \dots \chi_1 \, \chi_0\rangle$, and the qubit state $|q_{N-1} \dots q_1 \, q_0\rangle$:

$$q_j = \sum_{i \le j} \chi_i \qquad (mod\ 2) \,. \tag{47}$$

The only thing lacking is mapping the action of creation and annihilation operators. The first thing to implement is the phase change present in the expressions of creation and annihilation operators (35,33). The phase change of the jth qubit depends on the parity of all the $j-1$ qubits, but in this mapping, luckily, the parity of the $j-1$ qubits is given directly by the value of the $j-1th$ qubit. This means we introduce the necessary phase just by applying a Z Pauli gate on the $j-1th$ qubit. For example, if the parity of all the qubits with index lower than j is

1, then the Z gate will introduce a -1 on qubit j.
Due to the fact that the state of each qubit no longer means if the orbital is occupied, the next step is looking for an alternative to the operators Q^+ and Q^-. As mentioned before, to know the occupation of the jth orbital, we need to take into account the value of the $j-1th$ qubit. If we have $|0\rangle_{j-1}$, then the value of qubit j will accurately reflect the occupation of orbital j, being $|0\rangle_j$ if it is unoccupied and $|1\rangle_j$ if it is occupied. This implies that in this case we can directly apply Q^+ when simulating $a_j^\dagger$. On the other hand, if the previous qubit is $|1\rangle_{j-1}$, and the jth qubit is occupied (then $|0\rangle_j$), to simulate $a_j^\dagger$, we will have to apply Q^- on qubit jth. Then the operator $Q^\pm$ is represented by

$$P_j^\pm = Q_j^\pm \otimes |0\rangle\langle 0|_{j-1} - Q_j^\mp \otimes |1\rangle\langle 1|_{j-1} = \frac{1}{2}(X_j \otimes Z_{j-1} \mp iY_j)\,. \tag{48}$$

One last thing that must be taken into account is the fact that changing the occupation number of the jth qubit changes the parity of all the qubits with index bigger than j. We then need to update the value of all these qubits. This is accomplished by a string of X operators acting on all qubits with index greater than j. Then the creation and annihilation operators in the parity mapping are simulated as follows,

$$a_j^\dagger \to X_{j+1}^{\leftarrow} \otimes P_j^+ = \frac{1}{2}(X_{j+1}^{\leftarrow} \otimes X_j \otimes Z_{j-1} - iX_{j+1}^{\leftarrow} \otimes Y_j)\,, \tag{49}$$

$$a_j \to X_{j+1}^{\leftarrow} \otimes P_j^- = \frac{1}{2}(X_{j+1}^{\leftarrow} \otimes X_j \otimes Z_{j-1} + iX_{j+1}^{\leftarrow} \otimes Y_j)\,, \tag{50}$$

where

$$X_i^{\leftarrow} = X_{N-1} \otimes X_{N-2} \otimes \ldots X_{i+1} \otimes X_i\,, \tag{51}$$

which is called the update operator. We were trying to reduce the number of Pauli gates needed respect to the Jordan-Wigner mapping but we have just exchanged a string of Z gates for a string of X operators. This mapping also scales linearly with the size of the system, it scales as $O(N)$.

The Bravyi-Kitaev mapping: While in the Jordan-Wigner mapping the occupancy was stored locally (with locally meaning that we knew if an orbital was occupied just by looking at the corresponding qubit) and the parity was stored non-locally (with non-locally meaning that in order to know the parity of an orbital we had to look at the state of all the qubits with lower index than the corresponding qubit), in parity mapping the occupancy was stored non-locally and the parity was stored locally. This transform tries a different approach, where both, occupancy and parity, are stored non-locally. It can be understood as a halfway point. Instead of storing part of the information locally and the rest completely non-locally, having then to add strings of gates that affect all the remaining qubits, it tries to find a balance between the locality and non-locality of the occupation and parity information in order to improve the efficiency. The basic idea is that some qubits ($|q_j\rangle$) store partial sums ($\sum_{s=k}^{l} \chi_s$) of occupation numbers. It can also be understood as if a qubit stored a set of orbitals, meaning that it stores the parity of the set of occupation numbers corresponding to that set of orbitals.

First, we are going to see the structure of the matrix which allows to transform a fermionic state to a qubit state in this mapping, it is built in a recursive way, its structure is the following:

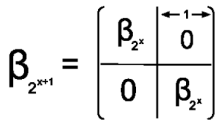

Figure 5: Bravyi-Kitaev matrix [15].

Following this structure, the matrix of the transformation in the case $x = 1$, knowing that $\beta_1 = 1$, when $x = 0$, is

$$\beta_2 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} . \tag{52}$$

Following the structure shown in figure (5), matrices for $x = 1, 2$ will be

$$\beta_{2^2} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} , \qquad \beta_{2^3} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} . \tag{53}$$

We can see that only odd indexes store information of other orbitals, for an odd value of j, it also stores information of orbitals with index less than j. Qubits with even value of j only hold the occupation state of orbital j. This can be seen in the following equation, being $|\chi_3\chi_2\chi_1\chi_0\rangle$ the fermionic state that we want to encode, we have

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \chi_3 \\ \chi_2 \\ \chi_1 \\ \chi_0 \end{pmatrix} = \begin{pmatrix} \chi_3 + \chi_2 + \chi_1 + \chi_0 \\ \chi_1 \\ \chi_1 + \chi_0 \\ \chi_0 \end{pmatrix} . \tag{54}$$

Reconstructing the creation and anihilation operators is not as trivial as in the previous methods. To extract the information from this coding we need to define three sets of qubits, the parity set, the update set and the flip set.

1. *The parity set.* One thing we need to know when simulating the action of creation and anihilation operators is whether or not it is necessary to introduce the phase -1 when acting on orbital j. The parity set of a qubit j is the set of qubits in the Bravyi-Kitaev basis that will tell us if this change of -1 is needed. The parity of this set of qubits has the same parity as the set of qubits with index less than j, this is why this set is called the parity set, denoted as $P(j)$.
   The parity of a qubit j is determined by the sum of the states of all the previous qubits, then the matrix determining the parities will have the following structure:

$$[\pi_n]_{ij} = \begin{cases} 1 \text{ if } i > j \\ 0 \text{ otherwise} \end{cases} \tag{55}$$

This matrix is very similar to the one that gives us the parity mapping but with 0 in the diagonal. This matrix must be applied onto the fermionic states, so we will have $\vec{p}_n = \pi_n \vec{\chi}_n = \pi_n \beta_n^{-1} \vec{q}_n$. Being β^{-1} the matrix that maps qubit strings in the Bravyi-Kitaev basis to the occupation number basis. The parity of a qubit j is given by the set of qubits with non-zero entries on the jth row. In the case of $n = 8$ we would have:

$$\vec{p}_8 \vec{q}_8 = \pi_8 \beta_8^{-1} \vec{q}_8 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} = \begin{pmatrix} q_6 + q_5 + q_3 \\ q_5 + q_3 \\ q_4 + q3 \\ q_3 \\ q_2 + q_1 \\ q_1 \\ q_0 \\ 0 \end{pmatrix} . \tag{56}$$

So for example the parity set of qubit with index 6 is $P(6) = \{q_5, q_3\}$.

2. *The update set.* In the same way when applying the parity mapping we had to include a string of X gates to update the parity of all the qubits with index greater than j when the occupation number of j changed, now when the occupation of a qubit changes we also have to update a set of qubits. This set of qubits that must be changed is called the update set, denoted by $U(j)$. Any qubit that stores information involving qubit j, this means that include partial sums depending on the occupation number j, is going to be included in the update set of index j. Since the transformation matrix is upper triangular, only qubits with index $i > j$ will be affected. Besides this, as mentioned before, even qubits do not store information of other orbitals. This means that only odd qubits will be part of the update set. This can be visualized in the transformation matrix. The update set of qubit with index j, $U(j)$, is formed by the non-zero entries in column j above the main diagonal. For example if we want to know $U(1)$ we would have:

$$\begin{matrix} 7 \\ 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \\ 0 \end{matrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow U(1) = \{q_3, q_7\} . \tag{57}$$

3. *The flip set.* This is the set of qubits that determine whether qubit j and orbital j are equal or opposite. This happened also in the parity mapping, when the gate that was gonna be applied (either $^+$ or $^-$) to j depended on the value of the previous qubit. So we had two options, for example if the orbital j was occupied, so $\chi_j = 1$, if qubit $j-1$ is $|0\rangle_{j-1}$, then $|1\rangle_j$, being equal the value of the qubit and the orbital. But if we had $|1\rangle_{j-1}$,

then $|0\rangle_j$, being the value the opposite. This is what the flip set help us to determine. It is denoted by $F(j)$. Since qubits with even index store just the information of the orbital with same index, then their flip set is empty. However, for odd indexes, we need to know which occupation states are included in each partial sum to transform back to the occupation state. This is done by looking at the inverse transformation matrix, defined by

$$\beta^{-1} = \begin{array}{c} \begin{matrix} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} = \begin{pmatrix} q_7 + q_6 + q_5 + q_3 \\ q_6 \\ q_5 + q_4 \\ q_4 \\ q_3 + q_2 + q_1 \\ q_2 \\ q_1 + q_0 \\ q_0 \end{pmatrix} . \tag{58}$$

In order to know if the occupation state of qubit j has the same value as the occupation of the orbital, we need to take into account the rest of qubits that contribute to the qubit state j. This would be for example in the case of $j = 7$, $F(7) = \{q_6, q_5, q_3\}$.

Once all these sets have been determined, it is possible to represent the creation and annihilation operators in the Bravyi-Kitaev basis.
In the case of a qubit with even j it is simple. The first thing is to apply $Q^{\pm}$ as usual (Q^+ for $a^\dagger$ and Q^- for a). Then, update the update set with a bit flip (as in the previous mapping, this is accomplished by applying X gates) and finally, introduce or not a negative sign depending on the value of the parity set (by introducing Z gates). This is

$$a_j^\dagger = X_{U(j)} \otimes Q_j^+ \otimes Z_{P(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{P(j)}) , \tag{59}$$

$$a_j = X_{U(j)} \otimes Q_j^- \otimes Z_{P(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} + iX_{U(j)} \otimes Y_j \otimes Z_{P(j)}) . \tag{60}$$

The size of $U(j)$ and $P(j)$ scales as $O(\log j) \leq O(\log n)$.
Simulating the creation and annihilation operators for odd indexes is not that easy. While for even indexes the flip set was the empty set, now it may have non-zero parity. In the case where the parity is non-zero, we would have to apply the creation operator to the qubit state where the annihilation operation is applied to the fermionic state and vice versa. It is necessary to define the following operators onto the even and odd states of a set S in order to take this into account

$$E_S = \frac{1}{2}(I + Z_S) , \tag{61}$$

$$O_S = \frac{1}{2}(I - Z_S) . \tag{62}$$

We are now able to define how the operator $Q^{\pm}$ is represented in the Bravyi-Kitaev basis for odd indexes, this is

$$\Pi_j^{\pm} = Q_j^{\pm} \otimes E_{F(j)} - Q_j^{\mp} \otimes O_{F(j)} = \frac{1}{2}(X_j \otimes Z_{F(j)} \mp iY_j) . \tag{63}$$

As in the case of even index, the update set of j has to be updated, it is done in the same way, with X gates. The only thing left is whether is it necessary to multiply by -1 depending on the parity of the parity set. $P(j)$ and $F(j)$ are not disjoint, and we need to take into account the parity of the qubits that belong to $P(j)$ but not to $F(j)$. The reason is that the relative sign in the $\Pi_j^{\pm}$ operator implicitly calculates the parity of the subset of the parity set that is also in the flip set of index j. We need to introduce the set $R(j)$, defined by

$$R(j) = P(j) \backslash F(j) \,. \tag{64}$$

This equation means that the set $R(j)$ is composed of all the qubits that belong to $P(j)$ but not to $F(j)$. Then the representation of the creation and annihilation operators in the Bravyi-Kitaev mapping for odd indexes will be

$$a_j^\dagger = X_{U(j)} \otimes \Pi_j^+ \otimes Z_{R(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{R(j)}) \,, \tag{65}$$

$$a_j = X_{U(j)} \otimes \Pi_j^- \otimes Z_{R(j)} = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} + iX_{U(j)} \otimes Y_j \otimes Z_{R(j)}) \,. \tag{66}$$

It is noticeable that the only difference between the anihilation and creation operators for even and odd indexes is the subset in which the Z gates are applied. Then we can summarize all these results in the following expressions

$$\rho_j = \begin{cases} P(j) & \text{if j even} \\ R(j) & \text{if j odd} \end{cases} \tag{67}$$

and finally

$$a_j^\dagger = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{\rho(j)}) \,, \tag{68}$$

$$a_j = \frac{1}{2}(X_{U(j)} \otimes X_j \otimes Z_{P(j)} + iX_{U(j)} \otimes Y_j \otimes Z_{\rho(j)}) \,. \tag{69}$$

We can see that instead of applying gate string to all the qubits of the system, this method focuses on sets of qubits whose size scales as $O(\log(N))$. This is what makes this mapping more efficient that the other two, instead of scaling as $O(N)$, it scales as $O(\log(N))$.

4.5 Ansatz selection

The choice of ansatz is crucial to get a good solution as it limits the accuracy of the simulation result. Choosing an ansatz that does not cover the solution of the problem will give us an invalid solution. The objective is to obtain an ansatz which provides high accuracy with few parameters (so that the classical optimizer works) and shallow circuits (the less circuits the less error due to noise we will have). There exist different ways of creating an ansatz, divided in two main groups, chemistry-inspired ansatz and hardware-efficient ansatz, in this subsection the more important ones are going to be shorty explained, a good review can be found in [16].

Chemistry-inspired ansatz They are based on quantum chemistry, meaning that every term in the ansatz describes a certain electron configuration. The first ansatz of this type studied is the Unitary Coupled Cluster Single and Double (UCCSD). It takes an initial state, usually a Hartree-Fock state, and makes it evolve using the exponenciated excitation operator.

This operator is based on the Hermitian cluster operator of coupled cluster theory. The operator used in UCCSD is defined as

$$T = T_1 + T_2 = \sum_{i\alpha} \theta_{i\alpha}(a_\alpha^\dagger a_i - a_i^\dagger a_\alpha) + \sum_{ij\alpha\beta} \theta_{ij\alpha\beta}(a_\alpha^\dagger a_\beta^\dagger a_i a_j - a_j^\dagger a_j^\dagger a_\alpha a_\beta)\,, \tag{70}$$

where $\theta_{i\alpha}$ and $\theta_{ij\alpha\beta}$ are cluster amplitudes. With this definition, we make the initial state evolve as follows

$$|\psi_{UCCSD}\rangle = e^{T_1+T_2}|\psi_{HF}\rangle \tag{71}$$

Its name, single and double excitation, comes from the fact that in equation (70) we have one term which contains all the single excitations and a second term which contains all the double excitations. More excitations could be needed depeding on the system to be simulated. In our problem we are able to truncate the operator at double ones. This ansatz gives as a result excessively large circuits with a too high number of parameters, which becomes worse as the number of particles increase.

Hardware-efficient ansatz While the previous type of ansatzs were created through the decomposition of a fermionic operator into quantum gates. Now we will see ansatzs based directly on quantum circuits are used. They lack of physical motivation. The main idea is to create a quantum circuit easy to run in the actual quatum computers that satisfies the condition of containing the solution state while maintaining a low number of parameters and quantum gates. Some examples of this type of ansatz would be the ones supported by qiskit (IBM platform used to elaborate the programs that also allows to run them in real quantum computers), for example $R_y R_z$ with linear entanglement (6).

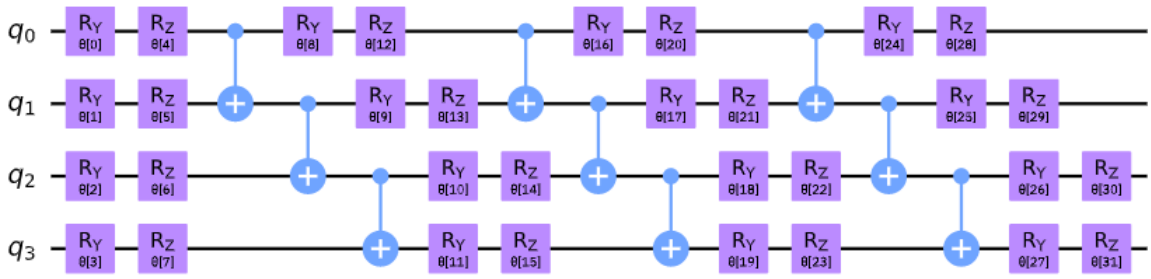

Figure 6: Linear entanglement

Another option is to create a circuit that considers all the possible states, for example in the case of having just one qubit, a circuit capable of generating any possible state would be

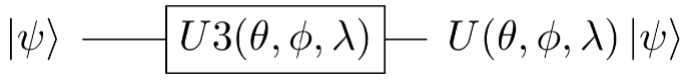

Figure 7: Universal ansatz (1 qubit).

Where

$$U3(\theta,\phi,\lambda) = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2}\theta & e^{i\lambda+i\phi}\cos\frac{\theta}{2}\theta \end{pmatrix}. \tag{72}$$

In the case we have 2 qubits, this circuit would be The efficiency of this circuit will be analysed

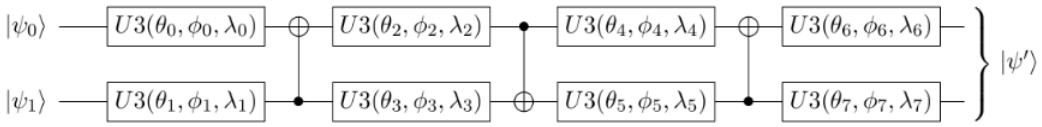

Figure 8: Universal ansatz (2 qubits).

in the results.

Something in between This type of ansatz is between the chemistry-inspired ansatz and the hardware-efficient ansatz. One algorithm from this group that has been shown that gives good results is the Adaptative Variational algorithm (ADAPT-VQE) [17]. This ansatz is built iteratively by adding fermionic operators. It achieves great accuracy with less gates than other types of ansatzs. It has been shown that it creates more compact and accurate wavefunctions ansatzs than UCCSD, although it requires a higher number of measurements. Based on this idea, other types of ansatzs have been created, such as the qubit-excitation-based adaptative variational quantum eigensolver (QBE-ADAPT-VQE) [18].

5 Experiments

The aim of this project was not only creating a VQE algorithm that calculated the ground state of different molecules but also studying how different choices influence the result. We have accomplished this for the H_2 molecule and explored its use for larger molecules. We have been able to analyse how the choice of ansatz, type of mapping and classical optimizer determine the result.
The algorithm has been programmed using *Qiskit* [19], an open-source software created for working with quantum computers. *Qiskit* also has a package with VQE algorithms already programmed. We have executed this algorithms in two different ways, simulating the behaviour of a quantum computer by using *Qiskit* on a classical computer and using a real quantum computer. We have been able to do so thanks to IBM Quantum, a platform that facilitates access to some of their quantum computers.
IBM quantum computers use superconducting circuits to implement qubits [20]. The fact that they need to be at really low temperatures (around a hundredth of a degree above absolute zero) makes them have a very big structure, see figure (9).

Figure 9: IBM Quantum Computer.

In this section we are first going to see the results obtained with the program we made, studying the results with different ansätze and the parity mapping. Then, we are going to execute the program already made by *Qiskit*, also with the parity mapping and adding the $UCCSD$ ansatz to the experiments. Finally, different mappings and the simulation of bigger molecules are going to be commented.

5.1 First experiment for H_2

In this section we are going to analyse the results obtained with the program we made, which is going to be called Program 1. It has been developed following the steps explained in the previous sections. First, the data of the molecule has been obtained from a driver (here we define the structure of the molecule, in this case the bond length). Here we can specify which mapper we want to use or what type approximations we want to include. It gives us the Hamiltonian written in terms of Pauli gates. Then, we create a function where the ansatz is defined. Once all this is done, the function that measures the expectation value of the Hamiltonian is written. This is done by constructing the adequate circuit using (4.3). Finally, a main function that calls all the previous ones and that includes the classical optimizer finds and stores the solution, this is the ground energy of the molecule for the distance selected (see Appendix).
The following graphics show the ground state energy of the H_2 molecule depending on the bond length. The simulations have been executed in a classical computer using *Qiskit* to simulate a quantum computer. Two cases have been studied, assuming a perfect behaviour and using a noisy simulator (the results of this one are going to be closer to the results of the actual device). For each simulation, two ansätze have been used, the one proposed in [1] and the universal ansatz for 2 qubits defined in (4.5). These results are shown in figure (10).

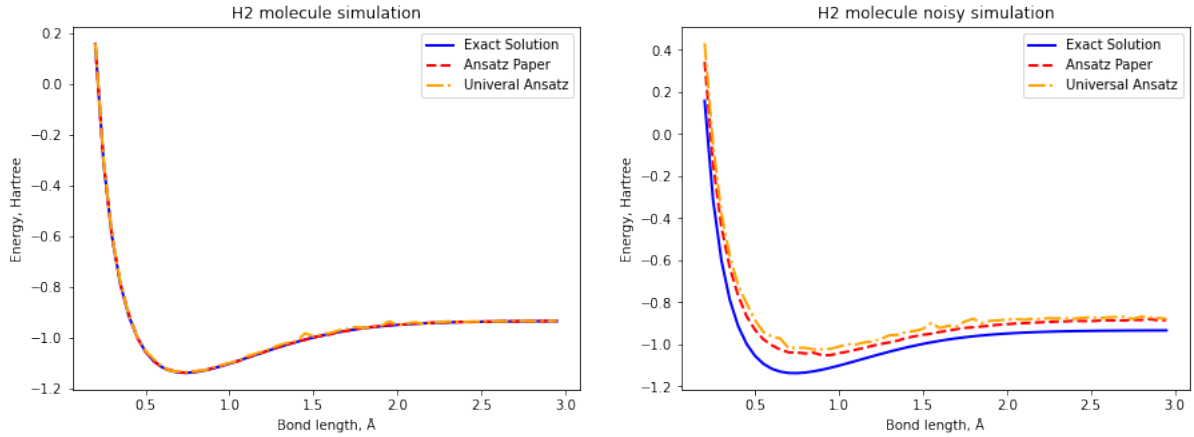

Figure 10: H_2 molecule simulations with parity mapper and my program.

We can see that the noisy simulation gives worse results that the first simulation. The accuracy of the program with each ansatz is studied in figure (11). Now, the energy is measured in $kcal/mol$, a result in considered acceptable if the error is below 1 $kcal/mol$.

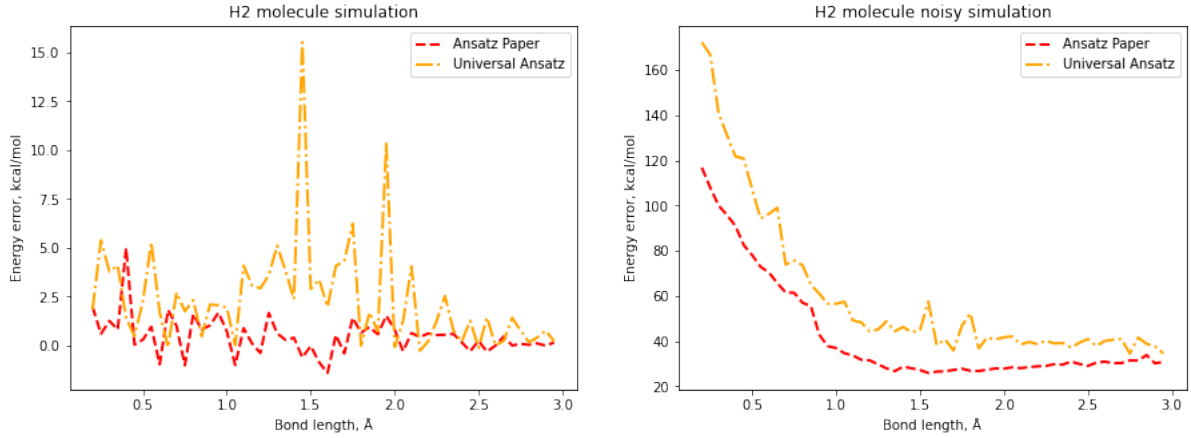

Figure 11: Energy differences from the exact solution with parity mapper and my program

This graphics show that although with perfect behaviour the precision is acceptable, the program does not reach chemical accuracy.
As mentioned before, the program has also been executed in a real quantum computer. Figure (12) show the results for the same two ansätze previously used.

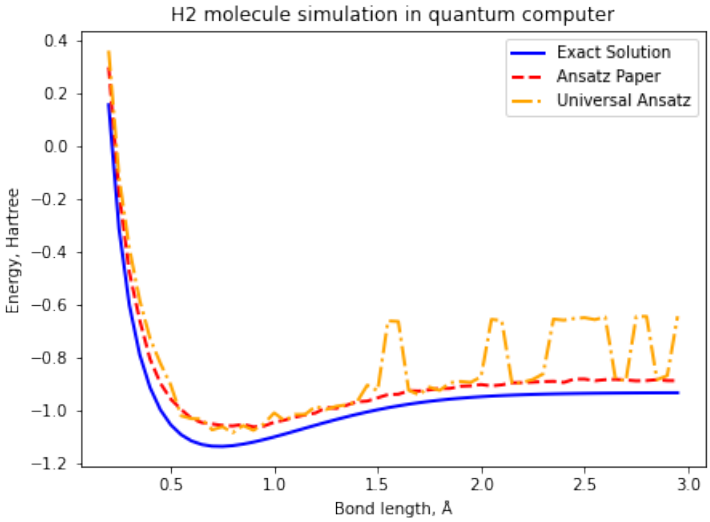

Figure 12: H2 molecule simulations in quantum computer with parity mapper and my program

Although for long distances the universal ansatz does not give good results, around the minimum they behave similarly to the noisy simulation from figure (10). Its accuracy in shown in figure (13).

EXECUTION TIME (min)	Ansatz Paper	Universal Ansatz
Simulation	28.47	121.22
Noisy simulation	45.63	317.32
Quantum computer	201.58	2592.17

Table 1: Comparison of the simulation execution time for different methods.

GROUND STATE ENERGY (Hartree)	Ansatz Paper	Universal Ansatz
Simulation	−1.13937	−1.13753
Noisy simulation	−1.04664	−1.03005
Quantum computer	−1.06333	−1.08500

BOND LENGTH (Ångstrom)	Ansatz Paper	Universal Ansatz
Simulation	0.745	0.735
Noisy simulation	0.760	0.835
Quantum computer	0.900	0.800

Table 2: Simulation results (estimated ground state energy and bond length) for the different tested methods.

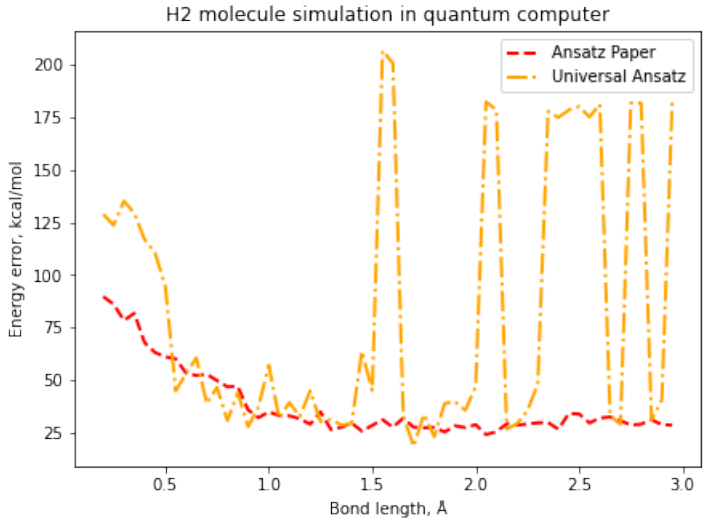

Figure 13: Energy differences from the exact solution with parity mapper and my program.

Apart from the accuracy, the execution time is also an important factor. In the following table we can see the time it took to each simulation with each ansatz to complete the program.

Finally, a comparison of the ground state found in each case is given in the following two tables.

5.2 Second experimemt for H_2

As mentioned before, *Qiskit* provides with a package with VQE algorithms already programmed. This program is going to be called Program 2. In this section we are going to study its performance both with the perfect behaviour and with the noisy simulation. In this case, appart from using the same ansätze as in the previous section, we are going to use the $UCCSD$ ansatz. This is eventually going to be executed in a real quantum computer.

Figure (14) shows the ground energy found for each bond length and each ansatz.

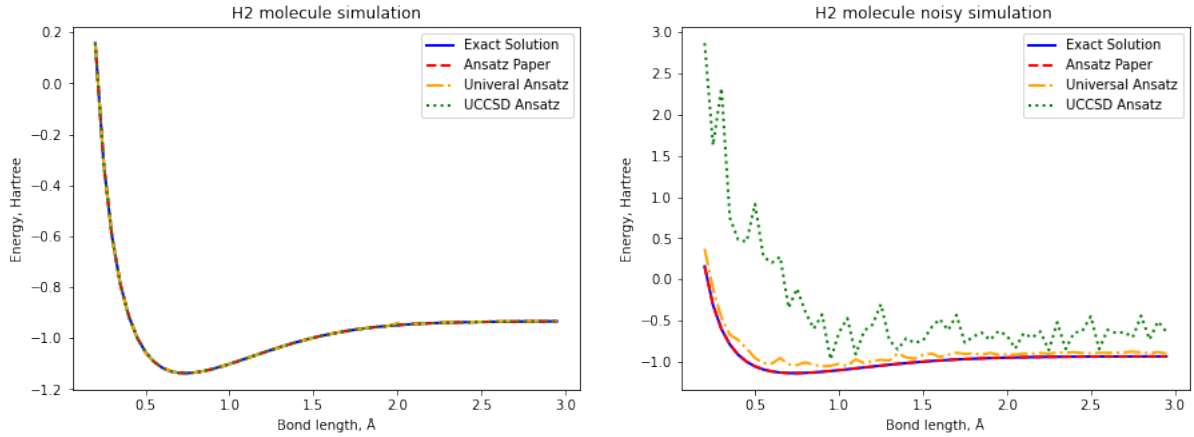

Figure 14: H2 molecule simulations with parity mapper and VQE python package.

We can see that the $UCCSD$ result for the noisy simulation does not give good results. The errors can be seen in figure (15)

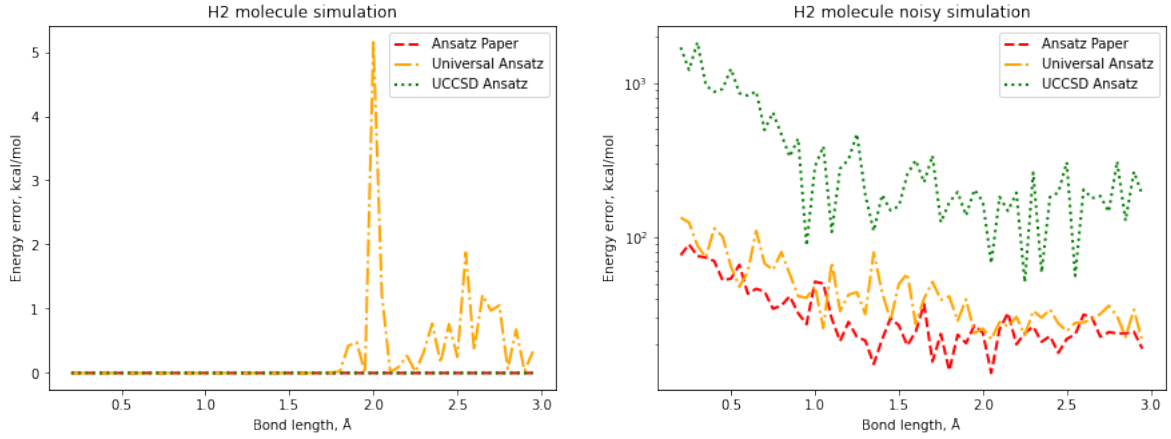

Figure 15: Energy differences from the exact solution with parity mapper and my program.

In the case of the noiseless simulation, all ansätze work perfectly around the ground state and

chemical accuracy is reached. On the other hand, noisy simulations are too inaccurate, especially the one performed with the $UCCSD$ ansatz.
Lastly, the simulations have been executed in a quantum computer, in this case only the ansatz from the paper and the universal ansatz have been used. Figure (16) shows the results, comparing the energy found at each distance for both ansätze.

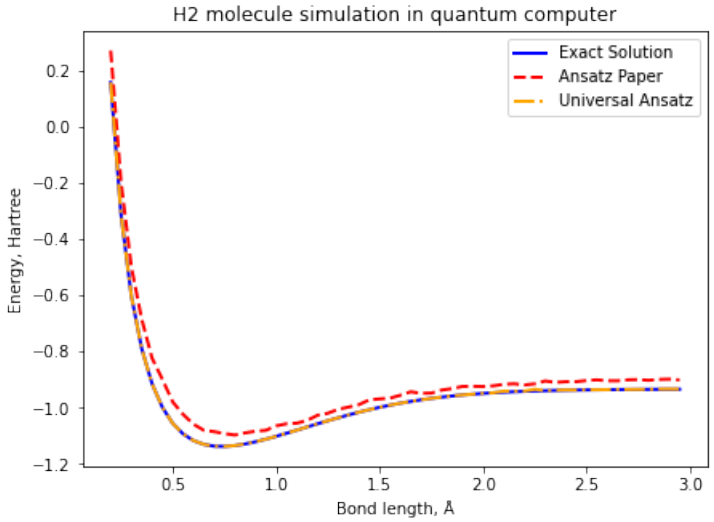

Figure 16: H_2 molecule simulations in quantum computer with parity mapper and the VQE Python package.

Again, the accuracy of these last simulation is shown in figure (17).

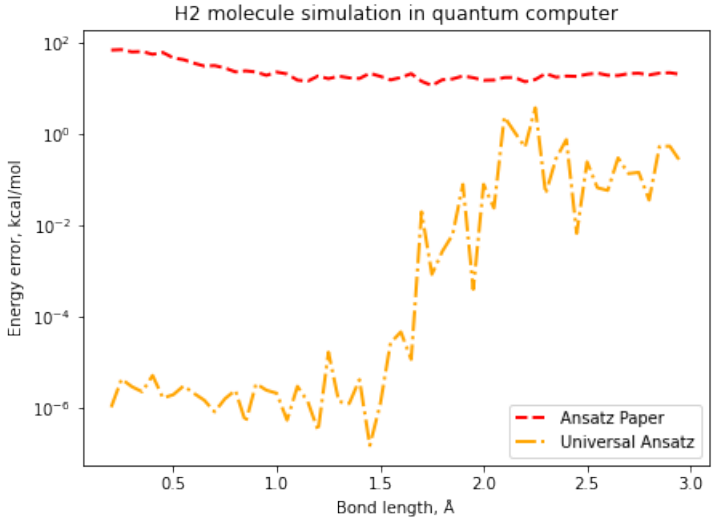

Figure 17: Energy differences from the exact solution with parity mapper and the VQE Python package (log units).

EXECUTION TIME (min)	Ansatz Paper	Universal Ansatz	UCCSD
Simulation	18.93	11.66	11.84
Noisy simulation	37.29	99.67	52.64
Quantum computer	95.57	3.67	—

Table 3: Comparison of the simulation execution time for different methods.

GROUND STATE ENERGY (Hartree)	Ansatz Paper	Universal Ansatz	UCCSD
Simulation	−1.13731	−1.13731	−1.13731
Noisy simulation	−1.09377	−1.09355	−1.04435
Quantum computer	−1.09663	−1.13712	—

BOND LENGTH (Ångstrom)	Ansatz Paper	Universal Ansatz	UCCSD
Simulation	0.735	0.735	0.735
Noisy simulation	0.715	0.755	0.685
Quantum computer	0.800	0.750	—

Table 4: Simulation results (estimated ground state energy and bond length) for the different tested methods.

We can see that the ansatz proposed in [1] does not work well, while the universal ansatz gives really good results. This can be due to the fact that the universal ansatz spans all states space, so it is easier to find the ground state.
The following tables show a comparison of the execution times of each program, together with the ground state energy and bond length found.

5.3 Different mappings

We have also executed these simulations with the Jordan-Wigner and the Bravyi-Kitaev mapping. We have seen that Parity and Bravyi-Kitaev mapping generate a Hamiltonian with the same number of Pauli gates, but, on the other hand, the Hamiltonian produced by the Jordan-Wigner mapping is longer. We would expect it to give less accurate results due to the increase of noise with the number of gates.
All the previous simulations with each type of ansatz have been run with each of the mappings, showing that the only one that find a good solution, besides from the Parity mapping, is the Bravyi-Kitaev mapping with the universal ansatz. It is not worthwhile showing all the failed results, so we are just going to look at this last one mentioned. Figure (18) shows the ground energy found at each distance comparing Program 1 with Program 2, both with noisy and noiseless simulations.

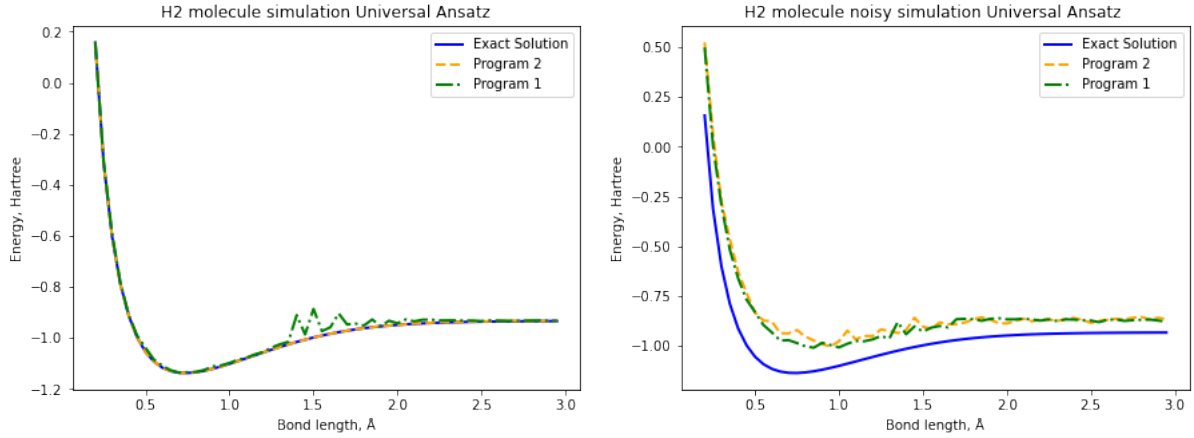

Figure 18: $H2$ molecule simulations with the Bravyi-Kitaev mapping and the Universal Ansatz.

The accuracy of these results is shown in figure (19).

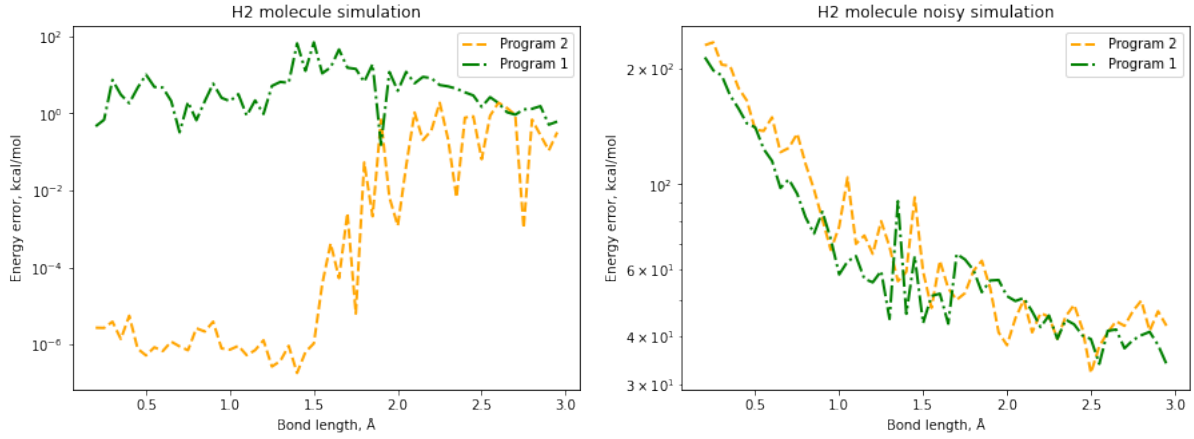

Figure 19: Energy differences from the exact solution with the Bravyi-Kitaev mapper and the Universal Ansatz comparing both programs.

We can see that with perfect behaviour the VQE python package gives really good results, while in the noisy simulation neither of them reaches chemical accuracy.
These simulations has also been run in a real quantum computer, the results are shown in figure (20).

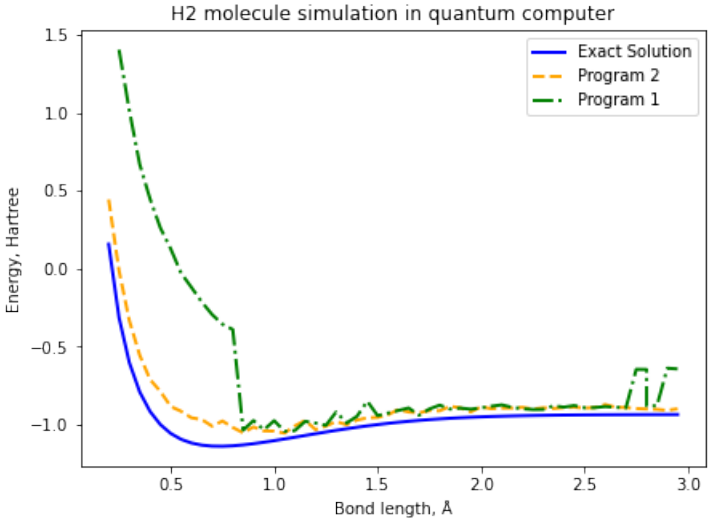

Figure 20: H_2 molecule simulations in quantum computer with the Bravyi-Kitaev mapper and both programs using the Universal Ansatz.

Here, Program 1 does not work very, maybe this could be because it is not being able no find the state that minimizes the energy.
Its accuracy can be seen in figure (21)

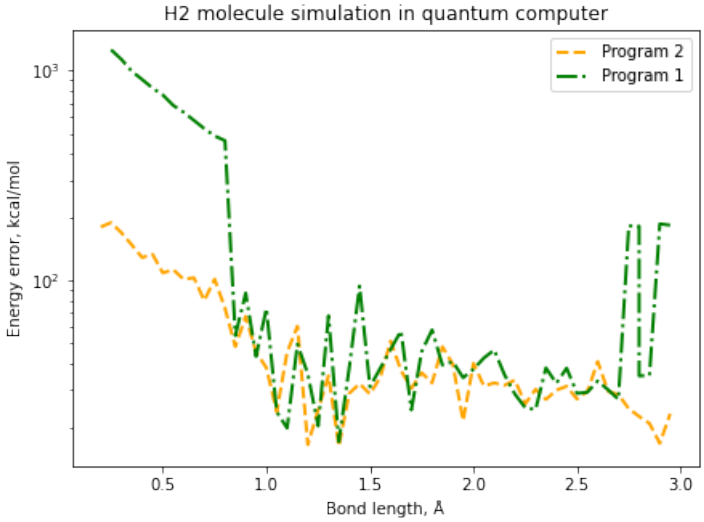

Figure 21: Energy differences from the exact solution with Bravyi-Kitaev mapper and Universal Ansatz comparing both programs.

Finally, we have the tables comparing the execution times and ground states found.

We can see that the execution time of Program 2 is smaller. The accuracy, however, is similar with both programs, at least for larger bond lengths.

Program 1	Execution Time (min)	Ground State Energy (Hartree)	Bond Length (Å)
Simulation	151.47	−1.1357398010788513	0.750
Noisy simulation	801.45	−1.009965704394788	0.850
Quantum computer	1865.92	−1.0415796647509152	1.05

Program 2	Execution Time (min)	Ground State Energy (Hartree)	Bond Length (Å)
Simulation	8.20	−1.137306029434759	0.735
Noisy simulation	104.63	−1.0085601307952317	0.805
Quantum computer	911.42	−1.0526563913799243	1.05

Table 5: Comparison of the simulation results (estimated ground state energy and bond length) and efficiency (execution time) for the different methods.

5.4 Bigger Molecules

Two atoms of hydrogen were the first atoms simulated with this type of programs. Lately, bigger molecules have been simulated, these have been LiH, BeH_2 and H_6. Using the *Qiskit* package, we have been able to simulate a molecule of LiH and study some of the properties of this program.
One of these properties has been the influence of the ansatz on the solution. We have executed a noiseless simulation using a circuit based ansatz and a fermionic based ansatz. The ansätze used are a chemistry-inspired one, $UCCSD$, and a hardware-efficient one, R_yR_x with linear entanglemen,t defined in section (4.5). The results can be seen in figure (22)

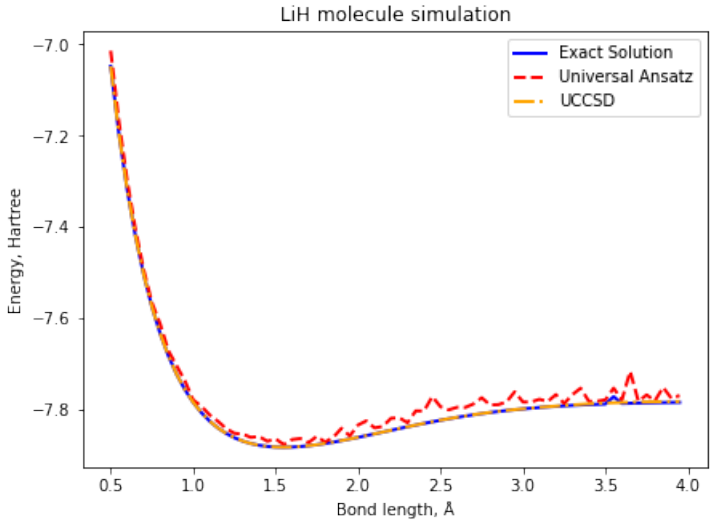

Figure 22: LiH molecule simulations in quantum computer with parity mapper using the VQE Python package.

The precision of the previous data can be seen in figure (23).

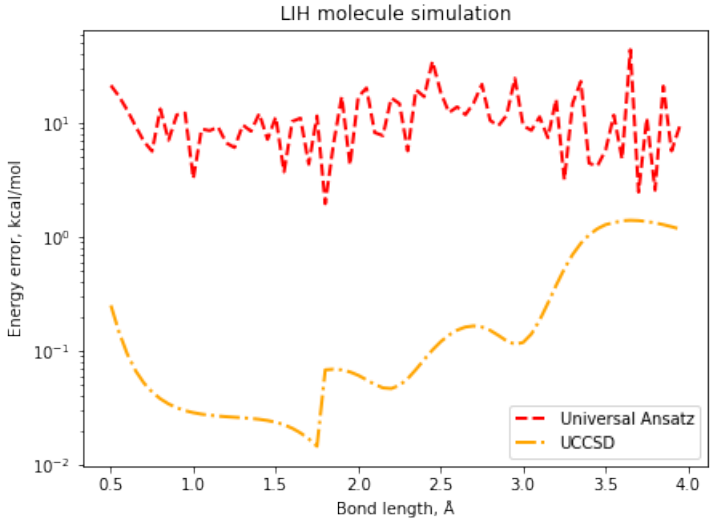

Figure 23: Energy differences from the exact solution with parity mapper using the VQE Python package.

It is clear that the $UCCSD$ ansatz gives way better results, it reaches chemical accuracy. We have seen that no chemical inspired ansätze seem to have difficulties finding the ground state when dealing with bigger molecules.
Another thing that should be mentioned is that the execution time of the program using $UCCSD$ ansatz is three hours, while the one using $R_y R_x$ ansatz is 45 min. This shows that the $R_y R_x$ ansatz is executed easily but gives less precise results.
Since this executions took a lot of time, different classical optimizers were also tested for the case of circuit based programs. We tried a couple of them based on gradient descent methods, but we did not see any improvement. Eventually, we used COBYLA, a linear approximation optimizer.

6 Discussion

During this work we have analysed in depth how variational quantum eigensolvers work. Starting from the basic concepts such as the definition of a qubit, we have been able to make a good revision of its major points.
The original idea was to reproduce the results from [1] and, from there, to study the influence of the choice of ansatz, optimizer and encoding. Having understood the mechanism of this algorithms, we managed to create a program that simulated a H_2 molecule and obtained its ground state and ground state energy. We have executed this program with different ansätze in order to study its efficiency. We found out that assuming perfect behaviour, the ansatz from [1] gave better results than the universal ansatz. In fact, the accuracy of the ansatz from the paper is near the limit of the chemical accuracy. We have seen that although the universal ansatz covers all the state space, it gives worse results. This can be related to the ability of the classical optimizer to find the minimum value of the energy since this ansatz has 24 parameters. There is a huge difference with the ansatz from the paper, which has just one parameter. When introducing the noisy simulation, again, the universal ansatz gives worse results. Now, this can also be caused by the fact that this ansatz is constructed with many more gates, so the noise will be bigger.
When executing the program in a real quantum computer, we verified that the ansatz from the paper gives better results, although not good enough to reach chemical accuracy. As mentioned before, the fact that the universal ansatz gives worse results could be caused by the number of parameters involved or to the number of gates needed. In conclusion, we have managed to develop a program that calculates the ground state of the H_2 molecule quite properly but not with the enough precision to be chemically accurate.

Afterward, we used the VQE python package, which includes the functions that allow us to execute VQE algorithms. Using these functions we have calculated the ground state of the H_2 molecule. Again, simulation this molecule, we have been able to study how different ansätze behave, now including the chemistry-inspired ansatz $UCCSD$ explained in section (4.5). We found out that when assuming perfect behaviour all of them reach chemical accuracy (at least around the minimum of the energy), but when making the noisy simulation, the results obtained are not good, especially with the $UCCSD$ ansatz. The precision of the circuit based ansätze in the noisy simulation is similar to the precision we obtained with the program we developed. The fermionic based ansatz, $UCCSD$, seems to be very sensitive to noise, giving very inaccurate results.
The simulation with the universal ansatz and the ansatz from the paper has been also launched in a quantum computer. Now, the results obtained with the universal ansatz are better than the ones obtained with the ansatz from the paper, reaching chemical accuracy. This could be because it samples better the state space so it finds better solutions. On the other hand, maybe this program behaves better with the noise and is more easily optimized than the program we developed. This would explain the difference in the results.

We also executed all these simulations with the other two types of mapping introduced in section (4.4.3), finding that the Bravyi-Kitaev mapping with the universal ansat was the only one that worked properly. Again, the simulations were executed assuming perfect behaviour and then using a noisy simulator. When assuming perfect behaviour only Program 2 reached chemical accuracy, although our program did not give bad solutions. On the other hand, when using a noisy simulator, the results were far from chemical accuracy.
Finally, these programs were launched in a quantum computer. For the lowest bond lengths, the results are not good, but eventually an approximated value of the ground state is reached. The

results are not considered chemically accurate.

In conclusion, when executing the programs in a real device, while Program 1 shows better results when using the ansatz from the paper [1], Program 2 finds a better solution with the universal ansatz. We have also seen that $UCCSD$ ansatz is too sensitive to the noise. Finally, universal ansatz is the only one that works for Bravyi-Kitaev mapping.

We have also studied the simulation of a LiH molecule, analysing how the type of ansatz and classical optimizer chosen affects the result. This has been done by using the *Qiskit* VQE algorithms.
First, we compared the accuracy of two different types of ansatz, the $UCCSD$ (chemistry-inspired) and the R_yR_x with linear entanglement (hardware-efficient). Both are described in section (4.5). We found that the $UCCSD$ assuming perfect behaviour gives way better results than the R_yR_x. In fact, it reaches chemical accuracy. It seems that when dealing with big molecules, hardware-efficient ansätze struggle with finding the ground state.

Future research could be focused on looking for more efficient ansätze, maybe trying to implement the newest ones (ADAPT-VQE for example). Trying to make the programs work when considering different mappings could be another way of continuing this work. Other option could be focusing on bigger molecules, studying the properties of different ansätze when using VQE python package in a real quantum computer.
A different thing that has not been mentioned in this work but that can be a good topic to study is the classical optimizer. Trying to look for a faster and more efficient method to optimize the function when having a large number of parameters is also a big challenge.

References

[1] O'Malley, P., Babbush, R., Kivlichan, I., Romero, J., McClean, J., Barends, R., Kelly, J., Roushan, P., Tranter, A., Ding, N., Campbell, B., Chen, Y., Chen, Z., Chiaro, B., Dunsworth, A., Fowler, A., Jeffrey, E., Lucero, E., Megrant, A., . . . Martinis, J. (2016). Scalable Quantum Simulation of Molecular Energies. *Physical Review X, 6*(3).

[2] *Mathematics as Metaphor. Selected Essays of Yuri I. Manin 77–78* (American Mathematical Society, 2007).

[3] Benioff, P. (1980). The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics, 22*(5), 563–591.

[4] Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical Physics, 21*(6–7), 467–488.

[5] Peruzzo, A., McClean, J., Shadbolt, P., Yung, M. H., Zhou, X. Q., Love, P. J., Aspuru-Guzik, A., O'Brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature Communications, 5*(1).

[6] Nielsen, M. A., Chuang, I. L. (2010). *Quantum Computation and Quantum Information* (10th Anniversary ed.). Cambridge University Press.

[7] Cohen-Tannoudji, C., Diu, B., Laloe, F. (1991). *Quantum Mechanics, Volume 1* (Volume 1 ed.). Wiley-Vch.

[8] Aspuru-Guzik, A., Dutoi, A. D., Love, P. J. Head-Gordon, M. (2005). Simulated Quantum Computation of Molecular Energies. *Science, 309*(5741), 1704-1707.

[9] Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X., Cincio, L., Coles, P. J. (2021). Variational quantum algorithms. *Nature Reviews Physics, 3*(9), 625–644.

[10] McClean, J. R., Romero, J., Babbush, R., Aspuru-Guzik, A. (2016).The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics, 18*(2), 023023.

[11] Maupin, O. G., Baczewski, A. D., Love, P. J., Landahl, A. J. (2021). Variational Quantum Chemistry Programs in JaqalPaq. *Entropy, 23*(6), 657.

[12] Tilly, J., Chen, H., Cao, S., Picozzi, D., Setia, K., Li, Y., Grant, E., Wossnig, L., Rungger, I., Booth, G. H. Tennyson, J. (2021). The Variational Quantum Eigensolver: a review of methods and best practices. arXiv:2111.05176 [quant-ph]

[13] Tranter, A., Sofia, S., Seeley, J., Kaicher, M., McClean, J., Babbush, R., Coveney, P. V., Mintert, F., Wilhelm, F., Love, P. J. (2015). The Bravyi-Kitaev transformation: Properties and applications. *International Journal of Quantum Chemistry, 115*(19), 1431–1441.

[14] Tranter, A., Love, P. J., Mintert, F., Coveney, P. V. (2018). A Comparison of the Bravyi–Kitaev and Jordan–Wigner Transformations for the Quantum Simulation of Quantum Chemistry.*Journal of Chemical Theory and Computation, 14*(11), 5617–5630.

[15] Seeley, J. T., Richard, M. J., Love, P. J. (2012). The Bravyi-Kitaev transformation for quantum computation of electronic structure. *The Journal of Chemical Physics, 137*(22), 224109.

[16] Fedorov, D. A., Peng, B., Govind, N. Alexeev, Y. (2022). VQE method: a short survey and recent developments . *Materials Theory, 6*(1).

[17] Grimsley, H. R., Economou, S. E., Barnes, E. Mayhall, N. J. (2019). An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications, 10*(1).

[18] Yordanov, Y. S., Armaos, V., Barnes, C. H. W. Arvidsson-Shukur, D. R. M. (2021). Qubit-excitation-based adaptive variational quantum eigensolver. *Communications Physics, 4*(1).

[19] Qiskit.org. 2022. qiskit.org. [online] Available at: ¡https://qiskit.org/¿ [Accessed 16 September 2022].

[20] Rasmussen, S., Christensen, K., Pedersen, S., Kristensen, L., Bækkegaard, T., Loft, N. Zinner, N. (2021). Superconducting Circuit Companion—an Introduction with Worked Examples. *PRX Quantum, 2*(4).

Appendix

First, we need to import all the necessary packages.

[ ]:
```
from qiskit import *
from numpy import *
import matplotlib.pyplot as plt
from qiskit_nature.drivers import Molecule
from qiskit_nature.drivers.second_quantization import (
    ElectronicStructureMoleculeDriver, ElectronicStructureDriverType)
from qiskit_nature.transformers.second_quantization.electronic import␣
↪FreezeCoreTransformer
from qiskit_nature.problems.second_quantization import ElectronicStructureProblem
from qiskit_nature.converters.second_quantization import QubitConverter
from qiskit_nature.mappers.second_quantization import ParityMapper,␣
↪JordanWignerMapper, BravyiKitaevMapper
from scipy.optimize import minimize
import numpy as np
from qiskit.algorithms.optimizers import COBYLA, SPSA, SLSQP, GradientDescent
from qiskit.opflow import TwoQubitReduction, Z2Symmetries
from qiskit import BasicAer, Aer
from qiskit.utils import QuantumInstance
from qiskit.quantum_info import Pauli
import time
```

This function obtains the data about the molecule we want to simulate. In the case of H2, it requests as an input the distance between the two atoms. As a result, it gives us the Hamiltonian written in terms of Pauli gates and some other properties of the molecule.

[ ]:
```
def get_qubit_op(dist):

    molecule = Molecule(
        # Coordinates in Angstrom
         geometry=[
            ["H", [0.0, 0.0, 0.0] ],
            ["H", [dist, 0.0, 0.0] ]        # One atom is placed in (0,0,0) and␣
↪the other in (dist,0,0), being dist the distance between both atoms
        ],
        multiplicity=1,                     # = 2*spin + 1
        charge=0,
    )

    # The driver takes the geometry of the molecule defined above and gives us␣
↪the intermediate data
    driver = ElectronicStructureMoleculeDriver(
        molecule=molecule,
        basis="sto3g",
        driver_type=ElectronicStructureDriverType.PYSCF)
```

```
    # The next line is needed in order to get the driver executed
    properties = driver.run()

    # Some properties of the molecule that are going to be needed are extracted␣
↪from the data
    num_particles = (properties
                         .get_property("ParticleNumber")
                         .num_particles)
    num_spin_orbitals = int(properties
                            .get_property("ParticleNumber")
                            .num_spin_orbitals)
    nuclear_repulsion_energy = properties.get_property("ElectronicEnergy").
↪nuclear_repulsion_energy

    # The problem is defined, the frozen core approximation is applied.
    problem = ElectronicStructureProblem(
        driver,
        [FreezeCoreTransformer(freeze_core=True)])

    # The Hamiltonian is quantized, in this case he second quantization is␣
↪applied
    second_q_ops = problem.second_q_ops()

    # Fermion to qubit mapping, the desired mapping has to be defined. Qubit␣
↪reduction is also applied.
    mapper = ParityMapper()
    hamiltonian = second_q_ops[0]
    converter =␣
↪QubitConverter(mapper,two_qubit_reduction=True,z2symmetry_reduction= "auto") ␣
↪#the two_qubit_reduction is just applied when possible
    reducer = TwoQubitReduction(num_particles)
    qubit_op = converter.convert(hamiltonian)
    qubit_op = reducer.convert(qubit_op)
    return qubit_op, nuclear_repulsion_energy, num_particles, num_spin_orbitals,␣
↪problem, converter
```

The next function defines the ansatz, in this case the ansatz paper is defined.

[ ]:
```
def define_ansatz(theta):
    q = QuantumRegister(2)
    c=ClassicalRegister(2)
    ansatz = QuantumCircuit(q,c)
    ansatz.rx(pi,q[1])
    ansatz.rx(-pi/2,q[1])
    ansatz.ry(pi/2,q[0])
    ansatz.cx(q[0],q[1])
```

```
    ansatz.rz(theta[0],q[1])
    ansatz.cx(q[0],q[1])
    ansatz.ry(-pi/2,q[0])
    ansatz.rx(pi/2,q[1])
    return (ansatz)
```

This next cell contains the function that calculates the expectation energy of the Hamiltonian.

[ ]:
```
def Energy_general(theta, R, num_qubits, gates_list_hamiltonian,␣
↪num_elements_hamiltonian):


    shots = 2**14                                    # Number of samples used for␣
↪statistics
    sim = Aer.get_backend('aer_simulator')      # The simmulator is chosen, it␣
↪mimics the execution of an actual device

    # Let's define some variables
    E_sim = []
    Energy_meas = []
    index_param = 0

    # This first loop goes through each term in the Hamiltonian
    for measurement in gates_list_hamiltonian:

        index_param += 1                                # This parameter helps us to␣
↪multiply each term by the correct coefficient
        n=num_qubits                                    # This parameter is the one␣
↪in charge to put each gate in the correct qubit
        q = QuantumRegister(num_qubits)
        c = ClassicalRegister(num_qubits)
        hamiltonian_circuit = QuantumCircuit(q,c)

        # This loop goes through each element of a specific term of the␣
↪Hamiltonian and creates the
        # circuit needed to measure its expectation value
        for sing_gate in measurement:
            n -= 1
            if sing_gate == Pauli('Z'):
                hamiltonian_circuit.measure(n,n)
            if sing_gate == Pauli('X'):
                hamiltonian_circuit.h(q[n])
                hamiltonian_circuit.measure(n,n)
            if sing_gate == Pauli('Y'):
                hamiltonian_circuit.sdg(q[n])
                hamiltonian_circuit.h(q[n])
                hamiltonian_circuit.measure(n,n)
```

```
        # Once the measurement circuit is built, it is merged with the ansatz
        qc = define_ansatz(theta).compose(hamiltonian_circuit)

        # Let´s run the simulation
        qc_trans = transpile(qc, sim)
        counts = sim.run(qc_trans, shots=shots).result().get_counts()      # The␣
↪results of the simulation are stored in the variable 'counts'

        # This calculates the probabilities for each term in the computational␣
↪basis
        probs = {}
        for output in ['00','01', '10', '11']:
            if output in counts:
                probs[output] = counts[output]/shots
            else:
                probs[output] = 0

        # This conditionals are the ones that calculate the mean value of each␣
↪term and that multiply
        # each of them by the correct coefficient of the Hamiltonian
        if measurement == Pauli('ZZ'):
            Energy_meas.append(float(R[index_param].real) *(probs['00'] -␣
↪probs['01'] - probs['10'] + probs['11']) )

        if measurement == Pauli('IZ'):
            Energy_meas.append(float(R[index_param].real) *(probs['00'] +␣
↪probs['01'] - probs['10'] - probs['11']) )

        if measurement == Pauli('ZI'):
            Energy_meas.append(float(R[index_param].real) *(probs['00'] -␣
↪probs['01'] + probs['10'] - probs['11']) )

        if measurement == Pauli('XX'):
            Energy_meas.append(float(R[index_param].real) *(probs['00'] -␣
↪probs['01'] - probs['10'] + probs['11']) )


    E_sim.append(np.sum(np.array(Energy_meas)))

    # The value that returns this function is the sum of the coefficients␣
↪multiplied by the expectation value
    # of each term, added to the coefficient that goes with the term II and to␣
↪the repulsion term.
    return E_sim[0] + float(R[0].real) + float(R[num_elements_hamiltonian].real)
```

This last cell executes the program. It includes the loop needed to go through all the distances we

want to consider. It also selects the backend. Here we get the final results, we obtain the bond length, the ground state energy and the angle that gives us the minimum of the energy function.

[ ]:
```
distances = np.arange(0.2, 3.0, 0.05)     # Array that defines the distances that␣
↪are going to be evaluated
theta = np.random.rand(1)                 # Initial value of the parameter of the␣
↪ansatz

# Some variables are defined
R = []
bond_length = []
energies = []
angles = []
gates_list_hamiltonian = []
gates_list_hamiltonian2 = []

# This loop determines the distances that are going to be evaluated
for dist in distances:
    (qubit_op, nuclear_repulsion_energy, num_particles, num_spin_orbitals,␣
↪problem, converter) = get_qubit_op(dist)

    # This lines define some variables that are going to be needed
    hamiltonian_matrix = qubit_op                                          #␣
↪Hamiltonian with its gates and coefficients
    nuclear_repulsion_term = nuclear_repulsion_energy
    coefficients_array = hamiltonian_matrix.coeffs                         # The next␣
↪two lines extract the coefficients from the Hamiltonian and store them in an␣
↪array
    coefficients_array = np.array(coefficients_array)
    R = np.append(coefficients_array, nuclear_repulsion_term)              # This is␣
↪an array consisting of the previous coefficients array and the repulsion term
    gates_list_hamiltonian = qubit_op.settings['primitive'].paulis         # This␣
↪extracts the Pauli strings that form the mapped Hamiltonian and stores them␣
↪into a list
    num_elements_hamiltonian = len(gates_list_hamiltonian)
    gates_list_hamiltonian2 = gates_list_hamiltonian[1:num_elements_hamiltonian]␣
↪    # The Pauli string consisting of II is removed from the previous list
    num_qubits = qubit_op.num_qubits

    # The optimization is performed, the function that calculates the mean value␣
↪of the Hamiltonian is called
    # and the desired optimizer is selected
    ret = minimize(Energy_general, theta, args = (R, num_qubits,␣
↪gates_list_hamiltonian2, num_elements_hamiltonian),method= 'COBYLA')

    # This stores the results of the optimization in the following variables
    val    = ret.fun
```

```
    ang = ret.x[0]

    # The results are stored in arrays in order to graphically represent them␣
↪later
    bond_length.append(dist)
    energies.append(val)
    angles.append(ang)

    # Let's print the results
    print('For the bond length ', dist, 'the ground state energy is',val)
    print('The minimum of the energy is reached when the ansatz takes the␣
↪parameter', ang)
```