COMPUTACIÓN CUÁNTICA APLICADA A LA QUÍMICA

QUANTUM COMPUTING APPLIED TO CHEMISTRY

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

TRABAJO FIN DE GRADO

CURSO 2023-2024

AUTOR

COLOMÁN SAMPRÓN GARCÍA

DIRECTORES

GUILLERMO BOTELLA JUAN

ALBERTO ANTONIO DEL BARRIO GARCÍA

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen (Español)

En los últimos años la computación cuántica se ha convertido en una de las áreas más prometedoras de la ciencia computacional. La química es la ciencia que se ocupa del estudio de la materia a nivel molecular y atómico, ciertos métodos computacionales clásicos han resultado de gran utilidad para el modelado y predicción del comportamiento de materiales. Pero estos métodos presentan grandes limitaciones en la resolución de problemas químicos complejos debido a la naturaleza exponencial de los mismos. Sin embargo, gracias a la computación cuántica, al aprovechar los principios de superposición, entrelazamiento y la capacidad de ejecutar múltiples cálculos simultáneamente, podemos emplear algoritmos que pueden resolver problemas químicos que serían imprácticos o imposibles para los computadores clásicos. Este trabajo propone ser una introducción a la computación cuántica orientada a la química, en él, resolveremos problemas químicos utilizando algoritmos cuánticos y ejecutaremos estos problemas tanto en simuladores con la ayuda de librerías orientadas a la computación cuántica como PennyLane.

Palabras clave. Computación cuántica, superposición, entrelazamiento, química cuántica, PennyLane, algoritmo VQE, Hamiltoniano molecular, algoritmo QPE.

Abstract (English)

In recent years quantum computing has become one of the most promising research areas in computational science. Chemistry is the science that deals with the study of matter at the molecular and atomic levels. Certain classical computational methods have been highly useful for modeling and predicting the behavior of materials. However, these methods have significant limitations in solving complex chemical problems due to their exponential nature. Nonetheless, thanks to quantum computing, by harnessing the principles of superposition, entanglement, and the ability to execute multiple calculations simultaneously, we can employ algorithms capable of solving chemical problems that would be impractical or impossible for classical computers. This thesis aims to provide an introduction to quantum computing focused on chemistry. We will solve chemical problems using quantum algorithms and run these problems simulators with the help of quantum computing libraries such as PennyLane.

Keywords Quantum computing, superposition, entanglement, quantum chemistry, PennyLane, VQE algorithm, molecular Hamiltonian, QPE algorithm.

1. Introducción

1.1. Breve historia de la computación cuántica y la química cuántica (español)

La mecánica cuántica ha sido tema de conversación para la comunidad científica durante poco más de un siglo. En los años ochenta del siglo pasado Paul Benioff definió por primera vez lo que es un computador cuántico, a través de una máquina de Turing teórica, y en el año 1982 Richard Feynman propuso una teoría más formal y estructurada para el uso de la mecánica cuántica en la resolución de problemas matemáticos mediante ordenador, así como las ventajas que estos algoritmos podían tener sobre métodos convencionales. Una vez sentadas las bases teóricas por los científicos que imaginaron esta tecnología David Deutsch en 1985 describió por primera vez cómo debía ser un computador cuántico. Sin embargo, todavía no estaba claro que un computador cuántico pudiese competir, a nivel de rendimiento, con un computador clásico.

En 1992 David Deutsch y Richard Jozsa propusieron un problema muy particular cuya solución clásica se ve exponencialmente mejorada mediante el uso de un algoritmo cuántico. Aunque esta solución se diese tan solo desde un marco teórico se podía empezar a intuir el potencial de la computación cuántica.

En 1994 Peter Shor teorizó un algoritmo cuántico capaz de factorizar enteros de manera eficiente. Un algoritmo que teóricamente podría romper el sistema criptográfico de clave pública RSA. Además, diseñó un sistema para la corrección de errores, uno de los puntos más débiles de la computación cuántica hasta nuestros días. A partir del algoritmo teorizado por Peter Shor la motivación para la investigación y el desarrollo de computadores cuánticos fue en aumento y poco después se diseñó el primer experimento cuántico con fines de comunicación y se programaron los primeros qubits. Primero se programó una máquina cuántica de 1 qubit del Instituto Tecnológico de Massachusetts, luego una de 2 qubits por parte de la Universidad de Berkeley y finalmente una de 3 qubits por parte de IBM-Almaden.

En la primera década del siglo XXI IBM se postuló como el líder en desarrollo de computación cuántica, en esta época también se crea el primer computador cuántico de 7 qubits en el Laboratorio Nacional de Los Álamos y se ejecuta por primera vez el algoritmo de Shor, en el año 2011 se vende por primera vez un computador cuántico fabricado por D-Wave Systems por 10 millones de dólares.

En los últimos años se ha producido una guerra por el desarrollo de los ordenadores cuánticos de tipo comercial y empresarial, así como los primeros prototipos de software cuántico. Este mercado fue tomado casi en su totalidad por IBM y D-Wave Systems. Ambos fueron mostrando procesadores estables cada vez más poderosos hasta que en el 2017 IBM toma la delantera con el procesador de 17 qubits, luego en 2019 con el IBM Q System One de 20 qbits y finalmente con el Osprey el 22 de septiembre del 2022.

En 2024 el desarrollo de la supremacía quántica, la cual expone la capacidad potencial de los dispositivos de computación cuántica para resolver problemas que los ordenadores clásicos prácticamente no pueden resolver, ha avanzado hacia una fase de "utilidad cuántica", donde las principales compañías buscan aplicaciones prácticas más allá de las pruebas conceptuales, IonQ por ejemplo, ha destacado por su enfoque en la escalabilidad y la modularidad, utilizando sistemas con qubits atrapados que según sus planes podrán alcanzar miles de qubits en el futuro cercano.

Por su parte la química cuántica es una rama de la química que aplica los principios de la mecánica cuántica para entender y predecir el comportamiento de los sistemas químicos. Junto con avances teóricos a principios del siglo XX, también surgen a finales del mismo, métodos computacionales que permiten realizar cálculos más complejos de estructuras y propiedades moleculares, como por ejemplo la aparición del método Hartree-Fock. El mayor problema que existe con estos métodos computacionales es que la complejidad de simular sistemas moleculares es muy alta. Simular con precisión las energías de atracción y repulsión de los elementos de los átomos y de los propios átomos dentro de una molécula con respecto a su geometría aumenta el tiempo de ejecución de forma exponencial a medida que se quiere simular moléculas más grandes y complejas. Propiedades físicas de los materiales como la polarización de la luz, el espín de los electrones o la corriente de anillos en superconductores son mucho más fáciles de representar y simular con qubits (2.2) y su representación geométrica la esfera de Bloch (2.9). Debido a esto podemos representar un electrón y todos sus posibles estados con un solo qubit, algo que en computación clásica nos requeriría más de un bit, es decir los recursos computacionales necesarios para simular moléculas son mucho menores en computadores cuánticos que en computadores clásicos. Debido a esto la utilización de la computación cuántica en este tipo de problemas puede ser de gran ayuda en la reducción del tiempo de ejecución de los mismos debido a algoritmos propios de la computación cuántica como VQE.

La química cuántica ha tenido un impacto profundo en la ciencia y la tecnología. Desde la comprensión de la estructura del ADN hasta el desarrollo de nuevos materiales y medicamentos y sigue evolucionando con el avance de la tecnología computacional y experimental.

1.2. Short history on quantum computing and quantum chemistry (English)

Quantum mechanics has been a topic of discussion for the scientific community for just over a century. In the 1980s, Paul Benioff first defined what a quantum computer is, using a theoretical Turing machine, and in 1982, Richard Feynman proposed a more formal and structured theory for using quantum mechanics to solve mathematical problems via computers, as well as the advantages these algorithms could have over conventional methods. Once the theoretical foundations were laid by the scientists who envisioned this technology, David Deutsch in 1985 described for the first time what a quantum computer should be like. However, it was still unclear whether a quantum computer could compete, in terms of performance, with a classical computer.

In 1992, David Deutsch and Richard Jozsa proposed a very specific problem whose classical solution is exponentially improved by the use of a quantum algorithm. Even though this solution was only theoretical, it began to hint at the potential of quantum computing.

In 1994, Peter Shor theorized a quantum algorithm capable of efficiently factoring integers, an algorithm that could theoretically break the RSA public-key cryptosystem. Additionally, he designed a system for error correction, one of the most significant weaknesses of quantum computing to this day. Following Peter Shor's algorithm, motivation for the research and development of quantum computers increased, and shortly after, the first quantum communication experiment was designed and the first qubits were programmed. First, a 1-qubit quantum machine was programmed by the Massachusetts Institute of Technology, followed by a 2-qubit machine by the University of California, Berkeley, and finally a 3-qubit machine by IBM-Almaden.

In the first decade of the 21st century, IBM emerged as the leader in quantum computing development. During this time, the first 7-qubit quantum computer was created at Los Alamos National Laboratory and Shor's algorithm was executed for the first time. In 2011, a quantum computer manufactured by D-Wave Systems was sold for the first time for 10 million dollars.

In recent years, there has been a race for the development of commercial and enterprise quantum computers, as well as the first quantum software prototypes. This market has been largely dominated by IBM and D-Wave Systems. Both companies have been demonstrating increasingly powerful and stable processors until IBM took the lead in 2017 with its 17-qubit

processor, then in 2019 with the IBM Q System One of 20 qubits, and finally with the Osprey on September 22, 2022.

By 2024, the development of quantum supremacy, which demonstrates the potential capability of quantum computing devices to solve problems that classical computers can practically not solve, has progressed to a phase of "quantum utility," where major companies are seeking practical applications beyond conceptual tests. For example, IonQ has stood out for its focus on scalability and modularity, using trapped qubit systems that, according to their plans, could reach thousands of qubits in the near future.

Quantum chemistry, for its part, is a branch of chemistry that applies the principles of quantum mechanics to understand and predict the behavior of chemical systems. Along with theoretical advances in the early 20th century, computational methods also emerged toward the end of the century, allowing for more complex calculations of molecular structures and properties, such as the development of the Hartree-Fock method. The major problem with these computational methods is the high complexity of simulating molecular systems. Accurately simulating the attraction and repulsion energies of atoms and their arrangement within a molecule in relation to its geometry exponentially increases the execution time as larger and more complex molecules are simulated. Physical properties of materials such as light polarization, electron spin, or ring currents in superconductors are much easier to represent and simulate with qubits (2.2) and their geometric representation, the Bloch sphere (2.9). Therefore, we can represent an electron and all its possible states with a single qubit, something that in classical computing would require more than one bit. This means that the computational resources needed to simulate molecules are much lower in quantum computers compared to classical computers. Consequently, using quantum computing for these types of problems can significantly reduce execution time due to quantum-specific algorithms like VQE. Quantum chemistry has had a profound impact on science and technology, from understanding DNA structure to developing new materials and medicines, and continues to evolve with advancements in computational and experimental technology.

2. Principios básicos de la computación cuántica

En este capítulo se introducirán los principios básicos de la computación cuántica, nos adentraremos en la teoría de la mecánica cuántica y explicaremos nociones básicas de matemáticas, más concretamente álgebra lineal. En primer lugar, presentaremos una notación muy usada en mecánica cuántica y que se utiliza a lo largo de este trabajo.

2.1. Notación de Dirac

La notación de Dirac, también conocida como notación bra-ket es la notación estándar para describir los estados cuánticos en la teoría de la mecánica cuántica. Esta notación también puede ser utilizada para denotar vectores abstractos en matemática pura. La notación $|\psi\rangle$ se denomina ket y representa el vector ψ de un espacio vectorial complejo como columna. Mientras que la notación $\langle\psi|$ se denomina bra y representa el conjugado de ψ cómo fila, así tenemos que $\langle\psi'|\psi\rangle$ denota el producto escalar dado por la siguiente ecuación:

$$\langle\psi,\psi'\rangle = \sum_{i=1}^{n} \psi_i \overline{\psi'_i} \tag{2.1}$$

Sin embargo, la notación $|\psi\rangle\langle\psi|$ denota el producto exterior. El resultado de este producto es una matriz de dimensiones *n* X *n* donde *n* es la dimensión del espacio complejo donde habita ψ.

2.2. Qubits

Para conocer los fundamentos de la computación cuántica primero debemos definir qué es un qubit. Un qubit es la unidad básica de información en la computación cuántica (del mismo modo que un bit es la unidad básica de información en computación clásica). Un qubit representa un espacio de Hilbert de dos dimensiones de números complejos $\mathbb{C}^2$. Los vectores de la base del espacio cuántico se denotan como $\{|0\rangle, |1\rangle\}$. A los que nos referiremos como los estados base:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \tag{2.2}$$

Un estado general de un solo qubit se describe como una superposición de las bases computacionales:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix} \in \mathbb{C}^2 \tag{2.3}$$

Donde α y β son coeficientes que satisfacen:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.4}$$

Aunque el qubit sea una superposición quántica mientras se ejecuta cualquier algoritmo, cuando se mide en la base computacional, se encontrará en el estado $|0\rangle$ o en el estado $|1\rangle$, no en una superposición. Estas mediciones *colapsan* a los estados $|0\rangle$ o $|1\rangle$ con

probabilidad $|\alpha|^2$ y $|\beta|^2$ respectivamente. Si hay n qubits en el sistema, el estado se describe por un vector en el espacio dimensional 2^n de Hilbert $(\mathbb{C}^2)^{\otimes n}$ formado tomando el producto tensorial de los espacios de Hilbert de los qubits individuales. Por ejemplo, para 10 qubits, el estado se describe por un vector en un espacio de Hilbert 1024-dimensional.

2.3. Orden tensorial de qubits

La comunidad de la física normalmente ordena el producto tensorial de n qubits con el qubit de menor nivel a la izquierda del producto tensorial:

$$|q\rangle = |q_0\rangle|q_1\rangle \dots |q_{n-1}\rangle = |q_0, q_1, \dots, q_{n-1}\rangle = \bigotimes_{i=0}^{n-1} |q_i\rangle \tag{2.5}$$

Sin embargo, muchos lenguajes de programación o librerías orientadas a la programación de computadores cuánticos (como Qiskit) utilizan el orden contrario:

$$|q\rangle = |q_{n-1}\rangle \dots |q_1\rangle|q_0\rangle = |q_{n-1}, \dots, q_1, q_0\rangle = \bigotimes_{i=n-1}^{0} |q_i\rangle \tag{2.6}$$

En otras palabras, si el qubit 0 está en el estado $|0\rangle$, el qubit 1 está en el estado $|0\rangle$, y el qubit 2 está en el estado $|1\rangle$, muchos libros de física representarían este estado como $|001\rangle$, mientras que en herramientas como Qiskit este estado se representaría como $|100\rangle$. Las diferencias en representación son importantes ya que afectan la manera en la que operaciones de varios qubits son representadas como matrices.

2.4. Entrelazamiento cuántico

Un sistema cuántico está entrelazado cuándo su estado cuántico no puede ser factorizado como un producto tensorial de sus constituyentes. Los estados pueden ser clasificados como un producto tensorial de partículas en sus estados base o un estado entrelazado:

- Los estados que son producto tensorial de partículas en estado base se pueden descomponer en productos tensoriales de menos qubits, por ejemplo:

$$\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.7}$$

- Los estados entrelazados no pueden descomponerse a un producto tensorial de estados. Por ejemplo, el estado de Bell:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.8}$$

Está entrelazado y solo puede medirse o en el estado $|00\rangle$ o en el estado $|11\rangle$, cada uno con una probabilidad de ½.

2.5. Definición de la esfera de Bloch

La esfera de Bloch describe un qubit en el espacio y es una forma específica de coordenadas *(figura 1)*. El vector $|\psi\rangle$, o la longitud de un qubit siempre es igual a 1, cualquier punto de la esfera de Bloch es un qubit que se puede expresar como:

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle \tag{2.9}$$

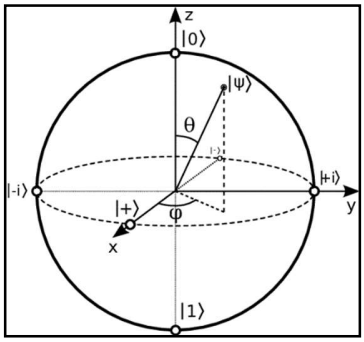


Figura 1. Esfera de Bloch: se ven los diferentes ejes de la esfera y los valores asociados a cada eje

Por ejemplo:

- Cuando α = 1 y β = 0, esto es el estado $|0\rangle$, se representa con un vector hacia el *norte* de la esfera siguiendo el eje de coordenadas z.
- Cuando α = 0 y β = 1, esto es el estado $|1\rangle$, se representa con un vector hacia el *sur* de la esfera siguiendo el eje de coordenadas z.

2.6. Las matrices de Pauli

Las matrices de Pauli son matrices fundamentales en física cuántica, ampliamente utilizadas en el contexto del momento angular intrínseco o espín, pero también tienen un rol crucial en la computación cuántica, donde representan operaciones cuánticas básicas en los qubits, en los circuitos cuánticos las matrices de Pauli se utilizan para describir tres de las puertas cuánticas más importantes (2.10), que permiten manipular el estado de los qubits.

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (2.10)$$

Las matrices de Pauli se utilizan como puertas cuánticas en los algoritmos cuánticos para realizar rotaciones sobre los ejes x, y, z de la esfera de Bloch, que es la representación geométrica de los qubits. Estas operaciones son esenciales para el control de qubits en sistemas de computación cuántica. Las matrices de Pauli son Hermíticas y Unitarias (2.11), es decir, las matrices elevadas al cuadrado dan como resultado la matriz Identidad 2 X 2:

$$\sigma_x^2 = \sigma_y^2 = \sigma_z^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (2.11)$$

Cada una de las matrices de Pauli es también igual a su inversa (2.12):

$$\begin{aligned} \sigma_x &= \sigma_x^{-1} \\ \sigma_y &= \sigma_y^{-1} \\ \sigma_z &= \sigma_z^{-1} \end{aligned} \quad (2.12)$$

2.7. Puertas cuánticas

Las puertas cuánticas son operadores unitarios ($U^\dagger U = UU^\dagger = \mathbb{1}$) que actúan sobre uno, dos o incluso tres qubits. La acción de la puerta cuántica sobre un estado cuántico corresponde a la multiplicación de la matriz que representa a la puerta con el vector que representa el estado cuántico: $U|q\rangle$. Dentro de las puertas cuánticas de un qubit, aparte de las matrices de Pauli $\sigma_x, \sigma_y, \sigma_z$ (2.10), cabe destacar la puerta de Hadamard o puerta H (2.13), con esta puerta conseguimos poner un qubit en superposición:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (2.13)$$

La puerta de Hadamard nos mapea el estado $|0\rangle$ al estado $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, también llamado estado $|+\rangle$ y el estado $|1\rangle$ al estado $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, también llamado estado $|-\rangle$.

Dentro de las puertas de dos qubits las más importantes son la puerta Controlled-not o CX (2.14) y la puerta SWAP (2.15).

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.14)$$

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.15}$$

También podemos tener puertas de tres qubits como la puerta Toffoli (también llamada CCX) o la puerta Controlled-Swap (CSWAP).

2.8. Circuitos cuánticos

Un circuito cuántico es una serie de instrucciones, puertas cuánticas, medidas y resets que se aplican a registros de qubits y pueden ser condicionados por computación clásica en tiempo real.

Se representan como una lista de qubits en vertical de los que sale de cada uno un "cable" en horizontal sobre el cual aparecen las puertas que van a transformar ese qubit. Las puertas que afectan a dos o tres qubits (como por ejemplo la CNOT), se aparecen con un punto sobre el cable que "sale" del qubit de control y otro "cable" en vertical hasta la puerta que afecta al qubit objetivo. (*Figuras 2 y 3).*

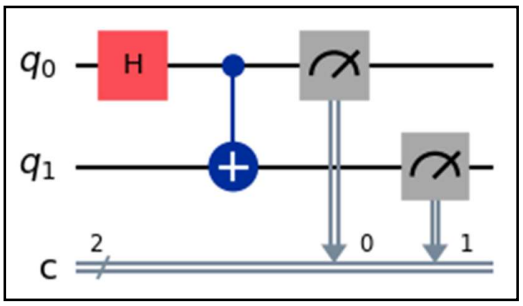


Figura 3. Circuito cuántico que crea un estado de Bell

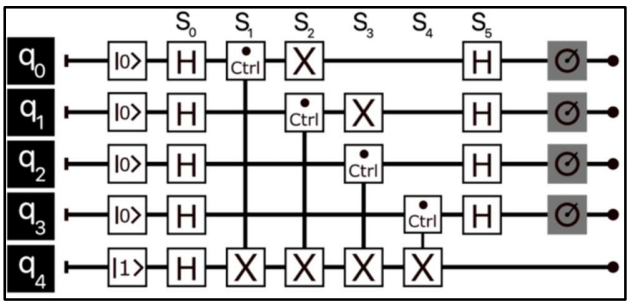


Figura 2. Circuito cuántico que efectúa el algoritmo de Deutsch-Jozsa

3. Contextualización del problema

En el siguiente apartado se expone una definición de la química cuántica y su influencia en diferentes campos e industrias, así como las ventajas de emplear computación cuántica en los problemas asociados con la misma.

3.1. Química cuántica

La química cuántica estudia el comportamiento de los átomos y las moléculas utilizando los principios y herramientas de la mecánica cuántica. Se basa en el concepto fundamental de que las partículas subatómicas, como electrones y protones, exhiben tanto propiedades de partícula como de onda, y su comportamiento no puede describirse completamente utilizando las leyes clásicas de la física. Los fundamentos de la mecánica cuántica y sus cinco postulados afectan directamente a la investigación de materiales y a la química computacional que se utiliza para, por ejemplo, encontrar nuevos medicamentos o permitir procesos más limpios y eficientes para transformar unas sustancias químicas en otras. La química cuántica ha sido fundamental para el desarrollo de la química moderna y ha permitido comprender fenómenos químicos a un nivel más profundo. Además, ha proporcionado herramientas para diseñar nuevos materiales y compuestos con propiedades específicas, así como para comprender y predecir el comportamiento de sistemas químicos complejos. (Keeper L. Sharkey; Alain Chancé 2022).

3.2. Ventajas de emplear computación cuántica

La investigación en química cuántica puede ser utilizada para calcular la evolución en el tiempo de un sistema de moléculas complejo, estimar el estado de mínima energía de una molécula o determinar la estructura de la banda electrónica de un material exótico. Como los computadores cuánticos tienen la misma naturaleza que estos problemas nos ofrecen una ventaja computacional potencial a la hora de simular las propiedades cuánticas de la materia.

En el caso de la simulación de moléculas complejas, la computación cuántica es capaz de simular estructuras moleculares y reacciones químicas con más precisión que los métodos clásicos. Los algoritmos cuánticos como el Algoritmo VQE (figura 10) permiten una representación más eficiente de los sistemas moleculares, reduciendo significativamente el tiempo necesario para obtener resultados precisos. También se necesita un menor número de qubits en comparación con los bits clásicos, los qubits pueden representar la superposición de estados. Esto significa una reducción en la cantidad de recursos computacionales necesarios para resolver problemas de química computacional, además, la capacidad de los qubits para representar la superposición y el entrelazamiento permite una descripción mejor de las moléculas, lo que mejora la exactitud de las simulaciones.

4. Entorno de trabajo

En el siguiente apartado se explica como emplear el entorno de trabajo utilizado para las simulaciones, así como la distinta funcionalidad que tiene el mismo.

4.1. PennyLane Quantum

PennyLane Quantum es una librería Python utilizada para programar computadores cuánticos. PennyLane permite la ejecución y el entrenamiento de programas cuánticos en varios backends.

El trabajo central de PennyLane es manejar la ejecución de programas de computación cuántica. El principio de diseño de PennyLane se basa en circuitos que pueden ser ejecutados en varios tipos de simuladores o dispositivos hardware sin tener que hacer cambios, los trabajos de optimización, comunicación con los dispositivos, compilación de circuitos para adaptarse al backend y elegir estrategias de gradiente se pueden hacer con funciones de la librería. La librería viene con dispositivos de simulación por defecto, pero está bien integrada con software y hardware externo para ejecutar circuitos cuánticos, como, por ejemplo, Qiskit de IBM, Cirq de Google, Forest de Rigetti o Strawberry Fields de Xanadu.

4.2. Método de empleo de PennyLane Quantum

En PennyLane las computaciones que necesiten la ejecución de uno o más circuitos cuánticos se representan como objetos *quantum node.* Un *quantum node* se utiliza para declarar el circuito cuántico y también enlaza la computación a un dispositivo específico que la ejecuta. *(Figura 4)*.

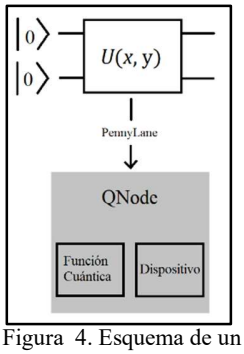


Figura 4. Esquema de un Qnode en PennyLane

Los *Quantum nodes* pueden interactuar con cualquiera de las librerías soportadas de machine learning o numéricas como NumPy, PyTorch, TensorFlow o JAX. Por defecto los *Quantum nodes* usan la interfaz NumPy.

Un circuito cuántico se construye como una función en Python *(figura 5)*.

```
import pennylane as qml

def circuito_sencillo(x, y):
    qml.RZ(x, cables=0) # Añado puerta Z al qubit 0
    qml.CNOT(cables=[0,1]) #Añado puerta CNOT a los qubits 0 y 1
    qml.RY(y, cables=1)#Añado puerta Y al qubit 1
    return qml.expval(qml.PauliZ(1))
```

Figura 5. Ejemplo de un circuito cuántico sencillo codificado en PennyLane que aplica una puerta Z sobre el qubit 0 (2.10), una CNOT sobre los qubits 0 y 1 (2.14), y una puerta Y sobre el qubit 1 (2.10).

En PennyLane se utiliza el término *wires* (cables) para referirnos a uno de los qubits. Las funciones cuánticas son un subconjunto de funciones Python, las funciones aceptan *inputs* clásicos y consisten en un conjunto de operadores cuánticos (puertas, medidas…). Las funciones pueden incluir estructuras de flujo clásicas como, por ejemplo, bucles for e instrucciones if. Una función cuántica siempre debe devolver un valor de medida o una tupla de valores de medida aplicando una función de medida al registro del qubit. Las funciones cuánticas son evaluadas en un dispositivo dentro de un *quantum node.*

Para ejecutar un circuito cuántico primero necesitamos especificar un dispositivo computacional. El dispositivo es una instancia de la clase *device* y puede representar un simulador o un dispositivo hardware. *(Figura 6).* Cuando creamos un dispositivo el nombre del dispositivo siempre debe de estar especificado y las especificaciones del dispositivo se pasan como argumentos, los dos argumentos más importantes de un dispositivo son los *wires* y los *shots*. El argumento *wires* es un número entero que nos indica el número de *wires* del dispositivo. Los *shots* son un número entero que nos indica cuantas veces debe ser evaluado nuestro circuito para estimar cantidades estadísticas.

Juntos una función cuántica y un dispositivo se utilizan para crear un *quantum node* que enlaza la función cuántica a un dispositivo crear un *quantum node* que enlace la función al dispositivo tal y como se muestra en la figura 7.

```
dispositivo = qml.device('default.qubit', cables = 2, iteraciones = 1000)
```

Figura 6. Creación de un dispositivo en PennyLane con 2 qubits

```
circuito = qml.QNode(circuito_sencillo, dispositivo)
```

Figura 7. Creación de un QNode en PennyLane con el dispositivo de la figura 6 y el circuito de la figura 5

El QNode puede ser utilizado para computar el resultado de un circuito cuántico como si fuese una función de python standard. Por ejemplo, en el caso de la figura 7 llamaríamos a nuestro circuito con los mismos argumentos que la función cuántica original

```
dispositivo = qml.device('default.qubit', cables = 2, iteraciones = 1000)

@qml.qnode(dispositivo)
def circuito_sencillo(x, y):
    qml.RZ(x, cables=0) # Añado puerta Z al qubit 0
    qml.CNOT(cables=[0,1]) #Añado puerta CNOT a los qubits 0 y 1
    qml.RY(y, cables=1)#Añado puerta Y al qubit 1
    return qml.expval(qml.PauliZ(1))

resultado = circuito_sencillo(0.543)
```

Figura 8. Ejemplo de código de un programa simple en PennyLane que emplea los ejemplos de las figuras 5, 6 y 7.

5. Algoritmos Cuánticos empleados para simular

Los algoritmos cuánticos son algoritmos diseñados para ser ejecutados en computadores cuánticos, aprovechando las propiedades únicas de la mecánica cuántica para realizar ciertos cálculos de manera más eficiente que las computadoras clásicas.

5.1. Hamiltoniano molecular

Para la simulación cuántica debemos entender que es un Hamiltoniano molecular. Un Hamiltoniano molecular es el nombre dado al operador al operador Hamiltoniano que representa la energía del sistema constituido por los electrones y el conjunto de núcleos de la molécula. Este es un operador autoadjunto, es decir hermítico, cuya ecuación de Schrödinger asociada juega un papel central en la química computacional para calcular propiedades de moléculas. Dado el operador Hamiltoniano H de un sistema cuántico, y su estado actual $|\psi_0\rangle$, el estado $|\psi_t\rangle$ después de un tiempo t viene dado por la solución de la ecuación de Schrödinger:

$$|\psi_t\rangle = e^{-iHt}|\psi_0\rangle \tag{5.1}$$

Se puede dar el caso de que el sistema para el que buscamos una solución sea independiente del tiempo. En estos casos tanto los orbitales atómicos de los electrones, que nos indican la región del espacio donde es más probable encontrar un electrón, son soluciones a la ecuación de Schrödinger para átomos en la mecánica cuántica y representan las áreas donde la densidad de probabilidad de encontrar un electrón es alta. En este tipo de casos la ecuación de Schrödinger independiente del tiempo cuenta con la información necesaria acerca del sistema, de manera que la ecuación a resolver es:

$$H|\psi\rangle = E_G|\psi\rangle \tag{5.2}$$

Donde E es la energía asociada a ese estado cuántico particular. Este valor se conoce como autovalor del Hamiltoniano. El operador Hamiltoniano actúa sobre el estado y genera como resultado el valor de la energía, E multiplicado por el estado.

5.2. Aproximación Born-Oppenheimer

La aproximación Born-Oppenheimer es una técnica fundamental en química cuántica que simplifica el tratamiento de sistemas moleculares al desacoplar (tratar por separado) los movimientos electrónicos de los movimientos nucleares. (debido a que el núcleo es mucho más pesado que el electrón y por ende su velocidad es mucho más lenta en comparación) (Born & Oppenheimer 1927). Si consideramos lo dicho anteriormente, entonces el Hamiltoniano que describe el movimiento de *N* electrones que se mueven dentro del campo de *M* cargas puntuales viene dado por:

$$H = -\sum_{i=1}^{N}\frac{1}{2}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j>1}^{N}\frac{1}{r_{ij}} \tag{5.3}$$

Donde Z_A corresponde al número atómico del núcleo A, la ecuación (5.4) corresponde a las distancias entre el *i*th electrón y *A*th núcleo, mientras que la ecuación (5.5) corresponde a las distancias entre el *i*th electrón y el *j*th electrón.

$$r_{iA} = |r_i - R_A| \tag{5.4}$$

$$r_{ij} = |r_i - r_j| \tag{5.5}$$

En la práctica, la mayoría de software orientado a la química computacional emplea la aproximación de BO ya que al desacoplar el problema en dos partes y no tener en cuenta las interacciones entre núcleos y electrones reduce la complejidad computacional en comparación a si tuviésemos que resolver la ecuación de Schrödinger para el sistema completo. En nuestro caso también la utilizamos para calcular el Hamiltoniano de las moléculas.

5.3. Estimación cuántica de fase (QPE)

El algoritmo Quantum Phase Estimation (QPE) es uno de los algoritmos que puede ser empleado en la simulación cuántica de problemas de química cuántica. Los algoritmos de estimación de fase operan tal que, dado un operador unitario *U*, un autovector $|\psi\rangle$, y un número de qubits de precisión *p*, nos permite extraer la fase ϕ:

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle \tag{5.6}$$

Para la implementación del algoritmo QPE se requieren $2^p - 1$ aplicaciones consecutivas del operador unitario *U* (basado en la precisión deseada) requiriendo un total de 2^n qubits, donde *n* corresponde al número de orbitales de spin. De esta manera el algoritmo de estimación cuántica de fase para la simulación cuántica puede ser generalizado tal que:

1. El Hamiltoniano del sistema H_S se codifica en un Hamiltoniano de qubits H_q.
2. Calculamos el operador unitario $U = e^{-iHt}$.
3. Consideramos un estado de entrada $|\psi\rangle$, expresado como la superposición de los autovectores del operador unitario *U*:

$$|\psi\rangle = \sum_i a_i|u_i\rangle \tag{5.7}$$

4. Inicializamos un primer registro $|0\rangle^n$.
5. Inicializamos un segundo registro empleando un ansatz.
6. Colocamos el primer registro en superposición.
7. Aplicamos el operador unitario controlado *U*, tantas veces como sea necesario.
8. Aplicamos Inverse Quantum Fourier Transform (QFT^{-1}) sobre el primer registro de qubits.

Si queremos poner el algoritmo en pseudocódigo quedaría de la siguiente manera:

Procedimiento para simular ($e^{-iHt}|\psi\rangle = e^{-iE_\psi t}|\psi\rangle$)

$|m\rangle \leftarrow |0\rangle^{\otimes p}$

$|n\rangle \leftarrow |\psi\rangle$

Para $i = 0; i < 2^p - 1; i += 1$ ***hacer***

$|n_{i+1}\rangle \leftarrow U|n_i\rangle$

Terminar para

$QFT_{2^p}^{-1}|m\rangle$

$p(a_z) = \langle m|P_z|m\rangle$

Terminar Procedimiento

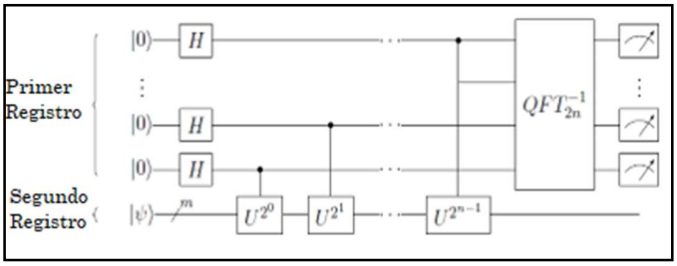


Figura 9 Circuito del algoritmo QPE

5.4. Variational Quantum Eigensolver (VQE)

Los algoritmos variacionales corresponden a un tipo de algoritmos híbridos clásico-cuánticos que hacen uso de técnicas de optimización clásica y computación cuántica. El objetivo de este tipo de algoritmos se centra en encontrar la mejor solución dentro de un conjunto de estados prueba parametrizados a los que se conoce como *ansatz*. La idea principal de este tipo de algoritmos es preparar un estado cuántico que dependa de un grupo de parámetros, para luego minimizar una determinada función de coste a través del ajuste de dichos parámetros. La ventaja de este tipo de algoritmos es su adaptabilidad y robustez en comparación con algoritmos como el QPE en dispositivos ruidosos.

El algoritmo VQE está basado en el teorema de Rayleigh-Ritz. De acuerdo con este teorema podemos hallar una cota superior $E_\psi \geq E_0$ para el valor esperado de H de un ansatz. De esta manera un estado de prueba $|\psi\rangle$ y un operador H, la energía esperada de ese estado viene dada por:

$$E_\psi = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle} \tag{5.8}$$

Por consiguiente, cuando la aproximación al estado fundamental $|\psi\rangle$ depende de un parámetro de ajuste θ que nos permita minimizar la función de energía E_ψ, podemos en efecto obtener una cota superior para E_0. Si la aproximación del estado fundamental es suficientemente buena, el valor de la cota converge mucho mejor con el valor exacto de la energía del estado base.

Siguiendo esta idea entonces se puede establecer un algoritmo que, al iterar sobre este principio, aproxime a una solución al problema de estados base y estados excitados. De esta manera el algoritmo VQE puede establecerse en pseudocódigo tal que:

Requiere: Tolerancia ϵ

$|\psi_i(\vec{\theta})\rangle \leftarrow |0\rangle^{\otimes n}$

Mientras $|E_{i+1} - E_i| \geq \epsilon$ ***hacer***

$|\psi_{i+1}(\vec{\theta})\rangle \leftarrow U(\vec{\theta})|\psi_i\rangle$ (actualizar estado cuántico)

$E(\vec{\theta}) \leftarrow \langle\psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle$ (calcular energía)

$\vec{\theta}_{i+1} \leftarrow min(\partial E/\partial\vec{\theta}_i)$ (optimizar parámetros)

Terminar mientras

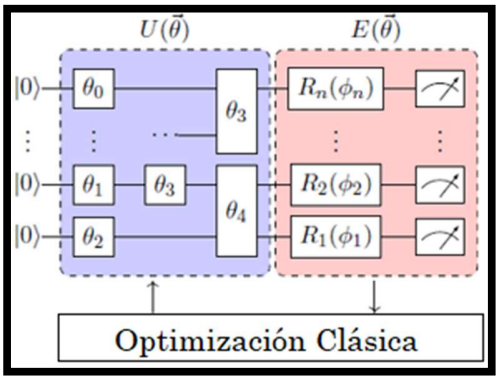
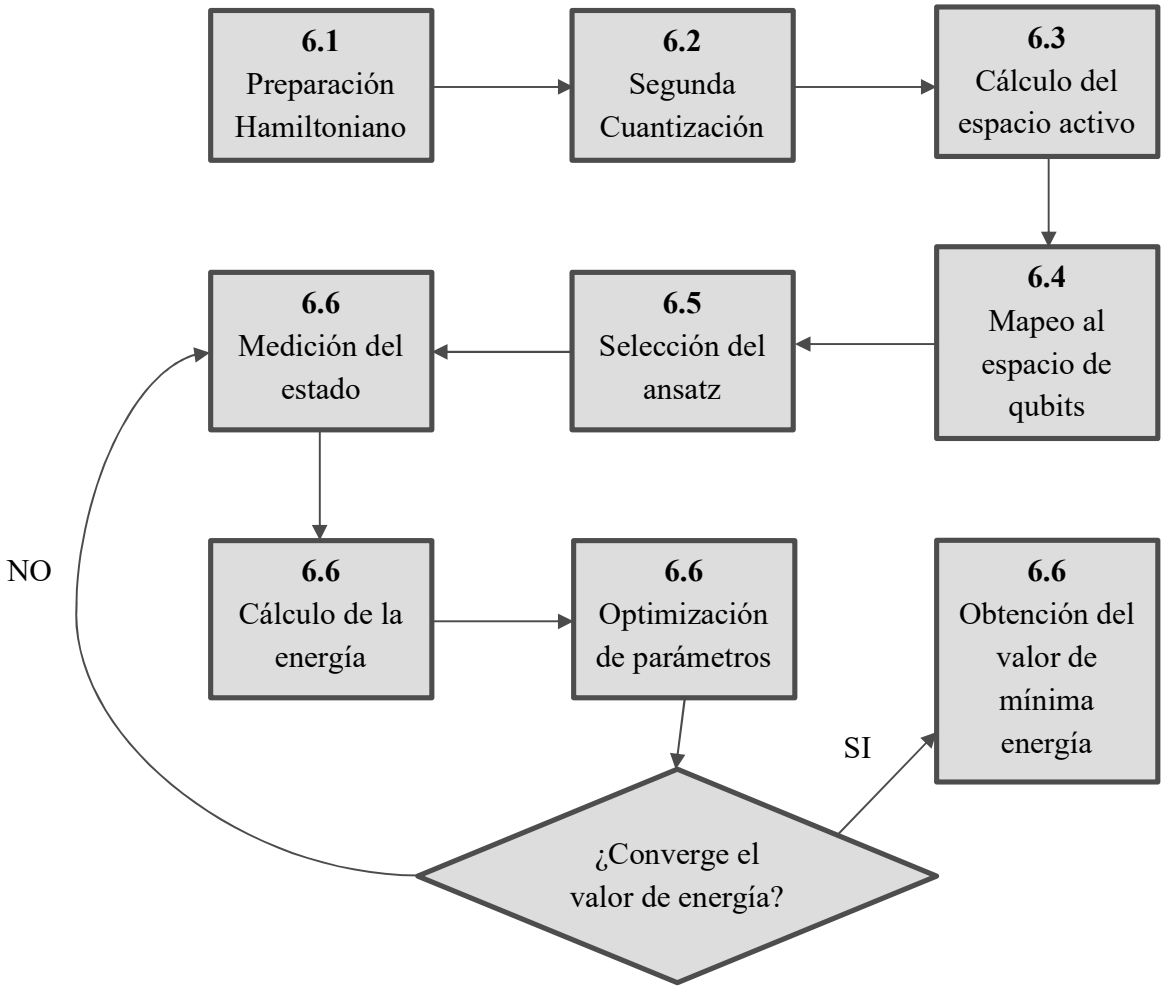


Figura 10. Algoritmo VQE

6. Metodología

En la siguiente sección se explicarán los pasos seguidos en nuestras simulaciones para calcular el Hamiltoniano asociado a las moléculas y ejecutar el algoritmo VQE para las mismas calculando así su estado de mínima energía. A continuación, se describe el diagrama de flujo que debemos seguir:



6.1. Preparación del Hamiltoniano molecular

La aproximación Born-Oppenheimer (5.3) es empleada para describir el Hamiltoniano correspondiente a un sistema atómico o molecular. La aproximación considera un conjunto de fermiones que tienen interacciones electrón-electrón, y electrón-

núcleo en proporción a las distancias que existen entre las partículas. Además, se considera que los núcleos son cargas puntuales que se comportan clásicamente.

Cuando planteamos el Hamiltoniano que describe nuestro sistema tenemos por objetivo, encontrar una descripción matemática lo suficientemente buena de nuestro sistema tal que nos permita encontrar con cierta precisión los auto estados de energía $|E_i\rangle$ y sus auto valores correspondientes E_i encapsulados dentro del Hamiltoniano electrónico H_e.

En este caso el Hamiltoniano de cada sistema estudiado es obtenido computacionalmente a través del uso del paquete de química cuántica, PySCF (Sun et al. 2017), basado en Python. Para cada sistema proponemos una geometría que describe las posiciones de cada uno de los átomos que conforman la molécula.

6.2. Segunda Cuantización

Una vez que se ha obtenido el Hamiltoniano que describe el sistema que deseamos estudiar el siguiente paso es transformarlo a la formulación de la segunda cuantización (Altland & Simons 2010).

Esta formulación es de gran utilidad cuando se trabaja con sistemas de muchas partículas, ya que facilita la descripción matemática de los procesos de interacción entre varias partículas. En este caso en lugar de enfocarnos en las propiedades individuales de las partículas, esta teoría emplea operadores de campo, que nos permiten describir la distribución de las partículas en el espacio y en el tiempo.

$$H = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \tag{6.1}$$

Donde h_{pq}, h_{pqrs} corresponden a los términos de interacción de uno y dos cuerpos respectivamente, y a, $a^\dagger$ corresponden a los operadores de creación y aniquilación que se describen en la teoría de segunda cuantización.

Cuando estamos preparando nuestro Hamiltoniano es importante tener en cuenta que los electrones que forman parte de nuestro sistema obedecen a ciertas reglas que deben ser tomadas en cuenta. Es así que necesitamos emplear los determinantes de Slater, que nos proveen de una herramienta matemática para construir una función de onda para un sistema con varios electrones. De esta manera los determinantes de Slater se construyen como elementos de una matriz, tal que:

$$|\psi\rangle_n = \frac{1}{\sqrt{n!}} \begin{vmatrix} |\chi_1\rangle_1 & \cdots & |\chi_1\rangle_n \\ \vdots & \ddots & \vdots \\ |\chi_n\rangle_1 & \cdots & |\chi_n\rangle_n \end{vmatrix} \tag{6.2}$$

Donde $|\chi_j\rangle_j$ representa un orbital de espín específico, y $\chi_j \in \{0, 1\}$ representa la ocupación de dicho orbital. El uso de los estados generados por los determinantes de Slater da lugar a la generación de un subespacio conocido como el espacio de Fock (Fock 1932).

Al aplicar la segunda cuantización empleando los determinantes de Slater para construir nuestro sistema, los elementos h_{pq}, h_{pqrs} vienen dados por las integrales de uno y dos cuerpos tal que:

$$h_{pq} = \langle \chi_p | T_e + V_{ne} | \chi_q \rangle = \int d\vec{x} \chi_p^*(\vec{x}) \left(-\frac{1}{2}\nabla^2 - \sum_A \frac{Z_A}{|R_A - r_i|} \chi_q(\vec{x}) \right) \tag{6.3}$$

$$h_{pqrs} = \langle \chi_p \chi_q | V_{ee} | \chi_r \chi_s \rangle = \int d\vec{x}_1 d\vec{x}_2 \frac{\chi_p^*(\vec{x}_1)\chi_q^*(\vec{x}_2)\chi_r(\vec{x}_2)\chi_s(\vec{x}_1)}{|r_1 - r_2|} \tag{6.4}$$

Los mismos que pueden ser calculados clásicamente empleando métodos numéricos. En este caso el operador correspondiente al Hamiltoniano de nuestro sistema molecular es calculado clásicamente empleando las herramientas del paquete de PySCF.

6.3. Cálculo del espacio activo

Cuando estamos realizando cálculos de química cuántica es importante tener en cuenta las dimensiones del espacio de Fock correspondiente a nuestro sistema, esto debido a que el poder computacional necesario para realizar cálculos crecerá exponencialmente a medida que se añadan dimensiones al espacio.

Para conservar los recursos necesarios dentro de cierto límite que nos permita realizar cálculos de manera más eficiente, es importante conservar el número de qubits necesarios para calcular nuestro sistema, al mínimo posible. Esto es posible gracias al uso de lo que se conoce como espacio activo. El espacio activo se considera como un subconjunto de los orbitales y electrones moleculares que pueden ser considerados relevantes para describir dicho sistema de manera precisa debido a que son los orbitales entre los que se pueden mover los electrones de la molécula.

En nuestro caso una vez que consideramos un determinado espacio activo, se considera que aquellas correlacionas seleccionadas determinaran el estado del sistema. De esta manera tendremos un conjunto de orbitales y electrones que consideramos no varían

en el tiempo y espacio, mientras que aquellos que forman el espacio activo serán simulados para determinar las características del sistema molecular seleccionado.

Tanto el entorno de Qiskit como el de Pennylane (que es el que estamos empleando) nos permiten seleccionar el número de electrones y orbitales que estamos tomando en cuenta para realizar nuestros cálculos. Por lo tanto, al momento de generar el Hamiltoniano perteneciente al sistema la información correspondiente al uso de un espacio completo, que toma en cuenta todas las interacciones, o un espacio activo, debe ser especificado.

6.4. Mapeo al espacio de qubits

Los computadores cuánticos actuales, que emplean puertas cuánticas utilizan en general un número limitado de puertas lógicas que operan en uno y dos qubits. Estas puertas lógicas consisten en una serie de pulsos de determinada duración que son las que permiten implementar el uso de puertas cuánticas. Por ello cuando queremos realizar operaciones en un computador cuántico es necesario codificar las operaciones necesarias para realizar el procedimiento en un conjunto sucesivo de puertas lógicas cuánticas que nos den como resultado una manipulación sobre el estado inicializado en el inicio del proceso.

Tal como se observa en los pasos anteriores, tras realizar dicha serie de procedimientos, obtenemos como resultado un Hamiltoniano. Para poder realizar el computo deseado es necesario entonces llevar nuestro problema a una base en la que las puertas empleadas para manipular los qubits funcionen también como operadores que medien la interacción entre los orbitales que conforman el sistema molecular. Dicho procedimiento se conoce como mapeo, y consiste básicamente en un cambio de base que se realiza sobre el Hamiltoniano que describe el sistema. Nosotros utilizaremos el mapeo de Jordan-Wigner, aunque existen otros mapeos como los de Bravyi y Kitaev.

Mapeo de Jordan-Wigner

Es una transformación matemática que nos permite almacenar la ocupación de cada orbital de espín en un qubit. Para ello si tomamos en cuenta que en la base computacional la desocupación de un orbital de espín puede representarse por $|0\rangle_k$ y la ocupación del mismo viene dada por $|1\rangle_k$, podemos expresar los operadores de creación y aniquilación tal que:

$$a_k|0\rangle_k = |1\rangle_k \tag{6.5}$$

$$a_k^\dagger |1\rangle_k = |0\rangle_k \tag{6.6}$$

Podemos entonces deducir la forma que posee cada uno de estos operadores y expresarlo utilizando los operadores de Pauli, de manera que podemos expresar a los operadores de creación y aniquilación tal que:

$$a_k = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \frac{X_k - iY_k}{2} = \sigma_k^- \tag{6.7}$$

$$a_k^\dagger = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \frac{X_k + iY_k}{2} = \sigma_k^+ \tag{6.8}$$

De manera que ahora tenemos nuestros operadores expresados en matrices unitarias que pueden ser implementadas en un computador cuántico.

En realidad, el mapeo Jordan-Wigner lo único que representa es en que orbital están los electrones y el espín que tienen. Imaginemos, por ejemplo, una molécula con tres orbitales activos y dos electrones, los dos primeros qubits se utilizan para representar el orbital más cercano al núcleo (el de menor energía), los dos siguientes el segundo más alejado, y los dos siguientes el más alejado. Si consideramos que en cada pareja de qubits que representan un orbital el primero de ellos representa un electrón con espín positivo y el segundo un electrón con espín negativo, el siguiente mapeo $|100100\rangle$ representaría que uno de los electrones está en el orbital de menor energía con espín positivo, y el otro está en el segundo orbital con espín negativo.

6.5. Selección del Ansatz

En el contexto del algoritmo VQE, un ”ansatz” se refiere a un circuito cuántico parametrizado. El objetivo del ansatz es generar un aproximado del estado base del sistema, y preparar el estado que será empleado en el proceso de iteraciones del algoritmo que optimizarán los parámetros del circuito. La generación del circuito cuántico dependerá de varios factores, entre ellos: el tipo de arquitectura seleccionada, el tipo de puertas seleccionadas para realizar las operaciones de uno y dos qubits, la conectividad entre qubits y el hardware cuántico disponible.

Tanto Qiskit como Pennylane poseen funciones que pueden ser empleadas para realizar la generación de este circuito de forma automática y eficiente. Una vez generado el Hamiltoniano que describe el sistema, la función utilizará la información referente al número de partículas y orbitales en conjunto con una técnica de mapeo seleccionada (en nuestro caso Jordan-Wigner) para generar el circuito correspondiente.

Es importante notar que la elección de un ansatz puede afectar significativamente la eficiencia y precisión del algoritmo de VQE. El diseño de un ansatz adecuado es en general un problema que debe ser lo suficientemente complejo como para capturar las correlaciones cuánticas necesarias para representar el problema y la viabilidad considerando los recursos de hardware cuántico disponible.

6.6. Estimación de la energía mínima

En el método de VQE, la medición juega un rol importante para estimar la energía de un sistema cuántico. En este paso es donde el hardware cuántico es utilizado para extraer información acerca del estado cuántico, lo que nos permite obtener una estimación de la energía del sistema.

Una vez que hemos calculado cada uno de requerimientos anteriores para el problema que deseamos estudiar y simular, esto es: un operador Hamiltoniano que representa el operador de energía total del sistema en que estamos interesados, y un ansatz que nos brinde una función de onda de prueba que se encuentre parametrizado; tenemos entonces las partes para proceder a la medición.

Para obtener estimaciones de la energía requerimos realizar medición sobre la función de prueba para obtener el valor esperado de cada término que forma parte del Hamiltoniano de nuestro sistema. El número de mediciones requeridas dependerá en la precisión deseada y las propias características del hardware cuántico del que se dispone.

Para cada término de nuestro Hamiltoniano, calculamos su valor esperado calculando el promedio de los resultados obtenidos en las mediciones correspondientes. El valor esperado del Hamiltoniano $\langle H \rangle$ se obtiene al sumar cada uno de los valores esperados de los términos individuales que forman parte del Hamiltoniano que describe nuestro sistema. El valor esperado que se ha obtenido a través de este proceso representa un estimado de la energía de la función de onda de prueba y los parámetros empleados para dicha medición. Consecuentemente, esta estimación de la energía puede ser empleada como una función de coste que nos permita optimizar los parámetros de nuestra función de prueba a través de un optimizador clásico.

En el paso de optimización, se emplean métodos de optimización clásicos para ajustar los parámetros de la función de prueba con el objetivo de minimizar la energía estimada. Este proceso implica una actualización iterativa de los parámetros seguido de la aplicación de medidas cuánticas hasta que se alcanza un criterio de convergencia. El

conjunto final de parámetros será tal que nos permita obtener un estimado muy cercano al del estado base de nuestro sistema.

Todo este proceso puede ser implementado a través de métodos y clases que se encuentran disponibles tanto en Qiskit como en Pennylane, con los cuales extraeremos los valores esperados del Hamiltoniano en la base de qubits que describe nuestro sistema e iteraremos sobre este, afinando los parámetros de optimización hasta dar con una respuesta que esté tan cerca como deseemos de la energía del estado base. En el caso de nuestras simulaciones utilizaremos un criterio de convergencia de unos $1X10^{-6}$ Hartrees.

.

7. Simulaciones

A continuación, vamos a simular tres moléculas distintas cada una más compleja que la anterior. El objetivo es obtener el estado de mínima energía de las moléculas, ya que posee gran valor para el estudio de las mismas.

7.1. Molécula H_2

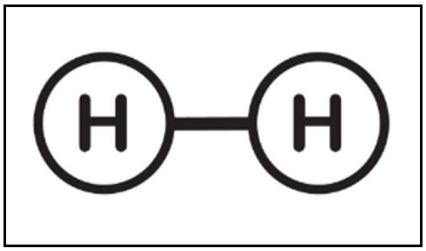


Figura 12. Molécula de hidrógeno

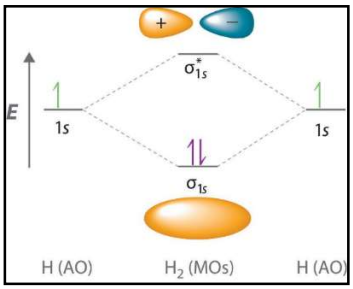


Figura 11. Orbitales Moleculares de la molécula de hidrógeno

El hidrógeno molecular H_2 es una molécula formada por dos átomos de hidrógeno. La distancia óptima de enlace para esta molécula es de 0,739Å.

En la simulación de H_2, se consideran dos electrones y dos núcleos de hidrógeno. Cada núcleo contiene un solo protón y un solo electrón. En la simulación cuántica de H_2 con el algoritmo de VQE, se consideran dos orbitales 1s y dos electrones que se distribuyen en estos orbitales (Figura 12). Esto permite modelar la molécula de hidrógeno y calcular propiedades como la energía de enlace y otras propiedades moleculares.

Construcción del Hamiltoniano Molecular

```
#Definición de átomos y coordenadas en el espacio de los mismos
#en la molécula
atomos = ["H", "H"]
coordenadas = np.array([[0.0, 0.0, -0.6614], [0.0, 0.0, 0.6614]])

molecula = qml.qchem.Molecule(atomos, coordenadas)
H, qubits = qml.qchem.molecular_hamiltonian(molecula)
print("El número de qubits es: ", qubits)
print("Hamiltoniano: ", H)
```

En el anterior código la función Molecule crea una estructura de datos donde guarda los átomos de la molécula, de los cuales con el carácter o caracteres que los simbolizan en la tabla periódica ya sabe las propiedades de los mismos, además también guarda la geometría de la molécula (las coordenadas en el espacio de cada átomo). La función molecular_hamiltonian calcula el hamiltoniano molecular de la molécula teniendo en cuenta la aproximación BO (5.3) realizando la segunda cuantización (apartado 6.2), calculando el espacio activo (apartado 6.3) y realizando el mapeo de Jordan-Wigner (apartado 6.4), por eso gracias a esta función ya sabemos el número de qubits que necesitamos para simular siguiendo el mapeo de Jordan-Wigner.

Hamiltoniano molecular H_2

```
Número de qubits necesarios para la simulación: 4
Hamiltoniano:
0.7808256947175569 * I(0) + 0.23730742579343955 * Z(0) + -0.4617382216448592 * Z(2) + 0.14068534985671294 * (Z(0) @ Z(2)) + 0.2
3730742579343944 * Z(1) + 0.1845824035123485 * (Z(0) @ Z(1)) + 0.04103888985278453 * (Y(0) @ X(1) @ X(2) @ Y(3)) + -0.041038889
85278453 * (Y(0) @ Y(1) @ X(2) @ X(3)) + -0.04103888985278453 * (X(0) @ X(1) @ Y(2) @ Y(3)) + 0.04103888985278453 * (X(0) @ Y
(1) @ Y(2) @ X(3)) + -0.4617382216448592 * Z(3) + 0.18172423970949747 * (Z(0) @ Z(3)) + 0.14068534985671294 * (Z(1) @ Z(3)) +
0.18172423970949747 * (Z(1) @ Z(2)) + 0.19181428727691374 * (Z(2) @ Z(3))
```

* Representación cuántica de la energía de la molécula de hidrógeno, donde cada qubit representa un electrón, o una interacción en el sistema. En el resultado se pueden ver las matrices de Pauli (2.10) que debemos aplicar (X, Y, Z, I) y a que qubits las debemos aplicar para obtener el Hamiltoniano. También cabe destacar que @ simboliza ⊗.

Algoritmo VQE

Lo siguiente que debemos hacer es aplicar el algoritmo VQE. Para eso necesitamos definir el circuito cuántico parametrizado "ansatz" (apartado 6.5) que prepara el estado de prueba de la molécula. Queremos preparar un estado de la forma:

$$|\psi(\theta)\rangle = \cos(\theta/2)\,|1100\rangle - \sin(\theta/2)\,|0011\rangle \qquad (7.1)$$

Este ansatz es bueno ya que en la aproximación de Hartree-Fock no se tiene en cuenta las fuerzas que genera un electrón sobre el otro, estas son mayores si los electrones están en el mismo orbital, así este ansatz "mitiga" el error cometido en la aproximación de Hartree-Fock. En el ansatz θ es el parámetro variacional que debe ser optimizado para encontrar la mejor aproximación al estado de mínima energía de la molécula. Según el mapeo de Jordan-Wigner, el primer término $|1100\rangle$ representa el estado de Hartree-Fock donde los dos electrones de la molécula ocupan los orbitales de menor energía. El segundo término $|0011\rangle$ codifica una doble excitación del estado de Hartree-Fock, donde dos partículas son excitadas de los qubits 0 y 1, a los qubits 2 y 3. El circuito cuántico para preparar el estado de prueba $|\psi(\theta)\rangle$ se representa en la figura 13:

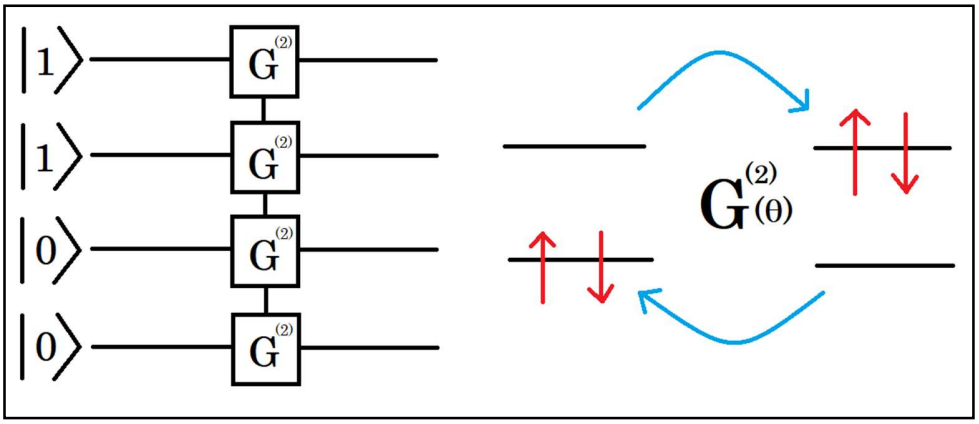


Figura 13. Ansatz que prepara el estado de prueba $|\psi(\theta)\rangle$

En la figura 13 la puerta $G^{(2)}$ se corresponde con la operación de doble excitación, que entrelaza los estados de cuatro qubits $|1100\rangle$ y $|0011\rangle$.

A continuación, se puede ver como implementar el circuito anterior en pennylane:

```
electrones = 2 #Nº de electrones de la molécula
hf = qml.qchem.hf_state(electrones, qubits)
```

Con el anterior fragmento de código obtenemos el estado Hartree-Fock de la molécula H_2 que en la representación Jordan-Wigner es el estado $|1100\rangle$.

Ahora crearemos el nodo de pennylane que implementará el circuito:

```
@qml.qnode(dispositivo)
def circuito(param, cables):
    qml.BasisState(hf, cables=cables) #Inicializamos los registros al estado Hartree-Fock
    qml.DoubleExcitation(param, cables=[0, 1, 2, 3]) #Añadimos las puertas de doble excitación al circuito
    return qml.expval(H)
```

Y ahora para definir nuestra función de coste tenemos que devolver el valor esperado del circuito que a su vez devuelve el valor esperado del Hamiltoniano:

```
def funcion_coste(param):
    return circuito(param, cables=range(qubits))
```

Ahora debemos minimizar la función de coste para encontrar el estado de mínima energía de la molécula H_2. Para eso necesitamos utilizar un optimizador clásico, En esta simulación utilizaremos el optimizador de descenso de gradiente. Llevamos a cabo la optimización con un máximo de 100 pasos intentando llegar a una convergencia de 10^{-6} para el valor de la función de coste.

Inicializamos el parámetro θ a 0, lo cual significa que empezamos desde el estado de Hartree-Fock. En cada iteración buscamos el parámetro θ que minimice el valor esperado del Hamiltoniano con un optimizador clásico, en este caso descenso de gradiente. El código del propio algoritmo VQE sería el siguiente:

```
theta = np.array(0.)

# Guarda los valores de la función de coste
energia = [funcion_coste(theta)]

# Guarda los valores del parámetro del circuito
angulo = [theta]

estado_optimizador = optimizador.init(theta)

for n in range(iteraciones_max):

    gradiente = jax.grad(funcion_coste)(theta)
    actualizaciones, estado_optimizador = optimizador.update(gradiente, estado_optimizador)
    theta = optax.apply_updates(theta, actualizaciones)

    angulo.append(theta)
    energia.append(funcion_coste(theta))

    convergencia = np.abs(energia[-1] - energia[-2])

    if n % 2 == 0:
        print(f"Paso = {n},  Energía = {energia[-1]:.8f} Ha")

    if convergencia <= conv_tolerancia:
        break

print("\n" f"Valor final del estado de mínima energía {energia[-1]:.8f} Ha")
print("\n" f"Valor óptimo del parámetro del circuito {angulo[-1]:.4f}")
```

Los resultados del algoritmo VQE son los siguientes:

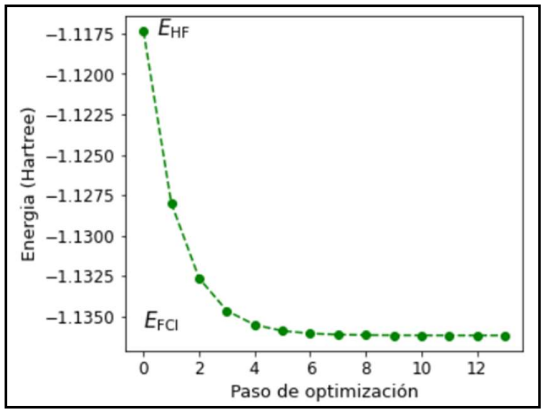


Figura 15. Energía de la molécula en cada paso del algoritmo VQE

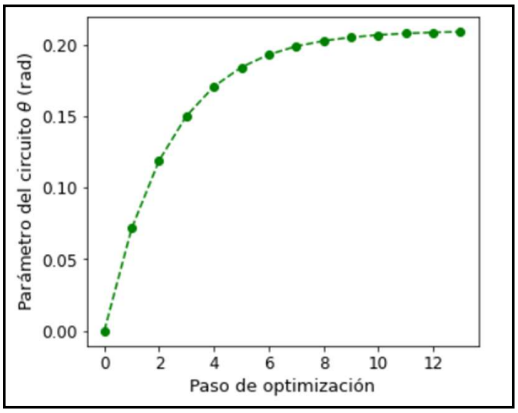


Figura 14. Valor del parámetro θ en cada paso del algoritmo VQE

Tabla 1. Resultados cada dos iteraciones del Algoritmo VQE para la molécula de Hidrógeno

Paso:	**Energía:**	**Parámetro angular θ:**
0	-1.12799983 Ha	0.0
2	-1.13466246 Ha	0.063
4	-1.13590595 Ha	0.172
6	-1.13613667 Ha	0.181
8	-1.13617944 Ha	0.198
10	-1.13618736 Ha	0.203
12	-1.13618883 Ha	0.208

*Nota 1 Hartree = 4.4e-18 Julios

En la anterior tabla se pueden ver los resultados obtenidos en cada uno de los pasos del algoritmo VQE para la molécula de H_2. En nuestro caso el algoritmo converge después de 12 iteraciones. El valor óptimo para el parámetro del circuito es $\theta^* = 0{,}208$, lo cual define el estado:

$$|\psi(\theta^*)\rangle = 0{,}994|1100\rangle - 0{,}104|0011\rangle$$

Es decir, este es el estado de mínima energía de la molécula H_2 en una aproximación de conjunto base mínimo.

7.2. Molécula H_3^+

El hidrógeno molecular protonado, catión trihidrógeno o Hodrógeno triátomico es uno de los iones más abundantes del universo. El catión H_3^+ posee gran importancia en la química en fase gaseosa del medio interestelar, por eso, a pesar de ser la molécula triatómica más simple, es de gran interés poder estudiar sus propiedades.

La molécula H_3 es lo que se conoce como una molécula de Rydberg debido a su gran inestabilidad ya que sus estados excitados (o estados de rydberg) poseen una vida media relativamente alta comparadas con sus periodos vibracionales y rotacionales. Sin embargo, el catión H_3^+ es estable.

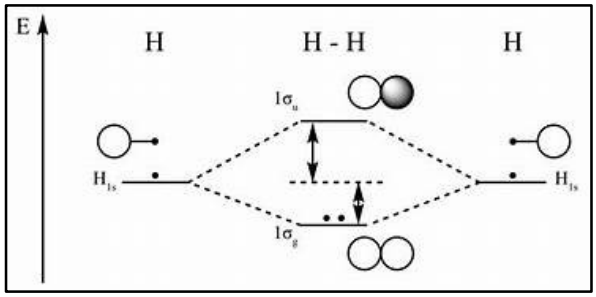


Figura 16. Diagrama de los orbitales moleculares de la molécula H_3^+

Construcción del Hamiltoniano molecular

```
atomos = ["H", "H", "H"]
coordenadas = np.array([[0.0102, 0.0442, 0.0], [0.9867, 1.6303, 0.0],[1.8720,-0.0085,0.0]])

H, qubits = qchem.molecular_hamiltonian(atomos, coordenadas, charge = 1)
print("El número de qubits necesarios para realizar la simulación es: {:}".format(qubits))
print("Hamiltoniano:")
print(H)
```

Ahora nuestra molécula no tiene carga neutra, por eso tenemos que especificar carga de +1, debido a que nuestra molécula es un catión al que le falta un electrón. La función molecular hamiltonian calculará el hamiltoniano molecular teniendo en cuenta la aproximación BO (5.3) realizará la segunda cuantización (apartado 6.2), calculará el espacio activo (apartado 6.3) y realizará el mapeo de Jordan-Wigner (apartado 6.4).

Cabe destacar de que a pesar de que esta molécula sea tan solo un átomo de hidrógeno más grande que la anterior el Hamiltoniano molecular será mucho más complejo, se puede ver perfectamente el crecimiento exponencial en la complejidad del problema, así como en los recursos computacionales que se necesitan para poder realizar estas simulaciones.

Hamiltoniano molecular H_3^+

```
El número de qubits necesarios para realizar la simulación es: 6
Hamiltoniano:
-0.29975061450116125 * I(0) + 0.2144549542316111 * Z(0) + -4.3201735952513864e-07 * (Y(0) @ Z(1) @ Y(2)) + -4.3201735952513864e
-07 * (X(0) @ Z(1) @ X(2)) + 3.661499299971449e-07 * (Y(0) @ Z(1) @ Z(2) @ Z(3) @ Y(4)) + 3.661499299971449e-07 * (X(0) @ Z(1)
@ Z(2) @ Z(3) @ X(4)) + -0.11489380123586843 * Z(2) + 0.10696109578702372 * (Z(0) @ Z(2)) + -0.11491204817793765 * Z(4) + 0.106
96270053341973 * (Z(0) @ Z(4)) + 0.21445495423161104 * Z(1) + 0.14703116579236286 * (Z(0) @ Z(1)) + 1.1014771324332692e-06 * (Y
(0) @ Y(2)) + 1.1014771324332692e-06 * (X(0) @ X(2)) + -9.3339218505510l4e-07 * (Y(0) @ Z(2) @ Z(3) @ Y(4)) + -9.33392185055101
4e-07 * (X(0) @ Z(2) @ Z(3) @ X(4)) + -4.320173595242713e-07 * (Y(1) @ Z(2) @ Y(3)) + 1.1014771324332692e-06 * (Z(0) @ Y(1) @ Z
(2) @ Y(3)) + -4.320173595242713e-07 * (X(1) @ Z(2) @ X(3)) + 1.1014771324332692e-06 * (Z(0) @ X(1) @ Z(2) @ X(3)) + 0.03599964
5767654506 * (Y(0) @ X(1) @ X(2) @ Y(3)) + -0.035999645767654506 * (Y(0) @ Y(1) @ X(2) @ X(3)) + -0.035999645767654506 * (X(0)
@ X(1) @ Y(2) @ Y(3)) + 0.035999645767654506 * (X(0) @ Y(1) @ Y(2) @ X(3)) + 3.661499299971449e-07 * (Y(1) @ Z(2) @ Z(3) @ Z(4)
@ Y(5)) + -9.333921850551014e-07 * (Z(0) @ Y(1) @ Z(2) @ Z(3) @ Z(4) @ Y(5)) + 3.661499299971449e-07 * (X(1) @ Z(2) @ Z(3) @ Z
(4) @ X(5)) + -9.333921850551014e-07 * (Z(0) @ X(1) @ Z(2) @ Z(3) @ Z(4) @ X(5)) + 0.03599969956877999 * (Y(0) @ X(1) @ X(4) @
Y(5)) + -0.03599969956877999 * (Y(0) @ Y(1) @ X(4) @ X(5)) + -0.03599969956877999 * (X(0) @ X(1) @ Y(4) @ Y(5)) + 0.03599969956
877999 * (X(0) @ Y(1) @ Y(4) @ X(5)) + -8.244716676554173e-07 * (Y(0) @ Z(1) @ Z(3) @ Y(4)) + -8.244716676554173e-07 * (X(0) @
Z(1) @ Z(3) @ X(4)) + 9.72916776978243e-07 * (Y(0) @ Z(1) @ Y(2) @ Z(4)) + 9.72916776978243e-07 * (X(0) @ Z(1) @ X(2) @ Z(4)) +
-0.11489380123586848 * Z(3) + 0.14296074155467822 * (Z(0) @ Z(3)) + -0.01812971961944379 * (Y(0) @ Z(1) @ Y(2) @ Z(3)) + -0.018
12971961944379 * (X(0) @ Z(1) @ X(2) @ Z(3)) + 0.015362548721459802 * (Y(0) @ Z(1) @ Z(2) @ Y(4)) + 0.015362548721459802 * (X
(0) @ Z(1) @ Z(2) @ X(4)) + -0.015363373193127462 * (Y(0) @ Y(3) @ Z(4) @ Y(5) @ Z(1) @ Y(2)) + -0.015363373193127462 * (Y(0) @
X(3) @ Z(4) @ X(5) @ Z(1) @ Y(2)) + -0.015363373193127462 * (X(0) @ Y(3) @ Z(4) @ Y(5) @ Z(1) @ X(2)) + -0.015363373193127462 *
(X(0) @ X(3) @ Z(4) @ X(5) @ Z(1) @ X(2)) + -0.01812930716156933 * (Y(0) @ Z(1) @ Z(2) @ X(3) @ X(4) @ Y(5)) + 0.01812930716156
933 * (Y(0) @ Z(1) @ Z(2) @ Y(3) @ X(4) @ X(5)) + 0.01812930716156933 * (X(0) @ Z(1) @ Z(2) @ X(3) @ Y(4) @ Y(5)) + -0.01812930
716156933 * (X(0) @ Z(1) @ Z(2) @ Y(3) @ Y(4) @ X(5)) + -0.11491204817793754 * Z(5) + 0.14296240010219974 * (Z(0) @ Z(5)) + 0.0
1813028007834631 * (Y(0) @ Z(1) @ Y(2) @ Z(5)) + 0.01813028007834631 * (X(0) @ Z(1) @ X(2) @ Z(5)) + -0.015363023691225795 * (Y
(0) @ Z(1) @ Z(2) @ Z(3) @ Y(4) @ Z(5)) + -0.015363023691225795 * (X(0) @ Z(1) @ Z(2) @ Z(3) @ X(4) @ Z(5)) + 0.106961095787023
72 * (Z(1) @ Z(3)) + 0.10696270053341973 * (Z(1) @ Z(5)) + 0.14296074155467822 * (Z(1) @ Z(2)) + -0.01812971961944379 * (Y(1) @
Y(3)) + -0.01812971961944379 * (X(1) @ X(3)) + 0.015362548721459802 * (Y(1) @ Z(3) @ Z(4) @ Y(5)) + 0.015362548721459802 * (X
(1) @ Z(3) @ Z(4) @ X(5)) + -0.01536337319312746 * (Y(1) @ X(2) @ X(3) @ Y(4)) + 0.01536337319312746 * (Y(1) @ Y(2) @ X(3) @ X
(4)) + 0.01536337319312746 * (X(1) @ X(2) @ Y(3) @ Y(4)) + -0.01536337319312746 * (X(1) @ Y(2) @ Y(3) @ X(4)) + 0.0181293071615
69328 * (Y(1) @ X(2) @ X(4) @ Y(5)) + 0.018129307161569328 * (Y(1) @ Y(2) @ Y(4) @ Y(5)) + 0.018129307161569328 * (X(1) @ X(2)
@ X(4) @ X(5)) + 0.018129307161569328 * (X(1) @ Y(2) @ Y(4) @ X(5)) + -8.244716676554173e-07 * (Y(1) @ Z(2) @ Z(4) @ Y(5)) + -
8.244716676554173e-07 * (X(1) @ Z(2) @ Z(4) @ X(5)) + 9.72916776978243e-07 * (Y(1) @ Z(2) @ Y(3) @ Z(5)) + 9.72916776978243e-07
* (X(1) @ Z(2) @ X(3) @ Z(5)) + 0.14296240010219974 * (Z(1) @ Z(4)) + 0.01813028007834631 * (Y(1) @ Z(2) @ Y(3) @ Z(4)) + 0.018
13028007834631 * (X(1) @ Z(2) @ X(3) @ Z(4)) + -0.015363023691225795 * (Y(1) @ Z(2) @ Z(3) @ Y(5)) + -0.015363023691225795 * (X
(1) @ Z(2) @ Z(3) @ X(5)) + -9.648559728958617e-12 * (Y(2) @ Z(3) @ Y(4)) + -9.648559728958617e-12 * (X(2) @ Z(3) @ X(4)) + 0.1
0724024533206666 * (Z(2) @ Z(4)) + 0.1638437446030772 * (Z(2) @ Z(3)) + 7.65781211838068e-07 * (Y(2) @ Y(4)) + 7.65781211838068
e-07 * (X(2) @ X(4)) + -9.648559728958617e-12 * (Y(3) @ Z(4) @ Y(5)) + 7.65781211838068e-07 * (Z(2) @ Y(3) @ Z(4) @ Y(5)) + -9.
648559728958617e-12 * (X(3) @ Z(4) @ X(5)) + 7.65781211838068e-07 * (Z(2) @ X(3) @ Z(4) @ X(5)) + 0.018867785017622783 * (Y(2)
@ X(3) @ X(4) @ Y(5)) + -0.018867785017622783 * (Y(2) @ Y(3) @ X(4) @ X(5)) + -0.018867785017622783 * (X(2) @ X(3) @ Y(4) @ Y
(5)) + 0.018867785017622783 * (X(2) @ Y(3) @ Y(4) @ X(5)) + 0.12610803034968943 * (Z(2) @ Z(5)) + -7.657715632783391e-07 * (Y
(2) @ Z(3) @ Y(4) @ Z(5)) + -7.657715632783391e-07 * (X(2) @ Z(3) @ X(4) @ Z(5)) + 0.10724024533206666 * (Z(3) @ Z(5)) + 0.1261
0803034968943 * (Z(3) @ Z(4)) + -7.657715632783391e-07 * (Y(3) @ Y(5)) + -7.657715632783391e-07 * (X(3) @ X(5)) + 0.16384447520
751738 * (Z(4) @ Z(5))
```

Se puede ver que el Hamiltoniano es mucho más complejo y que además vamos a necesitar seis qubits para poder llevar a cabo las simulaciones (dos más que en la simulación de H_2).

Algoritmo VQE

Lo primero que necesitamos para realizar el algoritmo (habiendo calculado ya el Hamiltoniano) es preparar nuestro circuito parametrizable (ansatz). Para ello empezaremos escogiendo el estado Hartree-Fock que es una buena aproximación al estado de mínima energía. El estado Hartree-Fock de la molécula H_3^+ sabiendo que necesitamos seis qubits sería $|110000\rangle$ en la representación Jordan-Wigner. Nuestro ansatz necesita llevarnos a un estado en superposición de la siguiente manera:

$$\alpha|110000\rangle + \beta|001100\rangle + \gamma|000011\rangle \tag{7.2}$$

Si seguimos la representación Jordan-Wigner α es que los dos electrones están en el orbital de menor energía y γ es que los dos electrones están en el orbital de mayor energía. Este ansatz es bueno ya que en la aproximación de Hartree-Fock no se tiene en cuenta las fuerzas que genera un electrón sobre el otro, estas son mayores si los electrones están en el mismo orbital, así este ansatz "mitiga" el error cometido en la aproximación de Hartree-Fock.

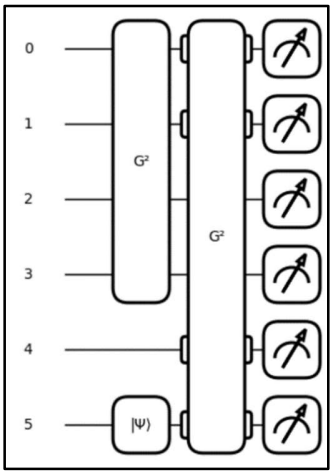
Para el algoritmo VQE también necesitamos que los parámetros α, β y γ puedan variar a medida que se vayan ejecutando los distintos pasos del algoritmo. Teniendo todo lo anterior en cuenta sabemos que tenemos que volver a utilizar las puertas de doble excitación ($G^{(2)}$) para conseguir ese ansatz. De esta manera teniendo en cuenta que los qubits 0 y 1 son inicializados a uno, y el resto son inicializados a 0. Primero deberemos aplicar las puertas de doble excitación sobre los qubits 0, 1, 2 y 3 y después deberemos aplicar las puertas de doble excitación sobre los qubits 0, 1, 4 y 5. Así obtendremos nuestro ansatz de la forma:

$$\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)|110000\rangle - \sin\left(\frac{\theta_1}{2}\right)|001100\rangle - \cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)|000011 \tag{7.3}$$

Que es lo que estábamos buscando. Así, el circuito que nos prepara este estado se implementaría de la siguiente manera:

```
def circuito(params, wires):
    qml.BasisState(hf, wires=wires)
    qml.DoubleExcitation(params[0], wires=[0, 1, 2, 3])
    qml.DoubleExcitation(params[1], wires=[0, 1, 4, 5])
    return qml.expval(H)
```

El ansatz sería el siguiente:



La implementación del propio algoritmo sería de la siguiente manera:

```
theta = np.array([0.0,0.0])

# Guarda los valores de la función de coste
energia = [funcion_coste(theta)]

# Guarda los valores del parámetro del circuito
angulo = [theta]


estado_optimizador = optimizador.init(theta)

for n in range(iteraciones_max):

    gradiente = jax.grad(funcion_coste)(theta)
    actualizaciones, estado_optimizador = optimizador.update(gradiente, estado_optimizador)
    theta = optax.apply_updates(theta, actualizaciones)

    angulo.append(theta)
    energia.append(funcion_coste(theta))

    convergencia = np.abs(energia[-1] - energia[-2])

    if n % 2 == 0:
        print(f"paso = {n},  Energía = {energia[-1]:.8f} Ha")



    if convergencia <= conv_tolerancia:
        break

print("\n" f"Valor final del estado de mínima energía {energia[-1]:.8f} Ha")
print("\n" f"Valor óptimo del parámetro del circuito {theta[0]:.8f} {theta[1]:.8f}")
```

Los resultados obtenidos son los siguientes:

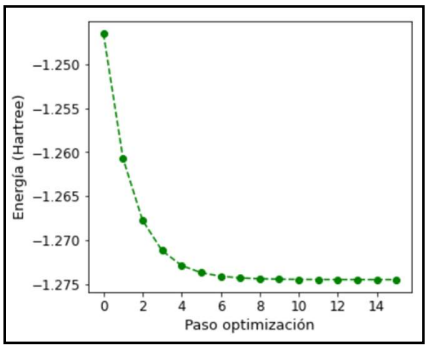


Figura 19. Resultado del algoritmo VQE en cada paso para la molécula H_3^+

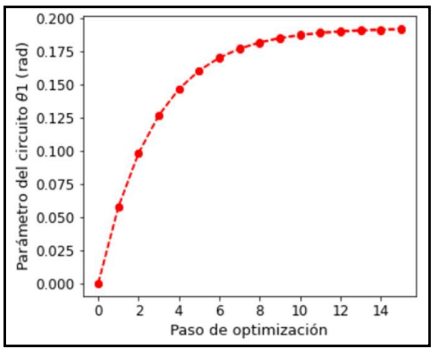


Figura 18. Resultados en cada paso del parámetro θ_1

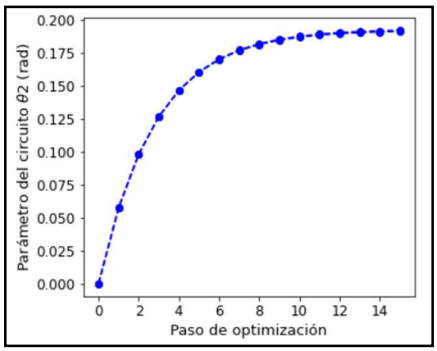


Figura 17. Resultados en cada paso del parámetro θ_2

Tabla 2. Resultados cada dos iteraciones del Algoritmo VQE para la molécula de trihidrógeno ionizado

Paso:	**Energía:**	$\boldsymbol{\theta_1}$**:**	$\boldsymbol{\theta_2}$**:**
0	-1.24670025 Ha	0.0	0.0
2	-1.26715671 Ha	0.061	0.064
4	-1.27365804 Ha	0.114	0.121
6	-1.27425241 Ha	0.137	0.145
8	-1.27439362 Ha	0.176	0.178
10	-1.27442718 Ha	0.182	0.185
12	-1.27443517 Ha	0.188	0.190
14	-1.27443707 Ha	0.191	0.192

En nuestro caso el algoritmo converge después de 14 iteraciones. Los valores óptimos para los parámetros del circuito son $\theta_1^* = 0{,}191$, $\theta_2^* = 0{,}192$ lo cual define el estado:

$$|\psi(\theta^*)\rangle = 0{,}9908|110000\rangle - 0{,}096|001100\rangle - 0{,}096|000011\rangle$$

Este es el estado de mínima energía de la molécula.

7.3. Molécula H_2O

La molécula de agua está formada por dos átomos de hidrógeno y uno de oxígeno. Esta molécula es esencial para los seres vivos, al servir de medio para el metabolismo de las biomoléculas. Estudiar la molécula de agua es muy interesante ya que aparte de ser esencial para la vida como se menciona anteriormente, el agua sirve como catalizador en incontables reacciones químicas.

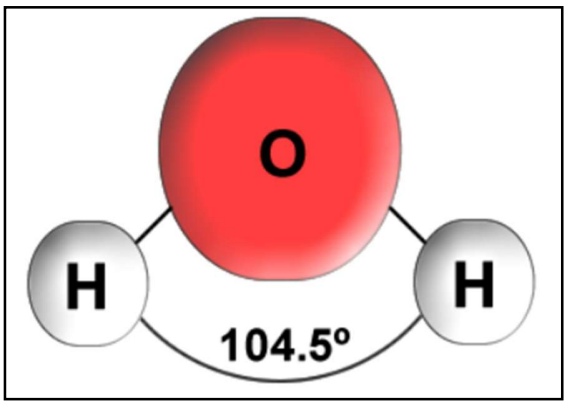


Figura 20. Molécula de agua

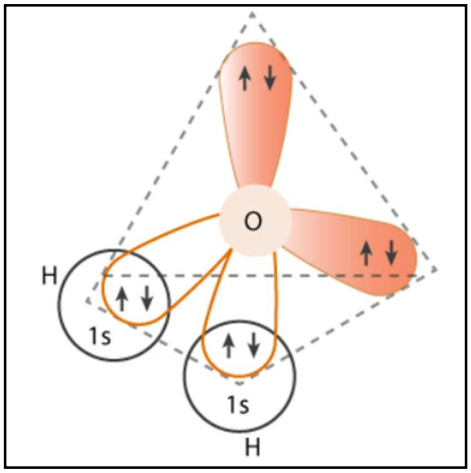


Figura 21. Orbitales en la molécula de agua

Estimación de recursos

La molécula de agua es mucho más compleja que los dos ejemplos anteriores, para empezar, tiene diez electrones, mientras que los dos ejemplos anteriores tan solo tenían dos. Además, el átomo de oxígeno presente en esta molécula le da a esta mucho más peso molecular, esto la convierte en una molécula mucho más difícil de estudiar y simular.

Como hemos visto en las anteriores simulaciones una molécula un poco más grande que otra, como es el caso de la molécula H_3^+ con respecto a la molécula H_2 (tan solo hay un átomo de hidrógeno más) ya hace que el Hamiltoniano molecular sea significativamente más complejo. Por eso podemos esperar que el Hamiltoniano molecular de la molécula de agua será mucho más complejo que los dos anteriores.

Algoritmos como QPE se están estudiando en química cuántica como potenciales salidas para problemas que son intratables con computadores clásicos. Uno de los problemas más estudiados de la química cuántica es la simulación del Hamiltoniano molecular (su energía esperada como hacemos en el algoritmo VQE, la energía de

interacción entre los átomos, la energía de repulsión entre dos electrones de cierto orbital…).

Para el estudio de las simulaciones de Hamiltonianos moleculares se suele teorizar el empleo del algoritmo QPE. El algoritmo QPE se puede utilizar para computar la fase asociada con un autoestado de un operador unitario. Para el propósito de la simulación cuántica, el operador unitario U puede ser elegido para compartir autovectores con un Hamiltoniano molecular. Por ejemplo, estableciendo $U = e^{-iH}$. Estimando así la fase de este operador unitario permite recuperar el correspondiente autovalor del Hamiltoniano.

Podemos usar PennyLane para realizar una estimación de los recursos que debemos emplear para simular el Hamiltoniano con el algoritmo QPE. Nos daremos cuenta de que, para la mayoría de casos de interés se necesitan más qubits y circuitos de más profundidad de lo que se puede implementar en Hardware existente. Para realizar la estimación de recursos con PennyLane debemos definir nuestra molécula:

```
atomos = ['O', 'H', 'H']
geometria = np.array([[0.00000000,  0.00000000,  0.28377432],
                      [0.00000000,  1.45278171, -1.00662237],
                      [0.00000000, -1.45278171, -1.00662237]], requires_grad=False)
```

*Nota: Las coordenadas de la molécula se corresponden con la "geometría de equilibrio" de la molécula de agua.

La función empleada por pennyLane necesita también las integrales electrónicas uno y dos (6.3 y 6.4) como entrada.

Por lo tanto, el siguiente paso es crear nuestro objeto molécula y computar las integrales electrónicas uno y dos que se puede hacer con la función electron_integrals:

```
molecula = qml.qchem.Molecule(atomos, geometria, basis_name='6-31g')
nucleo, uno, dos = qml.qchem.electron_integrals(molecula)()
```

Por último, utilizamos la función de la librería PennyLane para obtener nuestro resultado:

```
algoritmo = qml.resource.DoubleFactorization(uno, dos)
print(f'Puertas estimadas: {algoritmo.gates:.2e} \nQubits estimados: {algoritmo.qubits}')
```

Los resultados que obtenemos al ejecutar el anterior código son un total de 3,06e+08 puertas estimadas y 693 qubits estimados.

Estas estimaciones son obtenidas con un margen de error de 0,0016 Ha, que es el valor por defecto que utiliza PennyLane. Si cambiamos ese margen de error a un valor más alto el número de puertas y qubits necesarios disminuirá. Al ser el margen de error más alto la simulación del Hamiltoniano no debe ser tan precisa.

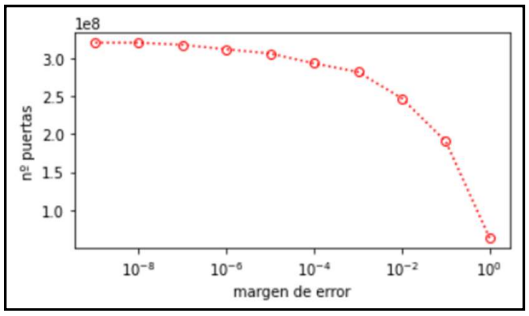


Figura 22. Estimación de puertas necesarias para realizar la simulación del Hamiltoniano usando el algoritmo QPE con respecto al margen de error

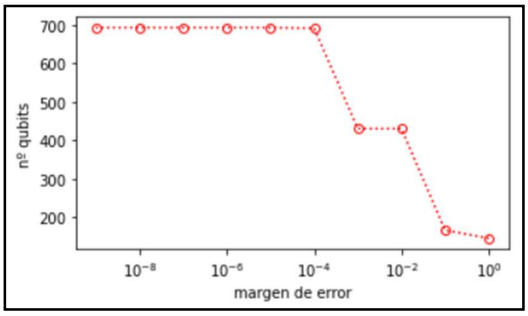


Figura 23. Estimación de qubits necesarios para realizar la simulación del Hamiltoniano usando el algoritmo QPE con respecto al margen de error

Construcción del Hamiltoniano molecular

La construcción del Hamiltoniano molecular se hace de la siguiente forma:

```
atomos = ["H", "O", "H"]
coordenadas = np.array([[-0.0399, -0.0038, 0.0], [1.5780, 0.8540, 0.0], [2.7909, -0.5159, 0.0]])

molecula = qchem.Molecule(atomos, coordenadas)
H, qubits = qchem.molecular_hamiltonian(molecula)
print("Número de qubits: {:}".format(qubits))
print("Hamiltoniano:")
print(H)
```

Con la función molecular hamiltonian calcularemos el hamiltoniano molecular teniendo en cuenta la aproximación BO (5.3), la función calculará la segunda cuantización (apartado 6.2), el espacio activo (apartado 6.3) y realizará el mapeo de Jordan-Wigner (apartado 6.4).

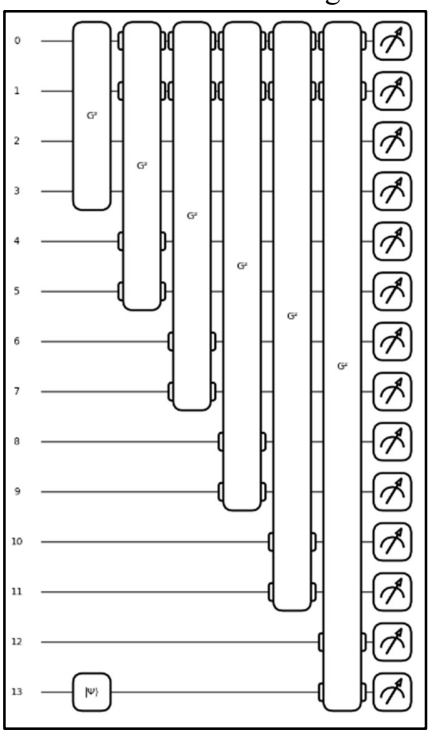
Esta vez nos da como resultado que necesitamos 14 qubits usando el mapeo Jordan-Wigner, nosotros estamos haciendo la simulación con un margen de error alto y por lo tanto nos da un número de qubits asumible. El hamiltoniano molecular de la molécula de agua entero ocuparía varias páginas de esta memoria, ya se pudo ver como el hamiltoniano de la molécula H_3^+ era significativamente mayor que el de la molécula H^2, por lo tanto, no es de extrañar que el Hamiltoniano de la molécula de agua (que es una molécula mucho más

grande y compleja) sea de un tamaño mucho mayor. De nuevo se vuelve a ver la naturaleza exponencial del problema.

Algoritmo VQE

Para el algoritmo necesitamos primero un circuito parametrizable “ansatz”, ahora encontrar un buen ansatz puede ser un problema debido a la mayor complejidad del problema. En el caso de la molécula del agua el estado de Hartree-Fock es el $|11111111110000\rangle$ y para obtener nuestro ansatz vamos a poner todos los posibles estados de los electrones en superposición, este no es el mejor ansatz para esta molécula y puede hacer que el tiempo de ejecución del algoritmo sea mayor, pero nos servirá en esta simulación.

Nuestro ansatz es el siguiente:



El algoritmo VQE se codifica de la siguiente manera:

```
theta = np.array([0.0,0.0,0.0,0.0,0.0,0.0])

# Guarda los valores de la función de coste
energia = [funcion_coste(theta)]

# store the values of the circuit parameter
angulo = [theta]


estado_optimizador = opt.init(theta)

for n in range(max_iteraciones):

    gradiente = jax.grad(funcion_coste)(theta)
    actualizaciones, estado_optimizador = optimizador.update(gradiente, estado_optimizador)
    theta = optax.apply_updates(theta, actualizaciones)

    angulo.append(theta)
    energia.append(funcion_coste(theta))

    convergencia = np.abs(energia[-1] - energia[-2])

    if n % 2 == 0:
        print(f"Paso = {n},  Energía = {energia[-1]:.8f} Ha")



    if convergencia <= conv_tolerancia:
        break

print("\n" f"Valor final del estado de mínima energía {energia[-1]:.8f} Ha")
print("\n" f"Valor óptimo del parámetro del circuito {theta[0]:.8f} {theta[1]:.8f}")
```

El resultado del algoritmo es el siguiente:

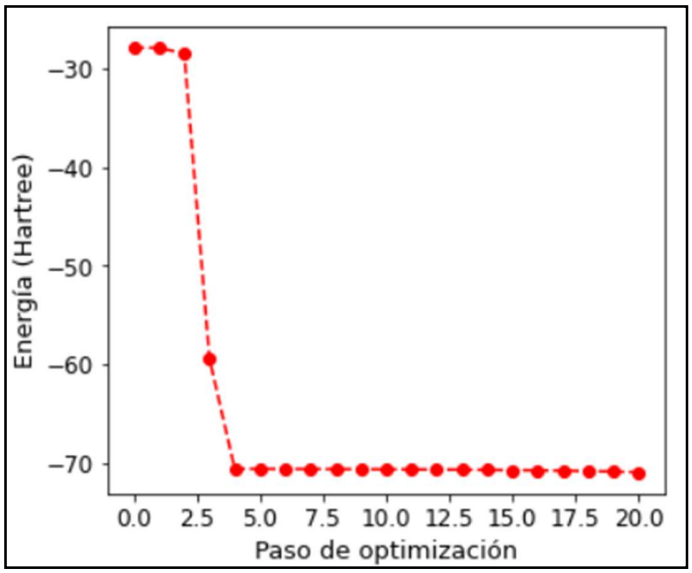


Figura 24. Resultados del algoritmo VQE para la molécula de agua

Tabla 3. Resultados cada dos iteraciones del Algoritmo VQE para la molécula de Agua

Paso:	**Energía:**
0	-27.87237035 Ha
2	-59.46917481 Ha
4	-70.62550134 Ha
6	-70.64087543 Ha
8	-70.65960136 Ha
10	-70.68293659 Ha
12	-70.71302737 Ha
14	-70.75376938 Ha
16	-70.81257055 Ha
18	-70.90376899 Ha
20	-70.96916301 Ha

Como se puede ver los resultados de esta simulación son mucho más interesantes que los de las otras dos, empezamos con un valor de energía muy alto y en las primeras iteraciones del algoritmo baja muy rápido hasta unos -70,6 Ha. A partir de ahí el algoritmo encuentra siempre un estado de menor energía de la molécula, pero cada vez baja menos la energía y se estabiliza en unos -71 Ha.

También cabe destacar que el tiempo de ejecución del algoritmo con la molécula de hidrógeno tardaba segundos en ejecutarse, lo mismo puede decirse de la molécula de trihidrógeno, sin embargo, la molécula de agua ha tardado 1 hora y 50 minutos en ejecutar el algoritmo. De nuevo se vuelve a ver la naturaleza exponencial del problema.

8. Discusión y Trabajo Futuro

8.1. Discusión, análisis y trabajo futuro (español)

Durante este trabajo de fin de grado se ha hecho un estudio de viabilidad de problemas de química computacional usando la computación cuántica que es un nuevo paradigma computacional, y llevar a cabo la simulación de tres moléculas diferentes. Se ha llevado a cabo una metodología propia usando la biblioteca de pennylane para extraer el estado de mínima energía de tres moléculas diferentes cada una de ellas más compleja, de 4, 6 y 14 qubits respectivamente. En nuestra metodología primero se ha obtenido el Hamiltoniano molecular de las mismas teniendo en cuenta la segunda cuantización, la aproximación de BO y el mapeo de qubits, después se ha obtenido un circuito parametrizable o ansatz diferente para cada una de las moléculas, por último, se ha ejecutado el algoritmo VQE en sucesivas iteraciones hasta un cierto rango de convergencia en el cual se utiliza un optimizador clásico (como se explica en el apartado 7 de la memoria).

Los recursos para simular estas moléculas crecen de forma exponencial, ya que como hemos comprobado simular la molécula de agua de forma precisa nos requiere 693 qubits. A día de hoy no hay computadores cuánticos comerciales con la suficiente mitigación de ruido que nos permitan efectuar esta simulación de forma correcta.

Los resultados que nos han dado en los casos del hidrógeno y el trihidrógeno son acordes con estudios realizados de estas moléculas, es decir, son unos resultados precisos. La simulación de las moléculas se vuelve muy compleja a medida que aumentan de tamaño, por eso, en el caso del agua nos ha dado que su energía mínima es de unos -71 Ha y en estudios de esta molécula se ha descubierto que es de -76 Ha, esto se puede deber a que no tenemos los suficientes qubits como para simularlo de forma precisa.

Queda como trabajo futuro realizar un estudio en profundidad de la librería PennyLane y sus posibilidades con otros ámbitos de computación cuántica como por ejemplo quantum machine learning, así como comparar PennyLane con estos problemas, por ejemplo, con otros lenguajes o librerías de computación cuántica como qiskit. Realizar el algoritmo VQE a la vez que se modifica la geometría de la molécula, esto es la distancia que separa sus átomos y el ángulo de uno con respecto del otro, si podemos hacer esto seremos capaces de prever por ejemplo la energía de disociación de una molécula y la distancia con la que se produce. De esta manera podríamos simular por ejemplo el calor y la luz que se generará de una reacción exotérmica.

Sería interesante también poder ejecutar estos problemas en computadores cuánticos reales y comparar los resultados obtenidos en computadores cuánticos basados en trampa de iones y basados en superconductores para ver el tipo de computador que se adapta mejor a este tipo de problemas. En el futuro también se podrían simular moléculas más grandes y complejas que pueden tener un gran interés estudiarlas, y al simularlas no solo mirar su estado de mínima energía, su estado más estable o los recursos necesarios para simularla, si no, que también se podrían simular otras propiedades como la energía de disociación o la estabilidad electrónica. Como ejemplo de esto último, la computación cuántica puede manejar la complejidad exponencial que enfrentan los métodos clásicos de computación de alto rendimiento (HPC) al tratar sistemas cuánticos.

Por ejemplo, una molécula que puede ser común para todos como la cafeína ($C_8H_{10}N_4O_2$) contiene 102 electrones, y para simular con precisión su estructura electrónica completa se necesitarían aproximadamente entre 1,000 y 10,000 qubits. Los métodos clásicos no pueden gestionar esta complejidad debido a las limitaciones computacionales y al crecimiento exponencial de los recursos necesarios.

IBM está avanzando en el desarrollo de sistemas cuánticos con bajos errores que se acercan a este rango de qubits. Sería interesante poder utilizar el procesador cuántico IBM Condor, que se lanzó en diciembre de 2023 para simular estas moléculas tan complejas. Estos sistemas están diseñados para reducir las tasas de error y aumentar la coherencia, lo cual es crucial para ejecutar códigos de simulación molecular precisos. La disponibilidad de hardware cuántico con entre 1,000 y 10,000 qubits y bajas tasas de error permitirá ejecutar códigos de simulación para moléculas complejas como la cafeína. Esto supera las limitaciones de la HPC clásica y abre nuevas posibilidades en química computacional, como el diseño de fármacos y el desarrollo de materiales avanzados. Por lo tanto, la computación cuántica ofrece una herramienta efectiva para abordar problemas que antes eran intratables, permitiendo simulaciones más precisas y eficientes de sistemas moleculares complejos. Por lo tanto, es un campo de investigación con un futuro prometedor y que puede cambiar el paradigma en una gran cantidad de industrias.

8.2. Discussion, analysis and future work (English)

During this undergraduate thesis, a feasibility study was conducted on computational chemistry problems using quantum computing, a new computational paradigm, and the simulation of three different molecules was carried out. A custom methodology was developed to extract the ground state of three increasingly complex molecules, consisting of 4, 6, and 14 qubits, respectively. In our methodology, the molecular Hamiltonian was first obtained, considering second quantization, the Born-Oppenheimer approximation, and qubit mapping. Next, a different parameterized circuit or ansatz was obtained for each molecule. Finally, the VQE algorithm was executed in successive iterations until a certain convergence range was achieved, using a classical optimizer (as explained in section 7 of the report).

The simulation of these molecules is not as precise as possible, as we have found that accurately simulating the water molecule requires 693 qubits. As of now, there are no commercial quantum computers with sufficient noise mitigation to perform this simulation correctly.

The results obtained for hydrogen and trihydrogen are consistent with studies conducted on these molecules, meaning they are accurate results. The simulation of molecules becomes very complex as their size increases. For example, in the case of water, we obtained a minimum energy of about -71 Ha, while studies of this molecule indicate it is -76 Ha. This discrepancy may be due to the insufficient number of qubits to simulate it accurately.

Future work includes conducting an in-depth study of the PennyLane library and its capabilities with other areas of quantum computing, such as quantum machine learning, and comparing PennyLane with other quantum computing languages or libraries like Qiskit. It would also involve performing the VQE algorithm while simultaneously modifying the geometry of the molecule—specifically, the distance between its atoms and the angle between them. If this can be achieved, we would be able to predict, for example, the dissociation energy of a molecule and the distance at which it occurs. This would allow us to simulate phenomena such as the heat and light generated by an exothermic reaction.

It would also be interesting to execute these problems on real quantum computers and compare the results obtained from ion trap-based quantum computers and superconducting-based quantum computers to determine which type of computer is better suited for these problems. In the future, it might also be possible to simulate larger and more complex molecules of significant interest. Not only would we look at their ground

state, most stable state, or the resources required for their simulation, but we could also simulate other properties such as dissociation energy or electronic stability. For example, quantum computing can handle the exponential complexity faced by classical high-performance computing (HPC) methods when dealing with quantum systems.

For instance, a molecule as common as caffeine ($C_8H_{10}N_4O_2$), which contains 102 electrons, would require approximately 1,000 to 10,000 qubits to accurately simulate its complete electronic structure. Classical methods cannot manage this complexity due to computational limitations and the exponential growth of required resources.

IBM is advancing in the development of quantum systems with low errors that approach this qubit range. It would be interesting to use the IBM Condor quantum processor, launched in December 2023, to simulate these complex molecules. These systems are designed to reduce error rates and increase coherence, which is crucial for executing precise molecular simulation codes. The availability of quantum hardware with 1,000 to 10,000 qubits and low error rates will enable the execution of simulation codes for complex molecules like caffeine. This surpasses the limitations of classical HPC and opens new possibilities in computational chemistry, such as drug design and advanced material development. Therefore, quantum computing offers an effective tool for addressing previously intractable problems, enabling more accurate and efficient simulations of complex molecular systems. It is a promising field of research with the potential to change the paradigm across many industries.

Bibliografía

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

Blinder, S. M. (2019). Introduction to the Hartree-Fock method. In *Mathematical Physics in Theoretical Chemistry* (pp. 1-30). Elsevier.

[on-line] Morten Hjorth-Jensen (2021) *Advanced Topics in Computational Physics: Computational Quantum Mechanics* (último acceso 11-09-2024)

URL: https://compphysics.github.io/ComputationalPhysics2/doc/LectureNotes/_build/html/intro.html

[on-line] Pablo García Fernández (2023) Física cuántica: Estructura de moléculas y sólidos (último acceso 12-09-2024)

URL: https://personales.unican.es/garciapa/solids/intro.html

Nam, Y., Chen, J. S., Pisenti, N. C., Wright, K., Delaney, C., Maslov, D., ... & Kim, J. (2020). Ground-state energy estimation of the water molecule on a trapped-ion quantum computer. *npj Quantum Information*, *6*(1), 33.

Sharkey, K. L., Chancé, A., & Khan, A. (2022). *Quantum Chemistry and Computing for the Curious: Illustrated with Python and Qiskit® code*. Packt Publishing Ltd.

Hasanein, A. A., & Evans, M. W. (1996). *Computational methods in quantum chemistry* (Vol. 5). World Scientific.

Aspuru-Guzik, A., Dutoi, A. D., Love, P. J., & Head-Gordon, M. (2005). Simulated quantum computation of molecular energies. *Science*, *309*(5741), 1704-1707.

Sun, Q., Berkelbach, T. C., Blunt, N. S., Booth, G. H., Guo, S., Li, Z., ... & Chan, G. K. L. (2017). PySCF: the Python-based simulations of chemistry framework. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *8*(1), e1340.

Altland, A. & Simons, B. D. (2010), Second quantization, 2 edn, Cambridge University Press

Born, M. & Oppenheimer, R. (1927), 'Zur quantentheorie der molekeln', Annalen der Physik 389(20), 457–484.

Fock, V. (1932). Konfigurationsraum und zweite Quantelung. *Zeitschrift für Physik*, *75*(9), 622-647.

Cao, Y., Romero, J., Olson, J. P., Degroote, M., Johnson, P. D., Kieferová, M., ... & Aspuru-Guzik, A. (2019). Quantum chemistry in the age of quantum computing. *Chemical reviews*, *119*(19), 10856-10915.

[on-line] Peruzzo, A., McClean, J., Shadbolt, P. *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nat Commun* **5**, 4213 (2014). (último acceso 12-09-2024)

URL: https://doi.org/10.1038/ncomms5213