UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Quantum Simulation

Bachelor's Degree Thesis
Submitted to the Faculty at the
Escola Tècnica Superior
d'Enginyeria de Telecomunicació de Barcelona
of the Universitat Politècnica de Catalunya
by

Albert López Escudero

In partial fulfillment
of the requirements for the
Degree in Electronics Engineering

Advisor: Vincenzo de Maio & Ivona Bandric
Reviewer: Javier Rodrigez Fonollosa

Barcelona, December 2024

Resum

Cada exemplar del Treball de Fi de Grau (TFG) ha de contenir un Resum, que és un breu extracte del TFG. En termes d'estil, el Resum hauria de ser una versió reduïda del projecte: una introducció concisa, un compendi dels resultats i les principals conclusions o arguments presentats en el projecte. El Resum no ha de superar les 150 paraules i cal que estigui traduït al català, castellà i anglès.

Resumen

Cada ejemplar del Trabajo de Fin de Grado (TFG) debe incluir un Resumenque es un breve extracto del TFG. En cuanto al estilo, el Resumen debería ser una versión reducida del proyecto: una introducción breve, un resumen de los resultados principales y las conclusiones o argumentos principales presentados en el proyecto. El Resumen no debe exceder las 150 palabras y debe estar traducido al catalán, castellano e inglés.

Summary

Each copy of the Bachelor's Thesis (TFG) must include a Summary, which is a concise abstract of the TFG. In terms of style, the Summary should be a condensed version of the project: a brief introduction, a summary of the main results, and the conclusions or key arguments presented in the project. The Summary should not exceed 150 words and must be translated to catalan, spanish and english.

A Dedication page may be included in your thesis just before the Acknowledgments page, but it is not a requirement.

Acknowledgements

It is appropriate, but not mandatory, to declare the extent to which assistance has been given by members of the staff, fellow students, technicians or others in the collection of materials and data, the design and construction of apparatus, the performance of experiments, the analysis of data, and the preparation of the thesis (including editorial help). In addition, it is appropriate to recognize the supervision and advice given by your advisor.

Revision history and approval record

Revision	Date	Author(s)	Description
1.0	dd/mm/yyyy	AME	Document creation
1.1	dd/mm/yyyy	AME, JPV	Error correction
2.0	dd/mm/yyyy	AME, MLO	Revised after review
4.0	dd/mm/yyyy	AME	Final version

DOCUMENT DISTRIBUTION LIST

Role	Surname(s) and Name
[Student]	
[Project Supervisor 1]	
[Project Supervisor 2 (if applicable)]	

Written by:		**Reviewed and approved by:**	
Date	dd/mm/yyyy	Date	dd/mm/yyyy
Name	Xxxxxxx Yyyyyyy	Name	Xxxxxxx Yyyyyyy
Position	Project Author	Position	Project Supervisor

Contents

List of Figures

List of Tables

Abbreviations

ETSETB Escola Tècnica Superior d’Enginyeria de Telecomunicació de Barcelona

EU European Union

GEF Grau en Enginyeria Física

GREELEC Grau en Enginyeria Electrònica de Telecomunicació

GRETST Grau en Enginyeria de Tecnologies i Serveis de Telecomunicació

CHAPTER 1

Introduction

An Introduction that clearly states the rationale of the thesis that includes:

1. Statement of purpose (objectives).
2. Requirements and specifications.
3. Methods and procedures, citing if this work is a continuation of another project or it uses applications, algorithms, software or hardware previously developed by other authors.
4. Work plan with tasks, milestones and a Gantt diagram.
5. Description of the deviations from the initial plan and incidences that may have occurred.

The minimum chapters that this thesis document should have are described below, nevertheless they can have different names and more chapters can be added.

1.1 Work goals

The primary objective of this project is to contribute a new methodology to the realm of molecular simulations by:

- Developing a quantum simulation program capable of efficiently modeling molecules using quantum algorithms.
- Introducing innovative code that implements adaptive circuit construction and advanced optimization strategies.
- Demonstrating the effectiveness of our approach by simulating selected molecular systems and comparing the results with classical computational methods.
- Enhancing the integration between quantum and classical computations to optimize both electronic and nuclear degrees of freedom.

1.2 Requirements and specifications

To achieve these objectives, the following requirements and specifications have been established:

- **Quantum Computing Frameworks**: Utilize quantum computing libraries such as PennyLane and JAX for quantum circuit simulation and automatic differentiation.
- **Algorithm Implementation**: Implement the VQE algorithm with adaptive circuit construction, allowing the selection of the most significant excitations based on energy gradients.
- **Optimization Techniques**: Employ efficient optimizers, including gradient-based methods like gradient descent and advanced techniques like the Quantum Natural Gradient optimizer.
- **Hamiltonian Construction**: Accurately build molecular Hamiltonians for various molecular geometries and ensure compatibility with standard quantum chemistry basis sets.
- **Visualization Tools**: Develop modules for visualizing energy convergence, parameter evolution, and molecular geometries throughout the optimization process.
- **Computational Resources**: Ensure the code is optimized for performance, making effective use of computational resources and supporting parallel execution where possible.
- **Extensibility**: Design the codebase to be modular and extensible, allowing future enhancements and adaptation to other molecular systems or quantum algorithms.

1.3 Methods and procedures

This project introduces innovative methods and procedures to enhance molecular simulations using quantum computing. The key innovations in our code are centered around adaptive circuit construction, advanced optimization strategies, and the seamless integration of quantum and classical computations.

Adaptive Circuit Construction

Traditional VQE implementations often rely on fixed ansätze, which may not efficiently capture the complexities of all molecular systems. Our code implements an adaptive approach where the quantum circuit is dynamically constructed by selecting excitations (operators) from a predefined pool based on their contribution to lowering the system's energy. This selection is guided by computing the energy gradients with respect to each operator, ensuring that only the most impactful excitations are included in the circuit. This method enhances computational efficiency and can lead to faster convergence to the ground state energy.

Advanced Optimization Techniques

Optimizing the parameters of the quantum circuit is crucial for the success of the VQE algorithm. Our implementation leverages both gradient-based and gradient-free optimization methods. We utilize optimizers like the *Gradient Descent Optimizer* and explore advanced techniques such as the *Quantum Natural Gradient* optimizer, which accounts for the geometry of the parameter space and can provide faster convergence. Additionally, our code extends the optimization process to include the nuclear coordinates, allowing simultaneous optimization of the electronic structure and molecular geometry.

Integration of Quantum and Classical Computation

Our code exemplifies a robust integration of quantum and classical computational techniques. By employing automatic differentiation tools provided by frameworks like JAX and PennyLane, we can efficiently compute gradients of the energy with respect to both circuit parameters and nuclear coordinates. This integration facilitates the optimization process and enables the handling of complex systems that are challenging for classical methods alone.

Innovations in Code Implementation

Key innovations in our code include:

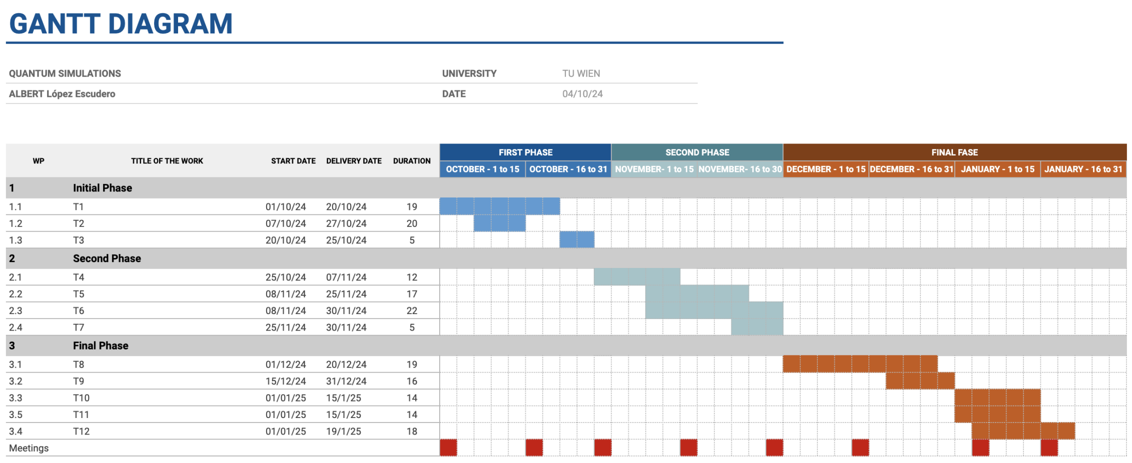
- **Dynamic Operator Selection**: A procedure to compute energy gradients for each operator in the pool and select the one with the highest impact, thus adaptively constructing the quantum circuit.
- **Hybrid Optimization Loop**: An iterative loop that updates both quantum circuit parameters and nuclear positions, enhancing the ability to find the global minimum energy configuration.
- **Efficient Gradient Computation**: Implementation of numerical methods to compute gradients with respect to nuclear coordinates, enabling geometry optimization within the VQE framework.
- **Visualization and Analysis Tools**: Development of comprehensive visualization functions to analyze the optimization process, including energy evolution plots and 3D representations of molecular geometries.
- **Modularity and Extensibility**: A modular code structure that allows for easy adaptation to different molecules, basis sets, and quantum devices, facilitating future research and development.

Utilization of Existing Frameworks and Contribution to the Field

While our work builds upon established quantum computing frameworks such as PennyLane and utilizes existing algorithms like the VQE, the innovations introduced in our code represent significant advancements in the field of molecular simulations. By enhancing the efficiency of quantum simulations and providing new methods for adaptive circuit construction and parameter optimization, this project contributes valuable tools and methodologies to researchers and practitioners in quantum chemistry and quantum computing.

1.4 Work plan

Normally the figures and tables are put in `\figure` and `\table` environments, that can float freely in the document. You can identify each float with a `\label`



GANTT DIAGRAM

QUANTUM SIMULATIONS		UNIVERSITY	TU WIEN
ALBERT López Escudero		DATE	04/10/24

WP	TITLE OF THE WORK	START DATE	DELIVERY DATE	DURATION
1	Initial Phase			
1.1	T1	01/10/24	20/10/24	19
1.2	T2	07/10/24	27/10/24	20
1.3	T3	20/10/24	25/10/24	5
2	Second Phase			
2.1	T4	25/10/24	07/11/24	12
2.2	T5	08/11/24	25/11/24	17
2.3	T6	08/11/24	30/11/24	22
2.4	T7	25/11/24	30/11/24	5
3	Final Phase			
3.1	T8	01/12/24	20/12/24	19
3.2	T9	15/12/24	31/12/24	16
3.3	T10	01/01/25	15/1/25	14
3.5	T11	01/01/25	15/1/25	14
3.4	T12	01/01/25	19/1/25	18
Meetings				

Figure 1.1: *Project's Gantt diagramGantt diagram of the project. For more information read the manual [**skalagantt**] of Skala..*

CHAPTER 2

State of the Art of the Technology Used or Applied in this Thesis

In recent years, computing has undergone significant evolution, reaching a point where improving the hardware of traditional devices presents considerable challenges. This has driven research in quantum computing, a technology that promises to revolutionize the field by enabling much more efficient calculations and superior processing capabilities. In applications such as molecular simulation, quantum computing has proven to be remarkably more efficient than classical computing, justifying investment in its development.

However, one of the main obstacles of quantum computing is the high cost associated with its devices. To run programs on a quantum computer, it must operate at extremely low temperatures, close to 0.1 Kelvin, which significantly increases the cost and technical complexity. Therefore, methods are being investigated to improve and optimize these devices, making them more accessible and viable for broader use.

Currently, one of the most practical ways to harness quantum computing's potential is through **quantum simulation**. Quantum simulation allows us to study complex quantum systems using either quantum simulators or classical computers. By emulating the behavior of quantum systems, we can explore quantum algorithms and applications effectively without necessarily requiring a fully functional quantum computer.

In this project, we focus on the operation of quantum simulators applied to molecular simulation, an area where quantum computing offers significant advantages. The main objective is to develop a program capable of simulating different molecules in a simple and efficient manner.

To this end, we will review the basic concepts of quantum mechanics that are essential to understand the fundamentals and potential of quantum computing, and explore the techniques of quantum simulation that allow us to harness quantum computing capabilities even with current technological limitations.

2.1 Quantum Simulation

Quantum simulation has emerged as an advanced and essential technique for studying complex quantum systems, especially those that are inaccessible or present great challenges for direct analysis using classical methods. Based on the proposal of Richard Feynman, who postulated that a computer built from quantum elements could overcome the limitations of classical computers in simulating quantum phenomena, quantum simulation has progressed significantly. It encompasses both digital and analog simulations and has expanded its applicability in various scientific areas.

There are mainly two approaches in quantum simulation: **Digital Quantum Simulation (DQS)** and **Analog Quantum Simulation (AQS)**. DQS employs the quantum circuit model, where systems are represented by qubits that evolve through quantum gates to reproduce the dynamics of the target system. This approach is universal, as it can, in principle, simulate any quantum system, although not always efficiently. On the other hand, AQS involves creating a quantum system that directly emulates the Hamiltonian of the system under study, allowing certain properties of the simulated system, such as time evolution, to be reproduced approximately. This method is particularly useful when a qualitative representation is required rather than high precision.

In addition to these approaches, there are algorithms inspired by quantum information theory that facilitate the classical simulation of quantum systems. Techniques such as **Matrix Product States (MPS)** and **Projected Entangled Pair States (PEPS)** allow representing particle systems on classical computers more efficiently than standard classical methods, optimizing the calculation of properties of complex quantum systems.

The applications of quantum simulation are broad and encompass multiple scientific fields. In condensed matter physics, it allows the study of models such as the Hubbard model and quantum phase transitions, fundamental for understanding phenomena like superconductivity. In quantum chemistry, it facilitates the calculation of molecular energies and complex chemical reactions. In high-energy physics and cosmology, it emulates particles in high-energy fields and cosmological phenomena. Furthermore, quantum simulation is instrumental in the analysis of open quantum systems and in the investigation of quantum chaos, allowing exploration of interactions with the environment and chaotic dynamics in the quantum realm.

However, quantum simulation faces significant challenges related to the precise control of the quantum simulator systems and the management of decoherence and errors, which can affect the accuracy of the results. The amount of required resources, such as the number of qubits and quantum gates, also depends on the size and complexity of the system to be simulated. It is estimated that quantum simulators require between 40 and 100 qubits to surpass the computational power of classical computers in specific problems. Despite these challenges, technological advances continue to improve the viability and efficiency of quantum simulation, promising to transform research in natural sciences and expand our understanding of quantum phenomena.

2.2 Key Concepts in Quantum Mechanics

It is essential to understand the difference between bits in classical computing and qubits in quantum computing to delve into this new technological paradigm.

In classical computing, the basic unit of information is the **bit**, which can take the value of 0 or 1. These bits are the foundation upon which conventional computers operate, processing information through combinations of these binary states.

In contrast, quantum computing uses the **qubit** or quantum bit as its basic unit. Unlike the classical bit, a qubit can exist in a superposition of states, meaning it can simultaneously represent the values 0 and 1 thanks to the principle of superposition in quantum mechanics. This property, along with phenomena such as quantum entanglement and interference, allows quantum computers to process information exponentially more efficiently for certain problems.

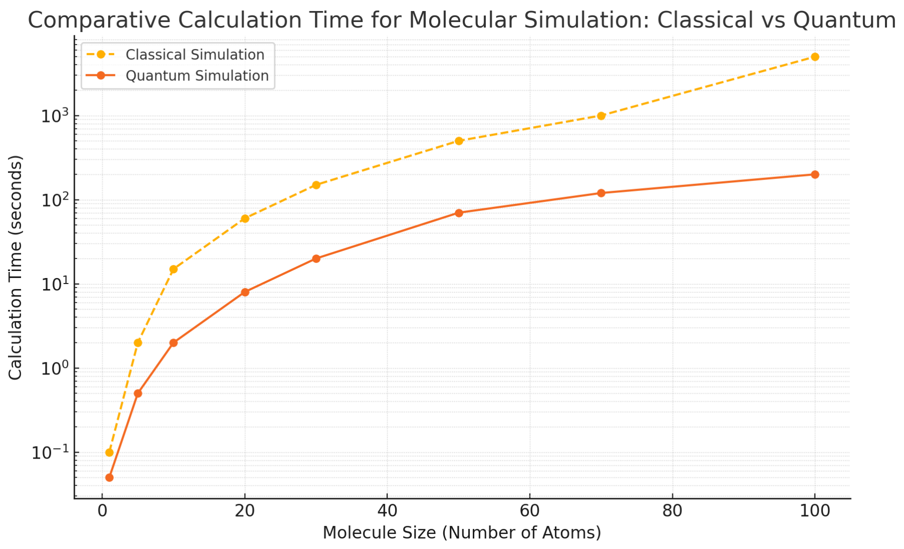


Figure 2.1: *Comparison of computation time for molecular simulations: classical vs quantum.*

Understanding how qubits operate and their differences from classical bits is essential to appreciate the revolutionary potential of quantum computing.

2.2.1 Qubit

The **qubit** is the basic unit of information in quantum computing. While the classical bit can only be in one of two states (0 or 1), a qubit can be in a superposition of both states simultaneously. This is due to the principle of quantum superposition, one of the fundamental characteristics of quantum mechanics.

Mathematically, a qubit is represented as a linear combination of the basis states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex numbers that satisfy the normalization condition $|\alpha|^2+|\beta|^2 = 1$. These coefficients indicate the probability amplitudes of finding the qubit in the states $|0\rangle$ or $|1\rangle$ upon measurement.

In addition to superposition, qubits can exhibit **quantum entanglement**, a property that allows creating strong correlations between qubits that cannot be explained by classical physics. Entanglement is essential for the computational power of quantum computers, as it enables processing and storing an exponentially larger amount of information than classical systems.

For example, while a classical system of n bits can represent one of 2^n possible state combinations, a quantum system of n qubits can represent a superposition of all those combinations simultaneously. This capability is what allows quantum computers to tackle complex problems more efficiently.

However, manipulating and maintaining qubits is a significant technical challenge. Qubits are extremely sensitive and can be affected by interactions with the environment, leading to **quantum decoherence**. To minimize this effect and preserve quantum properties, it is necessary to keep systems in controlled conditions, such as very low temperatures, close to absolute zero.

2.2.2 Quantum Superposition

Quantum superposition allows a quantum system to exist in multiple states simultaneously until a measurement is performed. This characteristic is key to the functioning of quantum computers, as it enables processing a large amount of information in parallel.

In quantum systems, superposition is combined with **quantum interference**, where the probability amplitudes of states can reinforce or cancel each other out. This phenomenon is exploited in quantum algorithms to increase the probability of obtaining the correct result. For example, in Grover's algorithm, constructive interference amplifies the probability of the desired state, significantly improving the efficiency of searching for elements in an unsorted database.

Superposition is especially useful in simulating complex molecular systems. Quantum computers can naturally model the superpositions of electronic states in molecules, which is crucial for studying chemical reactions and molecular properties that are difficult to address with classical methods due to the exponential growth of computational resources required.

2.2.3 Quantum Decoherence

Quantum decoherence is one of the main challenges in quantum computing. It refers to the loss of a system's quantum properties, such as superposition and entanglement, due to unwanted interactions with the environment. This loss causes the quantum system to transition toward classical behavior, affecting the accuracy and reliability of quantum calculations.

Qubits are extremely sensitive to external disturbances, such as electromagnetic fluctuations, vibrations, and temperature changes. These interactions can cause quantum states to mix with those of the environment, leading to a loss of coherence that is irreversible and degrades the stored quantum information.

To mitigate the effects of decoherence, various strategies are implemented:

- **System Isolation**: Designing physical systems that minimize unwanted interactions with the environment, using materials and techniques that protect qubits from external disturbances.
- **Quantum Error Correction**: Implementing error correction codes that allow detecting and correcting errors without directly measuring the qubit's state, thereby preserving quantum information.
- **Dynamic Control**: Applying techniques such as pulse refocusing and dynamic pulse sequences that actively compensate for disturbances and extend the coherence time of qubits.

Controlling and mitigating decoherence are essential for the advancement of quantum computing and its application in areas like molecular simulation, where the precision of calculations is fundamental.

2.3 The Hamiltonian in Quantum Mechanics

The Hamiltonian is a fundamental concept originating from classical mechanics, introduced by William Rowan Hamilton in 1833. Hamiltonian mechanics is a reformulation of classical mechanics that provides powerful tools for studying the dynamics of systems. The Hamiltonian function represents the total energy of the system, expressed in terms of generalized coordinates and momenta, and is given by the sum of the kinetic and potential energies.

In quantum mechanics, the **Hamiltonian operator** plays a central role in describing the energy and time evolution of quantum systems. It represents the total energy of the system, including both kinetic and potential energies, and is essential for formulating the Schrödinger equation.

2.3.1 Mathematical Definition

The Hamiltonian operator, commonly denoted as $\hat{H}$, is a self-adjoint operator acting on the Hilbert space associated with the quantum system. For a single particle in one dimension, the Hamiltonian is expressed as:

$$\hat{H} = \hat{T} + \hat{V}$$

where:

- $\hat{T}$ is the kinetic energy operator.
- $\hat{V}$ is the potential energy operator.

In terms of the position $\hat{x}$ and momentum $\hat{p}$ operators, these are defined as:

$$\hat{T} = \frac{\hat{p}^2}{2m} = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2}$$

$$\hat{V} = V(\hat{x})$$

Here, m is the mass of the particle, $\hbar$ is the reduced Planck constant, and $V(\hat{x})$ is the potential energy function depending on position.

2.3.2 Role in the Schrödinger Equation

The Hamiltonian is central to the Schrödinger equation, which describes how the quantum state of a system evolves over time. The time-dependent Schrödinger equation is expressed as:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$$

where $|\psi(t)\rangle$ is the state vector of the system at time t. For time-independent systems, the Schrödinger equation reduces to the eigenvalue equation:

$$\hat{H}|\psi\rangle = E|\psi\rangle$$

Here, E represents the eigenvalues of the Hamiltonian, corresponding to the allowed energy levels of the system, and $|\psi\rangle$ are the associated eigenstates.

2.3.3 Hamiltonian in Multi-Particle Systems

For systems with multiple particles, the Hamiltonian includes additional terms representing interactions between particles. For example, for a system of two particles, the Hamiltonian is expressed as:

$$\hat{H} = \hat{T}_1 + \hat{T}_2 + \hat{V}_1 + \hat{V}_2 + \hat{V}_{12}$$

where:

- $\hat{T}_1$ and $\hat{T}_2$ are the kinetic energy operators of particles 1 and 2, respectively.
- $\hat{V}_1$ and $\hat{V}_2$ are the individual potential energy operators.
- $\hat{V}_{12}$ represents the potential interaction between the two particles.

2.3.4 Importance in Quantum Simulations

In quantum simulations, especially in algorithms like the Variational Quantum Eigensolver (VQE), the Hamiltonian is decomposed into a sum of simpler terms, often expressed in terms of Pauli operators. This decomposition facilitates implementation on quantum circuits and allows estimating the system's energy through measurements on qubits.

Understanding the structure and properties of the Hamiltonian is essential for modeling and simulating quantum systems, as it determines the possible energies and dynamics of the system under study.

2.4 Algorithms

Once we have understood the basic concepts, we focus on the quantum algorithms used in particle simulation. These algorithms make use of quantum logic gates, which are detailed in Appendix A.

2.4.1 VQE: Variational Quantum Eigensolver

The VQE is a hybrid quantum-classical algorithm designed to find the minimum energy of a quantum system, such as a molecule. This algorithm combines quantum state preparation with classical optimization. It leverages the quantum properties of qubits to find the ground state of the molecule more rapidly, followed by classical optimization to refine the solution.

Stages of the Algorithm

El VQE is based on the variational principle, which states that the expected energy of any approximate state $|\psi(\theta)\rangle$ is always greater than or equal to the energy of the true ground state E_0:

$$E(\theta) = \langle \psi(\theta)|H|\psi(\theta)\rangle \geq E_0$$

1. Quantum State Preparation A parameterized quantum circuit known as an *ansatz* is used, defined by a set of adjustable parameters $\vec{\theta}$. This circuit applies a sequence of quantum gates, such as rotations and entangling gates, to generate a quantum state:

$$|\psi(\vec{\theta})\rangle$$

2. Measurement of the Expected Energy For a given quantum state $|\psi(\vec{\theta})\rangle$, the expected energy of the system under a Hamiltonian H is measured:

$$E(\vec{\theta}) = \langle\psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle$$

The Hamiltonian H is decomposed into terms of Pauli operators representing the electronic and nuclear interactions in the system.

3. Classical Optimization The parameters $\vec{\theta}$ are iteratively adjusted using a classical optimizer to minimize $E(\vec{\theta})$.

4. Iteration of the Process The steps of state preparation, measurement, and optimization are repeated until $E(\vec{\theta})$ converges to a minimum value. This value corresponds to the ground state energy of the system.

2.4.2 Adaptive Circuits

Adaptive circuits allow optimizing the design of quantum circuits for specific problems. They dynamically adjust their structure based on feedback and optimization criteria during the execution process.

An example is found in *Variational Quantum Algorithms* (VQA), which use classical optimization techniques to adjust the parameters of a quantum circuit and minimize a cost function. These algorithms are especially useful in current quantum devices, which are noisy and of limited size (NISQ).

For more information and practical examples on adaptive circuits:

https://pennylane.ai/qml/demos/tutorial_adaptive_circuits/

2.5 Optimizers

In the realm of quantum simulation algorithms like the **Variational Quantum Eigensolver** (VQE), optimizers are essential components that facilitate the minimization of the expected energy of a quantum system. Following the preparation of quantum states and the measurement processes described earlier, optimizers are employed in the classical computation stage to adjust the parameters of the quantum circuit, known as the *ansatz*.

The theoretical purpose of optimizers in this project is to solve a continuous optimization problem. They aim to find the optimal set of parameters $\vec{\theta}$ that minimize the cost function $E(\vec{\theta})$, which represents the expectation value of the Hamiltonian H with respect to the quantum state $|\psi(\vec{\theta})\rangle$:

$$E(\vec{\theta}) = \langle\psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle$$

Optimizers utilize mathematical techniques to navigate the high-dimensional parameter space effectively. Depending on the specific characteristics of the problem, different optimization methods can be employed:

- **Gradient-based methods**: These methods compute the gradient of the cost function with respect to the parameters and use this information to guide the search towards the minimum energy. Examples include gradient descent and its variants.
- **Gradient-free methods**: In cases where calculating the gradient is impractical or the cost function is noisy, gradient-free methods like Nelder-Mead or COBYLA can be used.
- **Second-order methods**: These methods, such as the BFGS algorithm, approximate the second derivatives (Hessian) of the cost function to achieve faster convergence.

Within the iterative loop of the VQE algorithm, the optimizer updates the parameters $\vec{\theta}$ after each quantum measurement based on the chosen optimization strategy. This process continues until convergence is achieved, meaning the expected energy $E(\vec{\theta})$ reaches a minimum value that approximates the ground state energy of the system.

Integrating optimizers into the quantum-classical workflow is crucial because they connect quantum computations with classical numerical methods. They enable effective exploration of the parameter space, addressing challenges such as multiple local minima and flat regions in the energy landscape that are inherent in quantum systems.

By employing appropriate optimization techniques, this project aims to enhance the efficiency and accuracy of molecular simulations. Optimizers play a pivotal role in leveraging the capabilities of quantum computing to achieve results that are difficult to obtain with classical computational methods alone, thus contributing to the advancement of quantum simulation despite current technological limitations.

CHAPTER 3

Methodology / project development

In this chapter, the methodology used in the completion of the work will be detailed. Its aim is to offer a thorough account of the approaches and techniques used, ensuring replicability and academic rigor. It will not only cover the research methods and measurement techniques employed but will also delve into the specifics of software and hardware development. Whether the project involves qualitative analysis, quantitative measurements, computational modeling, or physical prototyping, this chapter should elucidate how each component contributes to the overall objectives.

In addition to describing the methods themselves, the chapter will also provide justifications for why specific methods were chosen over others. For example, it may explain the choice of a particular programming language, statistical test, or experimental setup. The chapter will also address the limitations of the methodology and how these have been mitigated or accounted for. Readers should come away with a clear understanding of how the project's development has been carried out, why certain choices were made, and how these methods serve to fulfill the initially established objectives.

3.1 Elección framework

Para poder efectuar la decision de que framework utliziar. Estubimos comparando las documentaciones de los dos frameworks de simulación cuantica del mercado. PennyLane y Quiskit. Estos son los frameworks mas completo, con carateristicas similares, que existan en el momento de la cración de este proyeto. Después de estar revisando la documentación, al final nos decantamos por el uso de PennyLane debido a 2 razones.

La primera razón fue la cantidad de documentación referente a la simulación quantica. Una vez empezamos a buscar como otra gente estaba utilizando estos recursos, nos dimos cuenta, que para el ambito de simulación molecular. La documentación existente, tanto teorica pero sobretodo practica era sustancialmente mayor. Disponienso así de mas ejemplos para poder empezar a desarrollar nuestro proyecto.

La segundo razón de nuestra elección fue los grandes cambios que efectua regularmente

Qiskit, nos dimos cuenta, que Qiskit, es una herramienta que promete ser muy buena, aun asi, historicamente ha tenido grandes cambios significativos en su estructura.

Es por estas razones, que finalmente, este proyecto se ha desenvolupado con el framewerk, PennyLane. A continuación observaremos como ha sido el desarrollo del proyecto y explicaremos el porque de las decisiones tomadas.

3.2 Estructuración del Proyecto

3.2.1 Organización del Código

```
                    quantum_simulation_project/
    config/
         config_functions.py: Funciones de configuración para el proyecto.
         molecules.json: Datos de moléculas.
         __pycache__: Archivos de caché de Python.
    main.py: Archivo principal del programa.
    modules/
         ansatz_preparer.py: Preparación del ansatz cuántico.
         hamiltonian_builder.py: Construcción del Hamiltoniano molecular.
         molecule_manager.py: Gestión de datos moleculares.
         opt_mol.py: Optimización molecular.
         optimizer.py: Algoritmos de optimización.
         visualizer.py: Herramientas de visualización.
         __pycache__: Archivos de caché de Python.
    temp_results_autograd/
         energy_evolution.png: Gráfica de la evolución de la energía.
         filtered_report_autograd.txt: Informe filtrado de resultados.
         final_geometries_3D.png: Imagen de las geometrías finales en 3D.
         nuclear_coordinates.png: Coordenadas nucleares.
         output.txt: Salida de datos del programa.
         profile_output_autograd.txt: Perfil de salida de Autograd.
    test/: Directorio para pruebas.
```

Estructura Modular

Después de decidir la interfaz a utilizar, con la primera version de codigo implementada, decidimos hacer una redistribución para que el proyecto fuera mas preciso, con una estructura mas modular, ofreciendo la posibilidad de poder añadir de forma senzilla mas lineas de codigo.

Descripción de Archivos y Directorios

- **Archivo Principal (`quantum_simulation.py`)**: Explica su función como punto de entrada del programa.
- **Módulos Auxiliares (`mol_optimizer.py`)**: Detalla las funciones y clases incluidas, y cómo se dividen las responsabilidades.
- **Archivos de Datos (`data/`)**: Describe cómo y dónde se almacenan los datos de entrada y resultados.
- **Dependencias (`requirements.txt`)**: Menciona la importancia de gestionar las dependencias para la reproducibilidad.

3.2.2 Herramientas y Tecnologías Utilizadas

Control de Versiones

El control de las distintas versiones, se efectuó con el uso de Git, permitiendo un seguimiento detallado de los cambios y facilitando la colaboración en el proyecto de los supervisores de forma continua. Principalmente, al inició del proyecto, se utilizo una unica rama para desenvolupar el proyecto e ir conociendo las possibilidades del framework. Una vez se tuvo una version estable, se crearon ramas para poder hacer pruebas y desarrollos de nuevas funcionalidades, las primeras ramas creadas furon las distintas versiones de las interfazes. En cada una de las ramas se depuro el codigo para que utilizaran corrieran el mismo codigo en las distintas interfazes. Finalmente, se volvierón a unir con a la rama principal, unicamente los cambios de la interfaz que habia salido que era la mas adecuada para el proyecto.

3.3 Desarrollo e Implementación

3.3.1 Algoritmos y Métodos Implementados

Variational Quantum Eigensolver (VQE)

El **Variational Quantum Eigensolver (VQE)** es un algoritmo híbrido cuántico-clásico diseñado para encontrar la energía del estado fundamental de un sistema cuántico, como moléculas en química cuántica. Combina un circuito cuántico parametrizado con un optimizador clásico para minimizar la expectativa del Hamiltoniano del sistema. Este enfoque es particularmente relevante en la estimación de energías moleculares debido a las limitaciones de los dispositivos cuánticos actuales y la complejidad computacional de los métodos clásicos exactos.

Principio del VQE: El VQE se basa en el principio variacional, que establece que la energía esperada de cualquier estado aproximado $|\psi(\theta)\rangle$ es siempre mayor o igual a la energía del estado fundamental real E_0:

$$E(\theta) = \langle \psi(\theta)|H|\psi(\theta)\rangle \geq E_0$$

Donde H es el Hamiltoniano del sistema y θ representa un conjunto de parámetros variacionales. El objetivo es encontrar los valores óptimos de θ que minimizan $E(\theta)$, acercándose lo más posible a E_0.

Relevancia en la estimación de energías moleculares: La estimación precisa de las energías moleculares es esencial para comprender las propiedades químicas y predecir reacciones. Los métodos clásicos exactos, como la Interacción de Configuraciones Completa (FCI), son computacionalmente inviables para sistemas grandes debido al escalamiento exponencial con el tamaño del sistema. El VQE ofrece una alternativa eficiente al aprovechar las capacidades de los dispositivos cuánticos para explorar espacios de estados cuánticos más grandes con recursos computacionales reducidos.

Implementación en el código: En el código proporcionado, el VQE se implementa a través de varias funciones y pasos clave:

1. **Preparación del ansatz cuántico:**

   La función `prepare_ansatz` construye el circuito cuántico utilizando las excitaciones seleccionadas y los parámetros actuales:

   ```
   def prepare_ansatz(params, hf_state, selected_excitations, spin_orbitals):
       qml.BasisState(hf_state, wires=range(spin_orbitals))
       for i, exc in enumerate(selected_excitations):
           if len(exc) == 2:
               qml.SingleExcitation(params[i], wires=exc)
           elif len(exc) == 4:
               qml.DoubleExcitation(params[i], wires=exc)
   ```

   Este ansatz comienza desde el estado de Hartree-Fock y aplica una serie de operaciones de excitación (simple y doble) parametrizadas por θ.

2. **Definición de la función de costo:**

   Dentro de `update_parameters_and_coordinates`, se define la función de costo que calcula el valor esperado del Hamiltoniano:

   ```
   @qml.qnode(dev, interface=interface)
   def cost_fn(params):
       prepare_ansatz(params, hf_state, selected_excitations, spin_orbitals)
       return qml.expval(hamiltonian)
   ```

   Esta función es esencial para evaluar $E(\theta)$ dado un conjunto de parámetros.

3. **Optimización de los parámetros:**

   Se utilizan optimizadores clásicos, como `GradientDescentOptimizer`, para ajustar los parámetros y minimizar la energía:

   ```
   params, energy = opt.step_and_cost(cost_fn, params)
   ```

   Este paso actualiza los parámetros θ para encontrar el mínimo de $E(\theta)$.

4. **Ciclo de optimización:**

   El bucle principal dentro de `optimize_molecule` itera sobre:

   - Construcción del Hamiltoniano molecular.
   - Cálculo de los gradientes de los operadores.
   - Selección del operador más significativo basado en los gradientes.
   - Actualización de los parámetros del ansatz y las coordenadas nucleares.

   Este ciclo continúa hasta que se alcanza el criterio de convergencia o se completa el número máximo de iteraciones.

Requerimientos para ejecutar el VQE:

- **Hamiltoniano molecular preciso:** Representa las interacciones electrónicas y nucleares del sistema.
- **Ansatz adecuado:** Capaz de aproximar el estado fundamental del sistema.
- **Capacidad para evaluar el valor esperado de la energía:** Utilizando mediciones cuánticas.
- **Optimizador clásico eficiente:** Para ajustar los parámetros del ansatz y minimizar la energía.

Preparación del Estado de Hartree-Fock

El estado de Hartree-Fock es una aproximación donde los electrones se consideran independientes, ocupando orbitales moleculares en un campo medio creado por todos los demás electrones. Este estado se utiliza comúnmente como punto de partida en métodos post-Hartree-Fock y algoritmos cuánticos como el VQE.

Proceso de generación del estado inicial:

1. **Cálculo del número de electrones y orbitales:**

   La función `initialize_molecule` determina el número de electrones y orbitales de spin basándose en los símbolos atómicos y las coordenadas iniciales:

```
def initialize_molecule(symbols, x_init, charge=0, mult=1, basis_name='sto-3g'):
    ...
    electrons = molecule.n_electrons
    n_orbitals = len(molecule.basis_set)
    spin_orbitals = 2 * n_orbitals
    ...
    return electrons, spin_orbitals
```

Aquí, se considera que cada orbital espacial puede albergar dos electrones con espines opuestos, lo que resulta en el doble de orbitales de spin.

2. **Generación del estado de Hartree-Fock:**

   Utilizando la función `generate_hf_state`, se crea el estado de referencia de Hartree-Fock:

```
def generate_hf_state(electrons, spin_orbitals):
    hf_state = qml.qchem.hf_state(electrons, spin_orbitals)
    print(hf_state)
    return hf_state
```

   Esta función genera un vector binario donde los primeros n bits (siendo n el número de electrones) están en estado '1' (ocupado) y el resto en '0' (vacío), representando la ocupación de los orbitales en el estado de Hartree-Fock.

Conceptos teóricos relacionados:

- **Aproximación de Hartree-Fock:** Simplifica el problema de muchos electrones al asumir que cada electrón se mueve en el potencial promedio creado por los demás, ignorando la correlación electrónica.
- **Determinante de Slater:** Forma matemática que garantiza que la función de onda sea antisimétrica bajo el intercambio de electrones, cumpliendo con el principio de exclusión de Pauli.
- **Orbitales moleculares:** Combinaciones lineales de orbitales atómicos que los electrones pueden ocupar en una molécula.

Requerimientos:

- Información precisa sobre la geometría molecular y los números atómicos para calcular el número de electrones.
- Selección de una base atómica adecuada para representar los orbitales moleculares.
- Uso de herramientas computacionales (como PennyLane) para generar el estado de Hartree-Fock.

Construcción del Hamiltoniano Molecular

El Hamiltoniano molecular es una representación matemática que describe la energía total del sistema molecular, incluyendo las contribuciones de energía cinética y las interacciones entre electrones y núcleos. Es fundamental para cualquier cálculo de estructura electrónica y simulaciones de química cuántica.

Proceso de construcción del Hamiltoniano:

1. **Definición de la geometría molecular:**

   La geometría se especifica mediante los símbolos atómicos y las coordenadas cartesianas de cada átomo en la molécula:

   ```
   symbols = ['H', 'H']  # Ejemplo para una molécula de hidrógeno
   x_init = np.array([0.0, 0.0, 0.0,  # Coordenadas del primer átomo
                      0.0, 0.0, 0.74])  # Coordenadas del segundo átomo
   ```

2. **Elección de la base atómica:**

   Se selecciona una base, como 'sto-3g', que es un conjunto de funciones de base predefinidas para representar los orbitales atómicos de manera simplificada.

3. **Cálculo de integrales electrónicas:**

   Las integrales de uno y dos electrones se calculan para obtener los elementos de matriz del Hamiltoniano en la base escogida. Esto incluye:

   - Integrales de energía cinética de los electrones.
   - Integrales de potencial nuclear-electrón.
   - Integrales de repulsión electrón-electrón.

4. **Construcción del Hamiltoniano:**

   La función `build_hamiltonian` genera el Hamiltoniano molecular utilizando las funciones de PennyLane:

   ```
   def build_hamiltonian(x, symbols, charge=0, mult=1, basis_name='sto-3g'):
       x = np.array(x)
       coordinates = x.reshape(-1, 3)
       hamiltonian, qubits = qml.qchem.molecular_hamiltonian(
           symbols, coordinates, charge=charge, mult=mult, basis=basis_name
       )
       h_coeffs, h_ops = hamiltonian.terms()
       h_coeffs = np.array(h_coeffs)
       hamiltonian = qml.Hamiltonian(h_coeffs, h_ops)
       return hamiltonian
   ```

Esta función realiza:

- Cálculo de las integrales electrónicas.
- Transformación del Hamiltoniano al formato de operadores cuánticos (por ejemplo, utilizando la transformación de Jordan-Wigner).
- Devolución del Hamiltoniano como una suma de términos de operadores de Pauli con sus respectivos coeficientes.

Conceptos teóricos relacionados:

- **Hamiltoniano electrónico molecular:** Incluye términos de energía cinética electrónica, atracción núcleo-electrón y repulsión electrón-electrón.
- **Bases atómicas:** Conjuntos de funciones matemáticas (como funciones de tipo Gaussiano) utilizadas para aproximar los orbitales atómicos en cálculos cuánticos.
- **Segundo cuantización y transformación a qubits:** El Hamiltoniano se expresa en términos de operadores de creación y destrucción y luego se transforma a operadores de qubits para su implementación en circuitos cuánticos.

Requerimientos:

- **Geometría molecular precisa:** Necesaria para calcular las distancias y energías de interacción.
- **Selección adecuada de la base atómica:** Afecta la precisión y el costo computacional del cálculo.
- **Herramientas computacionales avanzadas:** Para el cálculo de integrales electrónicas y la construcción del Hamiltoniano (e.g., PennyLane QChem).
- **Transformación de operadores:** Uso de transformaciones como Jordan-Wigner o Bravyi-Kitaev para mapear operadores de fermiones a qubits.

Implementación detallada en el código: El código utiliza las capacidades de `qml.qchem` para automatizar muchos de los pasos complejos:

- `qml.qchem.molecular_hamiltonian` calcula las integrales electrónicas y construye el Hamiltoniano.
- El Hamiltoniano se descompone en términos que pueden ser medidos en un circuito cuántico.
- Se considera la carga y multiplicidad de la molécula, lo que permite estudiar moléculas neutras o iones con diferentes estados de espín.

Relación con los conceptos teóricos: La construcción del Hamiltoniano en el código refleja directamente los pasos teóricos necesarios para modelar un sistema molecular. Al transformar el Hamiltoniano a un formato compatible con qubits, se cierra la brecha entre la teoría química y la implementación práctica en un dispositivo cuántico.

Conclusión

El código proporcionado implementa el algoritmo VQE para estimar la energía del estado fundamental de una molécula, comenzando desde el estado de Hartree-Fock y construyendo el Hamiltoniano molecular a partir de la geometría y la base seleccionada. Cada paso del código está estrechamente relacionado con conceptos teóricos clave en química cuántica y computación cuántica, proporcionando una metodología completa que va desde la definición del sistema molecular hasta la optimización de los parámetros en un circuito cuántico. Esta implementación demuestra cómo los algoritmos cuánticos pueden aplicarse para resolver problemas complejos en química, aprovechando tanto la teoría como las herramientas computacionales modernas.

3.3.2 Optimización de Parámetros y Geometría

Selección de Operadores

Explica cómo se generó el pool de operadores y se seleccionaron las excitaciones más relevantes.

Optimización de Parámetros Variacionales

Detalla el proceso de ajuste de los parámetros del circuito cuántico para minimizar la energía esperada.

Actualización de Coordenadas Moleculares

Describe cómo se integró la optimización geométrica en el algoritmo, ajustando las posiciones atómicas para encontrar la geometría de menor energía.

3.3.3 Modificaciones y Decisiones Técnicas

Cambio de Optimizadores

Justifica la transición de optimizadores basados en JAX y Optax a los optimizadores nativos de PennyLane.

Uso de la Interfaz Autograd

Explica los beneficios de utilizar autograd para el cálculo automático de gradientes con arrays de NumPy.

Gestión del Estado del Optimizador

Detalla cómo se manejó el estado interno de los optimizadores de PennyLane y las simplificaciones realizadas en el código.

3.4 Registro y Análisis del Rendimiento

3.4.1 Registro de Tiempos de Ejecución

Motivación

Explica la importancia de medir los tiempos de ejecución para comparar el rendimiento de diferentes optimizadores y configuraciones.

Implementación

Describe la función `write_simulation_times` y cómo se aseguró la correcta escritura y actualización de los datos en el archivo `execution_times.csv`.

Manejo de Datos Duplicados

Detalla la solución implementada para evitar la superposición de columnas y actualizar los resultados de simulaciones previas.

3.4.2 Análisis de Resultados

Comparación de Optimizadores

Explica cómo se evaluó el desempeño de los diferentes optimizadores en términos de velocidad de convergencia y precisión.

Interpretación de Métricas

Describe las métricas utilizadas, como el tiempo total de optimización y el valor final de la energía.

3.5 Pruebas y Validación

3.5.1 Casos de Prueba

Moléculas Sencillas

Justifica la elección de moléculas como el hidrógeno molecular (H_2) para validar el funcionamiento básico del código.

Resultados Esperados vs. Obtenidos

Compara los resultados obtenidos con valores teóricos o de referencia para verificar la precisión de las simulaciones.

3.5.2 Verificación del Código

Depuración y Manejo de Errores

Describe las técnicas de depuración empleadas para identificar y corregir errores en el código.

Pruebas Unitarias

Explica si se implementaron pruebas unitarias o de integración para asegurar la correcta funcionalidad de los componentes clave.

3.6 Limitaciones y Consideraciones

3.6.1 Complejidad Computacional

Discute las limitaciones en términos de recursos computacionales y cómo afectan la escalabilidad a moléculas más grandes.

3.6.2 Aproximaciones Utilizadas

Reconoce las aproximaciones inherentes al método VQE y al uso de bases reducidas como STO-3G.

3.6.3 Dependencia de Herramientas Externas

Menciona cómo las actualizaciones en los frameworks o bibliotecas podrían afectar la reproducibilidad y estabilidad del proyecto.

3.7 Futuras Mejoras y Trabajos Relacionados

3.7.1 Optimización de Recursos

Propón estrategias para optimizar el uso de recursos, como el empleo de simuladores cuánticos más eficientes o la paralelización de tareas.

3.7.2 Extensión a Moléculas Más Complejas

Sugiere cómo podría ampliarse el enfoque para simular sistemas moleculares más grandes y complejos.

3.7.3 Integración con Hardware Cuántico

Explora la posibilidad de ejecutar las simulaciones en dispositivos cuánticos reales y los desafíos asociados.

3.8 Conclusión de la Metodología

3.8.1 Resumen de la Estrategia

Recapitula los pasos clave de la metodología y cómo contribuyen al cumplimiento de los objetivos del proyecto.

3.8.2 Reflexión Personal

Comparte aprendizajes significativos obtenidos durante el desarrollo y cómo las decisiones tomadas afectaron positivamente el resultado final.

Una vez decidido el framework, el siguiente paso fue estructurar el proyecto para que fuera escalable. Dentro del

3.9 Code Listings

The `tcblistings` environment of the `tcolorbox` package ([**tcolorbox**]) inserts code listings generated by the `listings` or `minted` packages into a `tcolorbox`, thereby achieving very well-presented and highly configurable listings.

As an example, the following is the listing of a program in Python language that implements the "Sieve of Eratosthenes" algorithm for calculating prime numbers less than a certain given number. Note that the "listings" package is capable of automatically interpreting and highlighting the language's syntax.

Python Example: Sieve of Eratosthenes

```python
def sieve_of_eratosthenes(limit):
  primes = []
  sieve = [True] * (limit + 1)
  for num in range(2, limit + 1):
    if sieve[num]:
      primes.append(num)
      for multiple in range(num*num, limit + 1, num):
        sieve[multiple] = False
  return primes

# Example usage:
if __name__ == "__main__":
```

```
limit = 30
print(f"The prime numbers up to {limit} are:
        {sieve_of_eratosthenes(limit)}")
```

CHAPTER 4

Results

This chapter should encompass your data analysis and findings. Additionally, include relevant tables, figures, and citations to support your results and interpretations. Here is a suggested list of topics to discuss:

4.1 Experiments and Tests

Describe the experiments conducted to assess the performance of your project. Explain how you collected and processed the data.

4.2 Data Visualization

Create visual representations of the results (e.g., scatter plots, bar charts). Interpret the visualizations and relate them to the research questions.

4.3 Limitations

Acknowledge any limitations in the data or analysis. Explain how these limitations may have influenced the results.

CHAPTER 5

Sustainability Analysis and Ethical Implications

Starting from the academic year 2023-24, the TFG regulations of ETSETB require the inclusion of a sustainability report in the project's documentation. This analysis involves an assessment of environmental, social, and economic impacts, as well as potential ethical implications resulting from the completion of the TFG. In the case where the TFG involves a product/service/system/building, etc., that could be implemented, the analysis should also address the impacts that the proposal would have during the various stages of its lifecycle.

Detailed instructions on what the sustainability report should contain and how to prepare it can be found on the ATENEA platform.

IMPORTANT: Please note that the previous chapter on "Project Budget" is now integrated into the sustainability analysis, specifically in the cells "Economic Cell/Development of BT" and "Economic Cell/Project Execution".

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

- Summarize the main results of your work.
- Discuss the degree of achievement in relation to the objectives set at the beginning of the work.
- Highlight the contributions of your work to the field of study.

6.2 Future Directions

- Identify areas for future research or development based on your work.
- Discuss possible ways to expand or improve the project.
- Consider questions that remained unanswered and opportunities for future exploration.

APPENDIX A

Logic Gates

A.1 Simple Logic Gates

Below are detailed the simple logic gates essential for constructing more complex quantum algorithms:

X Gate (Pauli-X) The **Pauli-X** gate is the quantum analog of the classical NOT gate. It performs a bit flip on the qubit, transforming the state $|x\rangle$ into $|\neg x\rangle$.

Representative Matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Effect on Basis States

- $X|0\rangle = |1\rangle$
- $X|1\rangle = |0\rangle$

Y Gate (Pauli-Y) The **Pauli-Y** gate performs a rotation of π around the y-axis. It transforms the state $|x\rangle$ into $i(-1)^x|\neg x\rangle$.

Representative Matrix

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Effect on Basis States

- $Y|0\rangle = i|1\rangle$
- $Y|1\rangle = -i|0\rangle$

Z Gate (Pauli-Z) The **Pauli-Z** gate is known as the phase inversion gate. It transforms the state $|x\rangle$ into $(-1)^x|x\rangle$.

Representative Matrix	Effect on Basis States
$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	• $Z\|0\rangle = \|0\rangle$ • $Z\|1\rangle = -\|1\rangle$

Hadamard Gate (H) The **Hadamard** gate creates an equal superposition of the computational basis states. It transforms the state $|x\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^x |1\rangle)$.

Representative Matrix	Effect on Basis States
$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	• $H\|0\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle) = \|+\rangle$ • $H\|1\rangle = \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle) = \|-\rangle$

A.2 Multi-Qubit Logic Gates

Controlled-NOT Gate (CNOT) The **CNOT** or **Controlled-X** gate is a two-qubit gate that flips the second qubit (target) if and only if the first qubit (control) is in the state $|1\rangle$. It transforms the state $|x, y\rangle$ into $|x, x \oplus y\rangle$, where $\oplus$ denotes the XOR operation.

Representative Matrix	Effect on Basis States
$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	• $\text{CNOT}\|00\rangle = \|00\rangle$ • $\text{CNOT}\|01\rangle = \|01\rangle$ • $\text{CNOT}\|10\rangle = \|11\rangle$ • $\text{CNOT}\|11\rangle = \|10\rangle$

Single Excitation Gate (*SingleExcitation*) This gate performs a rotation in the two-dimensional subspace $\{|01\rangle, |10\rangle\}$. It transforms the state $|10\rangle$ into $\cos\left(\frac{\phi}{2}\right)|10\rangle - \sin\left(\frac{\phi}{2}\right)|01\rangle$.

Representative Matrix

$$U(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{\phi}{2}\right) & -\sin\left(\frac{\phi}{2}\right) & 0 \\ 0 & \sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Effect on Basis States

It affects the subspace $\{|01\rangle, |10\rangle\}$, performing a rotation parameterized by ϕ.

Double Excitation Gate (*DoubleExcitation*) This gate performs a rotation in the subspace of states $\{|0011\rangle, |1100\rangle\}$. It specifically affects these states, leaving the others unchanged.

Representative Matrix

$$U(\phi) = \begin{pmatrix} I_{12} & 0 & 0 \\ 0 & \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & -\sin\left(\frac{\phi}{2}\right) \\ \sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{pmatrix} & 0 \\ 0 & 0 & I_2 \end{pmatrix}$$

Effect on Basis States

It performs a rotation parameterized by ϕ in the subspace $\{|0011\rangle, |1100\rangle\}$.

These gates are implemented in PennyLane as `qml.SingleExcitation` and `qml.DoubleExcitation`, and are essential in quantum chemistry algorithms such as the *Unitary Coupled-Cluster Singles and Doubles* (UCCSD).